Giulia Fanti*, Vasyl Pihur, and Úlfar Erlingsson

# Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries

**Abstract:** Techniques based on randomized response enable the collection of potentially sensitive data from clients in a privacy-preserving manner with strong local differential privacy guarantees. A recent such technology, RAPPOR [12], enables estimation of the marginal frequencies of a set of strings via privacy-preserving crowdsourcing. However, this original estimation process relies on a known dictionary of possible strings; in practice, this dictionary can be extremely large and/or unknown. In this paper, we propose a novel decoding algorithm for the RAPPOR mechanism that enables the estimation of "unknown unknowns," i.e., strings we do not know we should be estimating. To enable learning without explicit dictionary knowledge, we develop methodology for estimating the joint distribution of multiple variables collected with RAPPOR. Our contributions are not RAPPOR-specific, and can be generalized to other local differential privacy mechanisms for learning distributions of string-valued random variables.

**Keywords:** privacy-preserving analytics, differential privacy

DOI 10.1515/popets-2016-0015
Received 2015-11-30; revised 2016-03-01; accepted 2016-03-02.

## 1 Introduction

It is becoming increasingly commonplace for companies and organizations to analyze user data in order to improve services or products. For instance, a utilities company might collect water usage statistics from its users to help inform fair pricing schemes. Although user data analysis can be very beneficial, it can also pose a privacy threat. Collected data can reveal sensitive details about users, such as preferences, habits, or personal characteristics. It is therefore important to develop methods for analyzing the data of a population without sacrificing individuals' privacy.

A guarantee of *local differential privacy* can provide the appropriate privacy protection without requiring individuals to trust the intentions of a data aggregator [20]. Informally, a locally differentially-private mechanism asks individuals to report data to which they have added carefully-designed noise; this noise ensures that any individual's information cannot be learned, but an aggregator can correctly infer population statistics. The recently-introduced Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) is the first such mechanism to see real-world deployment [12].

RAPPOR is motivated by the problem of estimating a client-side distribution of string values drawn from a discrete data dictionary. Such estimation is useful in many security-related scenarios. For example, RAPPOR is reportedly used in the Chrome Web browser to track the distribution of users' browser configuration strings; this is done to detect anomalies symptomatic of abusive software [12, 15, 16].

Unfortunately, in its current state, the RAPPOR technology is of only limited utility. This is because it makes two simplifying assumptions that will certainly not always hold in practice:

**Assumption 1:** ***Aggregators only need to learn the distribution of a single variable, in isolation.*** In practice, aggregators may want to study the association between multiple variables because attributes are often more meaningful in association with other attributes. For example, in RAPPOR's application domain in the Chrome Web browser, an innocent-looking homepage or search-provider URL may become highly suspect if its use is strongly correlated with installation of software that is known to be malicious.

**Assumption 2:** ***Aggregators know the data dictionary of possible string values in advance.*** There are many scenarios in which both the frequencies of client-side strings and the strings themselves may be unknown. For instance, when collecting reports on installed software, it is unlikely that the names or hash values of all software will be known ahead of time, especially in the face of polymorphic software. Similarly, when studying user-generated data—manually-entered hashtags, for instance—the dictionary of possible strings cannot be known *a priori.*

***Corresponding Author: Giulia Fanti:** U.C. Berkeley, E-mail: gfanti@eecs.berkeley.edu
**Vasyl Pihur:** Google, vpihur@google.com
**Úlfar Erlingsson:** Google, ulfar@google.com

Lifting these two simplifying assumptions requires reasoning about “unknown unknowns.” The first assumption can only be removed by estimating the unknown joint distributions of two or more unknown variables that are observed only via differentially-private RAPPOR responses. Removing the second assumption requires learning a data dictionary of unknown client-side strings whose frequency distribution is also unknown. This process must additionally satisfy strong privacy guarantees that preclude the use of encryption or special encodings that could link individuals to strings. Furthermore, neither of these challenges admits a solution that is simultaneously feasible and straightforward. The naive approach of trying all possibilities incurs exponential blowup over the infinite domain of unknown strings, and is not even well-defined with regards to estimating joint distributions.

This paper provides methods for addressing these two challenges. First, regarding multivariate analysis, we present a collection of statistical tools for studying the association between multiple random variables reported through RAPPOR. This toolbox includes an expectation-maximization-based algorithm for inferring joint distributions of multiple variables from a collection of RAPPOR reports. It also includes tools for computing the variance of the distribution estimates, as well as testing for independence between variables of interest.

Second, regarding unknown data dictionaries, we introduce a novel algorithm for estimating a distribution of strings without knowing the set of possible values beforehand. This algorithm asks each reportee to send a noisy representation of multiple substrings from her string. Using our previously-developed techniques for association analysis, we build joint distributions of all possible substrings. This allows the aggregator to learn the data dictionary for frequent strings.

We demonstrate the practical efficacy of both contributions through simulation and real-world examples. For these experiments we have publicly-available analysis code.[1] While motivated by RAPPOR, and presented in that context, our contributions are general and do not depend critically on the RAPPOR encoding and decoding algorithms. Our methods can be generalized to other locally differentially-private systems that learn a distribution of discrete, string-valued random variables.

# 2 Background

A common method for collecting population-level statistics without access to individual-level data points is based on *randomized response* [27]. Randomized response is an obfuscation technique that satisfies a privacy guarantee known as *local differential privacy* [20]. We begin by briefly introducing local differential privacy and explaining how the RAPPOR system uses randomized response to satisfy this condition. Formally, a randomized algorithm $A$ (in this case, RAPPOR) satisfies $\epsilon$-differential privacy [10] if for all pairs of client's values $x_1$ and $x_2$ and for all $R \subseteq Range(A)$,

$$P(A(x_1) \in R) \leq e^{\epsilon} P(A(x_2) \in R).$$

Intuitively, this says that no matter what string user Alice is storing, the aggregator's knowledge about Alice's ground truth does not change too much based on the information she sends. Differential privacy is a property of an encoding algorithm, so these guarantees hold regardless of the underlying distribution.

RAPPOR is a privacy-preserving data-collection mechanism that makes use of randomization to guarantee local differential privacy for every individual's reports. Despite satisfying such a strong privacy definition, RAPPOR enables the aggregator to accurately estimate a distribution over a discrete dictionary (e.g., a set of strings).

The basic concept of *randomized response* is best explained with an example. Suppose the Census Bureau wants to know how many communists live in the United States without learning *who* is a communist. The administrator asks each participant to answer the question, “Are you a communist?” in the following manner: Flip an unbiased coin. If it comes up heads, answer truthfully. Otherwise, answer ‘yes’ with probability 0.5 and ‘no’ with probability 0.5. In the end, the Census Bureau cannot tell which people are communists, but it can estimate the true fraction of communists with high confidence. Randomized response refers to this addition of carefully-designed noise to discrete random values in order to mask individual data points while enabling the computation of aggregate statistics.

RAPPOR performs two rounds of randomized response to mask the inputs of users and enable the collection of user data over time. Suppose Alice starts with the string $X$ (e.g., $X$ = “rabbit”). The sequence of events in the encoder is as follows:

1. Hash the string $X$ twice ($h$ times in general) into a fixed-length Bloom filter, $B$.

[1] https://github.com/google/rappor

2. Pass each bit in the Bloom filter $B_i$ through a randomized response (giving $B'_i$) as follows:

$$B'_i = \begin{cases} 1, & \text{with probability } \frac{1}{2}f \\ 0, & \text{with probability } \frac{1}{2}f \\ B_i, & \text{with probability } 1-f \end{cases}$$

   where $f$ is a user-tunable parameter controlling the level of privacy guarantees. We refer to this noisy Bloom filter $B'$ as the *permanent randomized response* (PRR) for the value $X$, because this same $B'$ is to be used for both the current and all future responses about the value $X$.
3. Each time the aggregator requests a report, pass each bit $B'_i$ in the PRR through *another* round of randomized response (giving $X'_i$), as follows:

$$P(X'_i = 1) = \begin{cases} q, & \text{if } B'_i = 1. \\ p, & \text{if } B'_i = 0. \end{cases}$$

   We refer to this array of bits $X'$ as an *instantaneous randomized response* (IRR), and the aggregator only ever sees such bit vectors $X'$ for any reported value $X$. The smaller the difference between $q$ and $p$ (user-tunable), the stronger the privacy guarantee.

This process is visualized in Figure 1.

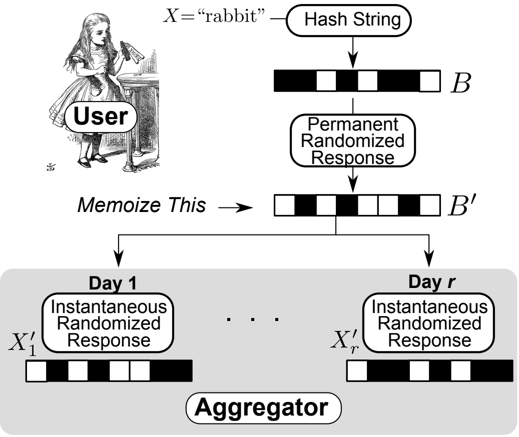
$X$="rabbit"
Hash String
$B$
User
Permanent Randomized Response
Memoize This
$B'$
Day 1
Day $r$
Instantaneous Randomized Response
Instantaneous Randomized Response
$X'_1$
$X'_r$
. . .
Aggregator

**Fig. 1.** Visualization of the RAPPOR mechanisms from [12]. Each user starts by hashing her true string into a Bloom filter $B$. This representation $B$ is used to generate a permanent randomized response (PRR) $B'$. Whenever the aggregator collects data (daily, for instance), the user builds a new instantaneous randomized response $X'$ from $B'$ and sends it to the aggregator.

The RAPPOR encoding scheme satisfies two different $\epsilon$-differential privacy guarantees: one against a one-shot adversary who sees only a single IRR, and one against a stronger adversary who sees infinitely many IRRs over time. The latter adversary is able to reconstruct $B'$ with arbitrary precision after seeing enough reports, which motivates the need for a PRR, but is unable to infer $B$ from a single copy of $B'$. In principle, users could always report $B'$ at every data collection, but this would create a unique tracking identifier.

If the set of possible strings is small and known prior to collection (e.g., country, gender, etc.), a simplified version of the algorithm, called Basic RAPPOR, is more appropriate. The single difference is that in step (1), Basic RAPPOR does not make use of Bloom filters, but deterministically assigns each string to its own bit ($h = 1$). In this case, the size of $B$ is determined by the cardinality of the set being collected. This also significantly simplifies the inference process to estimate string frequencies by the aggregator.

Despite strong report-level differential privacy guarantees, RAPPOR can approximate the marginal distribution of the measured variable(s) with high precision. One high-utility decoding scheme is described in [12], but the details of marginal decoding are not critical to understanding our present work.

# 3 Estimating Joint Distributions

Learning the distribution of a single variable is sometimes enough. More often, however, aggregators may be interested in learning the associations and correlations between *multiple* variables, all collected in a privacy-preserving manner. For example, suppose we would like to understand the relationship between installed software and annoying advertisements, e.g., to detect the presence of so-called *adware*. To do so, we might study the association between displayed advertisements and recently-installed software applications or extensions. If both of these variables are measured using the RAPPOR mechanism, the current literature does not describe how to estimate their *joint* distribution, although methods exist for estimating marginal frequencies of both variables individually.

In this section, we describe a general approach to estimating the joint distribution of two or more RAPPOR-collected variables. Inference is performed using the expectation-maximization (EM) algorithm [7], which produces asymptotically-unbiased estimates of joint probability distributions. These joint estimation techniques will play a key role in Section 4, in which we estimate data distributions over unknown dictionaries. Notice that EM is a general-purpose algorithm that can be applied to *any* differentially-private encoding mechanism; as such, the techniques in this section are not RAPPOR-specific.

## 3.1 Joint Distributions with the EM Algorithm

The EM algorithm is a common way to approximate maximum likelihood estimates (MLEs) of unknown parameters in the presence of missing or incomplete data. Here, the parameters to be estimated are the entries of a contingency table for two or more random variables. EM works by alternating iteratively fixing the parameters and computing the expected log-likelihood (expectation-step), and then maximizing the expectation of the log-likelihood with respect to the parameters (maximization-step). This eventually converges to a local maximum in the log-likelihood function, but it not guaranteed to give the true ML solution. The EM algorithm is well-suited to RAPPOR applications where true values are not observed and only their noisy representations are collected.

For the sake of clarity, we will focus on estimating the joint distribution of two random variables $X$ and $Y$, both collected using Basic RAPPOR introduced in Section 2. Extending this estimation to general RAPPOR requires careful consideration of unknown categories and will be discussed in the next section. Let $X' = \text{RAPPOR}(X)$ and $Y' = \text{RAPPOR}(Y)$ be the noisy representations of $X$ and $Y$ created by RAPPOR. Suppose that $N$ pairs of $X'$ and $Y'$ are collected from $N$ distinct (independent) clients. $X_i$ and $X'_i$ denote the $i$th data point and noisy representation, respectively. The conditional probability of true values $X$ and $Y$, given the observed noisy representations $X'$ and $Y'$ follows from Bayes' theorem:

$$P(X=x_i, Y=y_j|X',Y') = \frac{p_{ij} P(X',Y'|X=x_i,Y=y_j)}{\sum_{k=1}^{m}\sum_{\ell=1}^{n} p_{k\ell} P(X',Y'|X=x_k,Y=y_\ell)}.$$

where $m$ and $n$ are the number of categories in $X$ and $Y$, respectively. Here, $p_{ij}$ is the true joint distribution of $X$ and $Y$; this is the quantity we wish to estimate for each combination of categories $i$ and $j$. $P(X',Y'|X,Y)$ is the joint probability of observing the two noisy outcomes given both true values. $X'$ and $Y'$ are generated independently from $X$ and $Y$, respectively, and are therefore conditionally independent given $X$ and $Y$; that is, $P(X',Y'|X,Y) = P(X'|X)P(Y'|Y)$. Since the noise added through RAPPOR is predictable and mechanical, it is easy to precisely describe these probabilities. Without loss of generality, assume that $X = x_1$. In Basic RAPPOR, $x_1$'s Bloom Filter representation has a one in the first position and zeros elsewhere, so we have

$$\begin{aligned} P(X'|X=x_1) &= q^{x'_1}(1-q)^{1-x'_1} \times p^{x'_2}(1-p)^{1-x'_2} \\ &\quad \times \ldots \times p^{x'_i}(1-p)^{1-x'_i} \\ &\quad \times \ldots \times p^{x'_m}(1-p)^{1-x'_m}. \end{aligned}$$

The EM algorithm proceeds as follows:

1. **Initialize:** $\hat{p}^0_{ij} = \frac{1}{nm}, 1 \le i \le m, 1 \le j \le n$ (uniform distribution).
2. **E-Step:**
$$P(X_k=x_i, Y_k=y_j|X'_k,Y'_k;\hat{p}^t) = \frac{\hat{p}^t_{ij} P(X'_k,Y'_k|X_k=x_i,Y_k=y_j)}{\sum_{r=1}^{m}\sum_{\ell=1}^{n} \hat{p}^t_{r\ell} P(X'_k,Y'_k|X_k=x_r,Y_k=y_\ell)}.$$
3. **M-Step:** Set
$$\hat{p}^{t+1}_{ij} = \frac{1}{N}\sum_{k=1}^{N} P(X_k=x_i,Y_k=y_j|X'_k,Y'_k;\hat{p}^t)$$
4. **Repeat** steps 2-3 until convergence, i.e. $\max_{ij} |\hat{p}^{t+1}_{ij} - \hat{p}^t_{ij}| < \delta^*$ for some $\delta^* > 0$.

This algorithm converges to a local maximum of the log likelihood, parameterized by $p_{ij}$ [8].

**Proposition 3.1.** *The EM algorithm converges to the maximum-likelihood (ML) estimator of* $p_{ij}$, $1 \le i \le m$, $1 \le j \le n$.

(Proof in Appendix E.1)

Notice that ML estimators may be biased for finite sample sizes, but in this case, the EM algorithm estimator is asymptotically unbiased [26]:

**Proposition 3.2.** *Let* $\hat{p}_{ij,N}$ *denote the EM estimate of parameter* $p_{ij}$*, with* $1 \le i \le m$ *and* $1 \le j \le n$*, using* $N$ *samples. Then* $\hat{p}_{ij,N}$ *converges* $\hat{p}_{ij,N} \xrightarrow{a.s.} p_{ij}$.

(Proof in Appendix E.2)

## 3.2 The "Other" category

In the EM initialization step, we assume that we know all $n$ categories of $X$ and all $m$ categories of $Y$. In practice, the aggregator is unlikely to know all the relevant categories, and must make choices about which categories to include. Operationally, the aggregator would perform marginal analyses on both $X'$ and $Y'$ separately, estimate the most frequent categories, and use them in the joint analysis. The remaining undiscovered categories, which we refer to as "Other", cannot be simply omitted from the joint analysis because doing so leads to badly biased distribution estimates. In this section, we discuss how to handle this problem.

Suppose one ran the marginal decoding analysis separately on $X'$ and $Y'$, thereby detecting $m$ and $n$ top categories, respectively, along with their corresponding marginal frequencies. Note that $m$ and $n$ now represent the *detected* numbers of categories instead of the true numbers of categories. The "Other" categories for $X$ and $Y$ may constitute a significant amount of probability mass (computed as $1 - \sum_{i=1}^{m} \hat{p}_i$ and $1 - \sum_{j=1}^{n} \hat{p}_j$, respectively) which must be taken into account when estimating the joint distribution.

The difficulty of modeling the "Other" category comes from the apparent problem of estimating $P(X' = x' | X = \text{``}Other\text{''})$, i.e., the probability of observing a report $x'$ given that it was generated by any category other than the top $m$ categories of $X$. If we could estimate this probability, we could simply use the EM algorithm to estimate the joint distribution—an $(m + 1) \times (n + 1)$ contingency table in which the last row and the last column are the "Other" categories for each variable.

We use knowledge of the top $m$ categories and their frequencies to estimate this conditional probability. Let $c_s^m$ be the expected number of times that *reported* bit $s$ was set by one of the top $m$ categories in $X$. It is equal to $c_s^m = \left(\left(1 - \frac{f}{2}\right) q + \frac{fp}{2}\right) T_s + \left(\left(1 - \frac{f}{2}\right) p + \frac{fq}{2}\right) (N - T_s)$, where $N$ is the number of reports collected and $T_s = N \sum_{i=1}^{m} p_i I(B_s(x_i) = 1)$ represents the expected number of times the $s$th bit in $N$ Bloom filters was set by a string from one of the top $m$ categories. Here, $I$ is the indicator function returning 1 or 0 depending if the condition is true or not, $B(x_i)$ is the Bloom filter generated by string $x_i$ and $p_i$ is the true frequency of string $x_i$.

Given the above, the estimated proportion of times each bit was set by a string from the "Other" category is then $\hat{p}_s^o = \frac{c_s - \hat{c}_s^m(\hat{p}_i)}{N(1 - \sum_{i=1}^{m} \hat{p}_i)}$, where $c_s$ is the observed number of times bit $s$ was set in all $N$ reports. The conditional probability of observing any report $X'$ given that the true value was "Other" is $P(X' = x' | X = \text{``}Other\text{''}) = \prod_{s=1}^{k} (\hat{p}_s^o)^{x'_s} (1 - \hat{p}_s^o)^{1 - x'_s}$. We use this estimate to run the EM algorithm with "Other" categories and obtain unbiased joint distribution estimates. We generalize this estimation to other differential privacy mechanisms in Appendix D.

## 3.3 The Variance-Covariance matrix

Under mild regularity conditions, the asymptotic distribution of maximum likelihood estimates $(\hat{p}_{11}, \ldots, \hat{p}_{mn})$ is $\mathcal{N}\left((p_{11}, \ldots, p_{mn}), I^{-1}\right)$, where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian distribution with mean $\mu$ and variance-covariance matrix $\Sigma$, and $I$ is the information matrix [26]. This permits an aggregator to construct confidence intervals, test if any of the proportions are different from 0, or perform an overall test for the association between $X$ and $Y$.

In this case, the asymptotic variance-covariance matrix is given by the inverse of incomplete-data observed information matrix $I_{obs}$. To obtain an estimate of the information matrix, we would evaluate the second derivative of the observed-data log-likelihood function at our ML estimates $\hat{p}_{ij}$'s.

The log-likelihood function is the log of the probability of observing all $N$ reports, treated as a function of the unknown parameter vector $(p_{11}, \ldots, p_{mn})$:

$$\ell(p_{11}, \ldots, p_{mn}) = \sum_{k=1}^{N} \log \left( \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} P(X'_k, Y'_k | X_i, Y_j) \right).$$

The first derivative with respect to $p_{ij}$ is given by

$$\ell'_{p_{ij}} = \sum_{k=1}^{N} \frac{P(X'_k, Y'_k | X = x_i, Y = y_j)}{\sum_{o=1}^{m} \sum_{\ell=1}^{n} p_{ij} P(X'_k, Y'_k | X = x_o, Y = y_\ell)}.$$

The second derivative, also known as the observed information matrix (size $mn \times mn$), is given by

$$\ell''_{p_{ij}, p_{st}} = \sum_{k=1}^{N} \frac{P(X'_k, Y'_k | x_i, y_j) \cdot P(X'_k, Y'_k | x_s, y_t)}{\left(\sum_{o=1}^{m} \sum_{\ell=1}^{n} p_{ol} P(X'_k, Y'_k | x_o, y_\ell)\right)^2}.$$

Inverting this matrix and evaluating at the current ML estimates $\hat{p}_{11}, \ldots, \hat{p}_{mn}$ gives an estimate of the variance-covariance matrix $\hat{\Sigma}$. The $mn$ diagonal elements of $\hat{\Sigma}$ contain the variance estimates for each $\hat{p}_{ij}$ and can be directly used to assess how certain we are about them.

## 3.4 Testing for Association

It is often important to test for independence between categorical random variables. For two variables to be independent their joint distribution must be equal to the product of their marginals, i.e. $P(X,Y) = P(X)P(Y)$. The $\chi^2$ test is commonly used to test the independence of two or more categorical variables [1]. It compares observed cell counts to what is expected under the independence assumption. The formal test statistic is given by $\chi^2 = \sum_{i=1}^{mn} \frac{(O_i - E_i)^2}{E_i}$, where $E_i$ is expected number of cell counts under the independence assumption and $O_i$ is the observed number of cell counts. When $X$ and $Y$ are independent, this test statistic has a $\chi^2$ distribution with $(m-1)(n-1)$ degrees of freedom. However, we cannot use the $\chi^2$ test statistic because we do not observe exact cell counts $O_i$ of the co-occurrence of our random variables $X$ and $Y$. Instead, we have mean estimates and the corresponding variance-covariance matrix.

The Wald test is commonly used to evaluate how far parameter estimates are from those suggested by a null hypothesis [11]. In this context, we can test how significantly different our estimates are from those that would have been observed under the independence assumption. The Wald test uses statistic

$$T = (\hat{p} - \hat{\mu})^{\mathsf{T}} \hat{\Sigma}^{-1} (\hat{p} - \hat{\mu}), \qquad (1)$$

where $\hat{p}$ is a vector of $\hat{p}_{ij}$'s, $\hat{\mu}$ ($\hat{\mu}_{ij} = \hat{p}_i \hat{p}_j$) is a vector of products of marginals (i.e. the expected joint distribution if the variables are independent) and $\hat{\Sigma}$ is the estimated variance-covariance matrix. $\mathsf{T}$ indicates the transpose operation. Under the null hypothesis, this test statistic $T$ converges in distribution to a $\chi^2$ distribution with $(m-1)(n-1)$ degrees of freedom, since the parameter estimates are asymptotically normally distributed about their true values [26]. Indeed, under some regularity conditions [6], the Wald test is an asymptotically locally most powerful invariant test (asymptotically optimal) [11].

In summary, to perform a formal statistical test for independence between $X$ and $Y$, one would use the EM algorithm to estimate the joint distribution along with the variance-covariance matrix. Then, one would compute the Wald test statistic and compare it to the corresponding critical quantile $q_{1-\alpha}$ from the $\chi^2_{(m-1)(n-1)}$. We conclude that $X$ and $Y$ are not independent if $T > q_{1-\alpha}$ and state that there is no evidence for non-independence otherwise. The empirical power and Type I error rate of this test are studied in Appendix A.

## 3.5 Simulation Results

To illustrate our multivariable analysis of differentially private data, we generated synthetic RAPPOR reports for variables $X$ and $Y$, each with 100 unique categories.

**Table 1.** True joint distribution of $X$ (rows) and $Y$ (columns).

| | 1 | 2 | 3 | 4 | 5 | Other |
|---|---|---|---|---|---|---|
| 1 | 3.567 | 2.937 | 2.468 | 1.952 | 1.639 | 6.436 |
| 2 | 2.984 | 2.432 | 1.967 | 1.581 | 1.289 | 5.362 |
| 3 | 2.473 | 1.991 | 1.609 | 1.223 | 1.025 | 4.343 |
| 4 | 1.881 | 1.569 | 1.293 | 1.069 | 0.874 | 3.499 |
| 5 | 1.625 | 1.292 | 1.080 | 0.892 | 0.662 | 2.836 |
| Other | 6.380 | 5.292 | 4.311 | 3.495 | 2.809 | 11.863 |

**Table 2.** Estimated joint distribution of $X$ (rows) and $Y$ (cols).

| | 1 | 2 | 3 | 4 | 5 | Other |
|---|---|---|---|---|---|---|
| 1 | 3.306 | 2.952 | 2.413 | 2.018 | 1.806 | 6.691 |
| 2 | 3.045 | 2.292 | 2.043 | 1.588 | 1.286 | 5.302 |
| 3 | 2.336 | 2.173 | 1.587 | 1.115 | 0.916 | 4.450 |
| 4 | 1.902 | 1.506 | 1.354 | 1.087 | 0.887 | 3.510 |
| 5 | 1.763 | 1.233 | 1.188 | 0.873 | 0.615 | 2.801 |
| Other | 6.531 | 5.338 | 4.245 | 3.513 | 2.916 | 11.419 |

The marginal distributions of $X$ and $Y$ were discretized Zipfian distributions, and their (truncated) joint distribution is given in Table 1.

With 100,000 reports, we estimated the frequencies of 15 top categories for each $X$ and $Y$, on average. For the association analysis, we selected the top five categories from each variable's estimated marginal distribution. We first ignored the "Other" categories and assumed that $X$ and $Y$ had 5 unique values each. 10 Monte Carlo replications were performed; the first two panels of Figure 2 plot the estimated cell frequency against the true cell frequency for each of the ten trials and 25 cells. The estimated 25 proportions are poor estimates for both the true joint frequencies and the conditional frequencies $P(X = x, Y = y | X \in \text{top-5}, Y \in \text{top-5})$.

The bottom panels of Figure 2 show estimates when we account for the "Other" categories of $X$ and $Y$. The estimated joint distribution is now a $6 \times 6$ table, and the procedure produces unbiased estimates for the true joint frequencies (Table 2). Accordingly, it also produces 25 unbiased estimates for conditional frequencies.

Figure 3 shows the effect of privacy level $\epsilon$ and sample size $N$ on the reconstruction accuracy of this joint estimation. This suggests that operationally, we need a sample size of at least $N = 20,000$ in order to reliably reconstruct the joint distribution with privacy levels $\epsilon \leq 1$. This accuracy also depends on the size of the underlying dictionaries, which is explored in more detail in Appendix B.

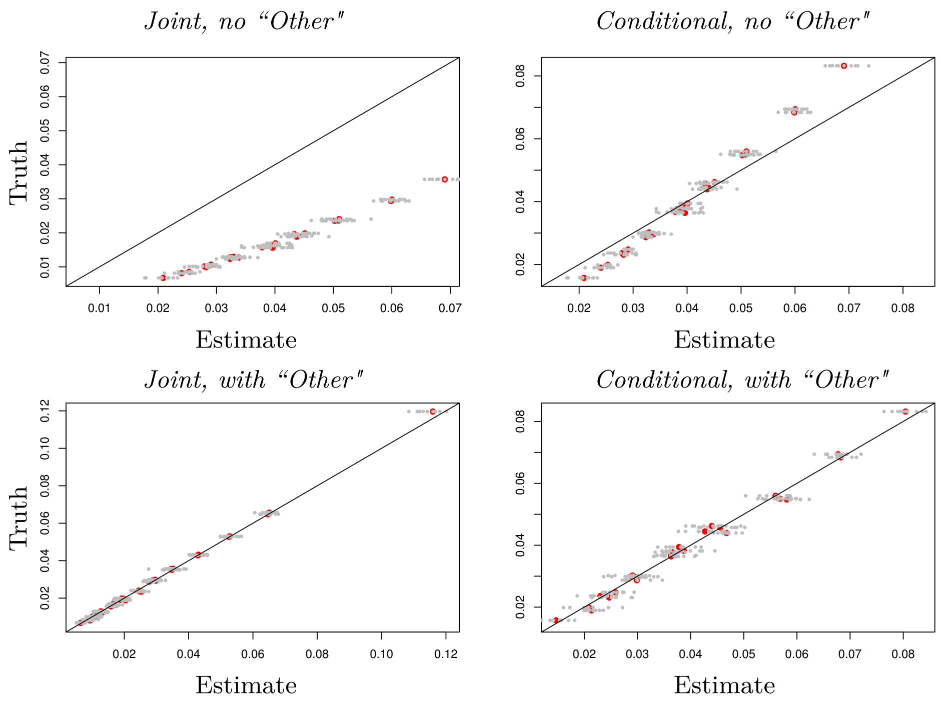

**Fig. 2.** True vs. estimated joint frequencies. Red dots show average estimates over 10 Monte Carlo runs. Grey dots show individual estimates. Top panels show how ignoring the "Other" category leads to biased joint and conditional distribution estimates. For the conditional distribution, there's a regression to the mean effect where high values are underestimated and lower values are overestimated. Accounting for the "Other" category fixes the problem, and estimates are close to the truth. Sample size is 100,000, $\epsilon = 3$.

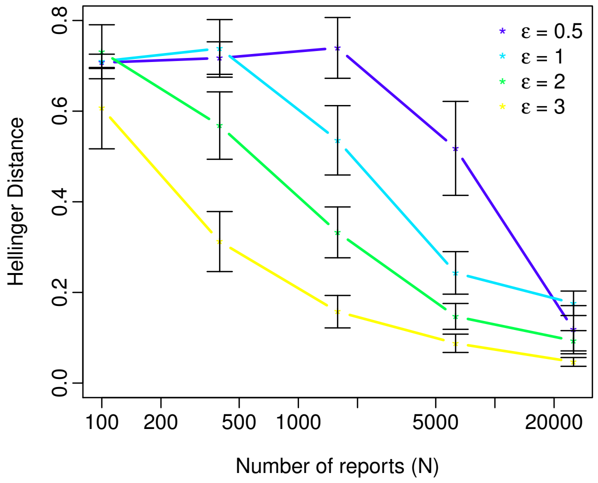

**Fig. 3.** Hellinger distance between the estimated and true joint distributions, parameterized by $\epsilon$ and the number of reports $N$. The true distribution has dictionary size $4 \times 5$.

## 3.6 Example: Google Play Store Apps

To demonstrate these techniques, we downloaded the public metadata for 200,000 mobile-phone apps from the Google Play Store. For each app, we obtained the app category (30 categories) and whether it is offered for free or not. This information can be summarized in a $30 \times 2$ contingency table. Applying a $\chi^2$ independence test to this contingency table would test whether different categories are statistically more likely to feature free apps. We use RAPPOR and our joint decoding approach to learn this distribution without direct access to the underlying data points. The data in this example is not particularly sensitive, but we were unable to find public datasets of sensitive, multivariate, categorical data, precisely due to the associated privacy concerns.

For each sampled app, we generated a simulated Basic RAPPOR report for both variables: app category and payment model. We used 30-bit reports for the category variables, and 1-bit reports for the Boolean payment model. We then performed a joint distribution analysis by estimating the $30 \times 2$ contingency table—i.e., the frequency of each combination of item category and payment model. Results are shown in the second panel of Figure 4. The green points show both true and estimated frequencies of free items for each category, while the brown points show the paid ones. Note that these are the 60 cell frequencies from the true and estimated contingency tables, not proportions of free or paid apps for each category. 95% confidence intervals are shown as horizontal bars for both sets of estimates and have proper coverage in all cases.

The top panel of Figure 4 shows the true and estimated paid rate for each category, computed as the proportion of paid apps for that category divided by the overall proportion of a category. This ratio estimate is less stable than the joint frequencies but follows the true rates closely for most app categories.

We perform a formal test for independence by computing the proposed $\chi^2$-test statistic $T = 107.093$, which has a p-value of $6.9523e - 11$. This is much smaller than 0.05 and we would therefore conclude that there are, in fact, statistically significant differences in paid rates between different app categories. This can be, of course, clearly seen from the top panel where categories are ordered in the descending prevalence of paid software, with proportions ranging from 30% to 4%.

# 4 RAPPOR Without a Known Dictionary

Suppose we wish to use RAPPOR to learn the ten most visited URLs last week. To do this, we could first create an exhaustive list of candidate URLs and then test each candidate against received reports to determine which ones are present in the sampled population. In this process, it is critical to include *all* potential candidates, since RAPPOR has no direct feedback mechanism for learning about missed candidates. Such a candidate list may or may not be available, depending on what is being collected. For instance, it may be easy to guess the most visited URLs, but if we instead wish to learn the most common tags in private photo albums, it would be impractical to construct a fully exhaustive list. In this section, we describe how to learn distribution-level information about a population without knowing the dictionary, i.e., the set of candidate strings, beforehand.

In order to provide a benchmark for future comparison, we pose the problem in terms of minimizing the Hellinger distance between the learned distribution and the true distribution. Hellinger distance captures the distance between two distributions of discrete random variables. For discrete probability distributions $P$ and $Q$ defined over some set $\mathcal{U}$, it is defined as

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i \in \mathcal{U}} (\sqrt{P(i)} - \sqrt{Q(i)})^2}.$$

This metric is appropriate because unlike Kullback-Leibler divergence, it is defined even when the two distributions have different support sets.

**Problem Statement:** Consider a collection of datapoints $\mathcal{X} = \{X_1, \ldots, X_N\}$ drawn i.i.d. from some discrete distribution $P$ defined over alphabet $\mathcal{U}$. In RAPPOR, $\mathcal{U}$ is the set of all strings of any length. Let $\mathcal{U}_P$ be the *unknown* subset of $\mathcal{U}$ that defines the true support of $P$, and suppose that $|\mathcal{U}_P| \ll |\mathcal{U}|$. In RAPPOR, $\mathcal{U}_P$ is the set of relevant strings, like webpages or names of browser plugins. Each datapoint is independently randomized using an $\epsilon$-locally-differentially-private mechanism, giving noisy datapoints $\mathcal{X}' = \{X'_1, \ldots, X'_N\}$. Design a mechanism $\mathcal{F}$ that takes as input $\mathcal{X}'$ and outputs a distribution $Q$, such that $H(P, Q)$ is as small as possible. The complexity of $\mathcal{F}$ should be constrained to scale according to $|\mathcal{U}_P|$ rather than $\mathcal{U}$.

To enable the measurement of unknown strings, we collect more information from clients, still using RAPPOR encoding. In addition to collecting a regular RAPPOR report of the client's full string, we collect RAPPOR reports generated from $n$-grams[2] selected randomly from the string. The key idea is to use co-occurrences among $n$-grams to construct a set of full-length candidate strings. For example, if the distribution contains only two words, "cats" and "dogs", we might measure randomly-selected bigrams, like "ca", "ts", "do", and "gs". The idea is to learn the *joint* distributions of these bigrams, to infer that "ca" and "ts" co-occur often, but "ca" and "gs" do not.

To analyze these co-occurrences, we use the joint distribution estimation algorithm developed in the previous section. Once we build a dictionary of candidate strings, we perform regular, marginal RAPPOR analysis on the full-string reports to estimate the distribution. In the limit, if our $n$-grams were as long as the string itself, we would be searching for candidates over $\mathcal{U}$, the space of all strings. By using small $n$-grams (2 or 3 characters long), we can significantly reduce the associated computational load, without compromising accuracy.

Our proposed method does not depend explicitly on the RAPPOR encoding mechanism. It only requires the aggregator to be able to learn joint distributions of string-valued random variables. Since EM is a general-purpose algorithm, it can be used to learn joint (or marginal) distributions irrespective of the encoding mechanism. Therefore, the method we propose in

[2] An $n$-gram is an $n$-character substring.

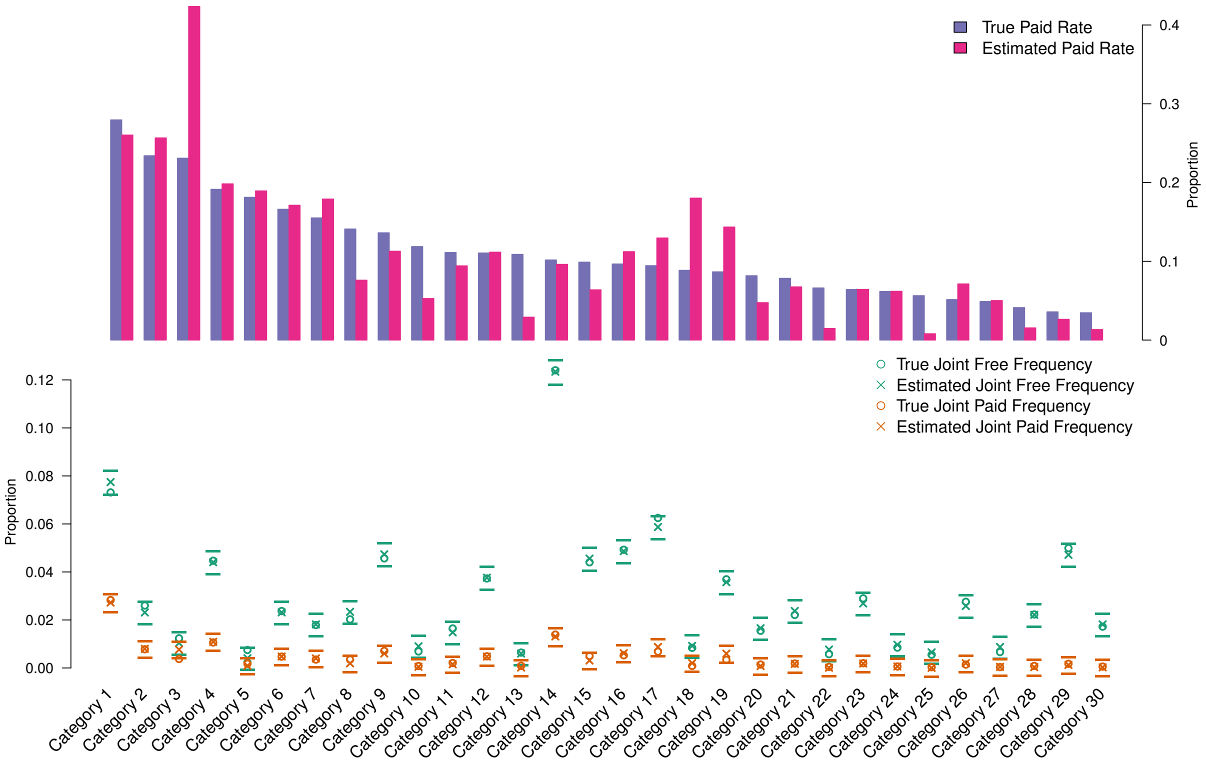

**Fig. 4.** Estimating the joint distribution of the categories of software items, and whether they are free or for purchase. Categories are ordered by the paid fraction, shown in the top panel. The bottom panel plots 60 true and estimated *joint* frequencies along with 95% confidence intervals shown as horizontal bars. Sample size is 100,000, and $\epsilon = 2$.

this section can be trivially applied to *any* differentially-private encoding.

Concretely, a client reporting string $x$ with local differential privacy budget $\epsilon$ would create a report, $X' = \text{RAPPOR}(x)$, by spending a third of her privacy budget (i.e., using differential privacy level $\epsilon/3$). The other two thirds of $\epsilon$ would be spent equally on collecting two $n$-grams, $G'_i = \text{RAPPOR}(n\text{-gram}(x, g_i))$, for $i \in \{1, 2\}$ at distinct random positions $g_1$ and $g_2$, where $n\text{-gram}(x, g_i)$ denotes the length-$n$ substring of $x$ starting at the $g_i$th character. The only limitations on $g_1$ and $g_2$ are that $g_1 \neq g_2$ and $g_1, g_2 \leq M - n$; one could choose partially overlapping $n$-grams. In our simulations, we partition the string into adjacent, non-overlapping $n$-grams. For instance, if our strings have at most $M = 6$ characters and our $n$-grams are two characters each, then there are only 3 bigrams per string; $g_1$ and $g_2$ are therefore drawn without replacement from the set $\{0, 2, 4\}$. In the original RAPPOR paper, each client would report a single randomized bit array $X'$. We instead collect $\{X', G'_1, G'_2, g_1, g_2\}$, where both $g_1$ and $g_2$, the two $n$-gram positions, are sent in the clear. To prevent leakage of information through the length of the string, $x$, the aggregator should specify a maximum string length $M$ and pad all strings shorter than $M$ with empty spaces. Strings longer than $M$ characters are truncated and hashed.

Note that one could use more than two $n$-grams. However, this would force each $n$-gram to use privacy level $\epsilon/(r + 1)$, where $r$ is the number of $n$-grams measured; this forces the client to send more data to achieve the same fidelity. Also, using more $n$-grams can significantly increase the complexity of estimating $n$-gram co-occurrences. For example, collecting 3 bigrams over the space of letters requires us to estimate a distribution over a sample space with $(26^2)^3$ possibilities. For this reason, we do not provide simulation results based on collecting more than two $n$-grams.

## 4.1 Building the Candidate Set

Let $N$ be the number of clients participating in the collection. The aggregator's reconstruction algorithm proceeds as follows:

1. **Build $n$-gram dictionary:** Start by building a subdictionary of every possible $n$-gram. If the alphabet has $D$ elements in it, this subdictionary will have $D^n$ elements. An example alphabet is $D = \{0-9, a-z, -, \_, .\}$.
2. **Marginal preprocessing:** Take the set of all reports generated from $n$-grams, $\{(G'_1)_i,\ (G'_2)_i\}_{i=1}^N$. Split this set into mutually exclusive groups based on the position from which they were sampled. There will be $M/n$ such groups.
3. **Marginal decoding:** For each position group, perform marginal analysis to estimate which $n$-grams are common at each position and their corresponding frequencies. This step uses the $n$-gram dictionary constructed in (1).
4. **Joint preprocessing:** Each pair of $n$-grams falls into one of $\binom{M/n}{2}$ groups, defined by the randomly-chosen positions of the two $n$-grams, $g_1$ and $g_2$. Split the reports into these groups.
5. **Joint analysis:** Perform separate joint distribution analyses for each group in (4) using the significant $n$-grams discovered in (3).
6. **$n$-gram candidates:** Select all $n$-gram pairs with frequency greater than some threshold $\delta$.
7. **String candidates (Graph-building):** Construct a graph with edges specified by the previously-selected $n$-gram pairs. Analyze the graph to select all $M/n$-node fully connected subgraphs which form a candidate set $C$.

Steps (3)–(7) are illustrated in Figure 5, but steps (6) and (7) require some more explanation. For simplicity assume that $M = 6$ and that we are collecting two bigrams from each client. For string $x$ with frequency $f(x)$, there could only be three different combinations of bigram pairs reported by each client: $(g_1, g_2) \in \{(0,2),\ (0,4),\ (2,4)\}$. If string $x$ is a true candidate, then we would expect the corresponding bigrams from *all three* pairings to have frequency of at least $f(x)$ in the relevant joint distributions. Additional frequency could come from other strings in the dictionary that share the same bigrams. In general, *all* $n$-gram pairs must have frequency greater than some threshold $\delta$ to produce a valid candidate. We take

$$\delta = \sqrt{\frac{p_2(1-p_2)}{(q_2-p_2)N}}, \tag{2}$$

where $q_2 = 0.5f(p+q) + (1-f)q$ and $p_2 = 0.5f(p+q) + (1-f)p$. This expression is designed to ensure that if an $n$-gram pair has no statistical correlation, then with high probability its estimated probability will fall below $\delta$. Indeed, $1.64\delta$ is a frequency threshold above which we expect to be able to distinguish strings from noise in our marginal analysis. We deliberately use a slightly lower threshold to reduce our false negative rate.

Step (7) is explained further in Figure 6. The idea is to construct a set of candidate strings by finding fully-connected cliques in a graph. Each $n$-gram at each position is treated as a distinct node. Edges are drawn between every valid $n$-gram pair from step (6). These edges may be due to true signal (solid lines) or noise (dotted lines), but the aggregator has no way of distinguishing *a priori*. Regardless of provenance, edges are only drawn between $n$-grams of different positions, so the resulting graph is $k$-partite, where $k = M/n$. Now the task simplifies to finding every fully-connected $k$-clique in this $k$-partite graph; each clique corresponds to a candidate string. If a string $x$ is truly represented in the underlying distribution, then the likelihood of *any* $n$-gram pair having a joint distribution below the threshold $\delta$ is small. Therefore, if even a single $n$-gram pair from string $x$ has a significantly lower frequency than $\delta$ after accounting for the noise introduced by RAPPOR, then it is most likely a false positive. Accordingly, the corresponding edge will be missing in the graph, and our clique-finding approach will discard $x$ as a candidate.

If executed naively, this clique-finding step can become a storage and computation bottleneck. Worst case, the number of candidates can grow exponentially in the number of bigrams collected. However, the problem of efficiently finding $k$-cliques in a $k$-partite graph has been studied [23].

Candidate strings can be further filtered based on string semantics and/or domain knowledge. For example, if it becomes apparent that what we are collecting are URLs, candidates that do not meet URL encoding restrictions can be discarded (e.g., strings with spaces in the middle).

## 4.2 Testing Candidate Strings

To estimate the marginal distribution of unknown strings, we use the set of full string reports $X'_1, \ldots, X'_N$ and candidate dictionary $C$ to perform marginal inference as described in the original RAPPOR paper. False positives in the candidate set $C$ will be weeded out in this step, because the marginal decoding shows that these strings occur with negligible frequency. The marginal analysis here differs from classical RAPPOR marginal analysis in two important ways:

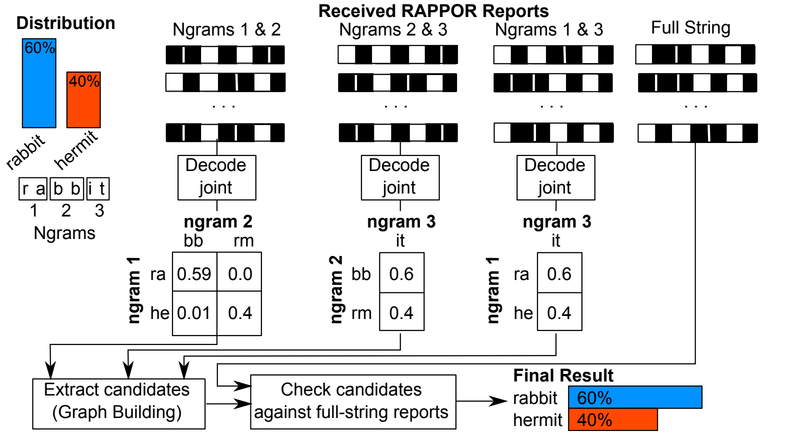

**Fig. 5.** Process for learning the distribution of a random variable without knowing the dictionary ahead of time. The aggregator computes pairwise joint distributions from the noisy reports generated at different $n$-gram positions. These pairwise joint distributions are used to generate a candidate string dictionary.

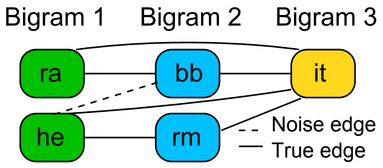

**Fig. 6.** The graph-building process for generating full string candidates identifies fully-connected cliques in a $k$-partite graph. In this example graph, the set of candidate strings would be $C = \{$rabbit, hermit, hebbit$\}$. The noisy false positive ("hebbit") gets removed by candidate testing (section 4.2).

(1) Reports $X'_1, \dots, X'_N$ are collected with stronger privacy by using privacy parameter $\epsilon/3$ instead of $\epsilon$.
(2) The estimated candidate set $C$ is unlikely to be as complete as an external knowledge-based set. With high probability, it will include the most frequent (important) candidates, but it will miss less frequent strings due to privacy guarantees imposed on $n$-gram reporting. On long-tailed distributions, a significant portion of distribution mass may fall below the noise floor. Set $C$ is also likely to be comprised of many false-positive candidates forcing a higher stress load on statistical testing that necessarily must be more conservative in the presence of a large number of tests.

The output of this step is the estimated marginal weights of the most common strings in the dictionary.

## 5 Results

We performed simulation studies and one real-world example to empirically show the utility of the proposed approach. We first discuss why this scheme does not alter RAPPOR's privacy guarantees.

**Privacy:** We split the privacy budget evenly between the $n$-grams and the full-string report. For instance, if we collect reports on two $n$-grams and the full string, each report will have privacy parameter $\epsilon/3$. The definition of local differential privacy implies that two independent measurements of the same datapoint, each with differential privacy parameter $\gamma$, will collectively have privacy parameter $2\gamma$. Dependent measurements contain less information than independent measurements, so the overall privacy parameter is at most $2\gamma$. Thus, our $n$-gram based measurement scheme provides the same privacy as a single RAPPOR report with differential privacy $\epsilon$. Note that local differential privacy guarantees hold even in the face of side information [19].

**Computational Efficiency:** The bottleneck of our algorithm is constructing the dictionary of candidate strings. This can be split into two phases: (a) computing $n$-gram co-occurrences, and (b) building the candidate dictionary from a $k$-partite graph of $n$-gram co-occurrences. Recall that $|D|$ is the size of our alphabet, and $r$ is the number of $n$-grams collected from each string. $N$ denotes the number of datapoints. Part (a) has complexity $O(N|D|^{nr})$ due to the EM algorithm. Part (b) depends on the size of the initial $k$-partite graph. If there are $p$ nodes in each of the partitions, this part has worst-case computational complexity $O(kp^{k-1})$. However, sparsity in this $k$-partite graph can reduce the complexity in practice.

These asymptotic costs can be prohibitive as the number of data samples increases. This is partially because the EM algorithm in phase (a) is iterative, and each iteration depends on every data element; this can lead to high memory constraints and lengthy runtimes. However, while the complexity of part (a) dominates part (b) in most usage scenarios, part (a) can also be parallelized more easily. For scalability, it would be most practical to use a parallelized implementation of EM estimation. We will also show how the parameter $\delta$ can be tuned to reduce the computational load of part (b) in exchange for a reduction in accuracy.

## 5.1 Simulated Results

In this section, we demonstrate empirically how the accuracy of this approach depends on various parameters: the privacy setting $\epsilon$, the number of reports $N$, the type of distribution being estimated, and n-gram parameters. We explored performance on a synthetic dataset of "hashes"—random, fixed-length character strings.

**Accuracy vs. $\epsilon$ and population size:** Figure 7 shows how accuracy of the method depends on $\epsilon$ and $N$. Here, $\epsilon$ refers to the differential privacy of the whole estimation scheme (including all n-grams and full string reports). Analogous plots for other distributions are included in Appendix C. These results suggest that the proposed method partially learns the unknown dictionary and distribution, but many strings may not be detected. Indeed, $\epsilon$ must be large ($\geq$ 3 or 4) to obtain meaningful estimates for populations with fewer than 100,000 users. Such values of $\epsilon$ are too large to offer substantial, general privacy-protection guarantees; this suggests that operationally, the proposed approach may too inefficient for smaller enterprises. More work is needed to understand how to improve the accuracy of these methods for a fixed number of users or reports.

To understand the source of this error, Figure 8 depicts a learned distribution, with and without the dictionary as prior knowledge. The most frequent strings' frequencies are estimated correctly, even without the dictionary; this implies that the error in Figure 7 between the true and estimated distributions stems mainly from learning an incomplete dictionary, rather than errors in distribution estimation.

The distribution used in this example (as well as subsequent tests), is a discretized power-law distribution with scaling exponent $\beta = -5$ over 100 random strings. We drew 100,000 strings from this distribution and encoded them as 128-bit RAPPOR reports, with parameters $p = 0.25$, $q = 0.75$, and $f = 0$; this corresponds to $\epsilon = 9$, per report. We limited ourselves to 100,000 strings for the sake of computational feasibility while exploring the parameter space. However, we show results in Sec. 5.2 from a larger trial on a real dataset with 1,000,000 simulated clients.

**Accuracy and $n$-gram length:** Figure 9 plots the Hellinger distance of our reconstructed distribution as a function of string length, for different sizes of $n$-grams.[3] This figure suggests that for a fixed string length, using larger $n$-grams gives a better estimate of the underlying dictionary. Intuitively, this happens for two reasons: (1) Reports generated from longer $n$-grams contain information about a larger fraction of the total string; we only collect two $n$-grams for communication efficiency, so the $n$-gram size determines what fraction of a string is captured by reports. (2) The larger the $n$-gram, the fewer $n$-gram pairs exist in a string of fixed length. In simulation, we observe that the likelihood of our algorithm missing an edge between $n$-grams is roughly constant, regardless of $n$-gram size. Therefore, if there are more $n$-gram pairs to consider with smaller $n$-grams, the likelihood that at least one of the edges is missing—thereby removing that string from consideration—is significantly higher.

This hypothesis is supported by Figure 10, which shows the false negative rate as a function of string size for different $n$-gram sizes (i.e., the fraction of dictionary strings that were not detected by the protocol). In all of these trials, we did not observe *any* false positives, so false negatives accounted for the entire discrepancy in distributions. Because our distribution was quite peaked (as is the case in many real-life distributions over strings), missing even a few strings caused the overall distribution distance to decrease significantly.

**Accuracy vs. computational costs:** As mentioned previously, graph-building becomes a bottleneck if the EM portion of the algorithm is properly parallelized and optimized. This stems from the potentially large number of candidate strings that can emerge while searching over the $k$-partite graph of $n$-grams. This number depends in part on the threshold $\delta$ used to select "significant" associations between $n$-grams. Choosing a larger threshold results in fewer graph edges and lower computational load, but this comes at the expense of more missed strings in the candidate set.

We examined the impact of the pairwise candidate threshold on accuracy. Setting this threshold to 0 recovers every string, while also greatly increasing the false positive rate, as well as the algorithmic complexity of finding those strings. Figure 11 plots the Hellinger distance of the recovered distribution against the number of edges in the candidate $n$-gram graph for various distribution thresholds. The number of edges in the $n$-gram k-partite graph indirectly captures the computational complexity required to build and prune candidates.

As expected, the computational complexity (i.e. number of edges in the candidate graph) decreases as the threshold increases. However, the accuracy decreases for very low thresholds. This is because each candidate string is treated as an independent hypothesis (the null

[3] We only generated one point using 4-grams due to the memory costs of decoding a dictionary with $26^4$ elements.

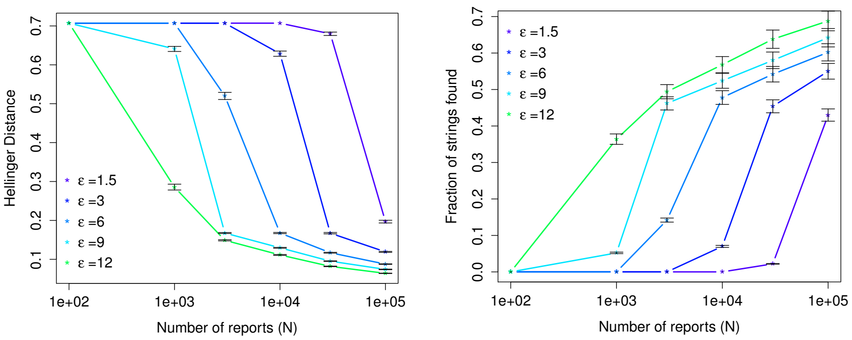

**Fig. 7.** Hellinger distance between the estimated distribution and the true distribution (left panel), and fraction of learned strings compared to the true dictionary size $|\mathcal{U}_P|$ (right panel), parameterized by $\epsilon$ and the number of reports $N$ (with 95% confidence-intervals). The underlying distribution is a discretized power-law distribution with scaling exponent $\beta = -5$ over 10 strings of 6 characters each.

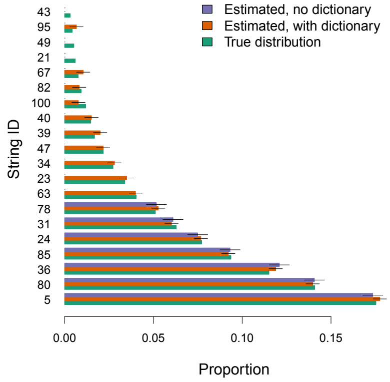

**Fig. 8.** Estimated distribution compared to the true distribution. Top 17 strings are learned with prior dictionary knowledge, compared to 7 strings without.

hypothesis being that the candidate is not significant). When testing for $M$ independent hypotheses with significance $\alpha$, *Bonferroni correction* is often used to reduce the significance of each individual test to $\alpha/M$ in order to account for the greater likelihood of seeing rare events when there are multiple hypotheses. Since lowering the threshold also increases the number of candidate strings, the resulting Bonferroni correction causes many true strings to fail the significance test. If we did not use Bonferroni correction, we would observe a high number of false positives. The optimal parameter setting is difficult to estimate without extensive simulation. However, we observe in simulation that the threshold in Eq. (2)—which is based on the statistics of the randomized response noise—appears close to a local optimum, and is likely a good choice in practice.

## 5.2 Estimating the Dictionary in Real-World Settings

To understand how this approach might work in a real setting, we located a set of 100 URLs with an interesting real-world frequency distribution (somewhat similar to the Alexa dataset [2]). We simulated measuring these strings through RAPPOR by drawing one million strings from the distribution, and encoding each string accordingly. We then decoded the reports using the methods in this paper.

All URL strings were padded with white space up to 20 characters, matching the longest URL in the set. In addition to full string reports, two randomly-chosen bigrams (out of 10) were also reported, all using 128-bit Bloom filters with two hash functions. Overall privacy parameters were set to $q = 0.75$, $p = 0.25$ and $f = 0$ (assuming one-time collection). This choice of parameters provides $\epsilon = 4.39$ or $\exp(\epsilon) = 81$ privacy guarantees, deliberately set high for demonstration purposes. Each of the collected reports—based on the string itself and two bigrams—were allotted equal privacy budgets of $\epsilon/3$, giving effective parameters $p = 0.25$ and $q = 0.32$.

Results are shown in Figure 12, where we truncate the distribution to the top 30 URLs for readability. Each URL's true frequency is illustrated by the green bar. The other three bars show frequency estimates for three different decoding scenarios. A missing bar indicates that the string was not discovered under that particular decoding scenario.

Under the first scenario, we performed an original RAPPOR analysis with $\epsilon = 4.39$ and perfect knowledge of the 100 strings in the dictionary. With 1 million reports, we were able to detect and estimate frequencies for 75 unique strings. The second scenario also assumes perfect knowledge of all 100 strings, but performs collection at $\epsilon/3 = 1.46$. This illustrates how much we lose purely by splitting up the privacy budget to accommodate sending more information. In this second scenario, 23 strings were detected, and their estimated frequencies are shown with blue bars.

In the third scenario, no prior knowledge of the dictionary was used. Each string and bigram was collected at privacy level $\epsilon/3 = 1.46$. Ten marginal bigram analyses for each bigram position returned 4, 2, 2, 3, 4, 5, 2, 1, 1, and 1 significant bigrams, respectively. After conducting joint distribution analysis on pairs of bigrams, we selected bigram pairs whose joint

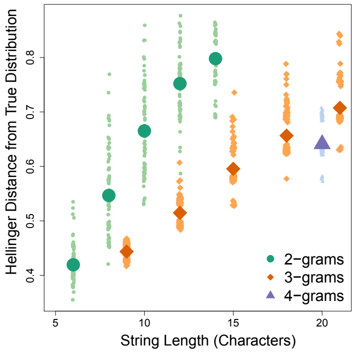

**Fig. 9.** Hellinger distance of learned distribution from true distribution as a function of total string length.

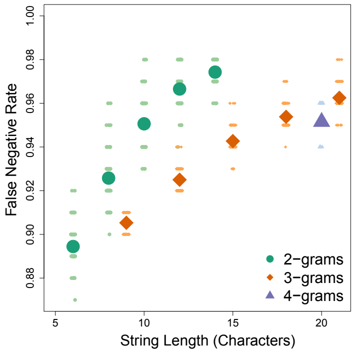

**Fig. 10.** False negative rate as a function of string length (i.e., fraction of unrecovered dictionary strings). Discrepancies between distributions are primarily caused by false negatives.

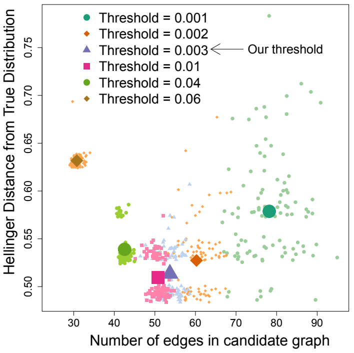

**Fig. 11.** Hellinger distance between the recovered and true distributions vs. the number of edges in the k-partite candidate graph. Clusters were generated using different edge-creation thresholds.

Effects of parameter choice on accuracy for a discretized power-law distribution with scaling $\beta = -5$ over 100 random strings. Sample size is 100,000, with $\epsilon = 9$ overall. This $\epsilon$ is consciously set high to highlight underlying trends. Each user reports two $n$-grams per string.

frequency was above the threshold cutoff of $\delta = 0.0062$. We then located the 10-cliques in the corresponding 10-partite graph, which produced 896 candidate strings. The final marginal analysis based on the full string reports (to weed out false positives) discovered the top five strings and estimated their frequency quite accurately (pink bars). There was also one false positive string identified by the analysis. We also reran the collection with trigrams, which produced only 185 candidate strings. Final marginal analysis resulted in only two strings with no false positives.

**A note on accuracy:** Our method fails to detect many of the distribution strings. While we make no claims of optimality, there is a well-studied fundamental tension between local differential privacy and data utility [9]. Since we collect each $n$-gram with privacy parameter $\epsilon/3$, our effective learning rate is significantly slower than that of regular RAPPOR. More problematically, estimating a multinomial distribution requires a number of samples that scales linearly in the support size of the distribution. So if we wish to estimate a distribution over an unknown dictionary of 6-letter words, in the worst case, we would need on the order of 300 million samples—a number that grows quickly in string length. Considering these limitations, it is not surprising that learning over an unknown dictionary performs significantly worse than learning over a known dictionary, regardless of algorithm. Our algorithm nonetheless consistently finds the most *frequent* strings, which account for a significant portion of the probability mass in many real-world distributions.

Critically, the natural drawback of this tradeoff is that an aggregator needs large amounts of data to effectively learn distributions. We want to emphasize that this is a property of the original RAPPOR mechanism, not of our post-processing methods. It is a significant problem that might be solved by using optimal encoding mechanisms or by using different privacy levels for different users, but these research questions are orthogonal to the present work.

## 6 Related Work

Since its introduction nearly a decade ago, differential privacy has become perhaps the best studied and most widely accepted definition of privacy [10]. When no party is trusted to construct a database of sensitive data, the notion of local differential privacy is often considered [9, 18, 20]. Research on local differential privacy is largely centered around finding algorithms that satisfy differential privacy properties [22, 24, 25], and improving the tradeoffs between privacy and utility [9, 18].

Our work follows a recent trend of using local differential privacy to learn a distribution's *heavy-hitters*—the most significant categories in a distribution [5, 12, 14, 21]. Several of these papers focus on the information-theoretic limits of estimating heavy-hitters while satis-

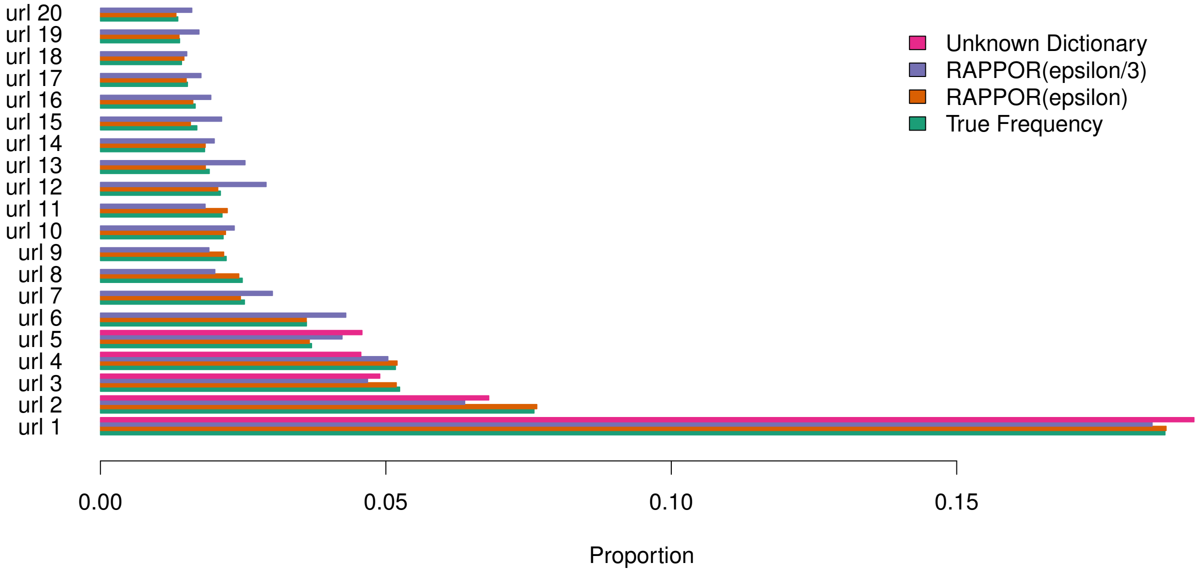

**Fig. 12.** Learning a distribution of URLs with and without the dictionary. The dictionary has 100 strings (top 20 are shown). From bottom up, the bars indicate: (1) the URL's true frequency, (2) the estimated frequency when collected at $\epsilon = 4.39$ with full dictionary knowledge, (3) estimated frequency with full dictionary knowledge, but data collected with $\epsilon/3 = 1.46$ (stronger privacy); this illustrates losses incurred by allocating 2/3 of the privacy budget to collecting two bigrams, (4) estimated frequency computed with no dictionary knowledge; each report and ngram was again encoded at $\epsilon/3$ privacy. The sample size was 1 million reports.

fying differential privacy. Our paper differs from existing work by asking new questions aimed at improving the practicality of the recently-introduced RAPPOR mechanisms [12]. Specifically, we consider two key questions: how to decode joint distributions from noisy reports, and how to learn distributions when the aggregator does not know the dictionary of strings beforehand.

Our work combats the recent notion that differential privacy is mainly of theoretical interest [3]. We have identified two main technical shortcomings of a differentially-private mechanism with practical, real-world deployment, namely RAPPOR, and provided solutions that address those shortcomings.

The question of estimating distributions from differentially private data is not new, with Williams *et al.* first making explicit the connection between probabilistic inference and differential privacy [17, 28]. This previous work is similar to our approach. There has even been some work on the related problem of releasing differentially private marginal distributions generated from underlying multivariate distributions [13].

However, previous work has focused on continuous random variables and learning the distribution of discrete data in a differentially-private manner (specifically, strings in an unknown dictionary, as we do here) has its own significant challenges. Such learning has been studied only recently, e.g., in work done concurrently with ours, [4], which describes sophisticated mechanisms whose characteristics are quite different from RAPPOR (e.g., offering no built-in longitudinal protection or defense against tracking). In comparison, our work builds directly upon RAPPOR, and therefore inherits its simple data-collection mechanisms, attack models, defenses, and other characteristics; thus, our work is directly suitable for use in existing RAPPOR deployments, without having to introduce, deploy, and build confidence in new privacy mechanisms.

## 7 Discussion

Privacy-preserving crowdsourcing techniques have great potential as a means of resolving the tensions between individuals' natural privacy concerns and the need to learn overall statistics—for each individual's benefit, as well as for the common good. The recently-introduced RAPPOR mechanism provides early evidence that such techniques can be implemented in practice, deployed in real-world systems, and used to provide statistics with some benefits—at least in the application domain of software security. In this paper, we consider two significant limitations of this original RAPPOR system—namely its inability to learn the associations between

RAPPOR-reported variables, and its need to known the data dictionary of reported strings ahead of time. We achieve those improvements without changing the fundamental RAPPOR mechanisms or weakening its local differential privacy guarantees.

That said, our new analysis techniques are not without shortcomings. Practically, the main limitation of our method is the accuracy vs. sample size tradeoff. We observe that in order to get meaningful dictionary estimation with a sample size of 20,000 users, the aggregator would need to set a privacy level of $\epsilon \geq 3$, which gives very weak privacy protection. This highlights the need for further work in designing mechanisms that allow privacy-preserving analytics at small sample sizes.

Another challenge we face is parameter selection and the allocation of privacy budgets. It may provide more utility, at similar overall levels of privacy, to collect $n$-grams with more relaxed privacy guarantees to get a better estimate of the candidate set, and then use stricter privacy settings when collecting full string reports. Because our algorithm's performance is distribution-dependent, it is difficult to estimate optimal settings theoretically. Moreover, searching over the complete parameter space is computationally challenging. We hope that the public release of our analysis mechanisms will encourage experimentation on both fronts.

# 8 Conclusions

Privacy-preserving crowdsourcing techniques based on randomized response can provide useful, new insights into unknown distributions of sensitive data, even while providing the strong guarantees of local $\epsilon$-differential privacy. As shown in this paper, such privacy-preserving statistical learning is possible even when there are multiple degrees and levels of unknowns. In particular, by augmenting the analysis methods of the existing RAPPOR mechanism it is possible to learn the joint distribution and associations between two or more unknown variables, and learn the data dictionaries of frequent, unknown values from large domains, such as strings. Furthermore, those augmented RAPPOR analysis techniques can be of value when applied to real-world data.

# Acknowledgment

The authors would like to thank Adam Smith for valuable comments and suggestions on how to improve the paper, as well as our anonymous reviewers. Special thanks to Andy Chu and Sooel Son for providing the Play Store data.

# References

[1] A. Agresti. *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2nd edition, 2002.

[2] Alexa. The top 500 sites on the web. http://www.alexa.com/topsites.

[3] Jane R Bambauer, Krish Muralidhar, and Rathindra Sarathy. Fool's gold: an illustrated critique of differential privacy. 2013.

[4] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *STOC*. ACM, June 2015, to appear.

[5] T-H Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies*, pages 140–159. Springer, 2012.

[6] Martin J Crowder. Maximum likelihood estimation for dependent observations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1976.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

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

[10] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[11] Robert F Engle et al. Wald, likelihood ratio, and lagrange multiplier tests in econometrics. *Handbook of econometrics*, 1984.

[12] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. *ACM CCS*, 2014.

[13] Stephen E Fienberg, Alessandro Rinaldo, and Xiaolin Yang. Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables. In *Privacy in Statistical Databases*, pages 187–199. Springer, 2011.

[14] Justin Hsu, Sanjeev Khanna, and Aaron Roth. Distributed private heavy hitters. In *Automata, Languages, and Programming*, pages 461–472. Springer, 2012.

[15] Google Inc. Google research blog: Learning Statistics with Privacy, aided by the Flip of a Coin. http://googleresearch.

blogspot.com/2014/10/learning-statistics-with-privacy-aided.html, .

[16] Google Inc. Unwanted Software Policy. http://www.google.com/about/company/unwanted-software-policy.html, .

[17] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. *arXiv preprint arXiv:1109.0105*, 2011.

[18] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, 2014.

[19] Shiva Prasad Kasiviswanathan and Adam Smith. A note on differential privacy: Defining resistance to arbitrary side information. *CoRR abs/0803.3946*, 2008.

[20] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, Washington, DC, USA, 2008.

[21] Joe Kilian, André Madeira, Martin J Strauss, and Xuan Zheng. Fast private norm estimation and heavy hitters. In *Theory of Cryptography*, pages 176–193. Springer, 2008.

[22] Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636. ACM, 2009.

[23] M Mirghorbani and P Krokhmal. On finding k-cliques in k-partite graphs. *Optimization Letters*, 7(6):1155–1165, 2013.

[24] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD International Conference on Management of data*, pages 735–746. ACM, 2010.

[25] Elaine Shi, T-H Hubert Chan, Eleanor G Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*, volume 2, page 4, 2011.

[26] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[27] Stanley Warner. Randomized response: A survey technique for eliminating evasive answer bias. *JASA*, 60(309):pp. 63–69, 1965.

[28] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *NIPS*, pages 2451–2459, 2010.

# A Evaluation of the Association Test

We evaluated the hypothesis-testing properties of our association statistic empirically. We considered pairs of random variables, which were chosen to be either independent or not. Our error rates using the association test from Section 3.4 allow us to study the Type I error and power $(1 - P(\text{Type II error}))$ of the test, under a null hypothesis of independence.

Consider two independent random variables, $X$ and $Y$. If we test a null hypothesis on these variables at a confidence level of $\alpha = 0.05$, we would expect to falsely reject the null hypothesis 5 percent of the time. More precisely, if the test statistic is continuous (as ours is), then the $p$-value is uniformly distributed between 0 and 1 if the null hypothesis is true.

Thus, in order to demonstrate that our proposed statistic can be used as a test of independence, we generated a pair of distributions of independent random variables, $X$ and $Y$ . In each trial, we drew $N = 10,000$ data points $\{(x_1, y_1), ..., (x_N, y_N)\}$. After encoding these data points with RAPPOR and then jointly decoding the reports, we obtain estimates $\hat{p}_{ij}$ and $\hat{\Sigma}$. We use these estimates to compute the Wald statistic, T, for a single trial, over 100 trials.

Since the null hypothesis is true by construction, the $p$-values of our test should have a distribution that is uniform. Therefore, the expected quantiles of our constructed dataset should be uniformly spaced between 0 and 1. Figure 13 plots these expected quantiles against our observed quantiles from the dataset. Because the points are well-represented by a linear fit with slope 1 and intercept 0, we conclude that our test statistic has the desired properties as a test of variable independence.

This allows us to compute the Type I error rate (false positive) as a function of sample size, dictionary size, and $\epsilon$, as shown in Figure 14. We used one variable with a fixed dictionary size of $n = 6$ strings, and we varied the other variable's dictionary size $m$. The Type I error appears to be upper-bounded by 0.05, which is consistent with our significance level. However, notice that these plots suggest that larger dictionaries have a lower rate of Type I error. This effect is due to the small sample size, which leads to small cell counts; it does not indicate that the test has better performance on large dictionaries. Indeed, we observe that as the sample size grows, the Type I error rate increases and stabilizes around $\alpha$.

Figure 15 shows the power $(1-P(\text{Type II error}))$ for a range of different dependent joint distributions. Distributions are generated as follows: two marginal distributions are drawn uniformly at random from the $m$- and $n$-dimensional probability simplexes, respectively. These marginals are used to compute the corresponding *independent* joint distribution. We then add a small, constant probability mass to the diagonal entries of the contingency table, and renormalize to generate a valid probability distribution. This allows us to control how much correlation we induce between the variables. 'Dist1' uses an added diagonal probability mass of 0.05, whereas

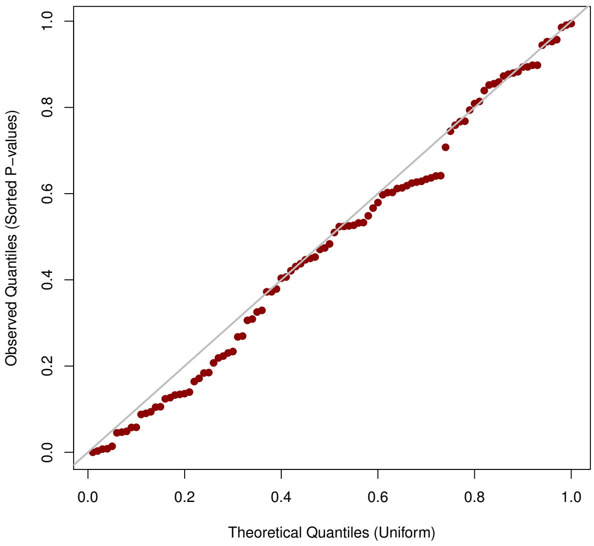

**Fig. 13.** Q-Q plot of the observed $p$-values for the Wald test versus the theoretical uniform distribution. As expected, $p$-values are uniformly distributed under the null hypothesis of independence.

'Dist2' adds a diagonal probability mass of 0.1. These measurements and distributions are non-exhaustive, but they suggest that the power of the test increases with $\epsilon$ and $N$, but decreases as the dictionary size grows. In particular, the test has low power for dictionaries with 10 strings. Also, the test has almost no power when $\epsilon \leq 1.5$; this confirms our intuition that these tests are only practical for large- to mid-sized enterprises.

# B Joint Distribution Estimation

The overall accuracy of joint estimation depends on the sample size $N$, the privacy level $\epsilon$, and the dictionary size. In Section 3.5, we showed the dependence of estimation accuracy on $\epsilon$ and $N$. Here, for a fixed $\epsilon = 1.5$, we varied the sample size $N$ and the dictionary size, and measured the Hellinger distance between the true distribution and the reconstructed distribution. We estimated a joint distribution of two random variables, and kept one dictionary size fixed to 6. The other variable's dictionary size was varied between 6 and 18. The results are included in Figure 16; this plot suggests that with privacy level $\epsilon = 1.5$, to achieve a fixed accuracy as the dictionary size increases by 6 strings, the necessary sample size can increase by as much as an order of magnitude. Thus, the proposed joint distribution estimation may only be practical for random variables with dictionary size $\approx 10$, unless sample sizes grow to $N \geq 100,000$.

# C Unknown Dictionary Results

In Section 5.1, we demonstrated the effect of privacy level $\epsilon$ and number of reports $N$ on accuracy for a power-law distribution. These results are distribution-dependent. Figures 17 and 18 illustrate the effect of $\epsilon$ and $N$ on the accuracy of our estimation for geometric and uniform distributions, respectively. Notice that here, $\epsilon$ is the *total* privacy budget for all ngram and string reports. The parameters of these experiments are the same as in Section 5.1, except for the underlying distribution; for the geometric distribution, we use parameter $p = 0.3$. These results suggest that random variables with higher entropy are more difficult to learn accurately for a given sample size.

# D Generalizing "Other" Estimation

It is straightforward to apply the EM algorithm to a more general class of locally-differentially-private mechanisms. However, in Section 3.2 we described how to estimate the "other" category explicitly in the context of RAPPOR. To estimate the "other" category for general randomization mechanisms that use symbol-wise randomized response, the aggregator once again uses knowledge of the top $m$ categories and their frequencies to estimate this conditional probability. Suppose that the differentially-private mechanism $\mathcal{Q}$ maps inputs from the space of strings to $k$ symbols from some alphabet $\mathcal{A}$. Let $c_s^m(i)$ be the expected number of times that the $s$th *reported* symbol was set to value $i \in \mathcal{A}$ by one of the top $m$ categories in $X$. $c_s^m(i)$ is mechanism-dependent, and depends on $N$ (the number of reports collected). The estimated proportion of times each symbol was set to value $i$ by a string from the "Other" category is then $\hat{p}_s^o(i) = \frac{c_s(i) - \hat{c}_s^m(i)}{N(1-\sum_{i=1}^m \hat{p}_i)}$, where $c_s(i)$ is the observed number of times symbol $s$ was set to $i$ in all $N$ reports. Because we have no knowledge of what strings are in the "Other" category, we treat it as a "single" string. The conditional probability of observing any report $X'$ given that the true value was "Other" is $P(X' = x' | X = \text{"}Other\text{"}) = \prod_{s=1}^k \hat{p}_s^o(x'_s)$. This estimate allows us to run the EM algorithm while modeling the

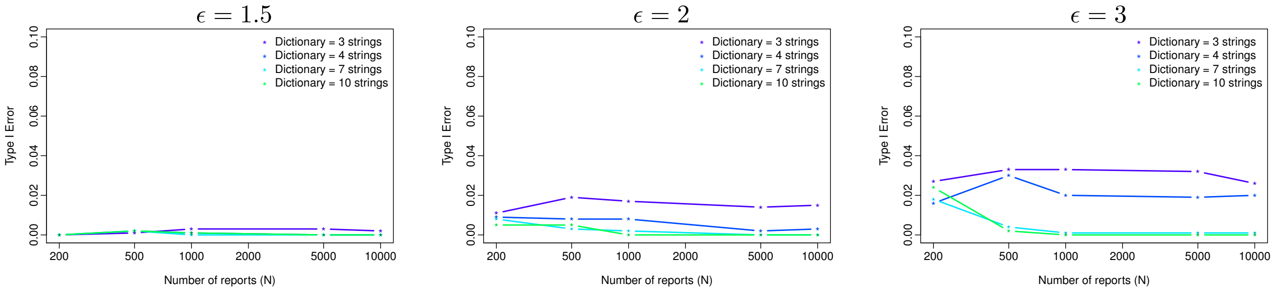

**Fig. 14.** Type I error of the Wald test for variable independence, at significance level $\alpha = 0.05$, averaged over 1000 trials. Measurements were taken from a bivariate distribution, with dictionary size $6 \times m$, where $m$ is varied from 3 to 10. Error rate is consistent with the expected rate of $0.05$. Larger dictionaries have lower error rates here because their cell frequencies are much smaller.

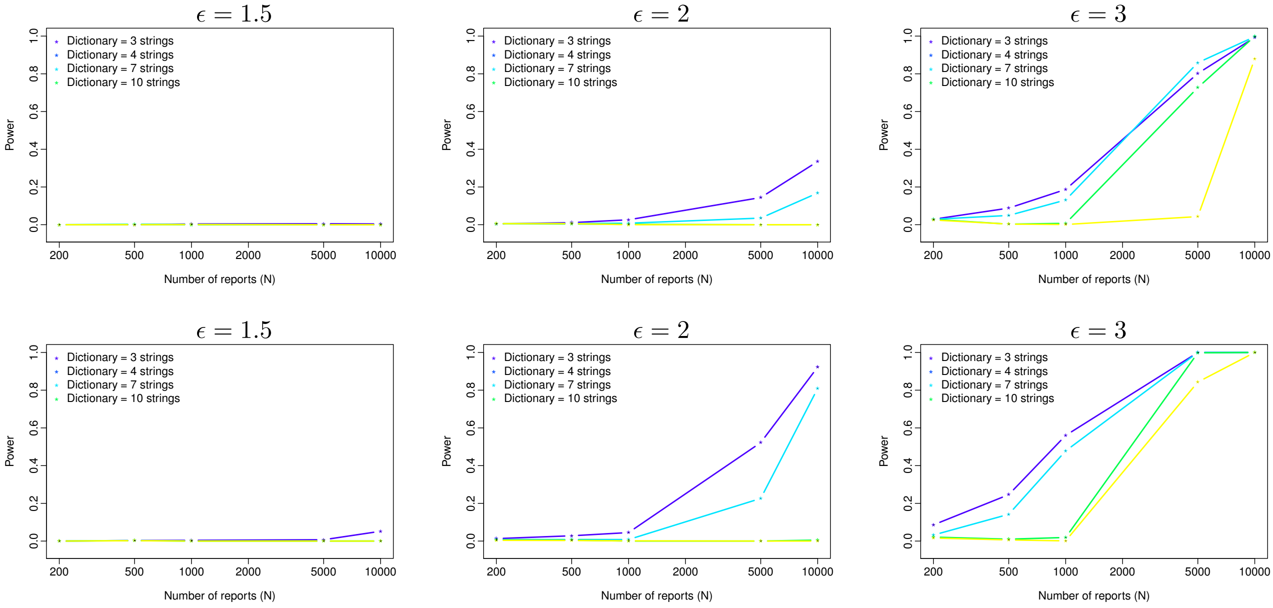

**Fig. 15.** Power of the Wald test for variable independence, at significance level $\alpha = 0.05$, averaged over 1000 trials. Measurements were taken from a bivariate distribution, with dictionary size $6 \times m$, where $m$ is varied from 3 to 10. Power grows as dictionary size decreases and $\epsilon$ increases. The method for generating dependent distributions is described in App. A; top row corresponds to "Dist1" (less dependency), and bottom row to "Dist 2" (more dependency).

unknown categories to obtain unbiased estimates for the known ones.

# E Proofs

## E.1 Proposition 3.1

It is enough to show that the log likelihood being maximized in the EM algorithm is concave. The log likelihood can be written as follows:

$$\mathcal{L}(X', Y'|p_{ij}) = \log\left(\prod_{k=1}^{N} P(X'_k, Y'_k|p_{ij})\right) =$$

$$\sum_{k=1}^{N} \log\left(\sum_{i,j} p_{ij} P(X'_k, Y'_k|X = x_i, Y = y_j)\right).$$

Since the log likelihood function is differentiable, we can show this by demonstrating that the Hessian is negative for all parameter values. For brevity in the rest of this proof, we will use $X'$ instead of $(X', Y')$, and we will use $p_i$, $1 \leq i \leq mn$ instead of $p_{ij}$. Differentiating, we get

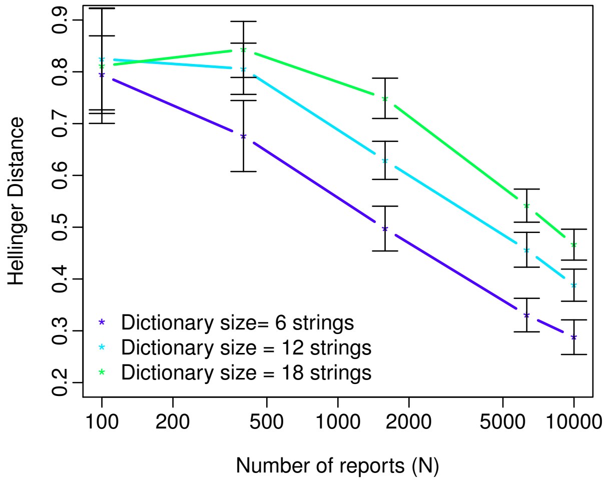

**Fig. 16.** Bivariate joint distribution reconstruction accuracy as a function of sample size $N$ for different dictionary sizes. One variable has dictionary size 6, and we vary the other. Results are averaged over 150 trials, and $\epsilon = 1.5$.

$\frac{\partial \mathcal{L}}{\partial p_i} = \sum_{k=1}^{N} \frac{P(X'_k|X=x_i)}{\sum_{j=1}^{mn} p_j P(X'_k|X=x_j)}$ and

$$\frac{\partial^2 \mathcal{L}}{\partial p_i \partial p_j} = \sum_{k=1}^{N} \frac{-P(X'_k|X=x_i)P(X'_k|X=x_j)}{\left(\sum_{\ell=1}^{mn} p_\ell P(X'_k|X=x_\ell)\right)^2}.$$

Since the denominator is strictly positive and the numerator is strictly negative (for all $i$ and $j$), we get that the log likelihood is concave. This implies that any local maximum is also a global maximum, so the EM algorithm converges to the maximum likelihood estimate of the parameters.

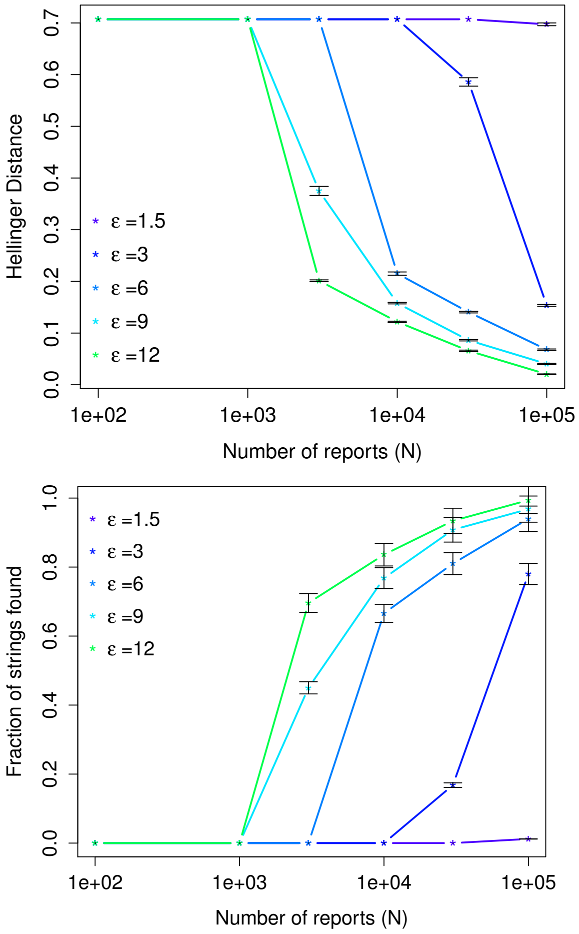

**Fig. 17.** (Geometric) Hellinger distance between the estimated distribution and the true distribution (top), and fraction of learned strings compared to the true dictionary size $|\mathcal{U}_P|$ (bottom), parameterized by $\epsilon$ and the number of reports $N$ (with 95% confidence-intervals). The underlying distribution is a geometric distribution over 10 strings of 6 characters each.

## E.2 Proposition 3.2

Since the EM algorithm produces an ML parameter estimate (Proposition 3.1), it is sufficient to verify the (mild) regularity conditions under which an ML estimate is asymptotically unbiased [26]:

1. Compactness: The parameter space $\Theta$ is compact.
2. Continuity: The probability $\mathbb{P}(X'; p_{ij})$ is continuous in $p_{ij}$. For economy of notation, we use $X'$ to denote the noisy variable pair $(X', Y')$.
3. Identifiability: $\mathbb{P}(X'; \hat{p}_{ij}) = \mathbb{P}(X'; p_{ij}) \iff \hat{p}_{ij} = p_{ij}$.
4. Boundedness: There exists a function $K(x')$ with $\mathbb{E}_{p_{ij}}[|K(X')|] < \infty$ and $\log(\mathbb{P}(x'; \hat{p}_{ij})) - \log(\mathbb{P}(x'; p_{ij})) \leq K(x')$ for all $x', p_{ij}$.

Condition 1 is met because the parameter space is precisely the probability simplex. The likelihood of observing reports $X'$ and $Y'$ parameterized by joint distribution $p_{ij}$ is precisely

$$\mathbb{P}(X', Y'; p_{ij}) = \sum_{i,j} p_{ij} \mathbb{P}(X', Y'|X = i, Y = j). \tag{3}$$

This is continuous in $p_{ij}$, so condition 2 is met. To show condition 3, suppose that $\mathbb{P}(X', Y'; \hat{p}_{ij}) = \mathbb{P}(X', Y'; p_{ij})$ for all $X', Y'$. This implies that

$$\sum_{i,j} \mathbb{P}(X', Y'|X = i, Y = j)(\hat{p}_{ij} - p_{ij}) = 0.$$

Notice that $\mathbb{P}(X', Y'|X = i, Y = j)$ is fixed and positive, which implies that $p_{ij} = \hat{p}_{ij}$ for all $i, j$. The other direction holds trivially by equation (3). To show condition

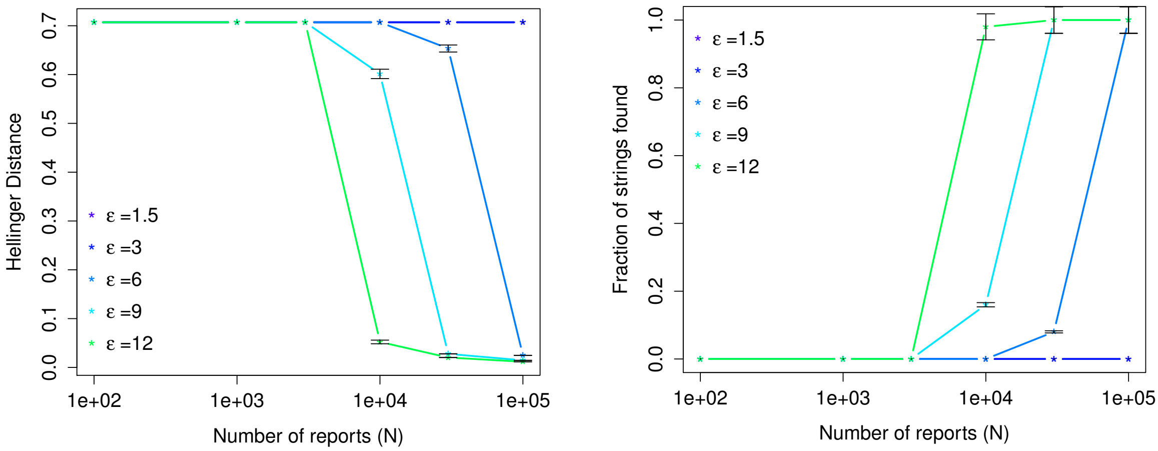

**Fig. 18.** (Uniform) Hellinger distance between the estimated distribution and the true distribution (left), and fraction of learned strings compared to the true dictionary size $|\mathcal{U}_P|$ (right), parameterized by $\epsilon$ and the number of reports $N$. The underlying distribution is a uniform distribution over 10 strings of 6 characters each.

4, notice that it is equivalent to

$$\frac{\mathbb{P}(X', Y'; \hat{p}_{ij})}{\mathbb{P}(X', Y'; p_{ij})} \leq e^{K(X',Y')}.$$

This gives

$$\begin{aligned}\frac{\mathbb{P}(X', Y'; \hat{p}_{ij})}{\mathbb{P}(X', Y'; p_{ij})} &= \frac{\sum_{i,j} \hat{p}_{ij} \mathbb{P}(X', Y' | X = i, Y = j)}{\sum_{i,j} p_{ij} \mathbb{P}(X', Y' | X = i, Y = j)} \\ &\leq \frac{\sum_{i,j} \hat{p}_{ij} e^{\epsilon} P^{max}_{X',Y'}}{\sum_{i,j} p_{ij} e^{-\epsilon} P^{min}_{X',Y'}} \quad (4) \\ &\leq \frac{e^{2\epsilon} P^{max}_{X',Y'}}{P^{min}_{X',Y'}}, \quad (5)\end{aligned}$$

where

$$P^{min}_{X',Y'} \equiv \min_{i,j} \mathbb{P}(X', Y' | i, j),$$

and

$$P^{max}_{X',Y'} \equiv \max_{i,j} \mathbb{P}(X', Y' | i, j),$$

and line (4) comes about from the definition of differential privacy. Thus, if we set $K(X', Y') = 2\epsilon + \log(P^{max}_{ij} / P^{min}_{ij})$, condition 4 is met. Therefore, we have that $p_{ij,N} \xrightarrow{a.s.} p_{ij}$.