Federated Heavy Hitter Analytics with Local Differential Privacy

YUEMIN ZHANG, The Hong Kong Polytechnic University, China
QINGQING YE†, The Hong Kong Polytechnic University, China
HAIBO HU, The Hong Kong Polytechnic University, China

Federated heavy hitter analytics enables service providers to better understand the preferences of cross-party users by analyzing the most frequent items. As with federated learning, it faces challenges of privacy concerns, statistical heterogeneity, and expensive communication. Local differential privacy (LDP), as the *de facto* standard for privacy-preserving data collection, solves the privacy challenge by letting each user perturb her data locally and report the sanitized version. However, in federated settings, applying LDP complicates the other two challenges, due to the deteriorated utility by the injected LDP noise or increasing communication/computation costs by perturbation mechanism. To tackle these problems, we propose a novel target-aligning prefix tree mechanism satisfying ϵ-LDP, for federated heavy hitter analytics. In particular, we propose an adaptive extension strategy to address the inconsistencies between covering necessary prefixes and estimating heavy hitters within a party to enhance the utility. We also present a consensus-based pruning strategy that utilizes noisy prior knowledge from other parties to further align the inconsistency between finding heavy hitters in each party and providing reasonable frequency information to identify the global ones. To the best of our knowledge, our study is the first solution to the federated heavy hitter analytics in a cross-party setting while satisfying the stringent ϵ-LDP. Comprehensive experiments on both real-world and synthetic datasets confirm the effectiveness of our proposed mechanism.

CCS Concepts: • **Security and privacy → Data anonymization and sanitization**.

Additional Key Words and Phrases: Local differential privacy, heavy hitter, federated analytics

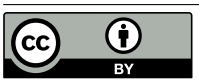
ACM Reference Format:
Yuemin Zhang, Qingqing Ye, and Haibo Hu. 2025. Federated Heavy Hitter Analytics with Local Differential Privacy. *Proc. ACM Manag. Data* 3, N1 (SIGMOD), Article 42 (February 2025), 27 pages. https://doi.org/10.1145/3709739

1 Introduction

Federated learning [28], which was introduced in 2016, allows developers to train machine learning models across multiple data parties without centralized data collection. Following the success of this computing paradigm, federated analytics (FA), which was coined by Google in 2020 for its Gboard application [44], involves utilizing federated technologies to answer more fundamental queries about decentralized data that often do not involve machine learning. Federated analytics mainly considers analytical queries, especially statistical queries [19], such as heavy hitters (i.e., the most frequent items) identification [12, 55, 69]. As with federated learning, FA faces the challenges of **privacy concerns, statistical heterogeneity, and expensive communication** [50]. Several

†Corresponding author.

Authors' Contact Information: Yuemin Zhang, ymin.zhang@connect.polyu.hk, The Hong Kong Polytechnic University, China; Qingqing Ye, qqing.ye@polyu.edu.hk, The Hong Kong Polytechnic University, China; Haibo Hu, haibo.hu@polyu.edu.hk, The Hong Kong Polytechnic University, China.



ACM 2836-6573/2025/2-ART42
https://doi.org/10.1145/3709739

privacy-preserving techniques (e.g., differential privacy, secure multi-party computing) have been investigated to tackle privacy concerns in statistical queries [1, 5, 56, 59]. However, most of them are restricted to a simplified scenario where the client is semi-trusted and holds users' raw data [15], or each client only includes a single user [56].

In practice, users are often distributed across different parties [46]. For instance, Amazon may require its branches in different regions (e.g., Europe and America) to collaboratively identify the top k items frequently purchased during the Christmas holiday, to develop more effective marketing plans. For the sake of privacy, users are not willing to upload their raw data to the branches since they may not be fully trusted. Local differential privacy [30] (LDP), as the golden standard for private data collection, allows users to perturb their data locally and send the sanitized data to the server, making it well-suited for privacy-preserving federated data analytics.

In this work, we study the problem of federated (a.k.a., cross-party) heavy hitter analytics in the context of LDP. In fact, LDP has been proposed and adopted in many distributed systems, such as Emoji and Safari usage data collection in Apple [3] and Chrome usage data analysis in Google [20]. While the privacy challenge of federated heavy hitter identification appears to be addressable by LDP, it unfortunately complicates the other two challenges, namely statistical heterogeneity and expensive communication/computation. In particular, as the data across different parties usually exhibit statistical heterogeneity (e.g., non-IID nature), simply aggregating the noisy data perturbed by the LDP mechanism may not accurately reflect the global representation of the data distribution. On the other hand, LDP mechanisms (e.g., unary encoding [20]) often involve encoding a single value into a vector (e.g., with a domain length $|\mathcal{X}| = 2^{64}$), resulting in high communication costs (e.g., up to $O(|\mathcal{U}| \cdot |\mathcal{X}|)$) in the server side where $|\mathcal{U}|$ is the user population size. Furthermore, such mechanisms identify heavy hitters by making $|\mathcal{X}|$ queries to estimate the frequency of every value in $\mathcal{X}$, however, issuing $|\mathcal{X}|$ queries is computationally infeasible [13, 34, 55]. As such, parties are not expected to transmit the large amount of user data (even being perturbed) to the central server, as it may consume large network bandwidth and excessive computation cost.

Fortunately, most of FA tasks can be disentangled into several sub-tasks, building on the natural basis that the overall results are composed of various partial results held by different parties and users. For example, in federated heavy hitter identification, an item is identified as a global top k item if the sum of its counts on Amazon Europe and Amazon America is ranked within the top k. As such, the central server can break down the task of federated heavy hitter identification into some sub-tasks and delegates them to each party instead of requiring them to upload the local datasets.

Following this design, a naive idea is to implement an existing LDP solution (e.g., PEM [55]) in each party to identify local heavy hitters and then the central server aggregates these results by counting. However, it is non-trivial to design such a mechanism with good utility in the federated setting, because of the non-IID nature of decentralized data. First of all, data heterogeneity across different parties can lead to skewed (and noisy) frequency distributions, resulting in many false positives. On the other hand, the non-IID issue also complicates the estimation and aggregation processes. The local heavy hitters identified within a party may not contribute to identifying the global ones. For instance, some globally infrequent items may be popular within a specific party, while overwhelming the other globally frequent ones and resulting in inconsistencies.

In this paper, we propose a target-aligning prefix tree mechanism for federated heavy hitter analytics, motivated by the typical using of a prefix tree (a.k.a, *trie*) structure to iteratively identify heavy hitters [12, 69] in the literature. In particular, we develop a two-phase process to align local and global targets. In the first phase, we design a shared shallow trie construction strategy, where each party independently estimates its trie with the same level. The aggregated leaves of this trie serve as a warm start for subsequent independent estimation of the second phase in each party, assisting in the filtration of false positive prefixes at a shallow level. Meanwhile, we introduce an

adaptive trie extension strategy, by integrating the frequency distribution and the LDP noise scale as constraints to enhance the estimation. This strategy serves to adaptively determine the prefixes to be extended at different levels within each party.

We further optimize the estimation accuracy by introducing a consensus-based pruning strategy. This is because, in the trie construction process, the extended domains are likely to contain numerous unnecessary candidates, resulting in large domain sizes and thus introducing excessive LDP noises. Firstly, globally infrequent prefixes or items do not serve as useful candidates. Secondly, some prefixes or items may be popular globally but infrequent in specific parties due to non-IID issues. Both types of prefixes fail to provide reliable frequency information. In FA settings, it is a natural idea to seek priori knowledge from other parties to improve the current party's estimation. Therefore, our consensus-based pruning strategy is designed to implement the second phase sequentially, so that each party adaptively prunes unnecessary candidates suggested by the previous one.

The key contributions of our study are summarized as follows:

- This is the first to explore stringent ϵ-LDP solution to the federated heavy hitter analytics under the cross-party setting. A novel target-aligning prefix tree mechanism is designed to adaptively extend the prefix tree at different levels.
- We further propose a consensus-based pruning strategy that enhances the alignment between the local and global targets. This strategy utilizes the prior knowledge from other parties to assist the pruning in the current party effectively.
- We conduct comprehensive experiments on real-world and synthetic datasets to confirm the superiority of our mechanism in comparison with existing solutions.

The remainder of this paper is organized as follows. We discuss related works in Section 2. In Sections 3 and 4, we provide necessary background information and the problem statement. In Sections 5 and 6, we detail the proposed mechanism. In Section 7, we present the experimental results. Finally, we conclude our paper in Section 8.

2 Related Work

Differential privacy (DP) [17] has been considered the golden standard for privacy protection in recent years, and has widely used in many tasks such as frequent subgraph mining [9], high-dimensional data publishing [8, 10, 47, 65], and learning problems [22]. As it always relies on a trusted server, LDP [30, 32] is proposed to allow users to locally perturb their data and report a noisy version. A number of studies have been conducted on the application of LDP, e.g., key-value data collection [62] and time-series data release [61], and several companies have implemented LDP in their products [3, 14, 20]. Some recent works study data poisoning attacks to LDP [6, 26, 48].

Federated analytics [44] allows individuals to collaboratively contribute to analytical tasks that do not require training, such as histogram construction and heavy hitters [50], in contrast to federated learning, which requires training a neural network. The problem of identifying privacy-preserving heavy hitters is a fundamental non-training application and has been studied under both DP and LDP settings. Zhu *et al.* [69] propose TrieHH that satisfies DP. It focuses on single-party settings and does not satisfy LDP. In contrast, our study focuses on and well addresses challenges raised in the multi-party setting while achieving a rigorous LDP guarantee. TrieHH++ [12] extends TrieHH and provides (ϵ, δ)-aggregated differential privacy. Chadha *et al.* [7] address a scenario where each user possesses multiple data points under aggregate DP. Li *et al.* [34] explore tracking heavy hitters on data streams at bounded memory expense under LDP. Wang *et al.* [55] introduce a protocol under LDP for identifying heavy hitters by discovering the popular prefixes with increasing lengths, named the prefix extending method (PEM), which is also designed for the single-party setting.

The existing studies that comply with LDP predominantly focus on a traditional local setting with a single server, disregarding the more practical scenario where users' data are distributed across multiple servers. The most relevant work [46] to ours employs a hierarchical approach to identify local and global heavy hitters in cross-party settings, but does not satisfy LDP. In this work, each user utilizes its own data and its prefix to select the random item X to add to the domain. It aims to address the out-of-domain issue when a user's prefix lies outside the predefined domain. However, it results in different output domains rely on each user's sensitive data, undermining the LDP guarantee. In response, our work introduces a federated heavy hitter analytics mechanism designed to detect heavy hitters across multiple parties while mitigating the negative impacts of non-IID data and satisfying ϵ-LDP.

3 Preliminaries

3.1 Local Differential Privacy

As the local variant of differential privacy (DP) [17], LDP [30] eliminates the need for a trusted data collector, making it a viable privacy model for numerous applications in distributed settings, such as frequency and mean estimation [52, 53], trajectory data synthesis and collection [16, 67], and preference ranking analysis [60]. The following provides a formal definition of LDP.

Definition 3.1 (ϵ-LDP). Given a privacy budget $\epsilon > 0$, a randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ provides ϵ-LDP if and only if, for any two inputs $x, x' \in \mathcal{X}$ and any possible output $y \in \mathcal{Y}$, the following inequality holds: $\Pr[\mathcal{M}(x) = y] \leq e^{\epsilon} \times \Pr[\mathcal{M}(x') = y]$.

The plausible deniability of LDP is achieved by perturbing users' data locally and is controlled by privacy budget ϵ. Similar to centralized settings, LDP enjoys the desired property of post-processing [18].

Theorem 3.2. *(Post-Processing) Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a mechanism that satisfies ϵ-LDP, and $\mathcal{F} : \mathcal{Y} \to \mathcal{Y}'$ be an arbitrary randomized mapping. Then $\mathcal{F} \circ \mathcal{M} : \mathcal{X} \to \mathcal{Y}'$ satisfies ϵ-LDP, where $\circ$ denotes the composition of $\mathcal{F}$ and $\mathcal{M}$.*

The post-processing property guarantees that no privacy leakage occurs when perturbed data undergo further processing.

3.2 Frequency Oracles

A *frequency oracle* (FO) is an LDP mechanism for frequency estimation. In this subsection, we briefly revisit three classic FOs, namely k-ary randomized response (k-RR) [29, 57], optimized unary encoding (OUE) [53], and optimized local hashing (OLH) [53].

***k*-RR.** Given a privacy budget ϵ, k-RR mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{X}$ satisfies ϵ-LDP, if, for any input $x \in \mathcal{X}$, the output $y \in \mathcal{X}$ is sampled from the following distribution:

$$\Pr[\mathcal{M}(x) = y] = \begin{cases} p = \frac{e^{\epsilon}}{|\mathcal{X}|-1+e^{\epsilon}} & \text{if } y = x, \\ q = \frac{1}{|\mathcal{X}|-1+e^{\epsilon}} & \text{otherwise.} \end{cases} \tag{1}$$

By Equation 1, k-RR consistently reports the original value (i.e., $y = x$) with a higher probability p. To estimate the frequency of $x \in \mathcal{X}$—that is, the proportion of users whose private value is x relative to the total user count—one counts the occurrences of x being reported, represented as c_x, and then computes $f_x^{k\text{-RR}} = \frac{c_x/n - q}{p - q}$, where n signifies the total number of users. The variance for this unbiased estimation [53] is $Var[f_x^{k\text{-RR}}] = \frac{|\mathcal{X}|-2+e^{\epsilon}}{(e^{\epsilon}-1)^2 \cdot n}$. It has been proven that k-RR achieves good performance for domain sizes k (i.e., $|\mathcal{X}|$) smaller than $3e^{\epsilon} + 2$ [53]. For a larger domain size, the state-of-the-art OUE [53] mechanism has been proven that can outperform k-RR.

OUE. Given a privacy budget ϵ, the OUE mechanism satisfies ϵ-LDP if it first encodes the input $x \in \mathcal{X}$ as a length-$|\mathcal{X}|$ one-hot binary vector $B = Encode(x) = [0, \cdots, 0, 1, 0, \cdots, 0] \in \mathcal{B}$, where only the x-th position is 1. It then perturbs each position of B, and the output $B' \in \mathcal{B}$ is sampled from the following distribution:

$$Pr[B'[x] = 1] = \begin{cases} p = \frac{1}{2} & \text{if } B[x] = 1, \\ q = \frac{1}{e^\epsilon + 1} & \text{if } B[x] = 0. \end{cases}$$

The perturbed vector is viewed as supporting an input x if $B[x] = 1$. Let c_x denote the counts of the supports of x among all the reported vectors, then frequency f_x^{OUE} of x is the same with k-RR. The variance of this estimation [53] is $Var[f_x^{\text{OUE}}] = \frac{4e^\epsilon}{(e^\epsilon - 1)^2 \cdot n}$. However, for significantly larger domain sizes, OUE can result in excessive communication costs. Therefore, OLH [53] is recommended.

OLH. Let $\mathbb{H}$ be a universal hash function family, with each $H \in \mathbb{H}$ outputting a value in $[d']$, where $d' = \lceil e^\epsilon + 1 \rceil$. Given a privacy budget ϵ, the OLH mechanism satisfies ϵ-LDP if it first encodes the input $x \in \mathcal{X}$ as $v = H(x)$, with H chosen uniformly at random. It then perturbs v, and the output $y \in [d']$ is sampled according to the following distribution:

$$\Pr[y = v'] = \begin{cases} p = \frac{e^\epsilon}{d' - 1 + e^\epsilon} & \text{if } v' = v, \\ q = \frac{1}{d' - 1 + e^\epsilon} & \text{otherwise.} \end{cases}$$

For each $x \in \mathcal{X}$, let $c_x = |\{u \mid H^u(x) = y^u\}|$ denote the number of reports supporting the input as x, where y^u represents the output of user u, perturbed by the randomly chosen hash function H^u. The frequency estimation of x and the variance of this estimation [53] are the same as the OUE mechanism. However, the computation cost of OLH is much larger, especially when the domain size is overly large [13, 55]. All the above three mechanisms can be used as an FO, each subject to different practical constraints. In addressing the heavy hitter problem, the FO is typically treated as a black box. Since the strategy for identifying heavy hitters being the primary challenge, which is also the most significant and impactful aspect.

3.3 Heavy Hitter Identification

Identifying heavy hitters under LDP in the local setting focuses on the most frequent items, especially when the item domain size is extremely large. Under such conditions, it is impractical to directly apply FOs (e.g., OUE), due to prohibitive communication and computation costs [13, 34, 55]. There are some existing works that consider the set-valued LDP setting (e.g., LDPMiner [41] and SVIM [54]), which also incur high communication costs. To reduce these costs while still achieving good utility, prefix-tree-based iterative algorithms are commonly employed [4, 51, 55]. For instance, PEM [55], the state-of-the-art heavy hitter identification mechanism under LDP, identifies heavy hitters by constructing a prefix tree iteratively. Concretely, PEM assigns the users into different groups to use an FO to report prefixes with different lengths of their encoded items. The server then sequentially processes each group's reports to determine which top t popular prefixes (where t refers to the extension number) should be extended in the next group for longer candidate prefixes. The heavy hitters are identified after the final group's estimations are completed.

Although the prefix-tree based algorithms are efficiently in identifying heavy hitters in local settings, the previous studies all consider a single-party setting, leaving a gap in the practical federated setting that the users are always distributed across multiple parties. In this setting, each party can only obtain the noisy answers from its groups, but cannot access the results from other parties.

4 Problem Statement and Challenges

4.1 Problem Statement

We consider a federated setting, where a central server wants to identify the heavy hitters in the data distributed across a set of parties $\mathcal{P} = \{P_1, \cdots, P_{|\mathcal{P}|}\}$. Each party $P_i \in \mathcal{P}$ comprises a distinct set of users U^i, without overlap with the other parties. Each user $u_j \in U^i$ holds an item $x_j \in \mathcal{X}$, where $\mathcal{X}$ is the item domain. Since the parties are not entirely trusted, users will locally perturb their private values via LDP mechanisms and only send noisy values to the parties. Upon receiving the sanitized data, each party computes and sends some partial results to the server for identifying the top-k federated heavy hitters, which are formally defined as follows.

Definition 4.1 (Top-k Federated Heavy Hitter). Given a set of parties $\mathcal{P}$ and the value domain $\mathcal{X}$, an item $x \in \mathcal{X}$ is a top-k federated heavy hitter if its frequency,

$$f_x = \frac{\sum_{P_i \in \mathcal{P}} |\{j | x_j = x, u_j \in U^i\}|}{\sum_{P_i \in \mathcal{P}} |U^i|},$$

is ranked within top k over frequencies of all possible values in $\mathcal{X}$ across all parties $\mathcal{P}$.

Intuitively, upon receiving perturbed data from the users, the party can directly send the local dataset to the central server for heavy hitter identification. However, this may produce the excessive communication cost. In practice, both the user population and the item domain are usually overly large, e.g., reaching the million level in recommendation scenarios [23, 35]. Assume we have $5,000,000$ users and $|\mathcal{X}| = 2,000,000$, the state-of-the-art FO, OUE [53], needs to encode an input value into a $2,000,000$-bit vector, then perturbs and submits it to the server. Thus, the communication cost in the server side is 1×10^{13} bits, which is unacceptable in many applications. One can use other communication efficient FOs, e.g., OLH, but communication costs are still large and decoding a single randomized data (via such mechanisms) on the server side requires a comprehensive scan of the entire domain $\mathcal{X}$, which is computationally infeasible for large data domains [13, 34, 55]. To mitigate the expensive communications and computations, the central server alternatively breaks down the task of federated heavy hitter identification into some sub-tasks and delegates them to each party instead of requiring them to upload the distributed data.

4.2 Straw-Man Solution and Challenges

In the literature, some existing works study the simplified scenario of federated heavy hitter identification under DP. A line of work assumes that the distributed parties are semi-trusted, so that they can collect users' raw data, inject DP noise into query results and then send the noisy version to the server [5, 27]. In the context of LDP, all the existing works assume each client only includes a single user, so that users can directly send the perturbed value to the central server [4, 51, 55].

In practice, users are always distributed among multiple untrusted parties. For instance, Amazon may require its branches in different regions (e.g., Europe and America) to collaboratively identify the top k items frequently purchased or liked during the Christmas holiday to develop more effective marketing plans. Under this circumstance, we alternatively turn to LDP. Specifically, users send the perturbed data to the party, which can then estimate some partial results (e.g., local heavy hitters [1]). Then parties send the partial results to the server for identifying federated heavy hitters.

This idea naturally leads to a straightforward solution, as described in Algorithm 1. Specifically, each party can employ existing approaches for heavy hitter identification (e.g., PEM or SPM [21]) to estimate the local heavy hitters, and then submit the identified top-k heavy hitters along with

[1]For ease of understanding, we refer to heavy hitters identified by a single party P_i as local heavy hitters.

Algorithm 1: PEM for Federated Heavy Hitters (FedPEM)

Input: All parties $\mathcal{P}$, value domain $\mathcal{X}$, maximum binary length m, granularity g, query k, and privacy budget ϵ
Output: Top-k federated heavy hitters R^k
// Party side:
Each party $P_i \in \mathcal{P}$ calls PEM(U^i, m, k, ε, g) (Algorithm 1 in [55]), where U^i denotes the set of users in the i-th party;
Each party P_i obtains the local top-k heavy hitters R_i^k and reports to server;
// Server side:
Initialize the federated heavy hitter dictionary $R^k = \{\}$;
Server aggregates and obtains the top-k federated heavy hitters R^k;
return R^k;

their estimated frequencies to the central server. The server then aggregates the local heavy hitters from different parties to determine the federated heavy hitters.

However, such a solution overlooks the non-IID problem, which poses challenges for accurate aggregation on the server side. Federated heavy hitter identification faces two significant issues related to non-IID data. First of all, data heterogeneity incurs varying frequency distributions across parties. For instance, the differing frequency distributions of values can lead to skewed frequency distributions of prefixes. However, existing mechanisms in LDP overlook this issue and consistently apply the same extension number of $t = k$ in each iteration when extending popular prefixes to a longer length. On the other hand, the non-IID nature of the data complicates the estimation and aggregation processes. The local heavy hitters identified within a party may not contribute to identifying the global [2] heavy hitters. For example, certain items may be frequent in Amazon but not popular, even not exist, in Shopee. Such items do not offer valuable information for federated heavy hitter identification and, to some extent, act as noise. This oversight can lead to false positives and the neglect of some significant popular prefixes. In the following sections, we will delve into these issues in more detail and introduce our mechanism to mitigate them.

5 Methodology

This section presents our proposed solution. We first introduce our design rationale in Section 5.1 and then overview our target-aligning prefix tree mechanism in Section 5.2. Then the implementation details are illustrated in Sections 5.3–5.5, followed by an analysis on privacy and utility in Section 5.6.

5.1 Design Rationale

The effectiveness of constructing a prefix tree (a.k.a., *trie*) for identifying heavy hitters has been verified in the typical LDP setting [51, 55]. Compared with other structures (e.g., multiple channels [4] and segment based structure [21]), the prefix tree encodes an item into a binary tree and thus utilizes the sequential feature for pruning. For example, given the item domain size 2^{64}, each item can be encoded into a 64-bit vector, based on which a 64-level binary trie can be constructed. The inherent sequential feature allows for the iterative pruning of unnecessary items by filtering prefixes (e.g., preserve only items whose first two-bit prefix is 00 or 10). This not only avoids the excessive communication cost but also reduces the scale of LDP noises by iteratively narrowing down the candidate domain. This inspires us to design an effective LDP-enabled prefix tree mechanism to accurately identify federated heavy hitters.

According to the Apriori property [2], the prefix of a popular value is likely to be popular as well. However, due to LDP noises and the non-IID nature of data in federated settings, some globally

[2]In the sequel, global and federated are interchangable.

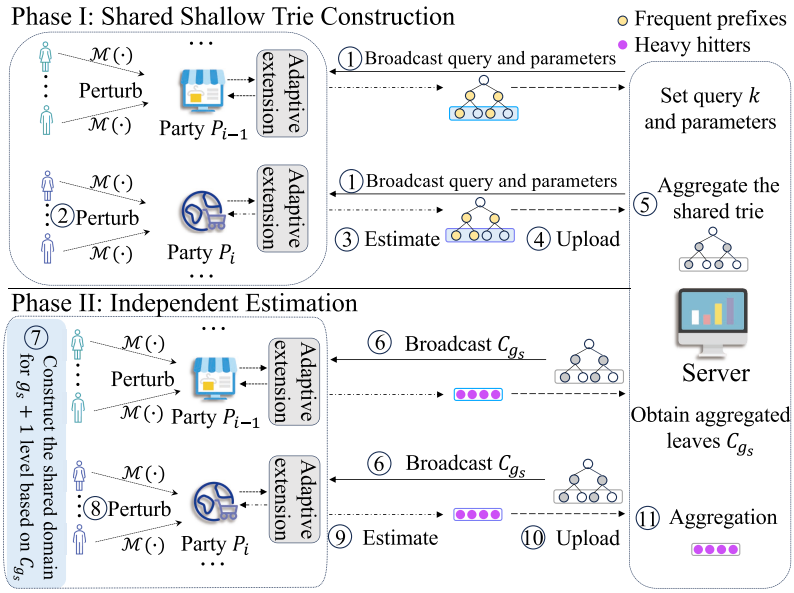


Fig. 1. An overview of the TAP mechanism.

frequent values might be overwhelmed by certain prefixes and pruned in local parties. This results in inconsistencies between local and global targets, especially at shallow levels. As such, we propose a strategy of constructing a shared shallow trie that aggregates frequent prefixes across parties at a shallow level, serves as a warm start, and aligns such inconsistencies collaboratively.

On the other hand, in federated settings, each party maintains its own trie by minimizing the local trie construction errors, which are inherently affected by both LDP noise and the underlying frequency distribution. Therefore, we propose an adaptive trie extension strategy. As opposed to existing studies that apply fixed extensions, we adaptively determine the extension sub-processes during trie construction. This approach not only reduces LDP noises but also mitigates the inconsistency between identifying local heavy hitters and global targets (i.e., covering necessary prefixes).

5.2 Mechanism Overview

The above design rationale leads to our target aligning prefix tree (TAP) mechanism. As shown in Figure 1, the TAP mechanism consists of two phases. In phase I, the server first broadcasts the query and parameters to all parties (step ①). Upon receiving them, all users in each party are divided into several groups uniformly at random based on the increasing prefix lengths. Each group corresponds to a level of the trie, and each user in that group reports a perturbed prefix of her value. In each party, some frequent prefixes estimated by the current group are then extended to construct the candidate domain for the next group, to estimate longer prefixes. Each user reports only once, which avoids dividing the privacy budget and thus achieving more accurate estimations. During phase I, the first few user groups in all parties collaboratively construct a shared shallow trie (steps ②–⑤), by aggregating the partial results from all parties, whose construction will be detailed in Section 5.3.

In phase II, the server first broadcasts the leaves of the constructed shared trie to each party (step ⑥), based on which the parties extend the same leaves to construct candidate domains (step ⑦). Then local heavy hitters in different parties can be identified (steps ⑧–⑩). Note that trie extensions in both phases are always adaptive to the noisy frequency distributions, which will

be further elaborated in Section 5.4. Finally, the server aggregates these findings to identify the federated heavy hitters (step ⑪).

5.3 Shared Shallow Trie Construction

As aforementioned, the non-IID problem may cause inconsistencies to the estimation among parties. Numerous infrequent values can share identical shorter prefixes, which might appear frequent at shallow levels. Due to LDP noises and the non-IID problem, values that are genuinely frequent across all parties might be overshadowed by these common prefixes and get prematurely pruned by local parties, especially at shallow levels. For instance, in Figure 2(a), there are two parties (i.e., party A and party B) that encode the entire domain into a trie (e.g., 2^{64} items represents by a 64-bit binary string) where the prefixes at the second level of the trie, namely the first two-bit prefix domain, are {00, 01, 10, 11}, and the query is for the top $k = 2$ heavy hitters. At the current level (shaded in blue), both parties select the yellow prefixes to be extended based on their own noisy observations. That is, party A extends the yellow prefixes 00, 01, and 11, whereas party B chooses 00 and 10. We can observe that at the current level of the trie, the globally frequent prefixes (in grey) are 00 and 10. Unfortunately, without intervention, party A will miss 10 (in pink) and further incur useless extensions at each deeper levels, which could introduce extra noise in identifying global heavy hitters. Therefore, we propose a strategy of constructing a shared shallow trie to mitigate this issue by aligning the local and global targets.

Algorithm 2 shows the workflow. The server assigns each party $P_i \in \mathcal{P}$ a granularity g and a shared trie level g_s (line 1). Each P_i instructs the first g_s groups of users to estimate the frequent prefixes at the initial g_s levels (lines 4–8). We apply adaptive extension here by considering the underlying frequency distribution and LDP noises, see Section 5.4. Then, each party sends the candidates with non-zero estimated counts at the g_s-th leves to the server (line 9). The server calculates the top k frequent prefixes (denoted as C_{g_s}) based on the reports (line 10). For instance, as shown in Figure 2(a), with $g_s = 2$ and $k = 2$, party A selects all prefixes as the candidates $C^A_{g_s}$ after completing the estimation at the second level (shown in blue), while $C^B_{g_s}$ consists of 00 and 10. They then report $C^A_{g_s}$, $C^B_{g_s}$, along with the counts of all prefixes to the server. The server aggregates and broadcasts the top $k = 2$ frequent prefixes, $C_{g_s} = \{00, 10\}$, to all parties. Then in phase II, after receiving these globally frequent prefixes C_{g_s}, each party constructs the prefix domain Λ_{g_s+1} by extending C_{g_s} and continues their estimations independently until they identify their own local heavy hitters.

5.4 Adaptive Trie Extension

In previous studies, the server extends the top $t = k$ (i.e., the extension number) frequent prefixes at each level to construct the candidate prefix domain for the next level. This extension number $t = k$ relies solely on the query k and is independent of the current frequency distribution. We argue that such fixed extensions, influenced by LDP noise and the inherent prefix frequency distribution, may not yield an appropriate perturbation domain for the next level. Concretely, the extension number t represents the number of child nodes to be extended, and the number of candidates grows exponentially with increasing t. Consequently, a large t results in a large candidate domain, including many unnecessary prefixes to be extended (i.e., those that are not prefixes of heavy hitters). When applying LDP perturbation, the noise scale increases with the domain size, leading to excessive LDP noise. Thus, the goal of adaptive trie extension is to strike a balance between the amount of important prefixes and the injected LDP noise. Specifically, for a given party and the candidate prefix domain Λ_h at level h, the party sorts the estimated frequencies $\hat{F}_h$ of all candidates after receiving the perturbed data from users, and then adaptively decides on the extension number

Algorithm 2: Shared Shallow Trie Construction (STC)

Input: All parties $\mathcal{P}$, maximum binary length m, granularity g, shared trie level g_s, query k, and privacy budget ϵ
Output: Globally frequent prefixes C_{g_s}

The server broadcasts granularity g and shared level g_s to all parties;
for $P_i \in \mathcal{P}$ **do**
  Initialize $C_0^i = \emptyset$, $h_0 = 0$;
  Divide its users U^i into g groups $\{U_1^i, ..., U_g^i\}$ randomly;
  for $h \in \{1, ..., g_s\}$ **do**
    Compute the prefix length of h-th level $l_h = \lceil h \times \frac{m}{g} \rceil$;
    Construct candidate domain Λ_h = `Construct` $(l_h, l_{h-1}, C_{h-1}^i)$;
    Get user reports to estimate $(C_h^i, \mathcal{I}_h^i)$ = `Estimate` $(l_h, \epsilon, \Lambda_h, U_h^i)$;
  Report prefix candidates $C_{g_s}^i$ and their counts $\mathcal{I}_h^i$ at g_s-th level to the server;
The server aggregates the top k frequent prefixes C_{g_s} from $\{C_{g_s}^1, ..., C_{g_s}^{|\mathcal{P}|}\}$ based on $\{\mathcal{I}_{g_s}^1, ..., \mathcal{I}_{g_s}^{|\mathcal{P}|}\}$;
return C_{g_s};
Procedure `Construct`(l_h, l_{h-1}, C_{h-1})**:**
  return $\Lambda_h = C_{h-1} \times \{0, 1\}^{l_h - l_{h-1}}$;
Procedure `Estimate`$(l_h, \epsilon, \Lambda_h, U_h)$**:**
  Each user in U_h reports her noisy prefix via the given FO and ϵ;
  Estimate frequencies of all prefixes $\lambda_h \in \Lambda_h$ with l_h bits length;
  Construct C_h as $t = k^* + \eta$ highest estimated values (with their counts $\mathcal{I}_h$) by solving Equations 2 and 3 (see Section 5.4);
  return $(C_h, \mathcal{I}_h)$;

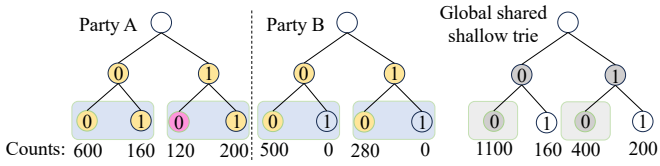
t. This decision considers two factors: a less frequent prefix among the top k serving as an anchor, and the amount of LDP noise. We first identify this anchor prefix as

$$\arg\max_{k^*} \frac{\sum_{1<j\le k^*} \hat{f}_j}{k^*} - \frac{\sum_{k^*<s\le k+1} \hat{f}_s}{k+1-k^*}, \tag{2}$$

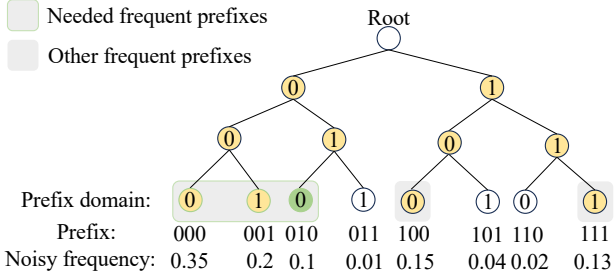
where $1 < k^* \le k$, $\hat{f}_j$ (resp. $\hat{f}_s$) denotes the noisy frequency of the j-th (resp. the s-th) popular prefix. In Equation 2, the first term calculates the average of the first k^* prefixes' frequencies (except for the largest one since it must be preserved), and the latter represents the average frequency of the other relatively infrequent prefixes within the top $k+1$ prefixes at current level. We include the $(k+1)$-th frequent prefix, which represents the upper bound of the average frequency of the rest prefixes and avoids dividing by zero. The underlying rationale for this objective function is that such an anchor prefix, along with any more frequent prefixes, will likely cover the least frequent prefix among the final top k heavy hitters at this level. The identification of k^* distinguishes the prefix with a significantly influential frequency, which serves as the boundary between the prefixes of top k at current level and others. However, the LDP noise must be took into account.

As our objective is to increase the cover rate of necessary prefixes at each level, it is natural to estimate how far the anchor prefix can drift under the LDP noise. Such an estimation, along with the anchor k^*, is considered an approximated worst case for the extension number that covers the most infrequent prefix among the needed prefixes. Since an estimated frequency can be approximated by a Gaussian distribution [55], let X_{k^*} and X_{k^*+x} denote the frequency variables of the k^*-th and (k^*+x)-th observed popular prefixes, respectively. We can calculate the drift distance η as follows:

$$\eta = \min(k, \mathbb{E}(x)), \tag{3}$$



(a) An instance of the shared shallow trie construction.



(b) An example of the adaptive trie extension.

Fig. 2. Toy examples of the TAP mechanism.

where the $\mathbb{E}(x)$ is the expectation of x, derived as

$$\mathbb{E}(x) = \sum_{x=k^*-k+1}^{\min(k,\pi_p^i-k)} x \cdot \Pr(X_{k^*} \le X_{k^*+x}),$$
$$\Pr(X_{k^*} \le X_{k^*+x}) = \Pr(X_{k^*} - X_{k^*+x} \le 0)$$
$$\approx \frac{1}{\sqrt{4\pi}} \int_{-\infty}^{0} \frac{1}{\sigma} \exp\left(-\frac{(t-(\hat{f}_{k^*}-\hat{f}_{k^*+x}))^2}{4\sigma^2}\right) dt,$$

where $\hat{f}_{k^*}$ and $\hat{f}_{k^*+x}$ are the observed noisy k^*-th and (k^*+x)-th frequencies and σ denotes the standard deviation of the FO used for frequency estimation. In the above equations, we approximately calculate the expectation that the k^*-th prefix drifts to the $(k^*+\eta)$-th position under LDP noise. Specifically, we approximate the means of the frequency variables X_{k^*} and X_{k^*+x} using the noisy k^*-th and (k^*+x)-th frequencies, setting the standard deviation according to the specific FO. Then we simulate the perturbation and calculate the probability that $X_{k^*} \le X_{k^*+x}$. Following this, we derive the expectation and set the drift distance to η accordingly. Note that we bound η that cannot exceed k since the drift distance should not be too large. Ultimately, we determine the extension number as $t = k^* + \eta$. Thus, the current party can extend these top t frequent prefixes to a longer length for further estimation. For example in Figure 2(b), consider the h-level prefix domain as $\{000, 001, 010, 011, 100, 101, 110, 111\}$ and the query $k = 4$. The necessary prefixes, namely, those of the top k heavy hitters at the h-th level, are 000, 001, and 010. After sorting the frequencies, we observe that the most frequent $k = 4$ prefixes are 000, 001, 100, and 111 in yellow, excluding the needed prefix 010, shown in green, which is ranked below 4 due to the LDP noise. In contrast to using a fixed extension number $t = k = 4$, which would overlook the necessary prefix, our adaptive strategy sets $t = k^* + \eta = 5$, thus encompassing the required ones.

Algorithm 3: Target-Aligning Prefix Tree Mechanism (TAP)

Input: Parties $\mathcal{P}$, maximum binary length m, granularity g, shared trie level g_s, query k, and privacy budget ϵ
Output: Top-k federated heavy hitters R^k

// Phase I: shared shallow trie construction
1 $C_{g_s} = STC(\mathcal{P}, m, g, g_s, k, \epsilon)$;
// Phase II: independent estimations with a warm start
2 **for** $P_i \in \mathcal{P}$ **do**
3 Based on the remaining $g - g_s$ groups, P_i constructs candidate domain in each level $h \in \{g_s + 1, ..., g\}$ for estimatation, i.e., $\Lambda_h^i = \texttt{Construct}(l_h, l_{h-1}, C_{h-1}^i)$,
4 $(C_g^i, \mathcal{I}_g^i) = \texttt{Estimate}(l_h, \epsilon, \Lambda_h^i, U_h^i)$
5 P_i uploads local heavy hitters C_g^i and counts $\mathcal{I}_g^i$ to server;
6 The server derives the top-k federated heavy hitters R^k from $\{C_g^1, ..., C_g^{|\mathcal{P}|}\}$ based on $\{\mathcal{I}_g^1, ..., \mathcal{I}_g^{|\mathcal{P}|}\}$;
7 **return** R^k;

5.5 Putting Things Together

We now summarize the entire pipeline of the proposed TAP mechanism. As shown in Algorithm 3, each party constructs the shared shallow trie to obtain the global frequent prefixes at the g_s-th level (line 1), as described in Algorithm 2. Then, each party continues the estimation process based on C_{g_s} (line 2). They extend these prefixes and use the corresponding users from each group to estimate the longer popular prefixes, repeating this procedure until they ultimately report the local heavy hitters at the g-th level to the server (lines 3-5). Note that the extensions are always adaptive. The server then derives the federated heavy hitters by counting (line 6).

The TAP mechanism utilizes the adaptive trie extension and the shared shallow trie construction to enhance the consistency of the local and global targets, enabling the identified local heavy hitters to provide much more valuable and reasonable statistical information for mining global heavy hitters across parties. The former can be applied within each party and relies on the current frequency distribution. The latter provides a warm start for each party by integrating global constraints extracted from all parties. These constraints can help participants collect more useful prefixes by disregarding certain prefixes that may appear frequent at initial levels within specific parties but are infrequent on a global view.

5.6 Privacy and Utility Analysis

We first prove that the proposed TAP mechanism satisfies ϵ-LDP.

Theorem 5.1. *The TAP mechanism satisfies ϵ-LDP if it uses an FO that satisfies ϵ-LDP.*

Proof. In the TAP mechanism, each party randomly divides its users into g groups, which does not raise privacy concerns. Each user in a group receives a candidate domain for perturbation and, given the privacy budget ϵ , perturbs their value via an FO. We first prove that the perturbation within a party satisfies ϵ-LDP, and then show that candidate domain construction does not consume any additional privacy budget.

For each user in the h-th group of a party, the report includes a level index h, which is decided independent of her private value, and the noisy prefix λ_h perturbed by an FO, denoted by $\mathcal{M}(\cdot)$, with privacy budget ϵ. For any two inputs $x_1, x_2 \in \mathcal{X}$, and for any specific tuple $\langle h, y\rangle \in \text{Range(TAP)}$, we have: $\max_{x_1,x_2,\langle h,y\rangle} \frac{\Pr[\text{TAP}(x_1)=\langle h,y\rangle]}{\Pr[\text{TAP}(x_2)=\langle h,y\rangle]} = \max_{x_1,x_2,\langle h,y\rangle} \frac{\Pr[\mathcal{M}(\lambda_h^1)=y]\cdot\Pr[h]}{\Pr[\mathcal{M}(\lambda_h^2)=y]\cdot\Pr[h]} = \max_{x_1,x_2,\langle h,y\rangle} \frac{\Pr[\mathcal{M}(\lambda_h^1)=y]}{\Pr[\mathcal{M}(\lambda_h^2)=y]} \leq e^\epsilon$, where λ_h^i denotes the prefix of x_i at the h-th level. $Pr[h] = 1/g$ is the probability that a user falls into the h-th group, which is chosen randomly and independent of the private value. The last inequality holds because the FO $\mathcal{M}(\cdot)$ used for perturbation satisfies ϵ-LDP.

In Phase I, the candidate domain is constructed independently at the first level to ensure privacy. For levels $1 \leq h \leq g_s$, domains are derived by extending the top t frequent prefixes from the $(h-1)$-th level, where g_s is determined independently. Since frequencies are perturbed with an FO satisfying ϵ-LDP, the resulting frequency distribution benefits from the post-processing theorem. Other parameters in Equations 2 and 3 (e.g., user count and FO variance) are non-private, so determining t also preserves LDP. Consequently, candidate domain construction at subsequent levels in Phase I also benefits from the post-processing theorem. In Phase II, the (g_s+1)-th level domain is extended based on sanitized global results at the g_s level. The privacy guarantee for levels $g_s+1 \leq h \leq g$ is consistent with that of Phase I. Thus, we complete the proof. □

We also provide a utility analysis on the adaptive extension. The potential failure of our adaptive extension would occur if t is consistently set to the same constant at all levels. Therefore, we analyze the probability under the worst case of when the adaptive extension consistently selects a constant across all levels.

Theorem 5.2. *Given the query k, the upper bound of the probability of event A that the adaptive extension strategy always sets the extension number t at g iterations to the same constant c is as follows:*

$$Pr[A] \leq (P_x)^g,$$

where $P_x = Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{3k+1}]$, $\phi(\cdot)$ denotes the CDF of the Gaussian distribution, δ_f denotes the maximum difference between two neighboring frequencies of the last $2k-k^*$ prefixes/items out of the frequent $2k$, and σ is the variance of the FO.

Proof. For the extension number t at each iteration, it is set to the same constant c when the selected $k^*+\eta = c$ (we assume $k^* \leq c$ has been determined since it highly depends on the distribution), that is, $Pr(k^*) \cdot Pr[min(k, \mathbb{E}(x)) = c - k^*]$. When $\mathbb{E}(x) \leq k$:

$$\begin{aligned}
&\Pr(k^* = c_1) \cdot \Pr[min(k, \mathbb{E}(x)) = c - c_1] \\
&= \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) \leq k] \cdot \Pr[\mathbb{E}(x) = c - c_1] \\
&= \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) \leq k] \cdot \Pr[\frac{(c-c_1)\cdot(1+c_1+c)}{2\sqrt{\pi}} \cdot \phi(\frac{-\delta_f}{2\sigma}) = c - c_1] \\
&= \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) \leq k] \cdot \Pr[\frac{(1+c_1+c)}{2\sqrt{\pi}} \cdot \phi(\frac{-\delta_f}{2\sigma}) = 1] \\
&= \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) \leq k] \cdot \Pr[\phi(\frac{-\delta_f}{2\sigma}) = \frac{2\sqrt{\pi}}{1+k^*+c}] \\
&\leq \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) \leq k] \cdot Pr[\phi(\frac{-\delta_f}{2\sigma}) \geq \frac{2\sqrt{\pi}}{1+k+c}] \\
&\leq Pr[\phi(\frac{-\delta_f}{2\sigma}) \geq \frac{2\sqrt{\pi}}{1+k+c}] \leq \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{1+3k}];
\end{aligned}$$

when $\mathbb{E}(x) > k$:

$$\begin{aligned}
&\Pr(k^* = c_1) \cdot \Pr[min(k, \mathbb{E}(x)) = k] = \Pr(k^* = c_1) \cdot \Pr[\mathbb{E}(x) > k] \\
&= \Pr(k^* = c_1) \cdot \Pr[\frac{(c-c_1)\cdot(1+c_1+c)}{2\sqrt{\pi}} \cdot \phi(\frac{-\delta_f}{2\sigma}) > k] \\
&= \Pr(k^* = c_1) \cdot \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2k\sqrt{\pi}}{(c-c_1)\cdot(1+c_1+c)}] \\
&\leq \Pr(k^* = c_1) \cdot \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2k\sqrt{\pi}}{k\cdot(1+k+c)}] \\
&= \Pr(k^* = c_1) \cdot \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{1+k+c}] \\
&\leq \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{1+k+c}] \leq \Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{1+3k}].
\end{aligned}$$

Since we have g iterations in total, we can obtain $Pr[t \geq c] \leq (P_x)^g$, where $P_x = Pr[\phi(\frac{-\delta_f}{2\sigma}) > \frac{2\sqrt{\pi}}{3k+1}]$. Thus, we complete the proof. □

We observe that when the query k, the maximum difference between two neighboring frequencies among the least frequent $2k - k^*$ items in the top $2k$, and the variance of the used FO are small, the probability of always setting t to a same constant at different iterations is very small.

6 Consensus-Pruned Target-Aligning Prefix Tree Mechanism

Although TAP mitigates the adverse effects of data heterogeneity and inconsistency issues via shared shallow trie construction and adaptive trie extension strategies, the non-IID nature of the data still complicates the estimation and aggregation processes. In phase II, each party independently continues to identify heavy hitters. Unfortunately, the local heavy hitters identified within a party may not contribute to identifying the global ones, which also incurs inconsistencies between local and global objectives. To this end, in this section, we propose the target-aligning prefix tree mechanism with the consensus-based pruning strategy (TAPS).

6.1 Sequential Estimation

The TAPS mechanism integrates TAP and a novel consensus-based pruning strategy that further enhances the target alignment across parties. This mechanism also consists of two phases. Similar to TAP, after phase I, the server first broadcasts the global top k frequent prefixes via the shared shallow trie (steps 1–6 in Figure 1). Then, in phase II, instead of identifying local heavy hitters independently, we propose our consensus-based pruning strategy. The motivation for this pruning strategy is that although TAP utilizes both global aggregated constraints and local frequency information in phase I to mitigate the inconsistencies, estimations in phase II do not consider the impact of data distributions from other parties.

Intuitively, such prior knowledge can assist the current party in filtering out some infrequent prefixes before its estimation, which reduces the domain size to reduce the LDP noises, thus leading to more accurate estimations. However, it is infeasible for a party's pruning procedure to receive assistance from all other parties simultaneously, as such information would necessitate completing the entire estimation or incur prohibitive communication costs. Therefore, we optimize the workflow in phase II of TAP to a sequential order, which circumvents this issue and mitigates the accumulation of errors. Since federated heavy hitter analytics represents a variant of the frequency estimation problem, we believe that the party with a larger user base is likely to contribute more significantly to pruning. Therefore, we sort the parties in descending order based on user population sizes. We use $\mathcal{P}^R$ to denote the sorted one, and each party sequentially performs the estimation in this order.

Assume the entire candidate domain of current level consists of all leaves in Figure 3. Each of the i-th party P_i initially receives some candidates from the server (sent by P_{i-1}, which has a larger user population) that may not need to be pruned entirely in the current party (step 6). Then, P_i employs a consensus-based test to validate these candidates and prunes the necessary ones (steps 7–8). Subsequently, it proceeds with the heavy hitter identification process within the pruned domain and collects pruning candidates of this level for the next party P_{i+1}. For the first party with the most users but without previous input, it independently performs the estimation. Upon completion of the identification process by P_i, it forwards the local heavy hitters and candidates for pruning to the server. The server then sends these candidates to P_{i+1}, initiating a repeat of this process. Note that adaptive trie extension is always employed during the frequency estimation by each party and level. Once the server obtains the local heavy hitters from the last party, it aggregates these findings to determine the federated heavy hitters.

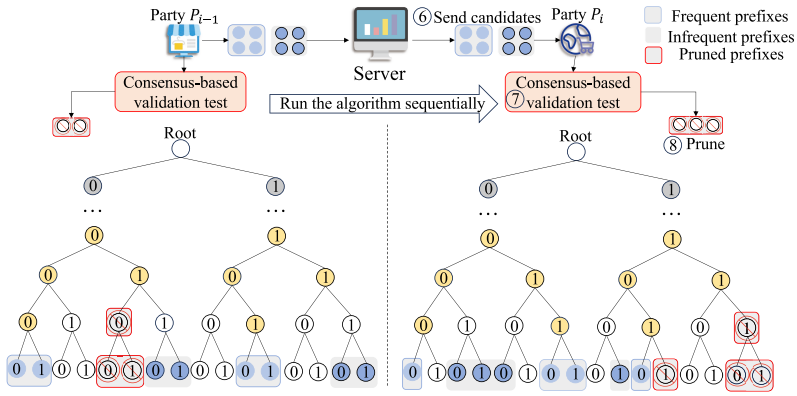


Fig. 3. Consensus-based Pruning Strategy.

6.2 Consensus-based Pruning Strategy

Due to the non-IID issue, the infrequent prefixes which can be pruned include two types. The first type refers to the globally infrequent items, which cannot be the targets. The second type includes items that are extremely infrequent in the current party (although it may be frequent in other parties), which contributes little to frequency estimation but increases LDP noise scale.

For the $(i-1)$-th party P_{i-1}, it needs to submit candidates of two types at the h-th level [3]. It first completes the estimation (based on the pruned domain) and obtains the noisy prefixes C_h^{i-1} (sorted by the descending frequencies) along with their estimated frequencies. Then it selects the pruning candidates as follows:

$$\begin{aligned} \Delta_h^{i-1} &= \{\Delta_{h,0}^{i-1}, \Delta_{h,1}^{i-1}\}, \\ \Delta_{h,0}^{i-1} &= \{\lambda^j | Rank(f_j) = |C_h^{i-1}| - j, \lambda^j \in C_h^{i-1}, 1 \le j \le 2k\}, \\ \Delta_{h,1}^{i-1} &= \{(\lambda^j, f_j) | Rank(f_j) = j, \lambda^j \in C_h^{i-1}, 1 \le j \le 2k\}, \end{aligned} \tag{4}$$

where $\Delta_{h,0}^{i-1}$ and $\Delta_{h,1}^{i-1}$ represent the infrequent and frequent candidate sets of P_{i-1} at h-th level, f_j denotes the frequency of the prefix λ^j, and $Rank(f_j) = j$ means f_j is j-th largest in C_h^{i-1}. Inspired by [54], we set $|\Delta_{h,0}^{i-1}|$ and $|\Delta_{h,1}^{i-1}|$ to $2k$. Because the goal here is to provide the next party with both exceptionally frequent and infrequent pruning candidates (specifically, the first k out of $2k$), thereby necessitating the inclusion of weaker candidate prefixes (the remaining ones in each $2k$) to introduce some randomness for the consensus-based validation test in P_i. Note that $|C_h^{i-1}| \succ 2k$ if t is not too small, since $|C_h^{i-1}| = t \cdot 2^{l_h - l_{h-1}}$, where $l_h = \lceil \frac{h \times m}{g} \rceil$. Then Δ_h^{i-1} is sent to the server, who forwards it to P_i.

For P_i, the consensus-based validation test is invoked to filter the candidates for pruning. The received Δ_h^{i-1} is regarded as the pruning predictions from P_{i-1}, which were estimated using most of its users U_h^{i-1} at level h that can be treated as the "training set". Thus, P_i validates Δ_h^{i-1} using a small fraction of its users U_h^i at level h, acting as the validation set for the consensus-based validation test. It then prunes the prefix domain for the 'test set' (i.e., the remaining majority of users in U_h^i). Concretely, as shown in Algorithm 4, P_i requires each user in $\hat{U}_{h,0}^i$ (resp. $\hat{U}_{h,1}^i$) to report her noisy prefix via an FO, then estimates the frequency of all prefixes in $\Delta_{h,0}^{i-1}$ (resp. $\Delta_{h,1}^{i-1}$) with the l_h bits

[3]Here we take the simplest example that only applies this pruning at the h-th level for ease of understanding.

Algorithm 4: Target-Aligning Prefix Tree Mechanism with the Consensus-based Pruning Strategy (TAPS)

Input: Parties $\mathcal{P}$, maximum binary length m, granularity g, shared level g_s, dividing ratio β, query k, and privacy budget ϵ
Output: Top-k federated heavy hitters R^k
// Phase I: shared shallow trie construction
1 $C_{g_s} = STC(\mathcal{P}, m, g, g_s, k, \epsilon)$;
// Phase II: continue the construction with pruning
2 Sort the parties in a descending order as $\mathcal{P}^R$;
3 **for** $P_i \in \mathcal{P}^R$ **do**
4 P_i initiates the pruning dictionary $D_i = \{\}$ and $C^i_{g_s} = C_{g_s}$;
5 **for** $g_s + 1 \le h \le g$ **do**
6 P_i initiates $\tilde{U}^i_h = U^i_h$ and $\hat{\Lambda}^{i-1,i}_h = \emptyset$;
7 **if** $g - g_s \le h \le g$ *or* $g_s + 1 \le h \le 2g_s$ **then**
8 **if** $i > 1$ **then**
9 P_i divides the users at h-th level into $\hat{U}^i_{h,0} = \hat{U}^i_{h,1} = \beta \cdot U^i_h$, and $\tilde{U}^i_h = U^i_h - \hat{U}^i_{h,0} - \hat{U}^i_{h,1}$;
10 P_i asks users in $\hat{U}^i_{h,0}$ and $\hat{U}^i_{h,1}$ to estimate the frequencies of pruning candidates in $D_{i-1}[h]$;
11 P_i obtains the pruning set $\hat{\Lambda}^{i-1,i}_h = \hat{\Lambda}^{i-1,i}_{h,0} \cup \hat{\Lambda}^{i-1,i}_{h,1}$ by Equations 5–8;
12 Λ^i_h=`Construct`$(l_h, l_{h-1}, C^i_{h-1})$;
13 $(C^i_h, \mathcal{I}^i_h)$ =`Estimate`$(l_h, \epsilon, h, \Lambda^i_h - \hat{\Lambda}^{i-1,i}_h, \tilde{U}^i_h)$;
14 **if** $i < |\mathcal{P}^R|$ **then**
15 Select $D_i[h] = \Delta^i_h$, where Δ^i_h is from Equation 4;
16 **else**
17 Λ^i_h=`Construct`$(l_h, l_{h-1}, C^i_{h-1})$;
18 $(C^i_h, \mathcal{I}^i_h)$ =`Estimate`$(l_h, \epsilon, h, \Lambda^i_h, \tilde{U}^i_h)$;
19 **if** $D_i.keys \neq \emptyset$ **then**
20 P_i uploads D_i to the server;
21 **if** $i < |\mathcal{P}^R|$ **then**
22 The server sends D_i to the next party;
23 P_i uploads local heavy hitters C^i_g and counts $\mathcal{I}^i_g$ to server;
24 The server derives the top-k federated heavy hitters R^k from $\{C^1_g, ..., C^{|\mathcal{P}|}_g\}$ based on $\{\mathcal{I}^1_g, ..., \mathcal{I}^{|\mathcal{P}|}_g\}$;
25 **return** R^k;

length (lines 9–10). Here, $\hat{U}^i_{h,0} = \hat{U}^i_{h,1} = \beta \cdot U^i_h$ represents the validation users of P_i for $\Delta^{i-1}_{h,0}$ and $\Delta^{i-1}_{h,1}$, respectively, where β is the dividing ratio. P_i then sorts the results by frequencies in ascending order to obtain $\hat{\Delta}^{i-1,i}_{h,0}$ and $\hat{\Delta}^{i-1,i}_{h,1}$.

Although P_i possesses the validated results, it is non-trivial to independently determine which prefixes should be pruned, since the entire frequency distribution at the h-th level is unknown. This uncertainty makes it difficult to discern the boundary between infrequent and frequent prefixes. While one could establish a fine-tuned threshold for pruning, the frequencies may be subject to instability due to the sampling errors from grouping users and the LDP noise. Moreover, the threshold might vary based on the underlying distributions, which differ across parties. Consequently, we rely on the rankings in both P_{i-1} and P_i, rather than using the frequencies directly, to adaptively decide the consensus pruning set, which mitigates the aforementioned issues. For the first type of infrequent prefixes (those are globally infrequent), we can filter the pruning prefixes as follows:

$$\arg\max_{k'} \quad \frac{|\hat{\Lambda}_{h,0}^{i-1,i}|}{k' \cdot (1+\epsilon)^{k'}} - \gamma \cdot \alpha^2, \tag{5}$$

$$\hat{\Lambda}_{h,0}^{i-1,i} = \Delta_{h,0}^{i-1}[:k'] \cap \hat{\Delta}_{h,0}^{i-1,i}[:k'], \tag{6}$$

where $1 \le k' \le k$, $\gamma = \left(1 - \frac{|U^{i-1}|}{\sum_{j=1}^{|\mathcal{P}|}|U^j|}\right)^2$, and $\alpha = \frac{k' - |\hat{\Lambda}_{h,0}^{i-1,i}| + 1}{k'+1}$. In Equation 5, the formula comprises two components: the intersection score and the penalty term. The intersection score, represented by $|\hat{\Lambda}_{h,0}^{i-1,i}|/k'$, indicates the proportion of the first common k' prefixes in $\Delta_{h,0}^{i-1}$ and $\hat{\Delta}_{h,0}^{i-1,i}$. To minimize the inclusion of too many false positives, this score is adjusted by dividing by $(1+\epsilon)^{k'}$, thus to refrain from always obtaining a large k' and provide the non-linear property. Because when ϵ is relatively small, which might result in inaccurate $\Delta_{h,0}^{i-1}$ and $\hat{\Delta}_{h,0}^{i-1,i}$ estimations, k' should not be too large. Regarding the penalty term, it consists of two factors. γ denotes the population confidence from the previous party, calculated according to the users' population. The more users in the P_{i-1}, the smaller γ value. The second factor, α, accounts for the non-intersection ratio within the first k' prefixes of $\Delta_{h,0}^{i-1}$ and $\hat{\Delta}_{h,0}^{i-1,i}$. Through the arg max operation, the chosen k' serves as an optimal boundary, thereby maximizing the consensus confidence of the infrequent prefixes ranked in first k' prefixes. Once the k' is determined, the intersection $\hat{\Lambda}_{h,0}^{i-1,i}$ becomes the consensus pruning prefix set for the first type. This set includes the infrequent prefixes selected by both P_{i-1} and P_i, achieving an optimal trade-off between consensuses and differences in $\hat{\Lambda}_{h,0}^{i-1,i}$, as defined by Equation 5.

For the second type of infrequent prefixes, i.e., extremely infrequent or even absent in the current party but may be frequent in other parties, we introduce the frequency contrast scores as the measurement:

$$\hat{\Delta}_{h,c}^{i-1,i} = \{(\lambda^j, f_j^c) | Rank(f_j^c) = j, f_j^c = \varphi(\lambda_j), \lambda^j \in \Delta_{h,1}^{i-1}, 1 \le j \le 2k\}, \tag{7}$$

where $\varphi(\lambda_j) = \frac{\Delta_{h,1}^{i-1}(\lambda_j)}{\hat{\Delta}_{h,1}^{i-1,i}(\lambda_j)+\tau}$, and $\tau = 1 \times 10^{-11}$ is to avoid dividing by zero, $\Delta_{h,1}^{i-1}(\lambda_j)$ (resp. $\hat{\Delta}_{h,1}^{i-1,i}(\lambda_j)$) denotes the frequency of the prefix λ_j in $\Delta_{h,1}^{i-1}$ (resp. $\hat{\Delta}_{h,1}^{i-1,i}$). In Equation 7, we initially calculate the frequency contrast score for each $\lambda_j \in \Delta_{h,1}^{i-1}$. This score quantifies the frequency discrepancy of λ_j between P_i and P_{i-1}. A higher f_j^c suggests that the prefix λ_j is more prevalent in P_{i-1} but significantly less so in P_i. For instance, if the best-selling item is only sold by the smallest party with the fewest users, e.g., the frequency of item X is 0.001 in the previous party and 0.7 in the current party (i.e., most sold), this score would be 0.001, which is relatively low. Conversely, if item Y's frequency is 0.7 in the previous party and 0.001 in the current party (i.e., almost nonexistent), the score would be 700, making it much more easier to identify. Then we arrange $\hat{\Delta}_{h,c}^{i-1,i}$ in descending order and feed it with $\hat{\Delta}_{h,1}^{i-1,i}$ into Equation 6:

$$\hat{\Lambda}_{h,1}^{i-1,i} = \hat{\Delta}_{h,c}^{i-1,i}[:k'] \cap \hat{\Delta}_{h,1}^{i-1,i}[:k']. \tag{8}$$

By substituting $\hat{\Lambda}_{h,0}^{i-1,i}$ with $\hat{\Lambda}_{h,1}^{i-1,i}$, we solve Equation 5 to derive the pruning set of infrequent prefixes $\hat{\Lambda}_h^{i-1,i} = \hat{\Lambda}_{h,0}^{i-1,i} \cup \hat{\Lambda}_{h,1}^{i-1,i}$. Note that because of descending order, the items with lower scores should not be selected. The current party then prunes the candidate domain, performs the estimation, and uploads its pruning candidates to the server for the pruning in the next party at this level as well (lines 11–15 in Algorithm 4).

We finally analyze the communication and computation costs of TAPS mechanism in comparison with two baselines (e.g., GTF and FedPEM) and the (infeasible) method of directly uploading

Table 1. Communication and computation costs.

	GTF	FedPEM	OUE	OLH	TAPS
Communication	$O(b \cdot k \cdot \lvert\mathcal{P}\rvert)$	$O(b \cdot k \cdot \lvert\mathcal{P}\rvert)$	$O(\lvert\mathcal{U}\rvert \cdot \lvert\mathcal{X}\rvert)$	$O(b \cdot \lvert\mathcal{U}\rvert)$	$O(b \cdot k \cdot \lvert\mathcal{P}\rvert \cdot g^*)$
Computation	$O(k \cdot \lvert\mathcal{P}\rvert)$	$O(k \cdot \lvert\mathcal{P}\rvert)$	$O(\lvert\mathcal{U}\rvert \cdot \lvert\mathcal{X}\rvert)$	$O(\lvert\mathcal{U}\rvert \cdot \lvert\mathcal{X}\rvert)$	$O(k \cdot \lvert\mathcal{P}\rvert)$

all noisy answers perturbed by an FO (i.e., OUE or OLH), to the central server. We assume that each pair of a prefix/item and its count consumes b bits, g^* represents the number of iterations where the pruning strategy is applied, $|\mathcal{U}|$ is the user population size, and $|\mathcal{X}|$ is the overall domain size. As shown in Table 1, the baselines incur a communication cost of $O(b \cdot k \cdot |\mathcal{P}|)$ and a computation cost of $O(k \cdot |\mathcal{P}|)$, as each party uploads only the top k local heavy hitters and the server only needs to count and sort them. The method of uploading user data (i.e., via OUE or OLH) for estimating heavy hitters is impractical due to the large user population and item domain, resulting in prohibitive communication costs increase linearly with $|\mathcal{U}|$ (for OUE), and excessive computation costs $O(|\mathcal{U}| \cdot |\mathcal{X}|)$ (for both OUE and OLH) [13, 55]. As a comparison, our mechanism TAPS incurs significantly less communication and computation costs. On the other hand, although TAPS still consumes a bit more communication costs [4]than two baselines GTF and FedPEM, its utility performance is significantly better than the baselines, as will be shown in Section 7. This can be attributed to our design of mitigating non-IID issues in federated settings.

6.3 Privacy Analysis

Finally, we establish the privacy guarantee of TAPS mechanism.

Theorem 6.1. *The TAPS mechanism satisfies ϵ-LDP if it uses an FO that satisfies ϵ-LDP.*

Proof. In the TAPS mechanism, the privacy guarantee during phase I remains consistent with that of TAP. In phase II, each party further splits each group of users at the first and last g_s levels into two set, namely, $\beta \cdot U_h$ for the consensus pruning test and $U_h - \beta \cdot U_h$ for the main estimation, where β is the independently dividing ratio. Each user within the respective set perturbs her value/prefix using an FO that satisfies ϵ-LDP, based on a predefined candidate domain. Similar to TAP, all candidate domains are established based on the sanitized results, thereby leveraging the post-processing property. Therefore, the TAPS mechanism satisfies ϵ-LDP. □

7 Performance Evaluations

7.1 Experimental Setting

Datasets. In the experiments, we use four real-world datasets (RDB, YCM, TYS, UBA) and one synthetic dataset (SYN). Each real-world dataset simulates heterogeneous parties across different application scenarios, e.g., RDB is designed to facilitate comparisons with baselines and simulate a typical heavy-hitter identification scenario that identifying the most frequent "out-of-vocabulary" words typed on keyboards [7]. RDB involves two parties — comments from Reddit [31] and movie reviews from IMDB [36]. YCM comprises four distinct datasets including Yahoo [66], CNN Dailymail [25, 45], Mind [58], and SWAG [64], which feature a variety of Q&A pairs, news articles, and video captions. TYS incorporates six parties, namely Twitter [24], Yelp [66], Scientific Papers [11], Amazon Arts [38], SQuAD [42, 43], and AG News [66], representing diverse text descriptions such as tweets, reviews, academic papers, Q&A, and news articles. UBA is the user behaviour dataset from Alibaba [39, 40, 68], which is one of the largest recommender system datasets from real industrial scenes. Each record in UBA represents a specific and private user-item interaction. We randomly

[4]The number of iterations g^* usually selects a small value, e.g., $g/2$.

Table 2. Datasets.

	Dataset description	# users	# total users	# unique items	# common items
RDB	Reddit [31] (Comments)	252,830	352,830	30,550	8,047
	IMDB [36] (Movie reviews)	100,000		15,470	
YCM	Yahoo [66] (Q&A)	812,300	1,336,048	79,971	3,879
	CNN Dailymail [25, 45] (News articles)	287,113		32,162	
	Mind [58] (Microsoft news)	123,082		17,309	
	SWAG [64] (Video captions)	113,553		7,656	
TYS	Twitter [24] (Tweets)	658,549	2,119,776	80,126	2,175
	Yelp [66] (Reviews)	649,917		34,866	
	Scientific Papers [11] (Papers)	349,119		27,372	
	Amazon Arts [38] (Reviews)	200,000		8,914	
	SQuAD [42, 43] (Q&A)	142,192		19,895	
	AG News [66] (News articles)	119,999		15,879	
UBA	UBA 0 (shopping interactions)	1,476,546	6,483,370	162,833	975
	UBA 1 (shopping interactions)	1,263,768		167,196	
	UBA 2 (shopping interactions)	1,246,972		167,309	
	UBA 3 (shopping interactions)	1,117,376		58,087	
	UBA 4 (shopping interactions)	774,626		9,203	
	UBA 5 (shopping interactions)	604,082		4,979	
SYN	SYN 0, Poisson ($\lambda = 10$)	220,000	780,000	5,484	276
	SYN 1, Poisson ($\lambda = 8$)	170,000		6,260	
	SYN 2, Zipf ($\alpha = 1.1$)	120,000		6,688	
	SYN 3, Zipf ($\alpha = 1.3$)	80,000		6,796	
	SYN 4, Poisson ($\lambda = 6$)	70,000		4,590	
	SYN 5, Poisson ($\lambda = 4$)	60,000		4,429	
	SYN 6, Zipf ($\alpha = 1.5$)	30,000		4,813	
	SYN 7, Zipf ($\alpha = 1.7$)	30,000		5,024	

select and allocate approximately 6.5 million users across six different parties. For synthetic dataset SYN, we follow the existing studies [33, 63] and use a Dirichlet distribution to allocate different item domains for eight parties from the Tmall [49] dataset, which contains a vast array of anonymous shopping logs, into each party. We divide and sample all items into $N = 6$ groups. For each party P_i, we sample $q \sim Dir_N(\beta = 0.5)$, and allocate a q_j proportion of the total items in group j to construct item domain in P_i, where β controls the imbalance level of the domain skew. Due to the small concentration parameter (0.5) of the Dirichlet distribution, some sampled item domains may not have any items of certain groups of item. We employ Zipf and Poisson distributions with different parameters to build the frequency distribution. Each user in a party holds only a single word or item, and multiple occurrences are sampled as one. We summarize the details in Table 2.

Baselines and Parameter Setting. The most related study focuses on the cross-party setting is [46]. Since it does not fully satisfy LDP, we substitute the GRRX mechanism proposed by the authors with the k-RR mechanism to ensure the LDP guarantee, denoted as GTF. Another baseline, FedPEM, is described in Algorithm 1. It directly employs PEM, the state-of-the-art mechanism in single party settings, in each party, and then uploads all the identified local heavy hitters and their counts to the server, which then aggregates and reports the overall results. For a fair comparison, we use k-RR as the FO (we also investigate the impacts of different FOs, see Section 7.3) and set maximum length $m = 48$. For k-RR, we assign a dummy item to out-of-domain items. We set the granularity $g = 24$ (i.e., step size $m/g = 2$) since g should not be too large (see Section 7.4). We recommend $g = 24$ or 12 depending on the population size. A large population can support a fine granularity while reducing the accumulated sampling errors. β determines the proportion of users for pruning candidates at each level, functioning as a test set. Therefore, we set $\beta = 0.1$, following the common practice in machine learning. We heuristically set the level of the shared shallow trie $g = \lfloor 0.25g \rfloor$ and assign 10% users for the estimations in this phase as a warm start to align all parties. All the mechanisms are executed 50 times and the average is plotted.

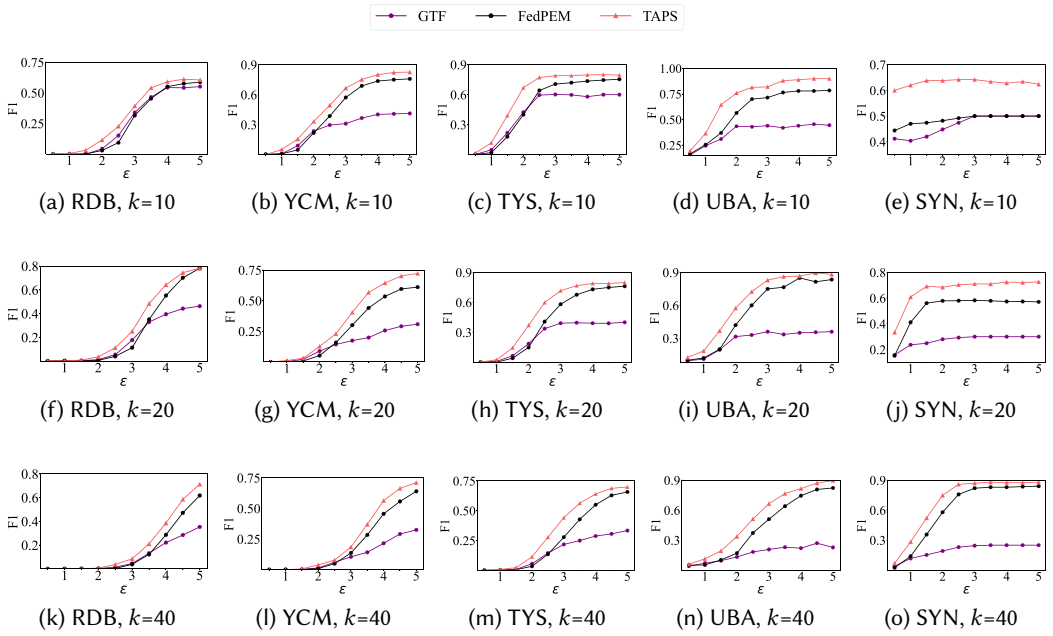


Fig. 4. F1 scores vs. privacy budget ϵ under different k.

Metrics. We evaluate the utility of the mechanisms using two metrics following the previous study [55]. The first one is the F1 score [37], which is the harmonic mean of precision and recall:

$$F1 = \frac{2pr}{p+r},$$

where $p = |R_T^k \cap R^k|/|R^k|$, $r = |R_T^k \cap R^k|/|R_T^k|$, R_T^k is the top k ground truths, and R^k is the estimated heavy hitters.

The other metric is the Normalized Cumulative Rank (NCR) [55], which assigns a higher penalty to missing the most frequent value than to others, by employing a quality function:

$$NCR = \frac{\sum_{v \in R^k} q(v)}{\sum_{v \in R_T^k} q(v)},$$

where $q(v) = k - rank(v)$ if v is ranked within top k, otherwise $q(v) = 0$. Both F1 and NCR scores fall within the range of $[0, 1]$, where larger scores indicate better performance.

7.2 Overall Results on Utility

We first evaluate the performance of the proposed TAPS mechanism over five datasets based on F1 score. We set privacy budget ϵ from 1 to 5, and the number of frequent items k from 10 to 40. As shown in Figure 4, although GTF can filter some noisy items when the privacy budget ϵ is small, it ignores the impacts of different quantities across parties and prunes too many inherent frequent but similar items, resulting in a poor performance. In the case of FedPEM, due to the independent estimation, it estimates many redundant prefixes or items and thus cannot lead to good performance. In contrast, our TAPS mechanism significantly outperforms baselines in all cases, even when k is relatively large and ϵ is small and as the number of parties grows. For example, when $\epsilon = 1$, TAPS achieves an F1 score of 0.283 on the SYN dataset, compared to 0.1395 for the

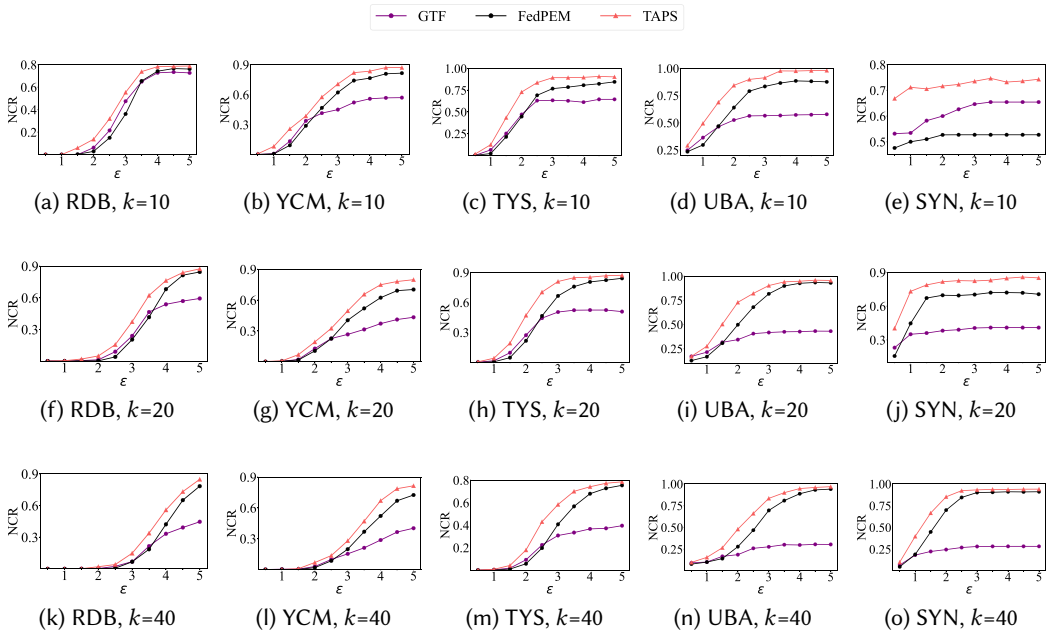


Fig. 5. NCR scores vs. privacy budget ϵ under different k.

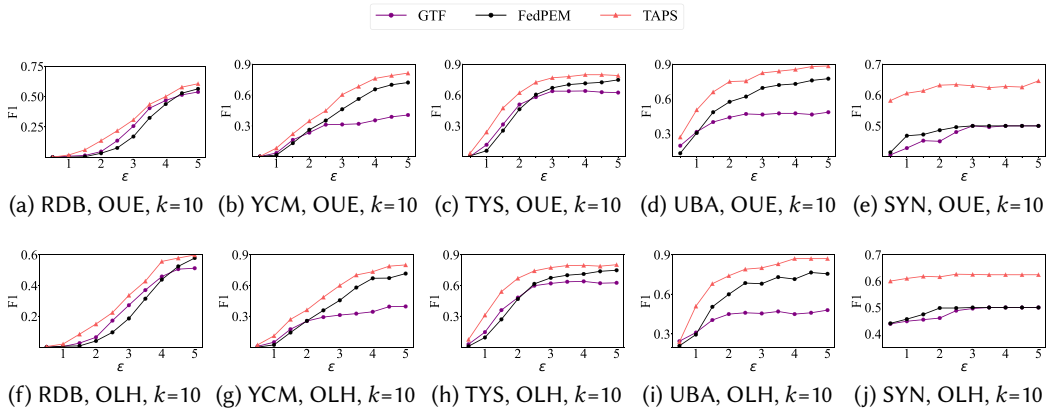
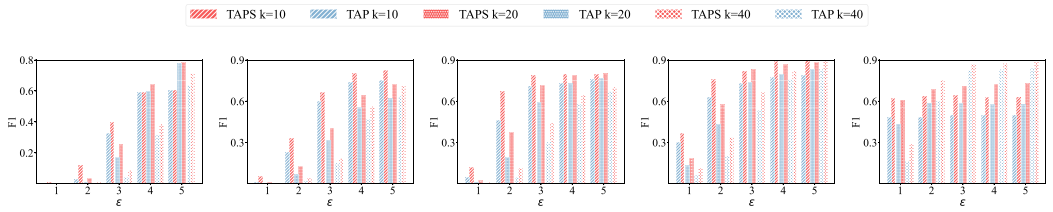


Fig. 6. F1 scores vs. privacy budget ϵ under OUE and OLH.

best baseline, demonstrating an improvement of over 100%. This highlights the effectiveness of our adaptive extension, shared trie construction, and the consensus-based pruning.

With regard to the NCR metric, TAPS always outperforms the baselines, as shown in Figure 5. The GTF mechanism achieves a higher NCR score in SYN when $k = 10$ compare to its F1 score. This discrepancy is due to the presence of some items in SYN whose frequencies are extremely high in certain parties. Therefore, with a relatively small $k = 10$ and a large number of parties in SYN, GTF can achieve a higher NCR score. However, as k increases, it drops many true negatives and thus diminishes its utility. Unlike the baselines, our TAPS mechanism benefits from the proposed strategies and consistently identify accurate heavy hitters, further demonstrating its effectiveness.



(a) RDB (b) YCM (c) TYS (d) UBA (e) SYN

Fig. 7. F1 scores of TAPS with or without the consensus-based pruning under varying privacy budget ϵ.

Table 3. F1 scores with varying step sizes ($\epsilon = 4$ and $k = 10$).

Dataset	Step size	GTF	FedPEM	TAPS
RDB	2	0.544	0.552	**0.592**
	4	0.392	0.632	**0.756**
	6	0.36	0.256	**0.382**
YCM	2	0.402	0.736	**0.80**
	4	0.392	0.632	**0.756**
	6	0.346	0.572	**0.656**
TYS	2	0.58	0.736	**0.798**
	4	0.642	0.69	**0.78**
	6	0.634	0.668	**0.748**
UBA	2	0.44	0.780	**0.89**
	4	0.50	0.755	**0.87**
	6	0.415	0.685	**0.815**
SYN	2	0.50	0.50	**0.628**
	4	0.50	0.50	**0.64**
	6	0.498	0.50	**0.63**

7.3 Impact Study on Different FOs

We further investigate the impact of different FOs on the performance. Specifically, for all the solutions, we substitute k-RR with two typical FOs, namely OUE [53] and OLH [53], respectively. Note that for OUE, we assign a dummy position in the encoded vector for the out-of-domain items or prefixes. As shown in Figure 6, the results are mostly consistent with those using k-RR. The proposed TAPS mechanism outperforms the baselines, which demonstrates the advantages and robustness of the proposed strategies. This indicates that TAPS is a general solution for federated hitter identification, and can be well adaptable to different FOs. As such, we can freely choose an appropriate FO according to the requirements of communication constraints and domain size of specific application scenario [53] to achieve better performance.

7.4 Ablation Study

We explore the impact of various parameters or components on performance, including different step sizes, adaptive trie extension, and shared shallow trie construction. Then, we assess our TAPS mechanism in addressing statistical heterogeneity. We also evaluate the effectiveness of our consensus-based pruning strategy and the robustness of the TAPS mechanism under varying degrees of data heterogeneity.

We first study the performance of TAPS and baselines under varying step sizes—different extension lengths $\lfloor m/g \rfloor \in \{2, 4, 6\}$, where m is the maximum length and g denotes the granularity (i.e., iterations). Note that we fix the privacy budget $\epsilon = 4$ and query $k = 10$. We can observe that in Table 3, TAPS consistently performs the best with its F1 scores shown in bold, which demonstrate the effectiveness of the proposed adaptive trie extension, shared shallow trie construction, and pruning strategies. Besides, the larger extension length also enhances the benefits of pruning a lot.

Table 4. F1 scores of TAPS with fixed/adaptive extension numbers ($\epsilon = 4$ and $k = 10$).

	$t = \lfloor k/2 \rfloor$	$t = k$	$t = 2k$	$t = 3k$	Adaptive t
RDB	0.424	0.574	0.582	0.546	**0.598**
YCM	0.644	0.786	0.782	0.762	**0.80**
TYS	0.753	0.769	0.781	0.785	**0.80**
UBA	0.89	0.89	0.885	0.87	**0.89**
SYN	0.624	0.614	0.602	0.60	**0.628**

Table 5. F1 scores of TAPS with or without the shared trie ($\epsilon = 4$ and $k = 10$).

	RDB	YCM	TYS	UBA	SYN
TAPS (w/o shared trie)	0.584	0.785	0.783	0.878	0.61
TAPS	**0.598**	**0.80**	**0.80**	**0.89**	**0.628**

Table 6. Average recall scores in addressing statistical heterogeneity ($\epsilon = 4$, $k = 10$, $\uparrow$: improvement over the best baseline).

	# parties	GTF	FedPEM	TAPS
RDB	2	0.304	0.302	**0.413** ($\uparrow$ **35.9%**)
YCM	4	0.262	0.255	**0.368** ($\uparrow$ **40.5%**)
TYS	6	0.296	0.283	**0.333** ($\uparrow$ **12.5%**)
UBA	6	0.295	0.294	**0.325** ($\uparrow$ **10.2%**)
SYN	8	0.162	0.162	**0.179** ($\uparrow$ **10.5%**)

Table 7. F1 scores with varying data heterogeneity controlled by β in SYN ($\epsilon = 4$ and $k = 10$).

	GTF	FedPEM	TAPS
$Dir_N(\beta = 0.2)$	0.384	0.40	**0.538**
$Dir_N(\beta = 0.5)$	0.50	0.50	**0.628**
$Dir_N(\beta = 0.8)$	0.50	0.552	**0.624**

Next, we investigate the adaptive trie extension strategy and set $\epsilon = 4$ and $k = 10$. Unless otherwise specified, we set the step size/extension length to 2 in the following experiments. We evaluate the TAPS mechanism with different fixed extension numbers from $\{\lfloor k/2 \rfloor, k, 2k, 3k\}$. In Table 4, we observe that the optimal fixed extension number varies across different datasets; however, our adaptive strategy consistently outperforms the fixed options, highlighting the benefits from adaptive trie construction. Additionally, we further assess the impact of the shared shallow trie construction on the accuracy of federated heavy hitters. Specifically, we remove the shared shallow trie construction step in TAPS and then compare the results with those by TAPS. As shown in Table 5, the TAPS with the shared shallow trie construction performs better across all datasets, which confirms its benefit.

We then evaluate the performance of TAPS on statistical heterogeneity by analyzing the average recall scores of global ground truths identified as local heavy hitters across different parties. As shown in Table 6, TAPS outperforms all baselines, achieving at least a 10% improvement over the best baseline across all datasets. This can be attributed to our shared shallow trie and adaptive extension strategies, which help align the heavy hitter targets. Additionally, the consensus-based pruning strategy mitigates statistical heterogeneity by leveraging prior knowledge from other parties.

We further validate the benefits of the consensus-based pruning strategy. As shown in Figure 7, the TAPS mechanism consistently outperforms its non-pruning version (TAP) across various datasets and queries k, underscores the value of our pruning strategy.

Then we study the impact of data heterogeneity on our TAPS mechanism's performance over the SYN dataset by varying the imbalance level of domain skewness. This is achieved by varying the Dirichlet distribution parameter $\beta \in \{0.2, 0.5, 0.8\}$, where a smaller β indicates more imbalance. As shown in Table 7 under $\epsilon = 4$ and $k = 10$, TAPS significantly outperforms baselines across various

Table 8. Scalability evaluation under varying user population in UBA ($\epsilon = 4$ and $k = 10$).

	F1 Scores			Communication Costs					Running Time				
User proportion	GTF	FedPEM	TAPS	GTF	FedPEM	TAPS	OUE	OLH	GTF	FedPEM	TAPS	OUE	OLH
25%	0.48	0.785	0.9	18 kb	18 kb	69 kb	> 2 PiB	$> 10^4$ kb	2.1105 s	2.0688 s	12.3642 s	> 72 h	> 72 h
50%	0.44	0.78	0.89	18 kb	18 kb	70 kb	> 2 PiB	$> 10^4$ kb	5.9571 s	6.2007 s	40.5691 s	> 72 h	> 72 h
75%	0.42	0.76	0.88	17 kb	17 kb	54 kb	> 2 PiB	$> 10^5$ kb	11.19 s	11.9729 s	79.9358 s	> 72 h	> 72 h
100%	0.44	0.78	0.89	17 kb	17 kb	75 kb	> 2 PiB	$> 10^5$ kb	17.5221 s	19.8624 s	128.0133 s	> 72 h	> 72 h

degrees of non-IID conditions. Both the adaptive trie extension and the consensus-based pruning strategies effectively mitigate the negative impacts of non-IID, leading to superior performance.

7.5 Impact Study on Scalability

We finally examine the scalability of our TAPS mechanism across varying user populations in the largest UBA dataset. We randomly sample different subsets of users, 25%, 50%, 75%, and 100%, in UBA to evaluate the F1 scores, communication costs, and running times. As shown in Table 8, our TAPS mechanism consistently outperforms the baselines across these varied populations, while maintaining competitive and acceptable communication and computation costs. The superior performance is because the shared shallow trie construction and the adaptive extension strategies help to preserve necessary prefixes by aggregating at a shallow level, and the consensus-based pruning strategy assists to leverage prior knowledge from other party to provide more informative frequency information. Although the communication cost of the TAPS mechanism is slightly higher than that of GTF and FedPEM, primarily due to the pruning strategy, it remains significantly lower than that of directly using OUE or OLH. Regarding the computation costs, TAPS requires more running time than GTF and FedPEM since sequential estimation consumes more time than the baselines that can be executed in parallel in different parties. However, it is still substantially less than that required by directly using OLH, demonstrating the practicality and robustness of TAPS across different user scales in real-world settings.

8 Conclusion

In this paper, we propose a novel target-aligning prefix tree mechanism satisfying ϵ-LDP, for federated heavy hitter analytics. We treat the estimation as a two-phase process, which applies the shared shallow trie construction and adaptive trie extension strategies. The proposed mechanism is further optimized by integrating a consensus-based pruning strategy, which enables each party to utilize (noisy) prior knowledge from the previous party to adaptively prune the candidate domain. To the best of our knowledge, this is the first solution to the federated heavy hitter analytics in a cross-party setting under the stringent ϵ-LDP. Extensive experiments on both synthetic and real-world datasets demonstrate the effectiveness of our mechanism.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No: 92270123, 62102334, 62372122 and 62072390), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15209922, 15208923, 15210023 and 25207224).

References

[1] Jayadev Acharya and Ziteng Sun. 2019. Communication Complexity in Locally Private Distribution Estimation and Heavy Hitters. In *International Conference on Machine Learning (ICML)*. PMLR, 51–60.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules. In *Proceedings of 20th International Conference on Very Large Data Bases (VLDB)*, Vol. 1215. 487–499.

[3] Differential Privacy Team Apple. 2017. Learning with Privacy at Scale. In *Apple Machine Learning Journal*.

[4] Raef Bassily and Adam Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*. 127–135.

[5] Jonas Böhler and Florian Kerschbaum. 2021. Secure Multi-Party Computation of Differentially Private Heavy Hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2361–2377.

[6] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. Data Poisoning Attacks to Local Differential Privacy Protocols. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*. 947–964.

[7] Karan Chadha, Junye Chen, John Duchi, Vitaly Feldman, Hanieh Hashemi, Omid Javidbakht, Audra McMillan, and Kunal Talwar. 2023. Differentially Private Heavy Hitter Detection using Federated Analytics. In *ICML Workshop on Federated Learning and Analytics in Practice (FL-ICML)*.

[8] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. In *Proceedings of the 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 129–138.

[9] Xiang Cheng, Sen Su, Shengzhi Xu, Li Xiong, Ke Xiao, and Mingxing Zhao. 2018. A Two-Phase Algorithm for Differentially Private Frequent Subgraph Mining. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 30. 1411–1425.

[10] Xiang Cheng, Peng Tang, Sen Su, Rui Chen, Zequn Wu, and Binyuan Zhu. 2020. Multi-Party High-Dimensional Data Publishing under Differential Privacy. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 32. 1557–1571.

[11] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[12] Graham Cormode and Akash Bharadwaj. 2022. Sample-and-Threshold Differential Privacy: Histograms and Applications. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1420–1431.

[13] Graham Cormode, Samuel Maddock, and Carsten Maple. 2021. Frequency Estimation under Local Differential Privacy. In *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 14. 2046–2058.

[14] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. 3574–3583.

[15] Xiaofeng Ding, Shujun Sheng, Huajian Zhou, Xiaodong Zhang, Zhifeng Bao, Pan Zhou, and Hai Jin. 2022. Differentially Private Triangle Counting in Large Graphs. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 34. 5278–5292.

[16] Yuntao Du, Yujia Hu, Zhikun Zhang, Ziquan Fang, Lu Chen, Baihua Zheng, and Yunjun Gao. 2023. LDPTrace: Locally Differentially Private Trajectory Synthesis. In *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 16. 1897–1909.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography (TCC)*. 265–284.

[18] Cynthia Dwork, Aaron Roth, et al. 2014. The Algorithmic Foundations of Differential Privacy. In *Foundations and Trends® in Theoretical Computer Science (TCS)*, Vol. 9. 211–407.

[19] Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Shanshan Han, Shantanu Sharma, Chaoyang He, Sharad Mehrotra, Salman Avestimehr, et al. 2023. Federated Analytics: A Survey. In *APSIPA Transactions on Signal and Information Processing*, Vol. 12.

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1054–1067.

[21] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. In *Proceedings on Privacy Enhancing Technologies (PETS)*, Vol. 2016. 41–61.

[22] Jie Fu, Qingqing Ye, Haibo Hu, Zhili Chen, Lulu Wang, Kuncan Wang, and Xun Ran. 2024. DPSUR: Accelerating Differentially Private Stochastic Gradient Descent Using Selective Update and Release. In *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 17. 1200–1213.

[23] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and

Directions. In *ACM Transactions on Recommender Systems (TORS)*, Vol. 1. 1–51.
[24] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. In *CS224N project report, Stanford*, Vol. 1. 2009.
[25] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1693–1701.
[26] Kai Huang, Gaoya Ouyang, Qingqing Ye, Haibo Hu, Bolong Zheng, Xi Zhao, Ruiyuan Zhang, and Xiaofang Zhou. 2024. LDPGuard: Defenses against Data Poisoning Attacks to Local Differential Privacy Protocols. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.
[27] Pranav Jangir, Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, Bhavish Raj Gopal, and Somya Sangal. 2023. Vogue: Faster Computation of Private Heavy Hitters. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 1–12.
[28] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and Open Problems in Federated Learning. In *Foundations and Trends® in Machine Learning*, Vol. 14. 1–210.
[29] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2014. Extremal Mechanisms for Local Differential Privacy. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS)*. 2879–2887.
[30] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What Can We Learn Privately?. In *SIAM Journal on Computing (SICOMP)*, Vol. 40. 793–826.
[31] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A Large Self-Annotated Corpus for Sarcasm. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*.
[32] Ninghui Li and Qingqing Ye. 2019. Mobile Data Collection and Analysis with Local Differential Privacy. In *Proceedings of the IEEE 20th International Conference on Mobile Data Management (MDM)*. 4–7.
[33] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated Learning on Non-IID Data Silos: An Experimental Study. In *2022 IEEE 38th international conference on data engineering (ICDE)*. 965–978.
[34] Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. 2024. Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory. In *Proceedings of the 50th ACM Conference on Management of Data (SIGMOD)*. 1–27.
[35] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *Proceedings of the 41st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1137–1140.
[36] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*. 142–150.
[37] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge university press.
[38] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 188–197.
[39] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2671–2679.
[40] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*. 2685–2692.
[41] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 192–203.
[42] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vol. 2. 784–789.
[43] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2383–2392.
[44] Daniel Ramage. 2020. Federated Analytics: Collaborative Data Science without Data Collection. Online. https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html.
[45] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1073–1083.

[46] Jiaqi Shao, Shanshan Han, Chaoyang He, and Bing Luo. 2023. Privacy-Preserving Federated Heavy Hitter Analytics for Non-IID Data. In *ICML Workshop on Federated Learning and Analytics in Practice (FL-ICML)*.
[47] Sen Su, Peng Tang, Xiang Cheng, Rui Chen, and Zequn Wu. 2016. Differentially Private Multi-Party High-Dimensional Data Publishing. In *Proceedings of the IEEE 32nd International Conference on Data Engineering (ICDE)*. 205–216.
[48] Xinyue Sun, Qingqing Ye, Haibo Hu, Jiawei Duan, Tianyu Wo, Jie Xu, and Renyu Yang. 2024. Ldprecover: Recovering Frequencies from Poisoning Attacks against Local Differential Privacy. In *Proceedings of the IEEE 40th International Conference on Data Engineering (ICDE)*. 1619–1631.
[49] Tianchi. 2018. IJCAI-15 Repeat Buyers Prediction Dataset. https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[50] Dan Wang, Siping Shi, Yifei Zhu, and Zhu Han. 2021. Federated Analytics: Opportunities and Challenges. In *IEEE Network*, Vol. 36. 151–158.
[51] Ning Wang, Xiaokui Xiao, Yin Yang, Ta Duy Hoang, Hyejin Shin, Junbum Shin, and Ge Yu. 2018. PrivTrie: Effective Frequent Term Discovery under Local Differential Privacy. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 821–832.
[52] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. In *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE)*. 638–649.
[53] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*. 729–745.
[54] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *2018 IEEE Symposium on Security and Privacy (SP)*. 127–143.
[55] Tianhao Wang, Ninghui Li, and Somesh Jha. 2019. Locally Differentially Private Heavy Hitter Identification. In *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Vol. 18. 982–993.
[56] Zibo Wang, Yifei Zhu, Dan Wang, and Zhu Han. 2022. FedFPM: A Unified Federated Analytics Framework for Collaborative Frequent Pattern Mining. In *IEEE Conference on Computer Communications (INFOCOM)*. 61–70.
[57] Stanley L Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. In *Journal of the American Statistical Association (JASA)*, Vol. 60. 63–69.
[58] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A Large-Scale Dataset for News Recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 3597–3606.
[59] Min Xu, Bolin Ding, Tianhao Wang, and Jingren Zhou. 2020. Collecting and Analyzing Data Jointly from Multiple Services under Local Differential Privacy. In *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 13. 2760–2772.
[60] Jianyu Yang, Xiang Cheng, Sen Su, Huizhong Sun, and Changju Chen. 2022. Collecting Individual Trajectories under Local Differential Privacy. In *Proceedings of the 23rd IEEE International Conference on Mobile Data Management (MDM)*. 99–108.
[61] Qingqing Ye, Haibo Hu, Kai Huang, Man Ho Au, and Qiao Xue. 2023. Stateful Switch: Optimized Time Series Release with Local Differential Privacy. In *IEEE Conference on Computer Communications (INFOCOM)*. 1–10.
[62] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-Value Data Collection with Local Differential Privacy. In *Proceedings of the 40th IEEE Symposium on Security and Privacy (SP)*. 317–331.
[63] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian Nonparametric Federated Learning of Neural Networks. In *International Conference on Machine Learning (ICML)*. 7252–7261.
[64] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[65] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. In *ACM Transactions on Database Systems (TODS)*, Vol. 42. 1–41.
[66] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-Level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 28.
[67] Yuemin Zhang, Qingqing Ye, Rui Chen, Haibo Hu, and Qilong Han. 2023. Trajectory Data Collection with Local Differential Privacy. In *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 16. 2591–2604.
[68] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 5941–5948.
[69] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. 2020. Federated Heavy Hitters Discovery with Differential Privacy. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 3837–3847.

Received July 2024; revised September 2024; accepted November 2024