Federated Heavy Hitters Discovery with Differential Privacy

Wennan Zhu
RPI

Peter Kairouz
Google

Brendan McMahan
Google

Haicheng Sun
Google

Wei Li
Google

Abstract

The discovery of heavy hitters (most frequent items) in user-generated data streams drives improvements in the app and web ecosystems, but can incur substantial privacy risks if not done with care. To address these risks, we propose a distributed and privacy-preserving algorithm for discovering the heavy hitters in a population of user-generated data streams. We leverage the sampling and thresholding properties of our distributed algorithm to prove that it is inherently differentially private, without requiring additional noise. We also examine the trade-off between privacy and utility, and show that our algorithm provides excellent utility while also achieving strong privacy guarantees. A significant advantage of this approach is that it eliminates the need to centralize raw data while also avoiding the significant loss in utility incurred by local differential privacy. We validate our findings both theoretically, using worst-case analyses, and practically, using a Twitter dataset with 1.6M tweets and over 650k users. Finally, we carefully compare our approach to Apple's local differential privacy method for discovering heavy hitters.

1 Introduction

Discovering the heavy hitters in a population of user-generated data streams plays an instrumental role in improving mobile and web applications. For example, learning popular out-of-dictionary words can improve the auto-complete feature in a smart keyboard, and discovering frequently-taken actions can provide an improved in-app user experience. Naively, a service provider can learn the popular elements by first collecting user data and then applying state-of-the-art centralized heavy hitters discovery algorithms (Cormode et al., 2003; Cormode and Hadjieleftheriou, 2008; Charikar et al., 2002). However, collecting and analyzing data from users can introduce privacy risks.

To overcome some of these risks, the service provider can use the central model of differential privacy (DP) to provide internal or external analysts with a privacy-preserving set of learned heavy hitters (Dwork et al., 2006b,a; Dwork, 2008; Dwork et al., 2010; Bhaskar et al., 2010; Dwork and Roth, 2014). However, this approach requires that users trust the service provider with their raw data. And even with a fully trusted service provider, tighter privacy regulations, such as Europe's General Data Protection Regulation (GDPR), the risk of hacks and other data breaches, and subpoena powers may encourage service providers to collect less data from their users.

The local model of DP (Warner, 1965; Evfimievski et al., 2004; Kasiviswanathan et al., 2011) addresses the above concerns by requiring users to perturb their data locally before sharing it with a service provider. Google (Erlingsson et al., 2014), Apple (Apple, 2017), and others (Ding et al., 2017; Kenthapadi and Tran, 2018) have deployed local DP algorithms. However, a large body of fundamental work shows that in the context of learning distributions and heavy hitters, local DP often leads to a significant reduction in utility (Kairouz et al., 2014; Wang et al., 2017; Bassily et al., 2017; Kairouz et al., 2016; Ye and Barg, 2018; Duchi et al., 2013; Cormode et al., 2018). As we show (e.g., Table 2), there are regimes where local DP is infeasible for practical use. Our goal is to provide practical algorithms that provide more privacy than prior approaches in such regimes, while maintaining sufficient utility (precision and recall).[1]

Our work builds on recent advances in federated learning (FL) (McMahan and Ramage, 2017; Konečnỳ et al., 2016; McMahan et al., 2017) to bridge the utility gap be-

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

[1]Whether or not a given approach provides sufficient privacy for a particular application is largely a domain-dependent policy question beyond the scope of this work; our goal is to expand the set of approaches available.

tween the local and central models of DP. Our proposed algorithm retains the essential privacy ingredients of FL: (a) no raw data collection (only ephemeral, focused updates from a random subset of users are sent back to the service provider), (b) decentralization across a large population of users (most users will contribute only 0 or 1 times), (c) interactivity in building an aggregate understanding of the population. However, unlike existing FL algorithms where the goal is to learn a prediction model, our work introduces a new federated approach that allows a service provider to discover the heavy hitters.

Contributions We develop an interactive heavy hitters discovery algorithm that achieves central DP while minimizing the data collected from users. In contrast to classical frequency estimation problems, our goal is to discover the heavy hitters but not their frequencies[2]. For example, in a smart mobile keyboard application, our algorithm allows a service provider to discover out-of-dictionary words and add them to the keyboard's dictionary, allowing these words to be automatically spell-corrected and typed using gesture typing.

We assume, without loss of generality,[3] that items (e.g., words) in user-generated data streams have a sequential structure (e.g., sequence of characters). Thus, we refer to items as sequences and leverage their sequential structure to build our algorithm. Our algorithm is interactive and runs in multiple rounds. In each round, a randomly selected set of users transmit a "vote" for a one element extension to popular prefixes discovered in previous rounds. The server then aggregates the received votes using a trie data structure, prunes nodes that have counts that fall below a chosen threshold θ, and continues to the next round.

We prove that our algorithm is inherently differentially private, and show how the parameters of the algorithm can be chosen to obtain precise privacy guarantees (see Theorem 1 and Corollary 1). When the number of users $n \geq 10^4$ and the sequences have a length of at most 10, our algorithm guarantees $(2, \frac{1}{n^2})$-differential privacy while achieving good utility (see Figure 2). See Table 1 for the DP parameters we can provide for various population sizes.

A key property of our algorithm is that it is sufficient for the service provider to receive only the set of extensions to the trie with votes that exceed a threshold θ, and the set of possible extensions is finite and known at the start of each round. A simple implementation of our algorithm would have the service provider directly receive each selected user's anonymous vote, and then immediately aggregate and threshold these votes in memory, with no persistence of the unaggregated votes.

However, our algorithm was explicitly designed to allow it to be implemented using aggregation schemes that further limit the information the service provider receives. In particular, a cryptographic secure sum protocol such as that of (Bonawitz et al., 2016) can be used to count votes, so the service provider never sees individual votes, only the aggregate sum over all users in the round (and only if a sufficient number of users participate). The service provider then is only trusted to apply the threshold θ. An intriguing open question is whether an efficient secure multi-party computation can be developed which also performs the thresholding. Another approach is to use the ESA architecture of (Bittau et al., 2017) to ensure shuffling and anonymization of the votes.

We have already discussed the privacy advantages of our approach compared to centralized approaches with DP that collect and store raw user data; undoubtedly such approaches could offer even higher utility, but we do not empirically assess this, as it is enough to show our algorithm achieves sufficient utility to be practical in many settings. Rather, we focus our empirical evaluation of utility on a comparison to local DP (in particular (Apple, 2017)), demonstrating that our algorithm obtains a strong central DP guarantee and high utility in settings where local DP performs poorly (see Table 2 for details). We use the Sentiment140 dataset, a Twitter dataset with 1.6M tweets and over 650k users Go et al. (2009). For Sentiment140, the top 200 words are recalled at a rate close to 1 with $\varepsilon = 4$ and $\delta < 5 \times 10^{-9}$.

Related work Federated learning (FL) (McMahan et al., 2017; Konečný et al., 2016; Bonawitz et al., 2019) is a collaborative learning approach that enables a service provider to learn a prediction model without collecting user data (i.e., while keeping the training data on user devices). The training phase of FL is interactive and executes in multiple rounds. In each round, a randomly chosen small set of online users download the latest model and improve it locally using their training data. Only the updates are then sent back to the service provider where they are aggregated and used to update the global model. Much of the existing works are in the context of learning prediction models. Our work differs in that it focuses on federated algorithms for the discovery of heavy hitters.

Differential privacy (DP) is a rigorous privacy notion that has been carefully studied over the last decade (Dwork et al., 2006b,a; Dwork, 2008; Dwork and Roth, 2014) and widely adopted in industry (Ding et al., 2017;

[2]Observe that once the popular items are discovered, learning their frequencies can be done using off-the-shelf DP techniques.

[3]Regardless of the items' data type, they can always be represented by a sequence of bits.

Apple, 2017; Kenthapadi and Tran, 2018; Erlingsson et al., 2014). It provides the ability to make strong formal privacy guarantees by bounding the worst-case information loss. There is a rich body of work on distribution learning, frequent sequence mining, and heavy-hitter discovery both in the central and local models of DP (Bhaskar et al., 2010; Bonomi and Xiong, 2013; Diakonikolas et al., 2015; Xu et al., 2016; Zhou and Lin, 2018; Kairouz et al., 2016; Wang et al., 2017; Bassily et al., 2017; Acharya et al., 2018; Ye and Barg, 2018; Avent et al., 2017; Bun et al., 2018; Cormode et al., 2018), and some recent works combine FL with central DP (Geyer et al., 2017; McMahan et al., 2018). The central model of DP assumes that users trust the service provider with their raw data while the local one gets away with this assumption. Thus, the utility loss is not as severe in the central model where the service provider may have access to the entire dataset. Our work bridges these existing models of privacy in that it allows an honest-but-curious service provider to learn the popular sequences in a centrally differentially private way, while only having access to minimal data: a randomly chosen user submits one character extension to an already discovered popular prefix.

Methods that provide DP typically involve adding noise, such as Gaussian noise, to the data before releasing it. In this work, we show that DP can be obtained without the addition of any noise by relying exclusively on random sampling and trie pruning which achieves k-anonymity. The connection between DP, random sampling, and k-anonymity has previously appeared in the literature (Chaudhuri and Mishra, 2006; Li et al., 2012; Gehrke et al., 2012). However, our approach and analysis are different in two fundamental ways. First, existing methods show how sampling and enforcing k-anonymity at the sequence level (in a centralized setting) can achieve central DP. When applied to our decentralized setting, such approaches have the disadvantage of revealing the entire sequences held by sampled users. On the contrary, our approach explores how interactivity, random sampling, and k-anonymity can achieve central DP while also drastically minimizing the data a user shares with the service provider. Second, our sampling method is different from existing methods that sample records from a centralized database in an i.i.d fashion (referred to as *Poisson sampling*). Under Poisson sampling, the number of chosen users can vary drastically across rounds, making such approach incompatible with existing federated learning production systems such as (Bonawitz et al., 2019). Instead, we sample (uniformly at random) a fixed number of users in each round. Combined with interactivity over rounds, this different sampling strategy makes our approach and proof techniques different from existing ones.

Our trie-based heavy hitters (TrieHH) algorithm exploits the hierarchical structure of user-generated data streams to interactively maintain a trie structure that contains the frequent sequences. The idea of using trie-like structures for finding frequent sequences in data streams has been explored before in (Cormode et al., 2003; Bassily et al., 2017). However, the work of Cormode et al. (2003) predates differential privacy and the TreeHist algorithm of Bassily et al. (2017) is non-interactive, relies on sketching, achieves local DP using the randomized response, and assumes the existence of public randomness. Our approach is interactive in nature, does not use sketching or offer local DP, and does not require public randomness. The only similarity between these two approaches is the use of a trie-like data structure that maintains a list of popular prefixes, a practice that is common for efficient discovery of heavy hitters (even under no privacy constraints). In fact, the differences between these two approaches lead to a fundamentally different privacy-utility trade-off and make private heavy-hitter discovery feasible even for small-to-moderate populations.

In Section 5, we compare TrieHH with Apple's Sequence Fragment Puzzle (SFP) algorithm, a state-of-the-art sketching based algorithm for discovering heavy hitters with local DP (Apple, 2017). Similar to TreeHist, SFP is also a count sketch based algorithm. However, instead of pruning by a tree structure, SFP estimates high frequency substring fragments and then stitches them together to get full length heavy hitters. We provide our source code implementation of SFP at `https://github.com/tensorflow/federated/tree/master/tensorflow_federated/python/research/triehh`, and a detailed description of this algorithm in Section D of the accompanying supplementary material.

2 Preliminaries

Model and notation We consider a population of n users $\mathcal{D} = \{u_1, u_2, \ldots, u_n\}$, where user i has a collection of items $\{w_{i1}, w_{i2}, \cdots, w_{iq}\}$. We abuse notation and use $\mathcal{D}$ to refer to both the set of all users and set of all items. Without loss of generality, we assume that the items have a sequential structure and refer to them as sequences. More precisely, we express an item w as a sequence $w = c_1 c_2 \ldots c_{|w|}$ of $|w|$ elements. For example, in our experiments (see Section 5), we focus on discovering heavy-hitter words in a population of tweets generated by Twitter users. Therefore, each user has a collection of words, and each word can be expressed as a sequence of ASCII characters. We assume that the length of any sequence is at most L.

For any set $\mathcal{D}$, we build a trie via a randomized algorithm $\mathcal{M}$ to obtain an estimate of the heavy hitters. We let $p_i(w)$ denote the prefix of w of length i. For a trie T and a prefix $p = c_1, c_2 \dots c_i$, we say that $p \in T$ if there exists a path $(\text{root}, c_1, c_2, \dots, c_i)$ in T. Also, let T_i denote the subtree of T that contains all nodes and edges from the first i levels of T. Suppose $(root, c_1, c_2, \dots, c_i)$ is a path of length i in T_i. Growing the trie from T_i to T_{i+1} by "adding prefix $(root, c_1, c_2, \dots, c_i, c_{i+1})$ to T_i" means appending a child node c_{i+1} to c_i.

Differential privacy A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private iff for all $\mathcal{S} \subseteq Range(\mathcal{M})$, and for all adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$:

$$P(\mathcal{M}(\mathcal{D}) \in \mathcal{S}) \leq e^{\varepsilon} P(\mathcal{M}(\mathcal{D}') \in \mathcal{S}) + \delta. \quad (1)$$

We adopt user-level adjacency where $\mathcal{D}$ and $\mathcal{D}'$ are adjacent if $\mathcal{D}'$ can be obtained by adding all the items associated with a single user from $\mathcal{D}$ (McMahan et al., 2018). This is stronger than the typically used notion of adjacency where $\mathcal{D}$ and $\mathcal{D}'$ differ by only one item (Dwork and Roth, 2014).

3 Single Sequence per User

In this section, we consider a simple setting where each user has single sequence. Much of the intuition behind the algorithm and privacy guarantees we present in this section carry over to the more realistic setting of multiple sequences per user.

We describe the proposed approach via a simple example (shown in Figure 1) where the goal is to discover popular words. Suppose we have $n = 20$ users and each user has a single word. Assume there are three popular words: "star" (on 3 devices), "sun" (on 4 devices) and "moon"(on 4 devices). The rest of the words appear once each. We add a "$" to the end of each word as an "end of sequence" (EOS) symbol. In each round, the service provider selects $m = 10$ random users, asks them to vote for a prefix of their word (as long as it is an extension of the prefixes learned in previous rounds), and stores the prefixes that receive votes greater than or equal to $\theta = 2$ in a trie. In the example in the figure, two prefixes "s" and "m" of length 1 grow on the trie after the first round. This means that among the 10 randomly selected users, at least two of them voted for "s" and at least another two voted for "m". Observe that users who have "sun" and "star" share the first character "s", so "s" has a significant chance of being added to the trie. In the second round, 10 users are randomly selected and provided with the depth 1 trie learned so far (containing "s" and "m"). In this round, a selected user votes for the length 2 prefix of their word only if it starts with an "s" or "m". The service provider then aggregates the received votes and adds a prefix to the trie if it receives at least $\theta = 2$ votes. In this particular example, prefixes "st", "su", and "mo" are learned after the second round. This process is repeated for prefixes of length 3 and 4 in the third and the fourth rounds, respectively. After the fourth round, the word "sun$" is completely learned, but the prefix "sta" stopped growing. This is because at least two of the three users holding "star" were selected in the second and third round, but less than two were chosen in the fourth one. The word "moon$" is completely learned in the fifth round. Finally, the algorithm terminates in the sixth round, and the completely learned words are "sun$" and "moon$".

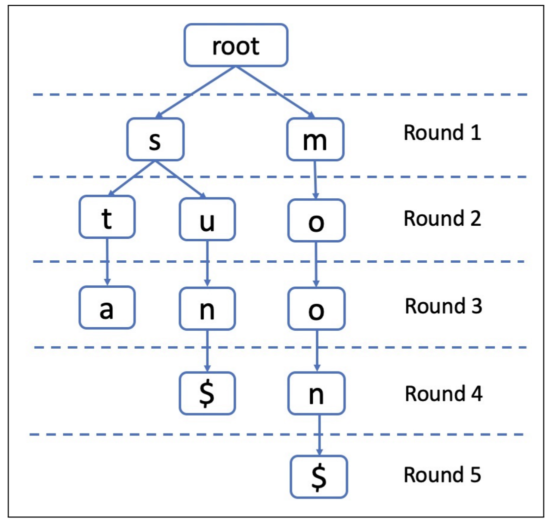

Figure 1: Example run of Algorithm 1.

Algorithm 1 Trie-based Heavy Hitters $\mathcal{M}(\mathcal{D}, \theta, \gamma)$

Input: A set $\mathcal{D} = \{u_1, u_2, \dots, u_n\}$ that have words $\{w_1, w_2, \dots, w_n\}$. A threshold θ. Batch size $m = \gamma\sqrt{n}$.
Output: A trie T.
Set $T = \{root\}$; $T_{old} = None$; $i = 1$;
while $T \mathrel{!}= T_{old}$ **do**
  Choose m users from $\mathcal{D}$ randomly to get a set $\tilde{\mathcal{X}}$ of sequences;
  $T_{old} = T$;
  $T = \mathcal{V}(\tilde{\mathcal{X}}, T, \theta, i)$; i++;
end while
return T;

To describe the algorithm formally, for a set of users $\mathcal{D}$, our algorithm $\mathcal{M}(\mathcal{D}, \theta, \gamma)$ runs in multiple rounds, and returns a trie that contains the popular sequences in $\mathcal{D}$. In each round of the algorithm, a batch of size $m = \gamma\sqrt{n}$ (with $\gamma \geq 1$) users are selected uniformly at random from $\mathcal{D}$. Note that there are interesting trade-offs between the utility and privacy with different choices of γ, which we will discuss later.

In the i^{th} round, randomly selected users receive a

Algorithm 2 Algorithm $\mathcal{V}(\tilde{\mathcal{X}}, T_{in}, \theta, i)$ to grow a trie by one level with a set of sequences.

Input: A set of sequences $\tilde{\mathcal{X}} = \{w'_1, w'_2, \ldots, w'_m\}$. An input trie T_{in} with i levels. A threshold θ.
Output: An output trie.
Initialize Candidates$[w'_j] = 0$ for all $w'_j \in \tilde{\mathcal{X}}$;
for each sequence w'_j in $\tilde{\mathcal{X}}$ that $|w'_j| \geq i$ and $p_{i-1}(w'_j) \in T_{in}$ **do**
  Candidates$[p_i(w'_j)]$++;
end for
return $T_{in} + \{p \mid \text{Candidates}[p] \geq \theta\}$;

trie containing the popular prefixes that have been learned so far. If a user's sequence has a length $i-1$ prefix that is in the trie, they declare the length i prefix of the sequence they have. Otherwise, they do nothing. Prefixes that are declared by at least $\theta \approx \log n$ selected users grow on the i^{th} level of the trie. Note that we grow at most one level of the trie in each round of the algorithm. Thus, if $c_1, \ldots, c_{i-1} \notin T_{i-1}$, then $c_1, \ldots, c_{i-1}, c_i$ cannot be in T_i. The final output of $\mathcal{M}$ is the trie returned by the algorithm when it stops growing. Algorithm 1 describes our distributed algorithm and Algorithm 2 shows a single round of the algorithm to grow one level of the trie.

Given the final trie, we extract the heavy-hitter sequences learned by Algorithm 1 by simply outputting the discovered prefixes from the root to leaves that end with \$ (the EOS symbol). Note that the non-EOS leaves also represent frequent prefixes in the population, which might still be valuable depending on the application.

Privacy guarantees Algorithm 1 has several privacy advantages: (a) randomly chosen users vote on a single character extension to an already discovered popular prefix, (b) the votes are ephemeral (i.e., never stored), and (c) a total of $L\gamma\sqrt{n}$ randomly chosen users participate in the algorithm. More importantly, sequences discovered by Algorithm 1 are k-anonymous with $k = \theta$, and as shown in the theorem below, the output of Algorithm 1 is inherently (ε, δ)-differentially private – without the need for additional randomization or noise addition.

Theorem 1. *When* $4 \leq \theta \leq \sqrt{n}$ *and* $1 \leq \gamma \leq \frac{\sqrt{n}}{\theta+1}$, *Algorithm 1 is* $(L\ln(1+\frac{1}{\frac{\sqrt{n}}{\gamma\theta}-1}),\ \frac{\theta-2}{(\theta-3)\theta!})$*-differentially private.*

Proof Sketch. Suppose $\mathcal{D}$ is obtained by adding w to a neighboring $\mathcal{D}'$ and assume $|w| = l$. We first decompose any $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ into $\mathcal{S}_0 \cup \mathcal{S}_1 \cup \ldots \mathcal{S}_l$, where $\mathcal{S}_0 = \{T \in \mathcal{S} | p_i(w) \notin T,\ \text{for } i = 1, 2, \ldots, l\}$ and $\mathcal{S}_i = \{T \in \mathcal{S} | p_1(w), \ldots, p_i(w) \in T \text{ and } p_{i+1}, \ldots, p_l \notin T\}$ for $i = 1, 2, \ldots, l$. Assume there are k users in $\mathcal{D}'$ that have prefix $p_i(w)$. Then we show that when k is large, the ratio between $P(\mathcal{M}(\mathcal{D}) \in \mathcal{S}_i)$ and $P(\mathcal{M}(\mathcal{D}') \in \mathcal{S}_i)$ is small so it could be bounded by e^{ε}. When k is small, $P(\mathcal{M}(\mathcal{D}) \in \mathcal{S}_i)$ is small enough so it could be bounded by δ. Intuitively, when k is large, it means prefix $p_i(w)$ is already popular in $\mathcal{D}'$, so the fact that $\mathcal{D}$ has one more user with this prefix does not affect the probability of it showing in the result too much. When k is small, the chance of prefix $p_i(w)$ showing up in the result is very small, even with an extra user with it in $\mathcal{D}$. □

The above result holds for a wide array of algorithm parameters (L, γ, and θ). The following corollary shows how precise privacy guarantees can be obtained by tuning the algorithm's parameters.

Corollary 1. *To achieve* (ε, δ)*-differential privacy, set* $\gamma = (e^{\frac{\varepsilon}{L}} - 1)\sqrt{n}/(\theta e^{\frac{\varepsilon}{L}})$ *and* $\theta = max\{10, \lceil e^{W(C_\delta)+1} - \frac{1}{2}\rceil, \lceil e^{\frac{\varepsilon}{L}} - 1\rceil\}$, *where* W *is the Lambert* W *function (Corless et al., 1996) and* $C_\delta = e^{-1}\ln(\frac{8}{7\sqrt{2\pi}}\delta^{-1})$. *Further, when* $n \geq 10^4$, *choosing* $\theta = \lceil \log_{10} n + 6 \rceil$ *ensures that Algorithm 1 is* $(\varepsilon,\ \frac{1}{300n})$*-differentially private* [4].

Table 1 shows how we can choose γ and θ to achieve $(\varepsilon, 1/(300n))$ and $(\varepsilon, 1/n^2)$ for various values of n. Since under Algorithm 1 the privacy loss can be large with probability δ (unlike mechanisms that rely on explicit noise addition), we focus (almost exclusively) on $\delta < 1/n^2$ in Section 5 where we conduct experiments on real data and compare to local differential privacy.

n	$L = 10$			
	$\delta \leq \frac{1}{300n}$		$\delta \leq \frac{1}{n^2}$	
	θ	γ	θ	γ
10^4	10	1.81	12	1.51
10^5	11	5.21	14	4.09
10^6	12	15.10	15	12.08
10^7	13	44.09	17	33.71

Table 1: Lower bound of θ and upper bound of γ to achieve $\varepsilon = 2$ in two cases: $\delta \leq \frac{1}{300n}$ and $\delta \leq \frac{1}{n^2}$.

Utility guarantees By the sampling nature of Algorithm 1, sequences that appear more frequently are more likely to be learned. The batch size m and threshold θ could be tuned to trade off utility for privacy. For a user set of size n, smaller m and larger θ achieve better privacy at the expense of lower utility, and vice versa.

To quantify utility under Algorithm 1, we examine the worst-case discovery rate of a sequence (probabil-

[4] In general, to get a $\delta \leq \frac{1}{n^a}$, by standard approximation of the Lambert function, we can choose $\theta \approx a(\ln n/\ln\ln n)$.

ity of discovering it) as a function of its frequency in the dataset. In particular, we consider the worst-case discovery rate which captures the probability of discovering a sequence assuming that it shares no prefixes with other sequences in the dataset. In the presence of such common prefixes, the discovery rate will only get better (see Section 5 for a comparison between worst-case discovery rates and ones that are achievable on real data).

Proposition 1. *Suppose a sequence appears W times in a dataset of n users where the longest sequence has length L. Then the worst-case discovery rate under Algorithm 1 is given by*

$$\left(\frac{1}{\binom{n}{m}} \sum_{i=\theta}^{\min\{W,m\}} \binom{W}{i} \binom{n-W}{m-i} \right)^L . \qquad (2)$$

Using Corollary 1 and Proposition 1, we can investigate how large the population should be if we want to discover sequences with high probability for a fixed ε. Figure 2 shows the relationship between sequence frequency and population size n if we want the worst-case discovery rate to be at least 0.9 for different ε's. Naturally, in order to be discovered with high probability, lower frequency sequences require larger population size, and vice versa. We also need larger populations for stronger privacy guarantees (smaller ε).

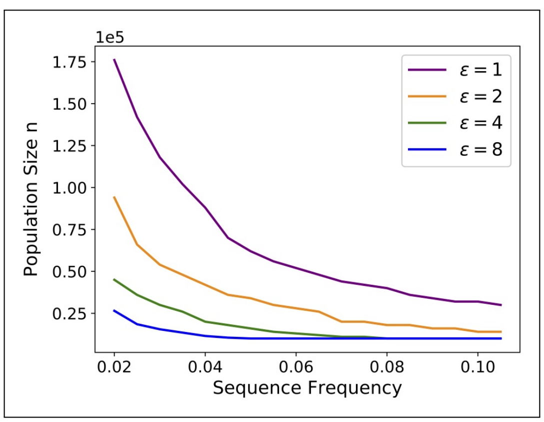

Figure 2: Minimum n required to ensure (via Proposition 1) a worst-case discovery rate greater than 0.9 for $L = 10$ and $\delta = 1/n^2$.

Remarks A few remarks are in order. First, in a production implementation of Algorithm 1, not all users may be online in every round of the protocol. In such a situation, the service provider will sample uniformly at random from available users. Therefore, assuming a strong adversary which knows the number and identities of online users in every round, the privacy guarantees will be determined by the number of online users. Second, Theorem 1 shows that the range of γ is: $[1, \sqrt{n}/(\theta+1)]$. Thus, $\gamma = 1$ is enough to achieve single digit epsilon, and if users are available, it could be increased up to $\sqrt{n}/(\theta+1)$ to achieve better utility. More importantly, this paper tackles the regime where $n \sim 10^5 - 10^7$ – see Table 1 for the choices of γ to get maximum utility in this setting. Even the upper bound on γ is not on the order of $\sqrt{n}$ (but rather 2 to 3 orders smaller than $\sqrt{n}$). For instance, $\gamma \approx 33$ when $\varepsilon = 2$, $\delta = 1/n^2$ and $n = 10^7$. Third, we study the communication cost of Algorithm 1 in Section E.1 of the accompanying supplementary material, but it is not the central quantity that this work focuses on.

4 Multiple Sequences per User

In this section, we consider the more general setting where each user could have more than one sequence on their device. Suppose the population is a set of n users $\mathcal{D} = \{u_1, u_2, \ldots, u_n\}$, and each user u_i has a set of sequences $\{w_{i1}, w_{i2}, \ldots, w_{iq}\}$.

Let $c_i(w_j)$ denote the number of appearances of w_j on u_i's device. We define the local frequency of w_j on u_i's device as $f_i(w_j) = c_i(w_j)/\sum_j c_i(w_j)$. Note that the sum of all the sequences' local frequencies on u_i's device is 1, i.e. $\sum_j f_i(w_j) = 1$. If a sequence w_j has 0 appearance on u_i's device, then $f_i(w_j) = 0$. Similarly, for a certain prefix p_j, let $c_i(p_j)$ denote the number of appearances of p_j on u_i's device. Then the frequency of p_j on u_i's device is $f_i(p_j) = c_i(p_j)/\sum_j c_i(p_j)$.

We are now ready to generalize Algorithm 1 to accommodate multiple sequences per user. In each round of the algorithm, we select a batch of m users from $\mathcal{D}$ uniformly at random. A chosen user u_i randomly selects a sequence $w_j \in u_i$ with probability $f_i(w_j)$, i.e., according to its local frequency. Thus, as in Algorithm 1, we still select m sequences from m users in every round. The voting step by these m sequences proceeded in the same way described in Algorithm 2. Algorithm 3 (in Section A of the supplementary material) shows the full algorithm.

Interestingly, the differential privacy guarantees we obtained in the single sequence setting also hold in the multiple sequence setting. This is formally stated in Corollary 2. To get this conclusion, we first provide the following more general (but intuitive) result.

Theorem 2. *Assume mechanism M achieves (ε, δ) record-level*[5] *DP on a dataset of size n. Consider a setting where we have n users and an arbitrary number of records per user. Then the mechanism that first selects 1 record per user (deterministically or randomly)*

[5]The difference between record-level and user-level DP is in the way neighboring datasets are defined. Under record-level DP, only a single record is varied when comparing $\mathcal{D}$ to $\mathcal{D}'$.

then applies M to the sampled dataset of size n achieves (ε, δ) user-level DP.

Corollary 2. *When $4 \le \theta \le \sqrt{n}$ and $1 \le \gamma \le \frac{\sqrt{n}}{\theta+1}$, Algorithm 3 (in Section A of the supplementary material) is $(L \ln(1 + \frac{1}{\frac{\sqrt{n}}{\gamma\theta} - 1}), \frac{\theta-2}{(\theta-3)\theta!})$-differentially private.*

5 Experiments

We now showcase the performance of the trie-based heavy hitters (TrieHH) algorithm on real data and compare it to Apple's Sequence Fragment Puzzle (SFP) algorithm, a state-of-the-art sketching based algorithm for discovering heavy hitters with local DP (Apple, 2017). We provide our source code implementation of both SFP and TrieHH at `https://github.com/tensorflow/federated/tree/master/tensorflow_federated/python/research/triehh`, and include a detailed description of SFP in Section D of the supplementary material. For a fair comparison between SFP and TrieHH, we "amplify" the local ε_{local} used by SFP to a central (ε, δ) used in TrieHH according to Theorem 5.3 of Balle et al. (2019). We also focus exclusively on the discovery stage of SFP and do not account for the count estimation stage. Since the trade-off between precision and recall could be tuned by a parameter T [6] under SFP, we compare TrieHH and SFP using precision, recall, and F_1 score. We use Sentiment140, a rich Twitter dataset (Go et al., 2009), and conduct three sets of experiments (see below for details). We run our experiments many times and report averaged utility metrics with 0.95 confidence intervals.

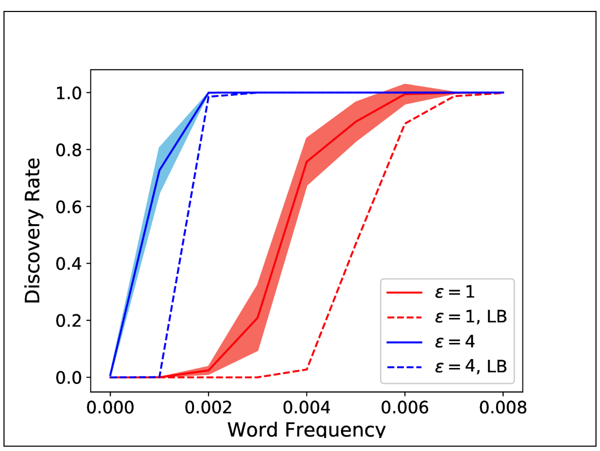

Figure 3: Frequency vs. discovery rate with the theoretical lower bound in the single word setting. $(\delta = 1/n^2)$

Single word per user: heavy hitters case To simulate this setting that each user has a single word using Sentiment140, we create a dataset by choosing the word with highest local frequency for each user and apply TrieHH on this dataset. Figure 3 shows the relationship between the word frequencies and the discovery rate using TrieHH. We limit L to 10, set $\delta = 1/n^2$, and choose θ and γ according to Corollary 1 to achieve various values of ε. The dashed lines represent the theoretical worst-case bounds on the discovery probability (presented in Section 3). Observe that there is a gap between the experimental results and the theoretical worst-case ones. This is because the theoretical bounds assume that sequences share no prefixes with others in the dataset, while in Sentiment140, many English words do share some prefixes. We also study the F_1 score of the K highest frequency words in the population. Figure 4 shows the F_1 score of the top K words vs. K with comparison to SFP. For SFP, $\varepsilon = 1 \to \varepsilon_{local} = 4.29$ and $\varepsilon = 4 \to \varepsilon_{local} = 4.96$. Observe that at $\varepsilon = 4$, the top 100 words have an F_1 score close to 1 under TrieHH, in comparison to an and F_1 score close to 0.2 under SFP.

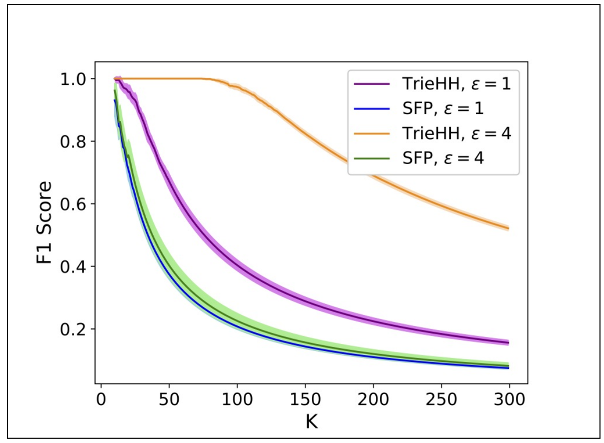

Figure 4: F_1 Score of the top K words in the single word setting. $T = 20$ for SFP.

Single word per user: out-of-vocab (OOV) case To simulate this setting using Sentiment140, OOV words are obtained by first scanning through the dataset and keeping only words that are made up of English letters and a few other symbols (such as "@" and "#") and then ensuring that these words do not belong to a highly tuned dictionary of over 260k words. After this pre-processing step, the frequencies of the OOV words are calculated and a dataset of size 6M is sampled according to those frequencies. Figure 5 shows the F1 score of the top K words for both TrieHH and SFP. Observe that the curves for both TrieHH and SFP are not monotonically decreasing for small K. This is because there are many long words in the top 10 to 20 of the OOV Twitter dataset (corresponding to usernames of trending Twitter users), and both algo-

[6] The parameters are proxies and do not necessarily represent the actual performance of Apple's system.

rithms perform worse for longer words. For larger K, the lengths of top words get smaller and more consistent. Table 2 shows recall at $K = 50$ and precision for both algorithms with different choices T for SFP. For SFP, $\varepsilon = 1 \rightarrow \varepsilon_{local} = 5.31$ and $\varepsilon = 4 \rightarrow \varepsilon_{local} = 5.99$ due to amplification. By increasing T for SFP, there is a gain of recall but the precision also drops dramatically. Some examples of interesting OOV words we have discovered include: "*hugs*", "*sigh*", ":'(", "@tommcfly", "@dddlovato", "#ff", "#fb", "b/c", "ya'll". The complete list of heavy-hitter OOV words and discovered ones are given in Section E.2 of the supplementary material.

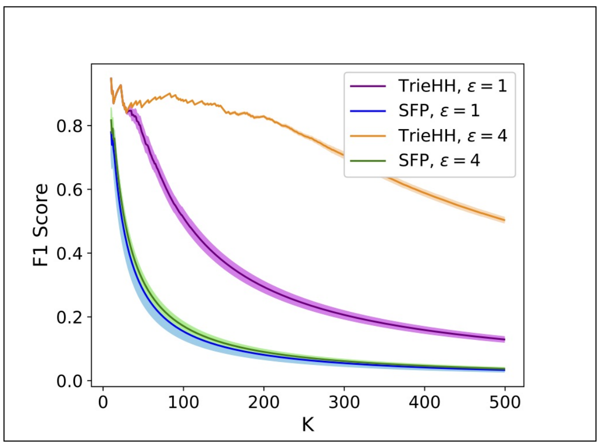

Figure 5: F1 Score of the top K words in the single word setting of OOV case ($\delta = 1/n^2$). $T = 20$ for SFP.

	$\varepsilon = 1$		$\varepsilon = 4$	
	Recall	Prec	Recall	Prec
TrieHH	0.65	1	0.76	1
SFP (20)	0.17	0.853	0.19	0.867
SFP (80)	0.25	0.494	0.325	0.456

Table 2: Comparison of recall at $K = 50$ and precision between TrieHH and SFP in the OOV setting for $\delta = \frac{1}{n^2}$ and $T = 20, 80$ under SFP.

Multiple words per user: heavy hitters case We use Sentiment140 as is for this experiment and calculate the population frequency of w_j by $F(w_j) = \frac{1}{n}\sum_i f_i(w_j)$. Similar to the single word setting, Figure 6 shows the relationship between the word frequency and the discovery rate using Algorithm 3. Note that in the multiple words setting, it is difficult to get a non-trivial lower bound on the discovery rate of Algorithm 3 because such bound heavily depends on the distribution of words. Figure 6 shows the discovery rate and Figure 7 shows the recall of the top K words. Observe that the top 200 words are recalled at a rate close to 1 with $\varepsilon = 4$ and $\delta < 5 \times 10^{-9}$

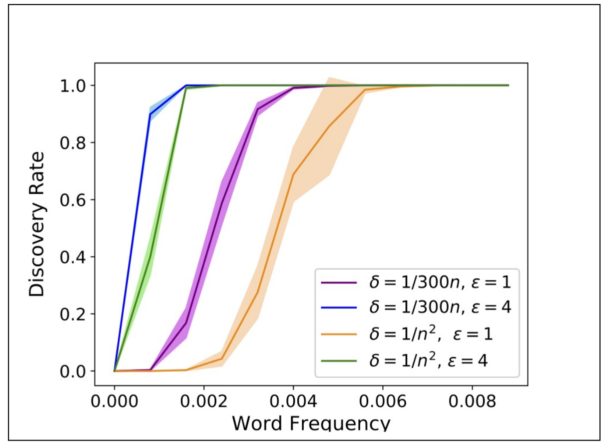

Figure 6: Sequence frequency vs. the discovery rate in the multiple words setting.

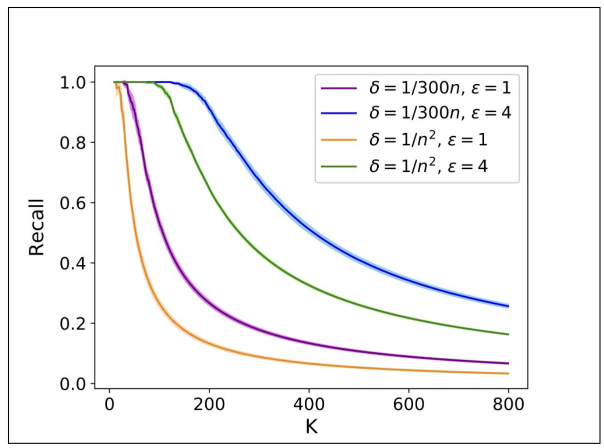

Figure 7: Recall of the top K words for different fixed ε in the multiple words setting.

6 Conclusion and Open Questions

We have introduced a novel federated algorithm for learning the frequent sequences, proved that it is inherently differentially private, investigated the trade-off between privacy and utility, and showed that it can provide excellent utility while achieving strong privacy guarantees. A significant advantage of this approach is that it eliminates the need to centralize raw data while also avoiding the harsh utility penalty of differential privacy in the local model. Many questions remain to be addressed, including (a) examining whether or not interactivity is necessary, (b) exploring secure multiparty computation and cryptographic primitives such as shuffling, threshold oblivious pseudorandom functions, and fully homomorphic encryption to provide stronger privacy guarantees, and (c) investigating the role of local plausible deniability (by allowing users to vote on wrong prefixes with small probability) and analyzing the privacy amplification gains obtained in the central model.

References

Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Communication efficient, sample optimal, linear time locally private discrete distribution estimation. *arXiv preprint arXiv:1802.04705*, 2018.

Apple. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. Blender: enabling local search with a hybrid differential privacy model. In *Proc. of the 26th USENIX Security Symposium*, pages 747–764, 2017.

Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. The privacy blanket of the shuffle model. *arXiv preprint arXiv:1903.02837*, 2019.

Raef Bassily, Uri Stemmer, Abhradeep Guha Thakurta, et al. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.

Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–512. ACM, 2010.

Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, pages 441–459, 2017. URL https://arxiv.org/abs/1710.00901.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. URL https://arxiv.org/abs/1902.01046.

Luca Bonomi and Li Xiong. Mining frequent patterns with differential privacy. *Proceedings of the VLDB Endowment*, 6(12):1422–1427, 2013.

Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, pages 435–447, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-4706-8. doi: 10.1145/3196959.3196981. URL http://doi.acm.org/10.1145/3196959.3196981.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

Kamalika Chaudhuri and Nina Mishra. When random sampling preserves privacy. In *Annual International Cryptology Conference*, pages 198–213. Springer, 2006.

Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the lambertw function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. *Proc. VLDB Endow.*, 1(2):1530–1541, August 2008. ISSN 2150-8097. doi: 10.14778/1454159.1454225. URL http://dx.doi.org/10.14778/1454159.1454225.

Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 464–475. VLDB Endowment, 2003. ISBN 0-12-722442-4. URL http://dl.acm.org/citation.cfm?id=1315451.1315492.

Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146. ACM, 2018.

Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. Differentially private learning of structured discrete distributions. In *Advances in Neural Information Processing Systems*, pages 2566–2574, 2015.

Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 429–438. IEEE, 2013.

Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and*

Trends in Theoretical Computer Science, 9(3–4):211–407, 2014.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006b.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29 (4):343–364, 2004.

Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In *Annual Cryptology Conference*, pages 479–496. Springer, 2012.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2879–2887. Curran Associates, Inc., 2014.

Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, pages 2436–2444, 2016.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, June 2011. ISSN 0097-5397. doi: 10.1137/090756090. URL `http://dx.doi.org/10.1137/090756090`.

Krishnaram Kenthapadi and Thanh TL Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. *arXiv preprint arXiv:1809.07754*, 2018.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 32–33. ACM, 2012.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

H Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, April 2017. URL `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`. Google AI Blog.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *Proc. of the 26th USENIX Security Symposium*, pages 729–745, 2017.

Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309): 63–69, 1965.

Shengzhi Xu, Xiang Cheng, Sen Su, Ke Xiao, and Li Xiong. Differentially private frequent sequence mining. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2910–2926, 2016.

Min Ye and Alexander Barg. Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, 2018.

Fengli Zhou and Xiaoli Lin. Frequent sequence pattern mining with differential privacy. In *International Conference on Intelligent Computing*, pages 454–466. Springer, 2018.