Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy

Zhan Qin[1,3*] zhanqin@buffalo.edu
Yin Yang[2] yyang@qf.org.qa
Ting Yu[1] tyu@qf.org.qa
Issa Khalil[1] ikhalil@qf.org.qa
Xiaokui Xiao[4] xkxiao@ntu.edu.sg
Kui Ren[3] kuiren@buffalo.edu

[1]Qatar Computing Research Institute, Hamad Bin Khalifa University, Qatar
[2]College of Science and Engineering, Hamad Bin Khalifa University, Qatar
[3]Department of Computer Science Engineering, State University of New York at Buffalo, USA
[4]School of Computer Science and Engineering, Nanyang Technological University, Singapore

ABSTRACT

In local differential privacy (LDP), each user perturbs her data locally before sending the noisy data to a data collector. The latter then analyzes the data to obtain useful statistics. Unlike the setting of centralized differential privacy, in LDP the data collector never gains access to the exact values of sensitive data, which protects not only the privacy of data contributors but also the collector itself against the risk of potential data leakage. Existing LDP solutions in the literature are mostly limited to the case that each user possesses a tuple of numeric or categorical values, and the data collector computes basic statistics such as counts or mean values. To the best of our knowledge, no existing work tackles more complex data mining tasks such as heavy hitter discovery over set-valued data.

In this paper, we present a systematic study of heavy hitter mining under LDP. We first review existing solutions, extend them to the heavy hitter estimation, and explain why their effectiveness is limited. We then propose LDPMiner, a two-phase mechanism for obtaining accurate heavy hitters with LDP. The main idea is to first gather a candidate set of heavy hitters using a portion of the privacy budget, and focus the remaining budget on refining the candidate set in a second phase, which is much more efficient budget-wise than obtaining the heavy hitters directly from the whole dataset. We provide both in-depth theoretical analysis and extensive experiments to compare LDPMiner against adaptations of previous solutions. The results show that LDPMiner significantly improves over existing methods. More importantly, LDPMiner successfully identifies the majority true heavy hitters in practical settings.

Keywords

Local Differential Privacy; Heavy Hitter

1. INTRODUCTION

Nowadays, with the advance of big data analytics, organizations have become increasingly interested in collecting and analyzing user data. For example, web browsers and mobile apps often collect system logs and usage patterns as a means to guide the development of future versions; crowdsourcing platforms, such as Mechanical Turk[1], also provide a convenient way to collect information from contributors. However, the collection of user data could incur significant privacy risks, as demonstrated in several past incidences, e.g., [15, 32], where accidental leakage of sensitive data led to public outrage, reputation damage, and legal actions against the data collector.

Local differential privacy (LDP) is the state-of-the-art approach to addressing the privacy concerns in data collection, which has been implemented in the Google Chrome browser [12]. Its main idea is to ensure that the data collector (i) never collects or possesses the exact values of any personal data, and yet (ii) would still be able to derive general statistics about the users. In particular, in LDP, each user locally perturbs her data under differential privacy [10], which is a strong and rigorous notion of privacy that provides plausible deniability; then, the user sends the perturbed data to the collector. As such, LDP protects both the users (against privacy risks) and the data collector itself (against damages caused by potential privacy breaches).

However, since LDP is a relatively new concept, existing solutions are mostly limited to obtaining statistics over a single numeric or categorical attribute, such as select-count [12], marginals [13] and histograms [3]. Similarly, multi-dimensional data analysis under LDP is also limited to simple numeric or categorical attributes, or the most basic types of aggregates, e.g., mean of each attribute [8]. As a consequence, existing solutions for LDP are inadequate for more

*This work was conducted while the first author was doing internship at Qatar Computing Research Institute.

[1]https://www.mturk.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS'16, October 24-28, 2016, Vienna, Austria

© 2016 ACM. ISBN 978-1-4503-4139-4/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2976749.2978409

complex types of data mining tasks. In particular, consider the problem of identifying *heavy hitters* over *set-valued data*, where (i) each user has a set of up to l items (e.g., web pages browsed, movies watched, locations visited, books purchased from an online store), and (ii) we aim to identify the top-k most frequent items among all users. Heavy hitter discovery is a well studied problem in data mining with numerous important applications, such as marketing analysis, cyber-attack detection and trend monitoring. As we show in Section 3, if we extend existing solutions to the heavy hitter problem, they would incur both prohibitively high communications overhead and low utility of result. One main reason for their inefficiency and ineffectiveness is that *each user reports to the data collector much information that is not relevant to the final result*, i.e., items not in the final top-k list. This incurs large communication costs, as well as a waste of the *privacy budget*, which is a key concept in differential privacy whose usage is negatively correlated with the utility of results.

To address the deficiency of the existing solutions, we propose LDPMiner, a novel algorithm for mining heavy hitters under local differential privacy. The main idea of LDPMiner is to employ a two-phase framework: the first phase identifies a candidate set for the top-k frequent items, and the second phase focuses the remaining privacy budget on refining the candidates, rather than spread it over all items in the universe. Both phases are non-trivial because (i) they operate on set-valued data, (ii) they involve only constant communication cost, and (iii) they achieve an error rate that is both empirically small and asymptotically optimal. In addition, there is synergy between the two phases: a user that reports the same items in both phases can do so with a reduced amount of perturbations, while still satisfying differential privacy. In-depth theoretical analysis and extensive experiments using real data confirm the effectiveness, efficiency, and practical value of LDPMiner.

The remainder of this paper is organized as follows. Section 2 provides the background on LDP and the existing LDP solutions for heavy hitter estimation. Section 3 formulates the problem and describes naive solutions. Section 4 presents the general framework of LDPMiner and elaborates on the algorithms used in the two phases of LDPMiner. Section 5 presents a thorough experimental evaluation. Section 6 surveys related work. Section 7 concludes the paper with directions for future work.

2. BACKGROUND

2.1 Local Differential Privacy

Local differential privacy (LDP) [22, 14] is a data collection framework based on ϵ-differential privacy [10]. Under this framework, each data contributor (e.g., a user of a website) locally perturbs her own data using a randomized mechanism that satisfies ϵ-differential privacy, before sending the noisy version of her data to a data collector. Specifically, a randomized mechanism $\mathcal{M}$ satisfies ϵ-differential privacy, if and only if for any two neighbor databases D and D' (explained shortly) that differ in exactly one record, and any possible output s of $\mathcal{M}$, we have the following inequality:

$$\frac{Pr[\mathcal{M}(D) = s]}{Pr[\mathcal{M}(D') = s]} \leq e^{\epsilon} \quad (1)$$

Intuitively, given an output s of mechanism $\mathcal{M}$, an adversary cannot infer with high confidence (controlled by ϵ) whether the input database is D or its neighbor D', which provides plausible deniability for individuals involved in the sensitive database. Here, ϵ is a system parameter called the *privacy budget* that controls the strength of privacy protection. A smaller ϵ signifies stronger privacy protection, since the adversary has lower confidence when it tries to distinguish between D and D'.

The concept of differential privacy was originally proposed for the setting where a trusted data curator, who possesses a database containing the exact data records from multiple individuals, publishes perturbed statistics derived from the database using a randomized mechanism. Hence, the definition of differential privacy involves the notion of "neighbor databases". In contrast, in LDP, there is no trusted data curator; instead, each individual perturbs her own data record under ϵ-differential privacy. In this situation, D and D' are two singleton databases, each containing exactly one record. Since two neighbor databases differ by exactly one record, essentially D and D' represent two arbitrary records. In the problem studied in this paper (described in Section 3), every record is a set of items, and each of D and D' represents such an item set.

An important property of differential privacy is *sequential composability*, which is elaborated by McSherry in [26]:

THEOREM 2.1. *Given t random mechanisms $\mathcal{M}_i$ ($1 \leq i \leq t$), each of which satisfies ϵ_i-differential privacy. Then, the sequence of $\mathcal{M}_i(D)$ satisfies $\left(\sum_{i=1}^{t} \epsilon_i\right)$-differential privacy.*

As pointed out in [26], *sequential composition* applies even when the t mechanisms are not independent, i.e., subsequent mechanisms can incorporate the outcomes of the preceding mechanisms. In other words, these mechanisms can be arbitrary functions of the input database and preceding outputs. Since record perturbation mechanisms in LDP satisfy differential privacy, they can also be sequentially composed. Accordingly, given a privacy budget ϵ, each user can partition ϵ into multiple portions and use each portion to release certain randomized information under differential privacy. This is the foundation of the proposed two-phase framework, elaborated in Section 4.

2.2 Existing LDP Solutions

An LDP solution includes both (i) a user-side data perturbation mechanism and (ii) an algorithm executed by the data collector for computing statistical information from the noisy data received from the users. The former one's main requirement is to satisfy differential privacy. Theoretically, every differentially private mechanism can be potentially applied to the LDP setting to be run at each user. In reality, however, most mechanisms for enforcing differential privacy are designed to run at a central data curator who has access to the exact records of all users, with the goal of publishing perturbed statistics based on these records. In LDP, neither assumptions holds, since the privacy protection algorithm is run by individual users who do not have access to other user's data, and that the information to be released is the perturbed data, not analysis results. Consequently, although the fundamental mechanisms for differential privacy such as the Laplace mechanism [10] and the geometric mechanism [16] can be applied to LDP, more sophisticated ones, e.g., for the frequent itemset mining [25], cannot be applied

to LDP as they require global knowledge of all users' data. Further, perturbing the data at the user side is only part of the solution; traditional differentially private mechanisms do not address the other important problem in LDP, i.e., computing statistics at the data collector based on individuals' noisy data. Hence, conventional solutions for enforcing differential privacy are largely inadequate for the LDP setting. In the following, we overview LDP solutions, which have attracted significant attention fairly recently.

Randomized Response. It has been pointed out (e.g., in [12]) that a classic *randomized response* (*RR*) technique commonly used in statistics can be adapted to LDP. Specifically, RR asks each user a sensitive question whose answer can be either yes or no, e.g., "Are you HIV positive?" The goals are (i) that each user answers the question with plausible deniability, and (ii) that the data collector can compute an unbiased estimate of the percentage of users whose answer is "yes" (resp. "no"). To do so, each user flips a coin before answering the question. If the coin turns head, the user provides her true answer (let's say it is "yes"); otherwise, she reports the opposite of her true answer (i.e., "no"). To adapt RR to satisfy LDP, we make the coin biased, with a probability p (resp. $1-p$) to turn head (resp. tail). It has been proven (e.g., in [12]) that RR satisfies ϵ-differential privacy with the following value of p:

$$p = \frac{e^{\epsilon}}{1+e^{\epsilon}} \tag{2}$$

Next we clarify how the data collector in RR estimates the percentage of users with an "yes" answer. If the data collector simply outputs the percentage of "yes" among the noisy answers (denoted by c), the results would be *biased*, due to the random perturbations performed at the users. To correct this, the data collector reports the following adjusted estimate:

$$c' = c \times c_{\epsilon}, \text{where } c_{\epsilon} = \frac{1}{1-2p} \tag{3}$$

RR is limited to the case where each user answers a binary question. Nevertheless, it is a fundamental building block for more sophisticated LDP solutions, explained below.

RAPPOR. RAPPOR [12, 13] extends RR to more complex data types at the user as well as more sophisticated statistics at the data collector. The most relevant problem addressed by RAPPOR is frequency estimation over categorical data. Specifically, let n be the total number of users; each user u_i ($1 \leq i \leq n$) possesses exactly one item v_i in a domain containing d possible items, and the data collector aims to estimate the frequency of each of the d items. In RAPPOR, user u_i represents v_i using a length-d bit vector, whose bits are all zero except for the v_i-th bit, which is one. Then, for each of the d bits, user u_i applies RR independently with a biased coin with probability p, whose value is clarified below. The data collector, upon receiving a length-d bit vector from each of the n users, computes an unbiased frequency estimate for each of the d items by applying RR independently.

To determine the value of p (i.e., the probability that a user reports the true value for each of the d bits), RAPPOR utilizes an important concept in differential privacy called *sensitivity*. Specifically, given an arbitrary function f, the sensitivity Δ_f of f is defined as

$$\Delta_f = max_{D,D'} \|f(D) - f(D')\|_1 \tag{4}$$

where D and D' are two arbitrary neighbor databases, and $\|\cdot\|_1$ represents $\mathcal{L}_1$ norm of a vector. In RAPPOR, D and D' are two arbitrary records, i.e., any two different items, and $f(D)$ (resp. $f(D')$) is the true value of the length-d bit vector corresponding to the item of D (resp. D'). Since such a bit vector contains a single bit of one, the maximum difference between $f(D)$ and $f(D')$ are two bits. Hence, the sensitivity of f is 2. Accordingly, [12] proves that RAPPOR ensures differential privacy with the following value of p:

$$p = \frac{e^{\frac{\epsilon}{2}}}{1+e^{\frac{\epsilon}{2}}} \tag{5}$$

Comparing Equations (2) and (5), the latter calibrates the noise according to the sensitivity of RAPPOR, i.e., 2. We use the same methodology to extend RAPPOR to our problem in Section 3. A main drawback of RAPPOR is its high communication overhead, i.e., each user needs to transmit d bits to the data collector, which can be expensive when the number of possible items d is large. For instance, in an application where each user reports its homepage to the data collector, the number of items d is the total number web pages in the entire Internet, leading to prohibitively high communication costs. Further, the original proposal of RAPPOR is limited to a single categorical attribute, and it cannot be directly applied to our problem with set-valued data.

Succinct histogram. Succinct histogram (SH) [3] addresses the heavy communication problem incurred by previous solutions such as RAPPOR. Specifically, SH targets the same setting described above, i.e., each user possesses exactly one item out of d possible items, and the data collector estimates the frequency of each item. In particular, in SH the data collector only reports items with relatively high frequency values (above a given threshold); for all other items, SH simply considers their frequency as zero. [3] proves that SH achieves asymptotically optimal accuracy. Interestingly, no previous work has compared SH with RAPPOR in terms of result accuracy; we provide both theoretical and experimental comparisons between SH and RAPPOR in this paper.

SH is based on two key ideas. First, instead of reporting d bits as is done in RAPPOR, in SH each user only reports one randomly chosen bit. Although not explicitly stated in [3], this idea only works when $d < n$, where n is the number of users, and we show how this is done later in the paper. Since each user reports only one bit, RR directly applies with parameter p computed using Equation (2), which is higher than that in RAPPOR (Equation (5)). Meanwhile, the communication cost of SH between each user and the data collector is $O(1)$ rather than $O(d)$ as in RAPPOR.

Second, in order for the first idea to work when $d > n$, SH applies random projection [5] on the data to reduce its dimensionality from d to $m = O(n)$ in a preprocessing step. Specifically, SH generates a $d \times m$ matrix Φ which each element is chosen independently and uniformly at random from two possible values: $\frac{1}{\sqrt{m}}$ and $-\frac{1}{\sqrt{m}}$. Note that elements in Φ are essentially binary values. Then, each user multiplies its length-d bit vector with Φ, obtaining a length-m vector. After that, she applies the first idea, i.e., choosing one random bit and then releasing it using RR.

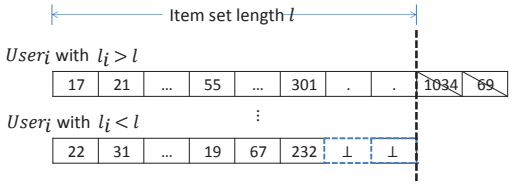


Figure 1: Itemset padding and truncation

Specifically, in [3], the authors prove that general histogram and heavy hitter estimation have the same asymptotic lower bound in terms of estimation error, and SH achieves a matching upper bound. However, [3] focuses on a different problem setting: each user has at most one item, and the heavy hitter set is defined with a given frequency threshold. Hence, the theoretical results in [3] do not apply to our problem. Extending SH to our problem effectively is non-trivial, which we elaborate in Section 4.2.

3. PROBLEM DESCRIPTION AND NAIVE SOLUTIONS

Problem description. This paper focuses on heavy hitter discovery and frequency estimation over set-valued data. Specifically, each user u_i possesses a set $\mathbf{v}_i$ of items. For simplicity, we assume that each user has a fixed number l of items, i.e., $\forall i, \mathbf{v}_i = l$, where l is a system parameter. If a user has less than l items, she pads her item set $\mathbf{v}_i$ with dummy items, which are ignored by the data collector, i.e., the latter does not compute frequency of the dummy item and does not consider it in the heavy hitter set. Conversely, if a user possesses more than l items, she randomly draws l samples without replacement from her item set, forming a new set with exactly l items. If items are already in random order in her original item set, the user can simply truncate the set to l items, as shown in Figure 1. Because such truncation loses information and introduces bias to the results, in general l should be set to a reasonable large value so that most users (i.e., except for a few outliers) have no more than l items in the original data.

Let n be the total number of users, d be the total number of different items, and f_j be the frequency of the j-th item (denoted as v_j), i.e., the portion of users possessing item v_j ($1 \leq j \leq d$). Formally, we have:

$$f_j = \frac{|\{u_i | v_j \in \mathbf{v}_i, 1 \leq i \leq n\}|}{n}$$

The data collector aims to find the top-k items with the highest frequency f_j, along with the frequency estimate for each such item. We assume that $k \ll d$, i.e., the result only contains the top few heavy hitters. Meanwhile, for many applications, k is also significantly smaller than l, the maximum number of items that each user has. The main goal is to obtain high accuracy of the results while satisfying LDP. Here result accuracy has two aspects. First, the relative ranking of the top-k items should be close to their actual ranking. Second, the error of the estimated frequencies for the reported heavy hitters should be minimized. We elaborate on the error metrics in Section 5.

Naive solutions. Next we describe two naive solutions obtained by adapting RAPPOR and SH described in Section 2.2 to our problem, respectively. We first focus on RAPPOR. Recall that RAPPOR requires each user to send a perturbed bit to the data collector, where the perturbation is based on RR (also described in Section 2.2) using a biased coin with probability p. The key, therefore, is to determine the value of p in our problem. To do so, we analyze the sensitivity of releasing a length-d bit vector at each user. Since each user possesses exactly l items, there are exactly l ones in the true bit vector; all other bits are zero. Therefore, two such bit vectors can differ by at most $2l$ bits, meaning that the sensitivity is $2l$. According to RAPPOR [12], we have:

$$p = \frac{e^{\frac{\epsilon}{2l}}}{1 + e^{\frac{\epsilon}{2l}}} \tag{6}$$

Similar to the case with categorical values, RAPPOR incurs a transmission cost of $O(d)$ for each user, which can be rather expensive for large domains. Furthermore, when l is relatively large, RAPPOR incurs rather high sensitivity, and, thus, heavy perturbation of the released bit vectors, leading to low accuracy.

Next we describe the adaptation of SH to our problem. Unlike RAPPOR, SH is limited to the case where each user possesses exactly one item. In order to handle set-valued data, we apply SH l times using sequential composition (refer to Section 2.1). Specifically, every user divides its privacy budget ϵ into l equal portions of $\frac{\epsilon}{l}$ each. Then, the user invokes SH l times, one for each of the l items she possesses, using privacy budget $\frac{\epsilon}{l}$. According to sequential composition, doing so for all l items satisfies ϵ-local differential privacy.

The data collector receives l independent data releases from each user. It then proceeds to generate l succinct histograms which correspond to l invocations of SH. After that, it aggregates these succinct histograms into one histogram. In particular, let f_j^k ($1 \leq j \leq d, 1 \leq k \leq l$) be the frequency estimate for item v_j from the k-th invocation of SH. The data collector returns $\sum_{k=1}^{l} f_j^k$ as the final frequency estimate for item v_j.

The above extension of SH incurs communication overhead $O(l)$ for each user, i.e., for applying SH l times with $O(1)$ transmissions each. Since $l \ll d$ in most applications, this method is less expensive in terms of communications compared to the adaptation of RAPPOR. Nevertheless, it can still be costly for large values of l, i.e., a user can possess a large number of items. Regarding result accuracy, when l is large, each portion of the privacy budget $\frac{\epsilon}{l}$ is rather small, leading to heavy perturbation and low accuracy similarly as in RAPPOR. Worse, aggregating multiple succinct histograms at the data collector also accumulates error. For this reason, the accuracy performance of this method is often lower than that of RAPPOR, as we show in the analysis in Section 4.2. Overall, neither naive approach is efficient (in terms of communications) or effective (in terms of result utility). Next we present the proposed solution LDPMiner, which addresses these problems.

4. LDPMINER

This section presents the proposed solution LDPMiner, which achieves both low communication overhead and high result accuracy. The main ideas include a novel two-phase framework, novel LDP mechanisms that are used as building blocks in these two phases, and further optimizations

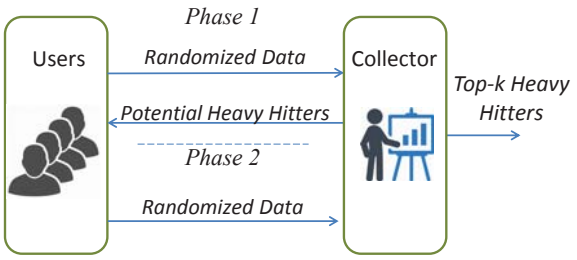


Figure 2: Two-Phase Framework in LDPMiner

for each of the two phases. In the following, Section 4.1 overviews the general framework of LDPMiner. Section 4.2 presents the building blocks. Sections 4.3 and 4.4 further optimize the two phases, respectively.

4.1 Two-Phase Framework

Recall from Section 3 that naive solutions incur high communication overhead and noisy results mainly due to the fact that each user possesses l items. In particular, naive RAPPOR has a sensitivity proportional to l, and naive SH divides the privacy budget into l shares; the amount of transmissions in naive SH is also $O(l)$. Observe that if we can reduce l, then we can effectively reduce the noise of both naive solutions as well as the transmission cost of naive SH. The main idea of the two-phase framework is to first filter the items and select $k_{max} = O(k)$ candidate heavy hitters in the first phase, and then focuses on refining the frequency estimates of these candidates in the second phase. Note that k is usually considerably smaller than l, as explained in Section 3. Figure 2 illustrates the proposed framework in LDPMiner.

Specifically, LDPMiner splits the privacy budget ϵ at each user into two parts ϵ_1 and ϵ_2, and allocates them to Phase I and Phase II respectively. The whole process satisfies ϵ-LDP according to the sequential composition property described in Section 2.1. Phase I addresses exactly the same problem as the original one, i.e., each user possesses a set of l items and reports them under ϵ_1-differential privacy, and the data collector computes the top k_{max} items with the highest frequency, and outputs these items as the candidate set. After the first phase finishes, the data collector broadcasts the set of candidate items to all users.

The task in Phase II also resembles the original problem, with the key difference that the universe of items becomes the k_{max} candidates, instead of the full set of d items. In other words, the final result consists of top-k heavy hitters that are chosen exclusively among the candidates; any item outside the k_{max} candidates are essentially considered a dummy item in Phase II, and LDPMiner does not compute the frequency of such an item. Another important distinction from Phase II and the original problem is that in the former, each user has already transmitted data in Phase I, and there can be overlap in the information released between the two phases. As we show later in Section 4.4, LDPMiner exploits such overlap between the two phases to further improve accuracy.

Note that the two-phase framework in LDPMiner is flexible: if we set $\epsilon_1 = \epsilon$, $\epsilon_2 = 0$ and $k_{max} = k$, then the second phase does not occur, and LDPMiner reduces to a single-phase approach. In our experiments, we compare the proposed two-phase LDPMiner with this single-phase alternative and demonstrate the benefits of the two-phase framework. It remains to clarify the LDP mechanisms used in Phase I and Phase II. These mechanisms are built upon a few basic blocks, described in the next subsection.

4.2 Building Blocks

In both phases, each user needs to report her items to the data collector, in order for the latter to derive frequency estimates and compute the set of candidates in Phase I and final heavy hitter sets in Phase II. We observe that the naive solutions described in Section 3 are ineffective and inefficient, because *each user tries to report all items she possesses*, which leads to high sensitivity in naive RAPPOR, as well as tiny privacy budget share for each item and high communication overhead in naive SH. One key idea in the LDPMiner building blocks, inspired by another technique [27] for multi-dimensional data analysis under LDP, is that *each user does not report all items; instead, she chooses a random item and reports it.* Such random sampling leads to biased item frequency estimates as each user now reports one instead of l items. To compensate for this, the data collector multiplies the estimated frequencies by l. As we show soon in our analysis, this leads to an unbiased estimate of true item frequencies. We call this approach *sampling randomizer*. Algorithm 1 shows the pseudo code for the proposed sampling randomizer method which implements the above idea.

Algorithm 1: Sampling randomizer algorithm

Input : Privacy parameter ϵ, item set $\mathbf{v}_i$ for each user u_i, number of heavy hitters k;
Output: Top-k heavy hitters HH_k;
1 **for** *user* $i \to n$ **do**
2 Pick j from range $[1, l]$ uniformly at random ; Apply a single-item randomizer to obtain $z^i = \mathcal{R}(\mathbf{v}_i[j], \epsilon)$;
3 Submit z^i to the data collector;
4 **end**
5 Data collector: Estimator $HH_k = \mathcal{E}(\sum_{i,j} z^i l, \epsilon)$;
6 Return HH_k;

Note that Algorithm 1 is a general framework for computing heavy hitters over set-valued data under LDP; in particular, the randomizer at each user (line 2) and the heavy hitter computation at the data collector (line 5) are left as black boxes. These black boxes can be fulfilled with RAPPOR, SH, or any method for heavy hitter estimation in the single-item setting. We call the sampling randomizer algorithm with RAPPOR (resp. SH) sampling RAPPOR (sampling SH) respectively. Next we compare the result accuracy of the naive solutions and the sampling-randomizer-based solutions through theoretical analysis.

Utility Analysis. Because the accuracy of top-k set is not easy to quantify, we analyze result utility in terms of the mean squared error of the item frequency estimates. Intuitively, for methods without special considerations for the heavy hitters such as the naive solutions (Section 3) and sampling randomizer (Algorithm 1), more accurate frequency estimates are expected to lead to more accurate heavy hitter estimation. A direct evaluation on the accuracy of the heavy hitter set is provided by the experiments in Section 5.

Analysis of naive RAPPOR. According to Section 3, the biased coin used in naive RAPPOR has probability $\frac{1}{e^{\epsilon/2l}+1}$ to turn heads. Using this probability into the similar deduction of original RAPPOR's utility analysis, we obtain

PROPOSITION 1. *For the private frequency estimation under naive RAPPOR and its estimator given above, for all ϵ, l, and n:*

$$\mathbb{E}\left[\|\tilde{f} - f\|_2^2\right] = \frac{e^{\epsilon/2l}}{n(e^{\epsilon/2l}-1)^2}$$

Analysis of naive SH. Under naive SH, for the simplicity and analysis, we assume that every item gets assigned to an interference-free channel. Hence, in each one of these channels, we run a frequency oracle and use the resulting frequency oracle to estimate the frequencies of all the items on the above list, which is an ideal scenario for SH. Based on the randomization procedures described in [3], we have the following result:

PROPOSITION 2. *For the private frequency estimation under naive SH, for all ϵ, l, and n:*

$$\mathbb{E}\|\tilde{f} - f\|_2^2 = \frac{1}{n^2}\left(fn\left(c_{\epsilon/l}^2 - 1\right) + 4\left(l - f\right)^2 n^2 \frac{e^{\epsilon/l}}{m(e^{\epsilon/l}+1)^2}\right)$$

where $m = \frac{\log(d+1)\log(2/\beta)}{\log(d)+\log(2/\beta)}(\epsilon/l)^2 n$, $c_{\epsilon/l} = \frac{e^{\epsilon/l}+1}{e^{\epsilon/l}-1}$, and β is a confidence parameter in SH.

Sampling RAPPOR and sampling SH. Unlike the naive approaches, in sampling-randomizer-based methods the randomizer enjoys full privacy budget ϵ, as well as the same sensitivity as in the single-item setting. We have the following result:

PROPOSITION 3. *For the private frequency estimation under sampling RAPPOR and its estimator given, for all ϵ, l, and n:*

$$\mathbb{E}\|\tilde{f} - f\|_2^2 = \frac{l^2 e^{\epsilon/2} + f(e^{\epsilon/2}-1)^2(l-1)}{n(e^{\epsilon/2}-1)^2}$$

PROPOSITION 4. *For the private frequency estimation under sampling SH, for all ϵ, l, and n:*

$$\mathbb{E}\|\tilde{f} - f\|_2^2 = \frac{l^2}{n^2}\left(\frac{fn}{l}\left(c_\epsilon^2 - 1\right) + 4\left(1 - \frac{f}{l}\right)^2 n^2 \frac{e^\epsilon}{m(e^\epsilon+1)^2}\right)$$

where $m = \frac{\log(d+1)\log(2/\beta)}{\log(d)+\log(2/\beta)}(\epsilon)^2 n$, and $c_\epsilon = \frac{e^\epsilon+1}{e^\epsilon-1}$,

Figure 3 plots the frequency estimation error obtained through the above analyses as a function of the privacy budget ϵ. Note that the vertical axis (i.e., the error) is in logarithmic scale. According the results, the sampling-randomizer-based methods have clear and consistent performance advantages over their naive counterparts. In particular, sampling RAPPOR obtains the lowest error in its reported frequency estimates.

In terms of communication costs, both versions of RAPPOR require each user to transmit a length-d bit vector to the data collector, leading to a communication cost of $O(d)$. Naive SH incurs $O(l)$ transmissions as explained in Section 3. Sampling SH, on the other hand, achieves constant communication cost since it invokes SH exactly once. Therefore, sampling RAPPOR and sampling SH provide different

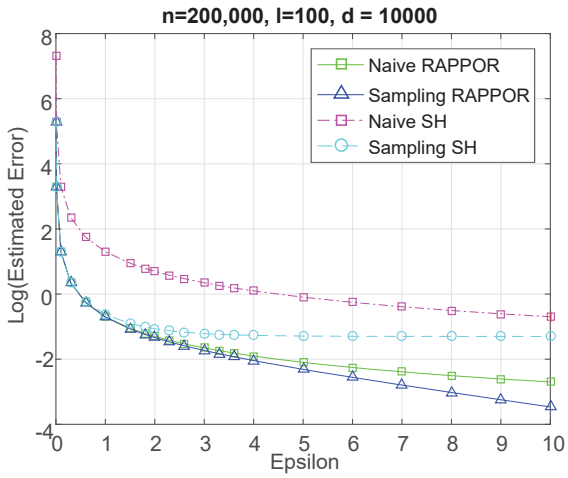


Figure 3: Error in the frequency estimates of basic solutions

tradeoffs between result accuracy and communication costs. Hence, we use both sampling-randomizer-based methods as basic building blocks in LDPMiner.

4.3 Design of Phase I

Phase I of LDPMiner can be implemented using either sampling RAPPOR or sampling SH, described in Section 4.2. We choose sampling SH since its communication cost is $O(1)$; sampling RAPPOR, on the other hand, incurs $O(d)$ transmissions which is prohibitively expensive for large domains.

Algorithm 2 shows the complete sampling SH algorithm. Note that the SH component is similar to the SH algorithm presented in [3], and here we present the full algorithm for completeness. In particular, in step 4, the random projection matrix Φ is generated by utilizing the Johnson-Lindenstrauss (JL) lemma [20], which states that any set of d points in a high dimensional space can be obliviously embedded into a pace of dimension $O(\log(d))$ such that this embedding preserves pairwise distances with probability at least $1-\beta$, where β is the confidence parameter. For this instance, $m = \frac{\log(d+1)\log(2\beta)\epsilon_1^2 n}{\log(ed/\beta)}$. Note that an element in $x = \Phi b_v$ is either $1/\sqrt{m}$ or $-1/\sqrt{m}$. And the utilization of JL transform requires for the columns of Φ to be $min(n,d)$-wise independent.

After that, the data collector aggregates each submitted individual reports z_t in its corresponding position t together:

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$$

and constructs a frequency oracle $FO = (\Phi, \bar{z})$. An item $v \in V$ whose frequency can be estimated through $\hat{f}(v) = \langle \Phi x_i, \bar{z}l \rangle - \gamma$, where γ is an approximation parameter. We refer the reader to [3] for further details, including certain parameter setting of frequency estimator.

Analysis. The following theorem establishes the correctness of sampling SH.

THEOREM 4.1. *The construction of the frequency oracle FO described above is ϵ_1-LDP.*

Algorithm 2: Sampling SH

Input : Privacy parameter ϵ_1, l-dimensional vector $\mathbf{v}_i$.
Output: Report pair (z_j, j).

1 Uniformly pick an element v from $\mathbf{v}_i$;
2 Uniformly pick a position t from 1 to m;
3 **if** $v \neq \perp$ **then**
4 $\quad x = \Phi b_v$;
5 $\quad$ Randomize the t-th bit x_t as follows:

$$z_t = \begin{cases} c_{\epsilon_1} m x_t & \text{with probability } \frac{e^{\epsilon_1}}{e^{\epsilon_1}+1} \\ -c_{\epsilon_1} m x_t & \text{with probability } \frac{1}{e^{\epsilon_1}+1} \end{cases}$$

$\quad$ where $c_{\epsilon_1} = \frac{e^{\epsilon_1}+1}{e^{\epsilon_1}-1}$;
6 **else**
7 $\quad$ Uniformly pick $z_t \in \{-c_{\epsilon_1}\sqrt{m}, c_{\epsilon_1}\sqrt{m}\}$;
8 **end**
9 Return the pair (z_t, t);

Proof. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be two arbitrary sets of items such that $|\mathbf{v}_1| = |\mathbf{v}_2| = l$. Let $\mathcal{A}$ denote Algorithm 2, and (z_t, t) be any possible output of $\mathcal{A}$. Note that $z_t \in \{-c_{\epsilon_1}\sqrt{m}, c_{\epsilon_1}\sqrt{m}\}$ and $t \in \{1, 2, \ldots, m\}$. We will prove the theorem by showing that

$$\frac{\Pr[\mathcal{A}(\mathbf{v}_1) = (z_t, t)]}{\Pr[\mathcal{A}(\mathbf{v}_2) = (z_t, t)]} \leq e^{\epsilon_1}.$$

Let $\Pr[t' \mid \mathbf{v}_i]$ denote the probability that Algorithm 2 picks $t = t'$ in Line 2, when $\mathbf{v}_i$ is the input. We have $\Pr[t' \mid \mathbf{v}_i] = 1/m$ for any t'. In addition, let $\Pr[z'_t \mid t' \wedge \mathbf{v}_i]$ denote the probability that, after Algorithm 2 picks $t = t'$ in Line 2 given $\mathbf{v}_i$ as the input, it outputs (z'_t, t') eventually. We have

$$\frac{1}{e^{\epsilon_1}+1} \;\leq\; \Pr[z'_t \mid t' \wedge \mathbf{v}_i] \;\leq\; \frac{e^{\epsilon_1}}{e^{\epsilon_1}+1}.$$

Therefore,

$$\begin{aligned} \frac{\Pr[\mathcal{A}(\mathbf{v}_1) = (z_t, t)]}{\Pr[\mathcal{A}(\mathbf{v}_2) = (z_t, t)]} &= \frac{\Pr[z_t \mid t \wedge \mathbf{v}_1] \cdot \Pr[t \mid \mathbf{v}_1]}{\Pr[z_t \mid t \wedge \mathbf{v}_2] \cdot \Pr[t \mid \mathbf{v}_2]} \\ &= \frac{\Pr[z_t \mid t \wedge \mathbf{v}_1]}{\Pr[z_t \mid t \wedge \mathbf{v}_2]} \;\leq\; e^{\epsilon_1}. \end{aligned}$$

Thus, the theorem is proved. □

To quantify the proposed solution's utility, we use a standard notion of approximation for frequent item mining. We use γ as an additive error to frequency f and f_k as the frequency of the k-th most frequent item in the dataset. Given $\beta > 0$, with probability at least $1 - \beta$, the algorithm is (γ, β, η)-approximate, if (1) *Soundness*: no item in the output has true frequency less than $f_k - \gamma$, (2) *Completeness*: all items with true frequency greater than $f_k + \gamma$ is in the output, and (3) *Accuracy*: The noisy frequency estimated differs at most η from the true frequency.

The following theorem establishes the soundness and completeness of sampling SH, and analyzes its accuracy.

Theorem 4.2. *For all $\beta > 0$, with probability at least $1 - \beta$, all items estimated by sampling SH mechanism in Phase I have their true frequencies $> \hat{f}_k - \gamma$, and all items with true frequency $> \hat{f}_k - \gamma$ are in the list, where $\gamma = \frac{2(e^{\epsilon_1}+1)}{e^{\epsilon_1}-1} \cdot \sqrt{\frac{\log(d/\beta)l}{n}}$.*

4.4 Design of Phase II

For Phase II, we can simply use sampling SH as in Phase I. Meanwhile, since the universe of items becomes the k_{max} candidates as explained in Section 4.1, sampling RAPPOR also becomes a viable solution, whose communication cost is now $O(k_{max}) \ll d$. We refer to LDMiner with sampling SH (resp. RAPPOR) in the second phase as LDPMiner-SH (resp. LDP-RAPPOR).

One complication in Phase II is that the trimming of users' answer still needs to be handled carefully. For example, the change of item set length after trimming also contains private information. Since user excludes non-candidate items from their answers, the change of item set length indicates the number of frequent items in a user's answer. This information will give the adversary additional advantage on certain items' probability distribution, violating LDP. LDPMiner uses a relatively conservative solution for this problem: after trimming, each user needs to pad her item set to l items with *dummy* items. For sampling RAPPOR, an *empty* item is a length-k_{max} binary vector filled with zeroes. After trimming, the user performs sampling RAPPOR or SH with privacy parameter ϵ_2.

Optimization. There is one optimization in Phase II that improves the accuracy of LDPMiner-SH, in which both phases apply the sampling SH algorithm. Recall from Section 4.2 that in sampling SH, each user u_i randomly chooses one item $v \in \mathbf{v}_i$ to report. The optimization applies when both phases select the same item. This happens when (i) u_i chooses item v in Phase I; then (ii) the aggregator identifies v as a top-k_{max} heavy hitter; finally (iii) in Phase II , v remains in $\mathbf{v}_i$, and u_i again chooses v to report in its second invocation of sampling SH. Observe that in (i) and (iii), u_i reports to the aggregator two different perturbed versions of the same item, i.e., $v \in \mathbf{v}_i$, under ϵ_1-LDP and ϵ_2-LDP, respectively.

In the LDPMiner algorithm descried so far, only the second perturbed version (reported in Phase II) is used at the aggregator, and the first one (reported in Phase I) is simply discarded. Consequently, the privacy budget ϵ_1 spent on v in Phase I is wasted. To avoid this waste, we apply a special perturbation technique in [29] in Phase II. In particular, this technique takes as input both the exact item v and its perturbed version reported in Phase I (using privacy budget ϵ_1), and outputs another perturbed version using privacy budget $\epsilon = \epsilon_1 + \epsilon_2$. According to [29], the noise added in the two phases are correlated, such that the two perturbed releases as a whole still satisfies ϵ-differential privacy, thus eliminating any waste of privacy budget, and improving the accuracy of the second perturbed version of v. In addition, LDPMiner applies consistent hashing [21] in the random selection of items in the two phases, so that each user is more likely to report on the same item in both phases.

Finally, the data collector integrates the users' reports from both phases into one frequency estimation to utilize the overlap in the information released between two phases. Note that to get an unbiased frequency estimation, the collector needs to adjust the frequency estimation generated from the Phase II following its estimation equations and then multiply $l-1$. After that, the final estimation is generated by aggregating frequency estimations from two phases together, i.e., $\hat{f} = (\hat{f}_1 + (l-1)\hat{f}_2)/l$. Then the top-$k$ heavy hitters in this final estimation is the output of the whole two phase framework.

Analysis. The following theorem establish the correctness of Phase II.

THEOREM 4.3. *The frequency estimation preocedure described in Section 4.4 is ϵ_2-LDP.*

PROOF. The frequency estimation procedure described in Section 4.4 can be regarded as an algorithm $\mathcal{B}$ whose input includes (i) a set $\mathbf{v}_i$ of l items, (ii) a privacy parameter ϵ_2, (iii) a set $\mathcal{V}_c$ of k_{max} candidate items. The algorithm first *trims* $\mathbf{v}_i$ by replacing any item not in $\mathcal{V}_c$ with a dummy item $\perp$. Let $\mathbf{v}_i'$ denote the version of $\mathbf{v}_i$ thus obtained. After that, the algorithm applies either sampling RAPPOR or SH on $\mathbf{v}_i'$ using $\mathcal{V}_c \cup \{\perp\}$ as the item alphabet, and then submits the output to the collector. We use $\mathcal{C}$ to denote the method applied on $\mathbf{v}_i'$. Note that $\mathcal{C}$ satisfies ϵ_2-LDP.

Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be any two item sets with l items before trimming, and o be any output of $\mathcal{C}$ submitted to the collector. We will prove the theorem by showing that

$$\frac{\Pr[\mathcal{B}(\mathbf{v}_1, \epsilon_2, \mathcal{V}_c) = o]}{\Pr[\mathcal{B}(\mathbf{v}_2, \epsilon_2, \mathcal{V}_c) = o]} \leq e^{\epsilon_2}.$$

Since the trimming procedure given $\mathcal{V}_c$ is deterministic,

$$\Pr[\mathcal{B}(\mathbf{v}_i, \epsilon_2, \mathcal{V}_c) = o] = \Pr[\mathcal{C}(\mathbf{v}_i', \epsilon_2, \mathcal{V}_c) = o].$$

Meanwhile, since $\mathcal{C}$ satisfies ϵ_2-LDP, we have

$$\frac{\Pr[\mathcal{B}(\mathbf{v}_1, \epsilon_2, \mathcal{V}_c) = o]}{\Pr[\mathcal{B}(\mathbf{v}_2, \epsilon_2, \mathcal{V}_c) = o]} = \frac{\Pr[\mathcal{C}(\mathbf{v}_1', \epsilon_2, \mathcal{V}_c) = o]}{\Pr[\mathcal{C}(\mathbf{v}_2', \epsilon_2, \mathcal{V}_c) = o]} \leq e^{\epsilon_2}.$$

Therefore, the theorem is proved. □

THEOREM 4.4. *The LDPMiner algorithm satisfies ϵ-LDP.*

PROOF. According to the proofs of Theorems 4.1 and 4.3, Phase I and Phase II of LDPMiner apply to the same input data. Specifically, for user u_i, her input to Phase I is clearly $\mathbf{v}_i$, i.e., her set of items. Meanwhile, $\mathbf{v}_i$ is also u_i's input to Phase II; note that the trimming and padding operations on $\mathbf{v}_i$ are a part of Phase II, which as a whole satisfies ϵ_2-LDP according to Theorem 4.3. Since Phase I and Phase II satisfy ϵ_1-LDP and ϵ_2-LDP, respectively, by sequential composition (Theorem 2.1), applying these two mechanisms in a sequence on the same data satisfies ϵ-differential privacy, where $\epsilon = \epsilon_1 + \epsilon_2$. Thus, the theorem is proved. □

Next we show the theoretical guarantees that with high probability, the reported frequencies of the heavy hitters are close to their true frequencies. First, we provide the error bound of the reported heavy hitter's frequency.

THEOREM 4.5. *For all $\beta > 0$, with probability at least $1 - \beta$, the noisy frequency estimated by sampling RAPPOR mechanism in Phase II differs at most η from the true frequency, where $\eta = \frac{l^2}{fn} \log(\frac{1}{2\beta})$.*

THEOREM 4.6. *For all $\beta > 0$, with probability at least $1 - \beta$, all items output by the two-phase have their true frequencies $> f_k - \gamma$, all items with true frequency $> f_k - \gamma$ are in the list, and all noisy frequencies differ by at most η from the corresponding true frequencies, where $\gamma = \frac{4(e^\epsilon+1)}{e^\epsilon-1}\sqrt{\frac{\log(d/\beta)}{nl}}$, and $\eta = \frac{l^2-l}{f\epsilon ne^2} ln(\frac{1}{2\beta}) + \frac{2(e^{\epsilon_1}+1)}{e^{\epsilon_1}-1}\sqrt{\frac{ln(d/\beta)}{nl}}$).*

Complexity analysis. In LDPMiner, the total communication cost between each user and the server is $O(\log(d)+k)$, where d is the total number of items in the domain, and k is the number of heavy hitters. Meanwhile, for each user, the total computation cost for generating a randomized report is $O(k + l)$. The additional computation/communication overhead of the proposed two-phase mechanism is constant compared with a single-phase mechanism.

5. EXPERIMENTAL EVALUATION

5.1 Setup

We design experiments to study the effectiveness of LDPMiner in terms of the accuracy of estimated heavy hitters. In particular, we want to understand (1) how much LDPMiner improves over simple extension of existing LDP mechanisms; and (2) how key parameters would affect the accuracy of LDPMiner's estimation of heavy hitters. Towards this goal, we run experiments over both synthetic and real datasets: the synthetic datasets allow us to observe the impact of data distributions on LDPMiner in a systematic way, while the real datasets would show the utility of LDPMiner in a more practical setting. Note that for a simple expression in figures, LDPMiner-RAPPOR and LDPMiner-SH are abbreviated as LM-RP and LM-SH respectively. And sampling RAPPOR and SH are abbreviated as RP and SH, since each of them outperforms other naive extensions of RAPPOR and SH respectively.

5.1.1 Datasets

Synthetic datasets. We generate two synthetic datasets following the normal distribution and the Laplace distribution respectively. Intuitively, the more "skewed" the heavy hitters are (i.e., the frequencies of heavy hitters are much higher than non-heavy hitters), the stronger the signal is, and thus the more easily for an LDP mechanism to identify heavy hitters and estimate their frequencies. Given the same mean and standard deviation, the Laplace distribution is more skewed than the normal distribution, which enables us to clearly understand the impact of data distribution on an LDP mechanism. In our experiments, the total number of items d is set to 1000. Both distributions are with a mean of 500 and a standard deviation of 100. Given l, the size of an item set, for each user we randomly pick l different items following the target distribution. Clearly for both datasets their heavy hitters are around item 500.

Real Datasets. We conduct experiments on two publicly available set-valued datasets.

- **AOL search log dataset [1].** This dataset contains the keyword search histories of a subset of AOL users. The real user names are replaced with a random user ID. We treat each keyword as an item. The dataset contains the search log of 647,377 users and 2,290,685 different keywords after removing stop words and performing word stemming. 90% of the users have fewer than 84 keywords in their logs.

- **Kosarak dataset [2].** This dataset contains the clickstream data of a Hungarian online news website. Each user is associated with a set of clicked URLs. There are 990,002 users and 41,270 different URLs. 90% of users have less than 66 URLs.

5.1.2 Experiment Parameters

The effectiveness of LDPMiner could be affected by several parameters.

n, the number of users. As a significant amount of noise is added to each individual user's data in local differential privacy, a large population would be needed to effectively remove the noise and reveal the true heavy hitters.

ϵ, the privacy budget. Though the tradeoff between ϵ and utility is clear, due to the nature of local differential privacy, we expect that a much larger privacy budget is needed (compared with the traditional centralized setting) in order to have reasonable estimations of heavy hitters.

k, the number of top heavy hitters. We expect that LDPMiner to perform well when k is relatively small. The reason is that in the second phase of LDPMiner the privacy budget has to be split to estimate the frequency of each candidate heavy hitters. Thus, a larger k would lead to less accuracy in frequency estimation.

In our experiments, we will vary the above parameters to study their impacts on LDPMiner. In all the experiments below, unless explicitly stated, we use the following default values for other relevant parameters: l: the 90-percentile of the item set sizes of all users; β: 0.01 (for the first phase of LDPMiner). For LDPMiner, by default, the privacy budget is split equally between the two phases.

5.1.3 Utility Metrics

Recall that the result of heavy hitter estimation is an ordered list of items along with their frequencies. Therefore, its quality should be measured in two aspects: (1) how accurately the estimation captures the actual set of heavy hitters as well as their ordering; and (2) how accurately the estimation captures the actual frequency of heavy hitters. In our experiments, we use the following two metrics to cover each aspect:

Relative Error (RE) [25]. It measures the error of estimated frequencies with respect to the actual frequencies of heavy hitters. Specifically, let $V = \{v_1, \ldots, v_k\}$ be the set of true top-k heavy hitters.

$$RE = Median_{v_i \in V} \frac{|f_{estimated}(v_i) - f_{actual}(v_i)|}{f_{actual(v_i)}}$$

Discounted Cumulative Gain (DCG). DCG measures the ordering quality of top heavy hitters estimated by the data collector [6]. The relevance, or gain, of an item v_i in the ranked list is measured using a graded relevance score rel_i defined as:

$$rel_{v_i} = \log_2(|d - |rank_{actual}(v_i) - rank_{estimated}(v_i)||)$$

Intuitively, the closer v_i's estimated rank is to its true rank, the larger is the gain. Given the true top k heavy hitters $v_1, \ldots, v_k$, the DCG of an estimated rank list is computed as:

$$DCG_k = rel_{v_1} + \sum_{i=2}^{k} \frac{rel_{v_i}}{\log_2(i)}$$

The discount factor $\log_2(i)$ is to give more weight to the gain of higher ranked items. Essentially, we care more about the correct ranking of important heavy hitters (those with high ranks). Finally, we normalize the DCG of an estimated ranking list by comparing it with the *Ideal DCG* (IDCG), which is the DCG when the estimated ranking list is exactly the same as the actual one (i.e., no estimation error):

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

It is easy to see that $NDCG_k$ is between 0 and 1 for all k, which allows us to compare the quality of estimated top-k heavy hitters across different k.

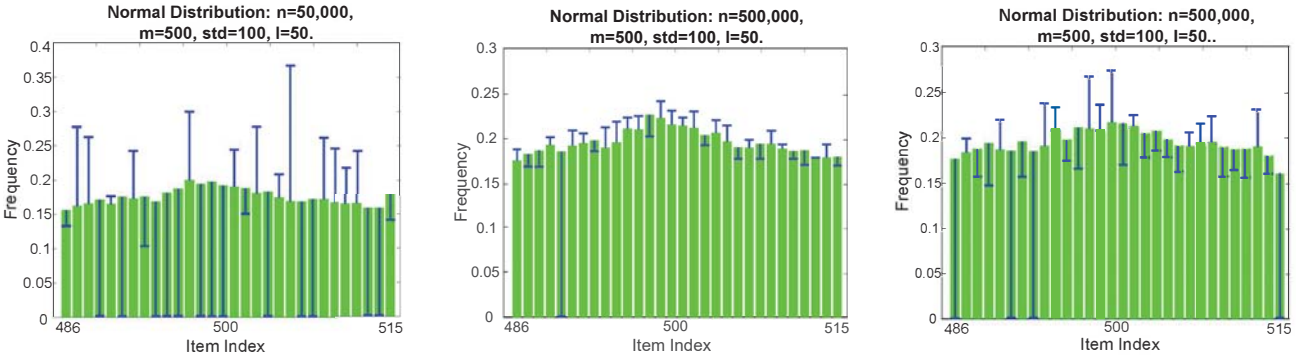
Competitors. For each experiment, we compare LDPMiner with the baseline approaches that extends RAPPOR and SH directly. As we show in figure 3, among the several extension schemes considered, sampling RAPPOR and SH offer the smallest variations, which we will use as baseline approaches. For LDPMiner, we consider both LDPMiner-SH and LDPMiner-RAPPOR (where SH and RAPPOR are used respectively in the second phase).

5.2 Experiments with Synthetic Datasets

We first study the impact of the number of users on the effectiveness of LDPMiner. Figure 4a shows a representative result of LDPMiner-RAPPOR over the synthetic dataset generated based on the normal distribution. We have 10,000 users, each with 50 items. As the mean of the normal distribution is 500, items 486 to 515 are the true top-30 heavy hitters. The green bars show these items along with their actual frequency, and the blue candlestick bars show their frequencies estimated by LDPMiner-RAPPOR with $\epsilon = ln(3)$. As a special notation, if an actual heavy hitter is missed by LDPMiner (not included in the estimated top heavy hitters), we set its candlestick bar to 0 (e.g., items 489 and 491), even though its estimated frequency is not 0. We can see that, when $n = 10000$, LDPMinor fails to capture many heavy hitters, and the reported frequencies are often far away from the real ones. The results of SH and RAPPOR are even worse than LDPMiner, which we do not show here due to space limit. Figure 4a shows the intrinsic challenge of local differential privacy for set-valued data. If an application cannot engage a significant population of users to contribute, the estimation of heavy hitters is inevitably of poor accuracy.

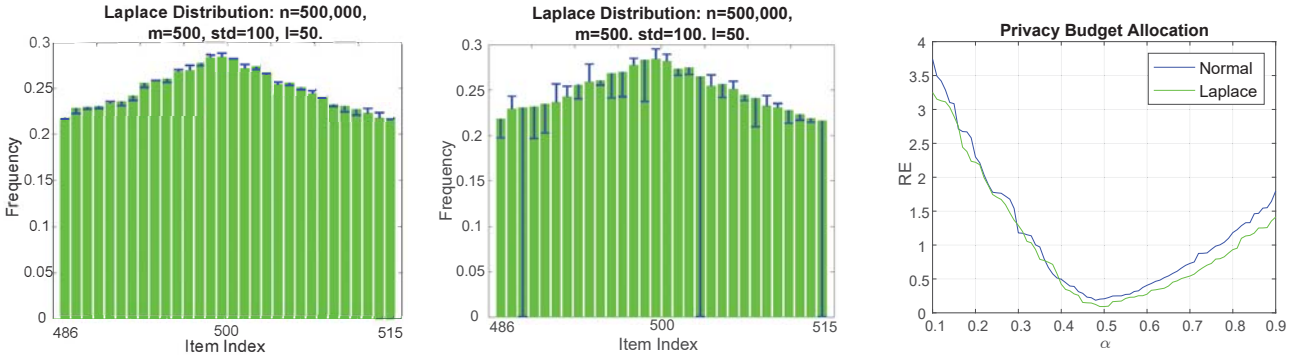
When the population size is increased, we see significant improvement of LDPMiner. Figure 4b shows a representative result of LDPMiner when we have 500,000 users instead. It is clear that the estimation becomes very accurate, not only that it only misses one heavy hitter but also the frequency estimation of each heavy hitter is very close to the real one. Meanwhile, figure 4c shows a representative result produced by sampling RAPPOR. Though it improves as well, it is clearly much inferior to that of LDPMiner, which shows the benefit of the proposed two-phase approach.

The skewness affects the frequency difference between heavy hitters and other items; hence, higher skewness makes it easier to find the correct results. LDPMiner's effectiveness would also be affected by the skewness of heavy hitters. Figure 5a shows a representative result of LDPMiner over the synthetic dataset generated from the Laplace distribution with 500,000 users. Compared with Figure 4c, the skewness of heavy hitters helps further improve estimation accuracy of LDPMiner: no actual heavy hitters are missed, and the frequency estimation error for each heavy hitter is further reduced. In contrast, Figure 5b shows a representative result of sampling RAPPOR over the same dataset. It does not show clear improvement over that in figure 4c. The reason is that because of the skewness, the first phase of LDPMiner is able to accurately capture the true top heavy hitters, which



(a) LDPMiner-RAPPOR (b) LDPMiner-RAPPOR (c) Sampling RAPPOR

Figure 4: Experimental results over synthetic datasets generated from a normal distribution



(a) LDPMiner-RAPPOR (b) Sampling RAPPOR (c) LDPMiner-RAPPOR

Figure 5: (a) and (b) show the experimental results over synthetic datasets following the Laplace distribution. (c) shows the measured Relative Error of two synthetic datasets (normal and Laplace) with different privacy budget allocation strategies

means even less privacy budget is spent on non heavy hitters in the second phase. For sampling RAPPOR, as the privacy budget is spread evenly over all items, it cannot take advantage of the skewness of heavy hitters.

In the above experiments, we evenly split the privacy budget between the two phases of LDPMiner. One may wonder what would be the impact if we split the budget differently. Intuitively, we can imagine that when we allocate too little budget to the first phase, according to Theorem 4.2, the candidate heavy hitter set would be much larger, which means that the second phase's privacy budget would need to be spread thin over all these candidate items. On the other hand, if the second phase gets too little budget, though the candidate set from the first phase would be smaller, their frequency estimation would be very noisy, due the limited budget we can spend on each candidate item. Figure 5c shows the Relative Error of the results of LDPMiner over both the normal-distribution and the Laplace-distribution synthetic dataset as described above with $n = 500,000$. x-axis is the percentage of the budget allocated to the first phase of LDPMiner. It is clear that the relative error is minimized when we split the budget roughly evenly. We have conducted the same experiments with other setting and using the NDCG metrics, obtaining similar observations. Thus, for the rest of the experiments in this paper, we always allocate the same amount of budget for each phase of LDPMiner.

5.3 Experiments with Real Datasets

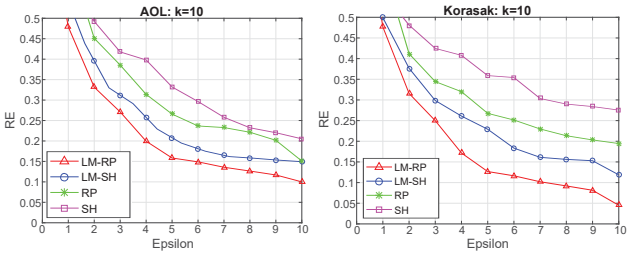
5.3.1 *The impact of ϵ*

In this experiment, for both datasets, we fix $k = 10$ and report the utility measures of different LDP mechanisms when varying ϵ. Figure 6 shows the relative errors over both datasets along with the change of ϵ. Not surprisingly, for all four algorithms, the larger ϵ, the lower their relative errors. Meanwhile, we see that LDPMiner-RAPPOR and LDPMiner-SH consistently achieve much lower relative errors than sampling RAPPOR and sampling SH. For most ϵ, the relative errors of single-phase approaches is almost twice that of LDPMiner.

Another observation is that LDPMiner-RAPPOR consistently outperforms LDPMiner-SH, which shows the utility gain by tolerating a little more extra communication cost (k bits vs. 1 bit per each user's response), when using RAPPOR instead of SH in the second phase of LDPMiner. We also observe that sampling RAPPOR is superior to sampling SH. However, this utility gain is associated with prohibitively communication cost. We have similar observations when the utility of estimated heavy hitters is measured using NDCG, as shown in Figure 7.

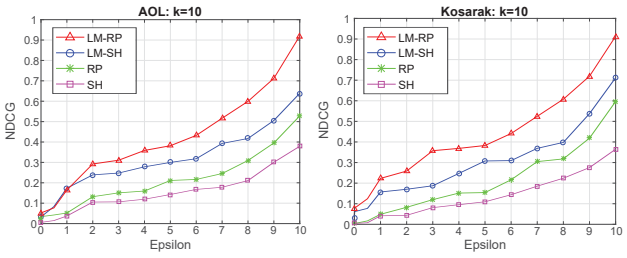
5.3.2 *The impact of k*

In this experiment, we fix $\epsilon = 3$ and vary the size of reported heavy hitters, i.e., k, to study its impact on the utility



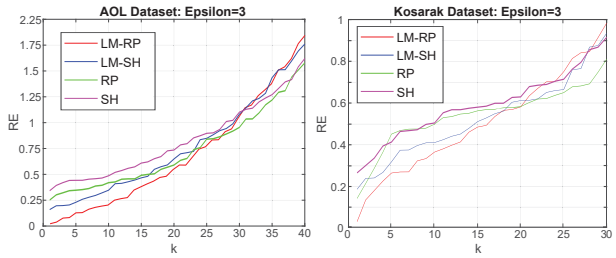
(a) AOL: RE (b) Kosarak: RE

Figure 6: Varying ϵ: Relative Error for Real Datasets



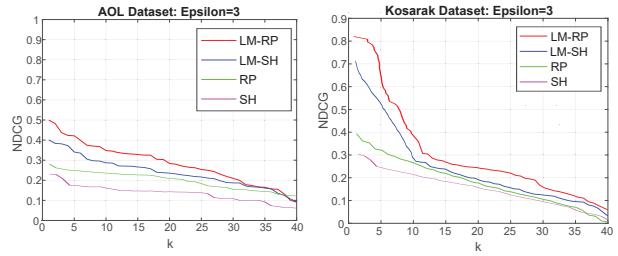
(a) AOL: NDCG (b) Kosarak: NDCG

Figure 7: Varying ϵ: Normalized Discounted Cumulative Gain for Real datasets



(a) AOL: RE (b) Kosarak: RE

Figure 8: Varying k: Relative Error for Real datasets



(a) AOL: NDCG (b) Kosarak: NDCG

Figure 9: Varying ϵ: Normalized Discounted Cumulative Gain for Real Datasets

of each algorithm. The results are shown in Figures 8 and 9. As expected, for all approaches their utility measures decrease when k increases. We also observe that the advantage of LDPMiner compared to single-phase approaches is much more significant when k is small. Further, when k keeps increasing (for example when $k > 30$ for the AOL dataset), the relative error of LDPMiner becomes even higher than single-phase based approaches. Intuitively, the proposed two-phase framework improves the utility by reducing the size of item sets from l to $O(k)$, the size of candidate heavy hitter set generated in the first phase. When k increases, the size of candidate sets becomes closer to l, and the utility gain due to the trimmed candidate set is offset by the utility loss caused by budget splitting. However, it is interesting to see that for the AOL dataset when k is larger than 30, though the relative errors of LDPMiner are higher than sampling RAPPOR, LDPMiner still achieves better NDCG. This observation is even clearer for the Kosarak dataset. Recall that the relative error reflects the quality of frequency estimation while NDCG reflects the quality of the ranked list of heavy hitters. The above observation suggests that when k increases, though the frequency estimation of LDPMiner deteriorates, it still manages to better preserve the relative orders among heavy hitters.

6. RELATED WORK

As discussed in Section 2, a series of works study the problem of frequency estimation following local differential privacy [3, 9, 19, 22]. After Warner's first study [28] on randomized response method, many works have explored a variety of perturbation mechanisms for achieving theoretical optimal utility. In [22], the local privacy model is first formalized. A recent work by Duchi et al. [9] shows an upper bound under the LDP setting, using information theoretic converse techniques to provide a minimax-error bound on convex statistical estimation. Hsu et al. [19] focus on estimating heavy hitters based on the general technique of random projection and concentration of measure. Following this work, Bassily et al. [3] propose an efficient protocol for succinct histogram estimation with information-theoretical optimal error. To handle the practical issues for private distribution estimation, RAPPOR [12, 13] is proposed to generalize Warner's randomizer from binary alphabets to k-ary alphabets. In [11], private estimation of frequent elements from stream data is studied in a setting called "pan-privacy", which can be considered as a variant of the LDP model.

Quite a few works focus on publishing histogram and count queries under the centralized differential privacy model. Hay et al. [17] generate differentially private histograms through hierarchical partitioning of data. Moreover, wavelet transform is utilized to handle multi-dimensional dataset in [30]. Li et al. [23] propose a query matrix framework that generalizes the above two works while incurring high computational complexity. Xu et al. [31] propose a two-phase kd-tree based spatial decomposition mechanism to publish histograms. Moreover, Li et al. [24] addresse the problem of releasing histograms for dynamic datasets using sparse vector technique. He et al. [18] propose a flexible policy where users can specify sensitive information for releasing cumulative histograms. In addition, there is also a rich literature on frequent itemset mining. Among them, several works are relevant to our work. Bhaskar et al. [4] also propose a two-phase based approach using a truncated frequency threshold to shrink the candidate list of frequent itemsets. This work also confirms the effectiveness of two-phase approach in a centralized model. Inspired by [4], Li et al. propose to improve the utility by constructing a basis set privately. In addition, Chen et al. [7] study the publication of set-

valued dataset under differential privacy. They present a tree structured partition mechanism in a top-down fashion. However, all above mechanisms require global knowledge of the dataset, which makes them not applicable under local differential privacy. We remark that though not all techniques developed for a centralized model is suitable to a local model, the ideas behind these technique still might be helpful in designing mechanisms in a LDP model.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we study heavy hitter estimation under local differential privacy over set-valued data. We first review existing LDP techniques and study direct extension of these techniques to handle set-valued data. Our theoretical and experimental analysis shows that such extensions would either suffer from high communication overhead or low result utility. To address the problem, we propose LDPMiner, a two-phase framework: a candidate set of heavy hitters is identified by data collector in the first phase, so that each participant is able to refine the frequency report of items in the candidate set in the second phase.Both theoretical analysis and extensive experiments confirm the utility, efficiency, and practicality of LDPMiner. A natural venue built on LDPMiner is to study frequent itemset mining under local differential privacy. The challenge is that direct adoption of existing frequent itemset mining algorithm would require iterative information exchange between users and data collectors, which would result in accumulation of dramatic noise due to the limited budget for each iteration.

8. ACKNOWLEDGMENTS

Xiaokui Xiao was supported by grant ARC19/14 from MOE, Singapore.

9. REFERENCES

[1] Aol search log. *http://www.gregsadetsky.com/aol-data/*.

[2] Spmf: An open-source data mining library. *http://www.philippe-fournier-viger.com/spmf*.

[3] R. Bassily and A. D. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135, 2015.

[4] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *SIGKDD*, pages 503–512, 2010.

[5] A. Blum, K. Ligett, and A. Roth. A learning theory approach to noninteractive database privacy. *JACM*, 60(2):12, 2013.

[6] C. Burges et al. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[7] R. Chen et al. Publishing set-valued data via differential privacy. *PVLDB*, 4(11):1087–1098, 2011.

[8] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Privacy aware learning. *J. ACM*, 61(6):38:1–38:57.

[9] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.

[10] C. Dwork. Differential privacy: A survey of results. In *Theory and applications of models of computation*, pages 1–19. 2008.

[11] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin. Pan-private streaming algorithms. In *ICS*, pages 66–80, 2010.

[12] Ú. Erlingsson et al. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

[13] G. Fanti et al. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *PoPETS*, issue 3, 2016, 2016.

[14] A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. *SICOMP*, 42(4):1494–1520, 2013.

[15] S. Hansell. Aol removes search data on vast group of web users. *New York Times*, 8:C4, 2006.

[16] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

[17] M. Hay et al. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.

[18] X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *SIGMOD*, pages 1447–1458, 2014.

[19] J. Hsu, S. Khanna, and A. Roth. Distributed private heavy hitters. In *Automata, Languages, and Programming*, pages 461–472. 2012.

[20] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[21] D. Karger et al. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *STOC*, pages 654–663, 1997.

[22] S. P. Kasiviswanathan et al. What can we learn privately? *SICOMP*, 40(3):793–826, 2011.

[23] C. Li et al. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.

[24] H. Li et al. Differentially private histogram publication for dynamic datasets: an adaptive sampling approach. In *CIKM*, pages 1001–1010, 2015.

[25] N. Li, W. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *PVLDB*, 5(11):1340–1351, 2012.

[26] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[27] T. T. Nguyên et al. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053*, 2016.

[28] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the ASA*, 60(309):63–69, 1965.

[29] X. Xiao, Y. Tao, and M. Chen. Optimal random perturbation at multiple privacy levels. *PVLDB*, 2(1):814–825, 2009.

[30] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.

[31] J. Xu et al. Differentially private histogram publication. *VLDBJ*, 22(6):797–822, 2013.

[32] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. 2008.