Journal of Information and Data Management, 2026, 17:1, doi: 10.5753/jidm.2026.7437
This work is licensed under a Creative Commons Attribution 4.0 International License.

Data-Driven User Allocation for Top-k Itemset Mining Under Local Differential Privacy

Renan Ribeiro de Oliveira [**Laboratório de Sistemas e Bancos de Dados (LSBD)** | ***renan.oliveira@lsbd.ufc.br***]
José S. Costa [**Laboratório de Sistemas e Bancos de Dados (LSBD)** | ***serafim.costa@lsbd.ufc.br***]
José M. Monteiro [**Laboratório de Sistemas e Bancos de Dados (LSBD)** | ***jose.monteiro@lsbd.ufc.br***]
Javam C. Machado [**Laboratório de Sistemas e Bancos de Dados (LSBD)** | ***javam.machado@lsbd.ufc.br***]

Laboratório de Sistemas e Bancos de Dados (LSBD) Departamento de Computação , UFC – Fortaleza – CE – Brazil

Received: 20 January 2026 • **Published:** 27 November 2026

Abstract The identification of Heavy-Hitters through Apriori-based frameworks is widely used to uncover meaningful patterns in transactional datasets and support decision making. However, achieving this while guaranteeing user privacy under Local Differential Privacy (LDP) poses significant challenges. In this context, domain scalability and set-valued user data hinder the performance of many existing LDP protocols. Prior work has addressed these problems through fixed-size transaction reporting protocols, but they often rely on hard-coded parameters, overlooking nuances across differing applications and dataset profiles. To address this limitation, we propose a data-driven approach for identifying the top-k most frequent itemsets under LDP. Through tailored queries, the integration of an optimization problem rooted in the underlying reporting mechanisms, and the modeling of data characteristics to guide sampling strategy and group partitioning decisions, our algorithm improves estimation accuracy across a wide range of privacy budgets and data profiles, achieving up to twice the utility compared to baselines.

Keywords: Data-Driven, Itemset mining, Local Differential Privacy, Heavy-Hitters, Optimization, Apriori

1 Introduction

Differential Privacy (DP), introduced by Dwork *et al*. [2006], has become a standard for protecting individual privacy in data analysis. Historically, DP has been implemented in a centralized setting, where a trusted curator collects raw data and applies privacy-preserving mechanisms before releasing aggregate insights. However, growing concerns over data breaches and misuse, as reported in Zhu *et al*. [2017]; K *et al*. [2021]; Frenkel and Isaac [2018]; Satariano *et al*. [2018] have motivated the adoption of Local Differential Privacy (LDP) frameworks Erlingsson *et al*. [2014], which shift privacy protection to the data source and remove the need for a trusted curator while preserving strong privacy guarantees.

A primary application of LDP is the private estimation of categorical frequencies, which extends to frequent itemset mining, a fundamental data mining problem with applications ranging from market basket analysis, association rule mining, and user behavior modeling. LDP-based methods enables pattern discovery and data-driven decision-making while preserving user privacy. Beyond traditional domains, it has been applied in large-scale systems such as Internet of Things (IoT) hardware diagnostics, where it integrates with AI techniques to detect recurring faults and co-occurring error patterns, enabling root cause analysis, failure prediction, and automated maintenance, as seen in Lin *et al*. [2020].

However, itemset mining raises significant privacy concerns, as publishing frequent itemsets can inadvertently reveal sensitive user information. Discovering frequent itemsets under LDP remains challenging: the exponential number of possible itemsets dilutes counts, making them difficult to distinguish from noise added by Frequency Oracles (FOs). Moreover, real-world datasets exhibit diverse distribution profiles, including differences in skewness and variance, aspects often overlooked in existing approaches. Effectively addressing this problem requires a data-driven solution that adapts to each dataset's underlying profile.

In Apriori-based itemset mining under LDP, user partitioning plays a critical role in itemset estimation, as it controls how the data are allocated across the differing steps of the framework. Estimation accuracy depends not only on the reporting protocol and the privacy budget used, but also on the number of users assigned to each step. Moreover, depending on the dataset, each step may contribute differently to the overall utility of the itemset mining process, highlighting that a hard-coded user partitioning setting is unlikely to be optimal across all applications.

In this paper, we build upon our prior DDSM framework Oliveira *et al*. [2025] and propose a data-driven partitioning strategy. We extend DDSM by identifying and exploiting key dataset characteristics obtained via a tailored private query to efficiently allocate users across the different steps of our new algorithm. A model is introduced to capture the correlation between these characteristics and the accuracy impact of each step in identifying the top-k items and itemsets under LDP, mitigating the sampling variance caused by splitting the user pool.

Contributions. Our main contributions are:

- We identify key dataset characteristics that affect sampling variance and show how they can be estimated through an LDP query.
- We propose a data-driven user partition strategy for the multi-queried LDP algorithm, that efficiently allocates users across algorithmic steps.
- We conduct a comprehensive experimental evaluation on both real-world and synthetic datasets from different application domains, benchmarking our technique against leading prior work.

2 Background And Definitions

2.1 Local Differential Privacy (LDP)

Under LDP, sensitive information v from each user is encoded by a randomized algorithm Ψ, and its output $\Psi(v)$ is sent to the aggregator, which is responsible for collecting all users' reports. Intuitively, LDP guarantees that, no matter what the value of $\Psi(v)$ is, it is approximately equally as likely to be a result of perturbing v as any other v' differing from v. Therefore, if $\Psi(v)$, instead of v, is collected, the users never reveal their private value v. The user's degree of privacy is controlled by the privacy budget ϵ. More formally,

Definition 1 (Local Differential Privacy Erlingsson *et al*. [2014]**).** *An algorithm $\Psi(\cdot)$ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon \geq 0$, if and only if for any pair of inputs (v, v'), and any set R of possible outputs of Ψ, we have*

$$Pr[\Psi(v) \in R] \leq e^{\epsilon} Pr[\Psi(v') \in R] \tag{1}$$

2.2 Frequency Oracle (FO)

Under LDP, frequency estimation is tackled with FO protocols, which estimate the frequency of sufficiently recurring values $v \in D$ in a domain D, while ensuring ϵ-LDP guarantees. Users apply noise locally with Ψ, and the aggregator estimates frequencies by applying Φ to the reports. We denote Φ as the (aggregator's side) estimation function applied by the aggregator to the collected reports, mapping them to frequency estimates over the domain D. These protocols are widely applicable to LDP tasks, as many problems can be reduced to the problem of frequency estimation.

2.2.1 Generalized Randomized Response (GRR)

Randomized Response Warner [1965], originally designed for binary domains, can be readily extended to larger domains Kairouz *et al*. [2016]. Each user reports their private value $v \in D$ with probability p, effectively reporting its true value. Otherwise, with probability $(1-p)$, the user sends a randomly chosen value $v' \in D$, effectively reporting noise. Formally, the user-end algorithm Ψ is defined as follows:

$$\forall_{x \in D} \Pr\left[\Psi_{\mathrm{GRR}_{(\epsilon)}}(v) = x\right] = \begin{cases} \dfrac{e^{\epsilon}}{e^{\epsilon} + |D| - 1}, & \text{if } x = v \\ \dfrac{1}{e^{\epsilon} + |D| - 1}, & \text{if } x \neq v \end{cases} \tag{2}$$

GRR satisfies ϵ-LDP since $p/q = e^{\epsilon}$. After receiving the vector of the n reports $\mathbf{r} = \langle x_1, x_2, \cdots, x_n \rangle$, where $x_i \in D$ is the reported value of the i-th user, the aggregator can estimate the frequency of any value $v \in D$ with Φ. Considering $C(n)$ as the frequency of value v in the response vector $\mathbf{r}$, the unbiased estimator for the frequency of $v \in D$ is $\Phi_{GRR_{(\epsilon)}}(v) := (C(v)/n - q)/(p - q)$.

Wang *et al*. [2017] Show that this is an unbiased estimation of the true count, and the variance of this estimation is

$$Var\left[\Phi_{GRR_{(\epsilon)}}(v)\right] = \frac{e^{\epsilon} + |D| - 2}{n(e^{\epsilon} - 1)^2}. \tag{3}$$

Since the variance (Equation 3) is linear in $|D|$, the protocol's accuracy decreases as the domain size increases.

2.2.2 Optimized Local Hashing (OLH)

The Optimized Local Hashing protocol Wang *et al*. [2017] tackles the limitation of large domains in GRR by applying a hash function that maps each input value to a smaller domain of size g. GRR is then applied to the hashed output. To minimize variance, g is set to $\lceil e^{\epsilon} + 1 \rceil$.

The hashing process uses a universal hash function family $\mathbb{H}$, where each $H \in \mathbb{H}$ maps a value from D to $\{1, \ldots, g\}$. The variance of the OLH estimator is given by

$$\mathrm{Var}\left[\Phi_{\mathrm{OLH}_{(\epsilon)}}(v)\right] = \frac{4e^{\epsilon}}{n(e^{\epsilon} - 1)}. \tag{4}$$

2.3 Padding and Sampling Frequency Oracle (PSFO)

One approach proposed in Qin *et al*. [2016] for handling set-valued inputs is to apply padding and sampling to the user's transaction before reporting their data with a standard FO. Conceptually, PSFO combines a padding and sampling (PS) function to process set-valued inputs and an FO for reporting data. The PS function works by filling transactions with dummy items until they reach a parameterized length. It then samples a single element from the resulting transaction.

Definition 2 (Padding and Sampling Function Qin *et al*. [2016]**)** *The padding and sampling function PS is parameterized by a positive integer ℓ and takes a set $v \subseteq I$ as input. It assumes the existence of ℓ dummy items $\perp_1, \perp_2, \ldots, \perp_\ell \notin I$. If $|v| < \ell$, $PS_\ell(v)$ adds $\ell - |v|$ unique random dummy items to the transaction v, then samples an element x at uniform random from it.*

A standard FO is used to report the element x sampled by PS. In this context, the FO's domain includes both the data domain and the dummy elements introduced during padding: $I \cup \{\perp_1, \perp_2, \ldots, \perp_\ell\}$. The aggregator obtains each item frequency estimate from the FO protocol and multiplies it by ℓ. PSFO standardizes transactions by padding them to a fixed length ℓ, ensuring each item x has an equal chance $1/\ell$ of being sampled and reported. This scaling of the estimates corrects the underestimation originated from the sampling step. While PS does not explicitly truncate transactions longer than ℓ, the scaling still assumes a transaction of length ℓ, effectively causing information loss for frequently occurring items in transactions longer than that. Scaling also amplifies the FO's noise by a factor of ℓ.

Definition 3 (Padding and Sampling Frequency Oracle Qin *et al*. [2016]**)** *A Padding and Sampling Frequency Oracle (PSFO) protocol is parameterized by a positive integer ℓ, a frequency oracle (FO), and a privacy budget ϵ. It consists of a pair of algorithms $\langle \Psi, \Phi \rangle$, defined as:*

$$PSFO(\ell, FO, \epsilon) := \left\langle \Psi_{FO(\epsilon)}(PS_\ell(\cdot)),\ \Phi_{FO(\epsilon)}(\cdot) \cdot \ell \right\rangle$$

Figure 1 illustrates the PSFO protocol functioning for a transaction v of size $|v| = 3$ with the limit size set to $\ell = 5$. The PS function pads the transaction to size $|v'| = \ell = 5$ and samples one item at uniform random. The sampled item is reported using a standard FO protocol, and the estimate is corrected by a factor of ℓ at the aggregator.

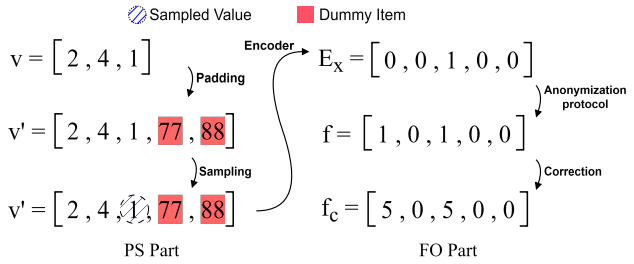

Figure 1. PSFO reporting process for a transaction v with size limit ℓ

2.4 Adaptive FO

When applying an FO with a PS function, Wang *et al*. [2018] show that GRR benefits from privacy amplification, whereas OLH benefits less. To address this, they introduce an adaptive protocol (Adapt) that selects GRR with privacy budget $\epsilon = \ln(\ell(e^{\epsilon} - 1) + 1$ if $d < e^{\epsilon}\ell(4\ell - 1) + 1$ and OLH with privacy budget ϵ otherwise.

3 Related Works

LDPMiner Qin *et al*. [2016], a pioneering method for querying relevant items under LDP, introduces a novel protocol for reporting set-valued data. It adopts a two-phase approach in which all users participate in both steps, each satisfying $(\epsilon/2)$-LDP, thereby ensuring ϵ-LDP via sequential composition. In this context, we use the term *utility* to denote estimation accuracy, reflecting the practical value of the privately reported data to the aggregator.

Wang *et al*. [2018] SVSM is an Apriori-based method; it improves upon LDPminer by addressing the itemset mining problem left open in its work, while also outperforming it in the item mining task. It introduces SVIM for frequent item mining and SVSM for itemset mining. Unlike LDPMiner, SVSM partitions users into groups, with each group answering one query under ϵ-LDP. The method consists of a multi-step query process involving multiple user groups and utilizing PSFO to estimate frequent items and itemsets.

While SVSM improves upon LDPMiner by introducing additional processing steps, it lacks adaptability. Its methodology relies on static parameters and fails to adjust to dataset characteristics, limiting its ability to optimize performance across diverse datasets. In particular, SVSM uses empirically fixed values of ℓ throughout its reporting steps, which hinders the opportunity to further exploit higher data utility in datasets with a minimum transaction size greater than one, as it enforces a suboptimal parameter. Moreover, SVSM adopts a fixed group partitioning approach that is independent of data characteristics. This methodology fails to account for high-variance datasets, where alternative user partition configurations that allocate larger samples to variance-sensitive steps could significantly reduce sampling error.

LDP-FPMiner Chen and Wang [2022] propose an FP-tree-based approach that exploits frequency consistency among itemsets, which is not done by SVSM. The method constructs and optimizes utility through a noisy FP-tree under LDP constraints. However, LDP-FPMiner requires prior knowledge of frequent items in the dataset, relying on SVIM for their estimation, whose utility is highly sensitive to the choice of transaction length ℓ.

PrivMiner Li *et al*. [2025] is designed to find potentially frequent items and then estimate top-k itemsets. It introduces a mechanism in which all possible records are partitioned into disjoint groups based on similarity to the true record. A key limitation is its strong dependence on the ℓ (record length) and g (number of groups) parameters, both of which are selected through heuristic strategies rather than principled optimization. The grouping parameter g in the group-wise estimation mechanism significantly influences the balance between privacy noise and data utility. A small g reduces distinguishability between groups and leads to information loss, whereas a large g increases variance and estimation error due to excessive noise injection.

Oliveira *et al*. [2025] DDSM adopts a data-dependent approach to itemset mining by leveraging auxiliary queries to collect metadata and optimize data utility. It formulates an optimization function and designs queries to compute optimal values for the size limit ℓ, enabling PSFO to report the optimal value for each query. Despite demonstrating consistent gains across diverse data profiles, DDSM exhibits limitations on datasets with high domain cardinality and skewed distributions. Moreover, it continues to rely on fixed user partitioning across the steps of its framework.

Limitations of baselines. Among the works that find top-k itemsets, all except DDSM have a strong dependence on parameters such as the transaction length limit ℓ. Moreover, all baselines rely heavily on population partitioning to execute the multiple phases. This partitioning is typically fixed uniformly or determined by heuristics, neither of which is guaranteed to be optimal across datasets with varying characteristics.

4 Defining the Problem

We consider a problem where each user holds a private subset of items. The data aggregator seeks to find the most relevant itemsets while ensuring each user's report satisfies ϵ-LDP. Let $D = \{v_1, v_2, \cdots, v_n\}$ be the dataset of the n users' transactions. Each transaction v_i is a subset of items, i.e, $v \subseteq I$, where I denotes the domain of all possible items in D. In this context, we define the relevance of an item or itemset in terms of its count, that is, the frequency with which it appears across the dataset. Hence, a lower count indicates less relevancy, and a higher count indicates higher relevancy. Ultimately, the aggregator's goal is to find the set IS of the top-k most relevant itemsets.

While existing FO protocols are effective for querying attribute counts in tabular datasets, where each user reports a single attribute value, set-valued data introduces additional challenges. Standard FO protocols struggle to scale in this context, as they rely on binary encoding to apply noise to the user's data. In this setting, encoding each transaction as a binary array over the domain $\mathcal{P}(I)$ (the power set of items in I) quickly becomes infeasible due to the exponential number of combinations. Additionally, FO protocols are only capable of identifying values that occur frequently, since the magnitude of the added noise is proportional to the square root of the population size Chan *et al*. [2012]. In set-valued datasets, although certain items and itemsets occur frequently, the transactions themselves are typically unique or relatively infrequent. Thus, directly encoding and querying full transactions under LDP would effectively yield only noise when no single transaction pattern appears often enough to stand out.

5 DHHS: Data-Driven Identification of Heavy-Hitter Itemsets

In this section, we describe the proposed algorithm in detail and explain how a data-driven approach is ensured throughout its design and implementation. We begin by formalizing the transaction length, i.e., the size of the report, optimization process, and the group partitioning strategy. We then show how this methodology is leveraged to guide and enhance each step of our algorithm.

5.1 Optimal Limit Size for Reporting

The PS function, specified in Definition 2, is capable of adding a significant amount of noise to the user's response. Depending on the parameter ℓ, some users may end up reporting way more noise than others. Overall, the amount of noise reported due to padding is data dependent on the number of user sets smaller than ℓ. For a transaction v and a transaction size limit ℓ used in the PSFO protocol, when $|v| < \ell$, there's a probability $P_{\text{Sdummy}} = \frac{\ell - |v|}{\ell}$ of the protocol feeding the added noise to the FO and a probability $P_{\text{Sreal}} = \frac{|v|}{\ell}$ of the protocol reporting a real value to the FO, strengthening the utility of the estimation. Although transactions of size $|v| > \ell$ have zero probability of sampling noise, since they will not be padded by the PS function, they can still impact the protocol's utility due to underestimation, as their items will be sampled with a probability bigger than $1/|v|$ while being scaled only by a factor of ℓ.

For instance, in Figure 2, the PS function receives a set v of size 3 and the size limit ℓ is set to 5. After padding, PS has a probability $P_{\text{Sdummy}} = 40\%$ of sampling a dummy item, and $P_{\text{Sreal}} = 60\%$ of sampling a real item.

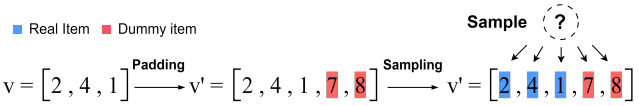

Figure 2. Illustration of the sampling done by the PS function

While larger transactions carry more utility in their reports for larger values of ℓ, smaller transactions, once padded, add too much noise to the user's private set. Similarly, when ℓ is small, larger transactions are underestimated in their reports, whereas smaller transactions have less noise padded into them. This dynamic underlies the choice of ℓ. Underestimation introduces bias, while padding increases variance by adding dummy items. Therefore, before applying a PSFO protocol, we propose to select an ℓ that balances this trade-off. This optimal value is closely tied to the dataset's transaction-size distribution.

We introduce an objective function to compute the optimal value L_{opt} for the PSFO protocol. It models the trade-off between utility losses and gains resulting from adjusting the transaction size limit. By formalizing this relationship, the objective function allows the identification of the value of ℓ that offers the best balance between noise and data utility. Loss is defined as $P_{\text{Sdummy}} f_l L$, the expected number of times PSFO samples a dummy item from a set of transactions of size l, multiplied by L, a limit size set for the PSFO protocol, and also the times at which the sample will be scaled. Likewise, Gain is defined as Gain $= P_{\text{Sreal}} f_l L$, the expected number of times PSFO samples a real item from a set of transactions of size l, multiplied by L. The objective function is computed as the sum of the differences between Gain and Loss for each transaction size l in the dataset's size distribution, denoted by f, where f_l indicates the count of transactions of size l within the f distribution. The optimization problem is then configured to find the value of L for PSFO that maximizes the total utility across all possible lengths $l \in [1, 2, \ldots, d]$:

$$L_{\text{opt}} = \arg\max_{L} \left(L \sum_{l=1}^{d} (P_{\text{Sreal}} f_l - P_{\text{Sdummy}} f_l) \right) \quad (5)$$

We refer to Eq. (5) as the function $\text{Opt}(\cdot)$, which computes the optimal transaction limit size for PSFO that maximizes its reporting utility given a transaction length distribution: $L_{\text{opt}} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(d)])$.

5.2 Optimal User Group Partitioning

In the proposed algorithm, the most frequent items and itemsets are identified throughout a multi-step process involving multiple queries. This approach works by partitioning the total user data pool into groups, each tasked with reporting specific information to the aggregator. This is similarly done in Filho and Machado [2023], where user partitioning improved utility compared to querying the entire dataset in every step while splitting the privacy budget to satisfy LDP.

In previous apriori itemset mining works, DDSM and SVSM Oliveira *et al*. [2025]; Wang *et al*. [2018], the group partitions are static parameters, determined empirically, such that a strong performance on a limited number of datasets dictates the partitioning setting. While this approach may yield good results on tested datasets, it overlooks their unique statistical characteristics, as it is unlikely that a single setting will be optimal for every dataset the algorithm encounters; in some cases, it may even be the worst.

In comparison to Subsection 5.1, where we proposed an optimization approach based on a hypothesis around the PSFO protocol, the data-driven group partitioning optimization emerged from empirical observations across datasets with different distributions. By analyzing how each step of the DDSM framework contributes to the overall final utility, we identified a nuanced interaction between dataset characteristics and the framework's ability to estimate the most frequent items and itemsets. In particular, for sparse datasets with a large item domain, where a large fraction of itemsets are infrequent, allocating more users to the steps that identify frequent candidates tends to improve utility. This behavior is mainly explained by sampling variance, which is higher in sparse datasets and makes them more sensitive to partitioning. By allocating more data in these steps, the framework captures a more representative sample of the set-valued patterns, reducing the sampling variance and enabling the FO to estimate their ranks more accurately.

In contrast, in dense datasets, where patterns are densely distributed, the gain from increasing the sample sizes in steps that construct candidates diminishes, as smaller subsets suffice for that task. In this scenario, allocating a larger frac-

tion of the users to these steps becomes inefficient, as it reduces data volume for subsequent steps. Additionally, we observed that standard deviation and skewness also influence how each step responds to changes in group size. In particular, the steps that directly estimate candidate frequencies benefit more from larger sample sizes when these values are low, whereas the gain becomes negligible when they are high.

Within this context, we propose modeling the relationship between the dataset characteristics and the framework's group partitioning to achieve more efficient user allocation across the aggregators' queries.

To model this correlation, we adopt a Random Forest regression model due to its robustness to noise and its ability to capture non-linear relationships. The model maps dataset characteristics to the optimal group partitioning. The model's input features include the dataset domain size, the privacy budget, and the standard deviation and skewness of the transaction size distribution. The output consists of the optimal proportions of users for each group, as determined by the algorithm.

Table 1. Synthetic datasets and their characteristics

Dataset	n	γ	σ	Distribution
D1	**500,000**	**5.00**	**200**	**Exponential**
D2	**500,000**	**2.50**	**20**	**Pos. Skewed Normal**
D3	**250,000**	**0.10**	**2**	**Normal**
D4	**10,000**	**1.00**	**2**	**Exponential**

The training dataset was constructed by executing the DDSM algorithm on four synthetic transactional datasets seen in Table 1 (based on Sharkfin's code Oliveira [2026]), while varying the partition configurations. Specifically, the algorithm was run with varying privacy budgets across 126 distinct partitioning settings per dataset, corresponding to all possible combinations of 5 groups, with individual partitions ranging from 10% to 90%. Based on the results, the ones with utility in the upper 80th percentile are selected to compose the model's training data, along with all dataset characteristics and their corresponding best partitioning settings.

In summary, the resulting dataset maps the dataset characteristics, the privacy budget, and the corresponding best group partitions that yield the highest utility. The dataset was split into disjoint training and test sets, where the Random Forest model was trained exclusively on the training set and evaluated for prediction accuracy on the test set.

Our proposed algorithm uses a six-group partitioning setup, where the first group serves to privately estimate the regression model's relevant statistics. To compute the starting group partition, Eq. 7 is introduced to determine the proportion allocated to group G_1. Here, p_{min} and p_{max} are the lower and upper bounds on the partition percentage, and c_{min} and c_{max} denote the domain cardinality thresholds. For domain sizes below c_{min}, the partition is fixed at p_{min}; for values between c_{min} and c_{max} it scales linearly; and for domain sizes exceeding c_{max} it caps at p_{max}.

$$\text{weight}(d) = \frac{\min\left(\lfloor \log_{10} d \rfloor + 1,\ c_{\max}\right) - c_{min}}{c_{\max} - c_{min}} \tag{6}$$

$$p(d) = \begin{cases} p_{\min} & \lfloor \log_{10} d \rfloor + 1 \leq c_{min} \\ p_{\min} + (p_{\max} - p_{\min}) \cdot \text{weight}(d) & \text{otherwise} \end{cases} \tag{7}$$

5.3 The DHHS Algorithm

Based on the methodology presented in this section, we propose a new algorithm for identifying the top-k items and itemsets under LDP, consisting of Data-driven Heavy Hitters Item Identification (DHHI) for frequent item mining and Data-driven Heavy Hitters ItemSet Identification (DHHS) for frequent itemset mining. The algorithm begins by identifying frequent items using DHHI, then estimates frequent itemsets using DHHS through a multi-step process. Except for the first step, which is used to fetch user transaction lengths to optimize user partitioning, each step performs one of two core aggregator operations: querying the distribution of users' private itemset sizes and estimating frequencies using the data from previous steps. A high-level overview is shown in Figures 3 and 4.

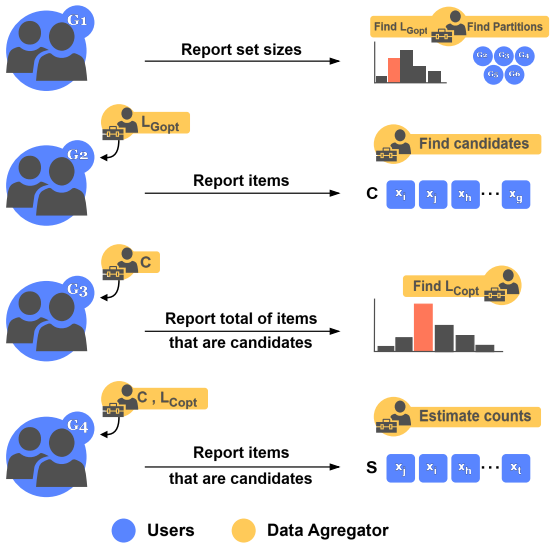

Figure 3. Flux Chart Illustrating the Steps for Item Mining Task (DHHI)

5.3.1 Acquiring Frequent Items (DHHI)

Step 1: Fetching The Size Distribution. Initially, the aggregator has no prior knowledge about the dataset beyond its domain. Using the logarithmic scaling function defined in Eq. 7, the first user group can be determined. Once G_1 is allocated, the aggregator queries it to estimate the distribution of user transaction lengths. By understanding how the number of items per transaction is distributed within G_1, the aggregator can generalize its statistical features to the entire dataset and use this privately obtained information to compute the optimal group partitions for the remainder of the steps and the optimal size limit for Step 2.

Each user in group G_1 reports the length l of their transaction v using OLH with privacy budget ϵ: $\Psi_{\text{OLH}(\epsilon)}(|v|)$. The aggregator then estimates the size distribution of the group transactions by computing $\Phi_{\text{OLH}(\epsilon)}(l)$ for all $l \in [1, \ldots, d_w]$, where $d_w = |I|/10$ is used to reduce noise while capturing a representative range of transaction lengths.

From the size distribution of G_1, the aggregator computes L_{Gopt} using Eq. 5, the optimal transaction size limit for Step 2. Lastly, the domain size $|I|$, privacy budget ϵ, standard deviation, and skewness from the distribution are used in the described regression model to predict the proportions for groups G_2, G_3, G_4, G_5, and G_6, which determine how the users are allocated at the subsequent steps.

Algorithm 1: DHHI

Input : Number of users n, privacy budget ϵ, item domain size d, target rank k
Output : Top-k frequent items S

Determine the first user group G_1:
  $G_1 \leftarrow$ Equation 7(d);
Estimate transaction size frequencies in G_1:
  $r_1 \leftarrow \Psi_{\text{OLH}(\epsilon)}(|v|) \mid v \subseteq I$;
  $f_1 \leftarrow \Phi_{\text{OLH}(\epsilon)}(l)\ \forall\, l \in [1, \ldots, d/10]$;
$L_{Gopt} \leftarrow$ Equation 5(f_1);
Using features from f_1, compute the group proportions of $G_2, \ldots, G_6$ using a regression model;
Estimate item frequencies in G_2:
  $r_2 \leftarrow \Psi_{\text{PSFO}(L_{Gopt},\text{Adap},\epsilon)}(v)$;
  $f_2 \leftarrow \Phi_{\text{PSFO}(L_{Gopt},\text{Adap},\epsilon)}(x)\ \forall\, x \in I$;
$C \leftarrow \text{Top}_{\lceil \log_{10} d \rceil k}(f)$;
Estimate frequencies of the number of candidate items in G_3 users' transactions:
  $r_3 \leftarrow \Psi_{\text{OLH}(\epsilon)}(|v \cap C|)$;
  $f_3 \leftarrow \Phi_{\text{OLH}(\epsilon)}(l)\ \forall\, l \in [1, \ldots, \lceil \log_{10} d \rceil k]$;
$L_{Copt} \leftarrow$ Equation 5(f_3);
Estimate candidate items' frequencies in group G_4 and apply the correction factor (Eq. 8):
  $r_4 \leftarrow \Psi_{\text{PSFO}(L_{Copt},\text{Adap},\epsilon)}(v \cap C)$;
  $f_4 \leftarrow u(L_{Copt}) \cdot \Phi_{\text{PSFO}(L_{Copt},\text{Adap},\epsilon)}(x)\ \forall\, x \in C$;
$S \leftarrow \text{Top}_k(f_4)$;
return S

Step 2: Finding Item Candidates. The data aggregator now shifts focus from metadata to querying item frequencies. The optimal length L_{Gopt} from Step 1 is provided to G_2, whose users report a sample from their private set v using PSFO: $\Psi_{\text{PSFO}(\text{L}_{\text{Gopt}},\text{Adap},\epsilon)}(v)$. The aggregator estimates the item frequencies by computing $\Phi_{\text{PSFO}(\text{L}_{\text{Gopt}},\text{Adap},\epsilon)}(x)$ and constructs the candidate set C of the zk most frequent items, where $z = \lceil \log_{10} |I| \rceil$ is our proposed scalar to extend the top-k range according to the datasets cardinality. This accounts for possible ranking shifts of candidates outside the top-k due to estimation noise. While C reliably identifies heavy hitters, their frequencies and rankings remain inaccurate due to the variance introduced by the large value of ℓ used in PSFO, which is addressed in the next step by deriving a new value for ℓ from the distribution of the candidate items in users' transactions.

Step 3: Item Candidate Set Size Estimation. The data aggregator now estimates frequencies solely over the zk candidates in C. Since the size distribution of candidate-containing subsets $\{|v_i \cap C| \mid i = 1, 2, \cdots, n\}$ differs from that of general transaction $\{|v_i|\}$, a new value for ℓ must be determined to optimize the reports estimation. To infer the new size distribution, each user in G_3 receives C and reports the number of candidates in their private set, i.e, $\Psi_{\text{OLH}(\epsilon)}(|v \cap C|)$. The aggregator computes $\Phi_{\text{OLH}(\epsilon)}(l)$ for all $l \in [1, 2, \ldots, zk]$ and derives $L_{Copt} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(zk)])$ for use in Step 4.

Step 4: Estimation And Adjustment. Given L_{Copt} and C, group G_4 reports items that are candidates from their private sets via $\Psi_{\text{PSFO}(\text{L}_{\text{Copt}},\text{Adap},\epsilon)}(v \cap C)$. The aggregator estimates the counts for all $x \in C$ by computing $\Phi_{\text{PSFO}(\text{L}_{\text{Copt}},\text{Adap},\epsilon)}(x)$. In this process, users whose transactions exceed L_{Copt} may

Algorithm 2: DHHS

Input : Number of users n, privacy budget ϵ, item domain size d, target rank k, Top-k items S
Output : Top-k frequent itemsets IS

$IC \leftarrow$ Equation (9)(S);
Estimate frequencies of the number of candidate itemsets in G_5 users' transactions:
  $r_5 \leftarrow \Psi_{\text{OLH}(\epsilon)}(|vs|) \mid vs = \{\mathbf{x} \mid \mathbf{x} \in IC, \mathbf{x} \subseteq v\}$;
  $f_5 \leftarrow \Phi_{\text{OLH}(\epsilon)}(l) \mid l \in [1, \ldots, \lceil \log_{10} d \rceil k]$;
$L_{ICopt} \leftarrow$ Equation 5(f_5);
Estimate candidate itemsets' frequencies in group G_6 and apply the correction factor Eq. (8):
  $r_6 \leftarrow \Psi_{\text{PSFO}(L_{ICopt},\text{Adap},\epsilon)}(vs)$;
  $f_6 \leftarrow u(L_{ICopt}) \cdot \Phi_{\text{PSFO}(L_{ICopt},\text{Adap},\epsilon)}(x)\ \forall\, x \in IC$;
$IS \leftarrow \text{Top}_k(f_6)$;
return IS

cause underestimation of the counts; to correct this, Wang *et al.* [2018] propose a correction factor assuming unreported counts follow a distribution similar to the reported ones, scaling each estimate by a factor of $u(L)$, defined as:

$$u(L) := \frac{\sum_{\ell=1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)\ell}{\sum_{\ell=1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)\ell - \sum_{\ell=L+1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)(\ell - L)} \quad (8)$$

In Eq. 8, the numerator denotes the total item count, while the denominator excludes items from sets exceeding L, accounting for those missed by PSFO. Since this correction is post-processing arithmetic relying on information from Steps 3 and 4, it raises no privacy concerns. Ultimately, the aggregator obtains S, the set of top-k items and their counts.

5.3.2 Acquiring Frequent Itemsets (DHHS)

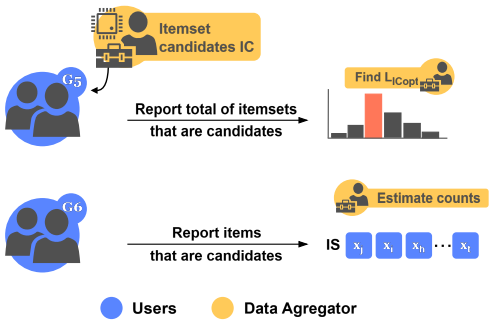

Figure 4. Flux Chart Illustrating the Steps for Itemset Mining task (DDSM)

Upon obtaining the set of frequent items and their estimated counts, the combinatorial complexity of the exponential number of potential itemsets is addressed by constructing a candidate itemset collection, denoted as IC. Let S be the k most relevant items returned by DHHI. Following the approximation strategy proposed in Wang *et al.* [2018], the frequency of an itemset $\mathbf{x}$ is estimated as $\tilde{f}_{\mathbf{x}} = \prod_{x \in \mathbf{x}} \Phi'(x)$, where $\Phi'(x) = 0.9 \frac{\Phi(x)}{\max_{x \in S} \Phi(x)}$ represents the normalized estimate of the itemset $\mathbf{x}$, and $\Phi(x)$ is the count of the item x estimated by DHHI. The factor 0.9 attenuates the contribution of the most frequent item, preventing itemsets that exclude it from being assigned unrealistically high frequencies.

$$\text{IC} := \left\{ \mathbf{x} : \mathbf{x} \subseteq S,\ 1 < |\mathbf{x}| < \log_z k,\ \prod_{x \in \mathbf{x}} \Phi'(x) \right\} \quad (9)$$

The top zk itemsets with the highest estimated counts form the candidates set IC. This is justified by the observation

that frequent itemsets often contain frequent items, whereas the converse, in which frequent itemsets are composed predominantly of infrequent items, is unlikely. Formally, IC is defined in Eq. 9 and truncated to satisfy $|IC| = zk$.

Step 5: Itemset Candidate Set Size Estimation. In this Step, G_5 receives IC and each user of the group reports the number of itemsets from their private set v that are present in IC, i.e., $\Psi_{\text{OLH}(\epsilon)}(|vs|)$, where $vs = \{\mathbf{x} \mid \mathbf{x} \in IC,\ \mathbf{x} \subseteq v\}$. The aggregator estimates the size distribution $\Phi_{\text{OLH}(\epsilon)}(l)$ for all $l \in [1, \ldots, zk]$, and finds an optimal value $L_{ICopt} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(zk)])$ for use in Step 6.

Step 6: Estimation And Adjustment. Finally, the aggregator provides L_{ICopt} and IC to group G_6, where each user uses PSFO with length limit L_{ICopt} to report itemsets from their set v belonging to IC: $\Psi_{\text{PSFO}(\text{L}_{\text{ICopt}},\text{Adap},\epsilon)}(vs)$. Frequency estimates $\Phi_{\text{PSFO}(\text{L}_{\text{ICopt}},\text{Adap},\epsilon)}(\mathbf{x}) \cdot u(L_{ICopt})$ are computed for all $\mathbf{x} \in IC$, yielding the top-k itemsets IS.

Algorithms 1 and 2 operate sequentially to privately estimate the most frequent items and itemsets. Algorithm 1 handles item mining: lines 1–2 determine the initial partition to allocate the user group G_1; lines 3–6 estimate the transaction size distribution of G_1, and computes its optimal limit size; line 7 applies the trained random forest model to predict the optimal group partitions for the remainder queries; lines 8–11 estimate the candidates; lines 12–15 estimate the candidate distribution and computes its optimal limit size; lines 16–20 estimate the candidate frequencies and returns the top-k items. The output S is then passed to Algorithm 2, where line 1 computes the candidate itemsets; lines 2–5 estimate the itemset candidate distribution, and compute its optimal limit size; lines 6–10 estimate the candidate frequencies and return the k most frequent itemsets IS. Together, the two algorithms compose the full DHHS pipeline, consuming a shared privacy budget ϵ across the six user groups.

6 Experimental Evaluation

To evaluate the performance of our new algorithm, we compare it against our previous version Oliveira *et al*. [2025] as well as its baseline Wang *et al*. [2018]. We assess both item and itemset estimates, as the accuracy of item estimates directly influences itemset estimation. In the experimental setup, the solutions are evaluated over ten independent runs on three real-world and one synthetic dataset. Our primary competitors are methods that identify both sets: top-k frequent items and itemsets. Among these, SVSM is recognized as the most important baseline in this domain. We also include DDSM due to its adaptability, particularly its ability to optimize transaction length, which improves performance in both top-k item and itemset mining tasks. SVSM, SVIM, DDIM, and DDSM are implemented following the specifications described in Wang *et al*. [2018]; Oliveira *et al*. [2025], while DHHI and DHHS follow the methodology outlined in Section 5.

Datasets. The data workloads are detailed below:

- **Netflix:** Consists of users' movie ratings, provided by Netflix for the KDD Cup 2007 Zhu [2018]. The dataset contains 480,189 users, each having up to 17,770 movie ratings. The number of ratings per user ranges from 1 to 17,653.
- **Accidents:** Contains traffic accident data from the National Institute of Statistics for Flanders (Belgium) Geurts *et al*. [2003], covering the period from 1991 to 2000. Composed of 340,183 records and 468 unique types of accidents, each record can have up to 51 accidents and as few as 18.
- **SharkFin:** A dataset made to simulate scenarios where transaction sizes are concentrated away from the unit size. It contains 515,596 users and a domain of 1,306 items Oliveira [2026]. The number of user-set items can range from 1 to 165.
- **Groceries:** Available at Nasrullah [2019]; Jha [2019], this dataset contains 9,835 transactions from customers' shopping lists, with 169 unique grocery items, with a sparse transaction size distribution where various transaction lengths within the domain are not observed.

As data partitioning across queries is the main contribution of this work, evaluating the proposed algorithm across multiple and distinct partitioning scenarios is imperative. The evaluation here is presented to independently reflect the experimental scope and contributions of this work. The datasets were selected to provide complementary statistical profiles, covering large and small populations, high and low cardinality, high and low sparsity data, exponential, positively skewed, and normal distributions. Although prior work includes several datasets also used here, certain scenarios remain underexplored, particularly restrictive settings in which the aggregator must identify frequent patterns in highly sparse, low-volume data. The Groceries dataset is included to evaluate performance under this condition, while Kosarak was excluded due to substantial overlap with characteristics already represented in the selected dataset pool.

Metric. To evaluate the identification of relevant items, we measure both the number of correctly identified items and their relative ranking. For this, we use Normalized Discounted Cumulative Gain (NDCG), a widely used metric that captures relevance and ranking accuracy. NDCG compares the estimated ranking to an ideal ranking in which all relevant items are present and ordered by their true importance. Scores are normalized between 0 and 1, with higher values indicating better performance. Given a ranked list of k items with scores $\{rel_1, rel_2, \ldots, rel_k\}$, NDCG is computed in Eq. 10 as the ratio between the Discounted Cumulative Gain (DCG) of the estimated ranking and the Ideal DCG (IDCG), where items are ideally sorted by descending relevance rel_i.

$$\textbf{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k}, \qquad \textbf{DCG}_k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)} \quad (10)$$

7 Results

We now present the findings from our experimental analysis. To evaluate the performance of each solution, we assess them in two main benchmarks: their effectiveness in identifying and ranking items and itemsets across varying levels of privacy strictness. This evaluation will provide insight into the impact of introduced noise in each solution and how well

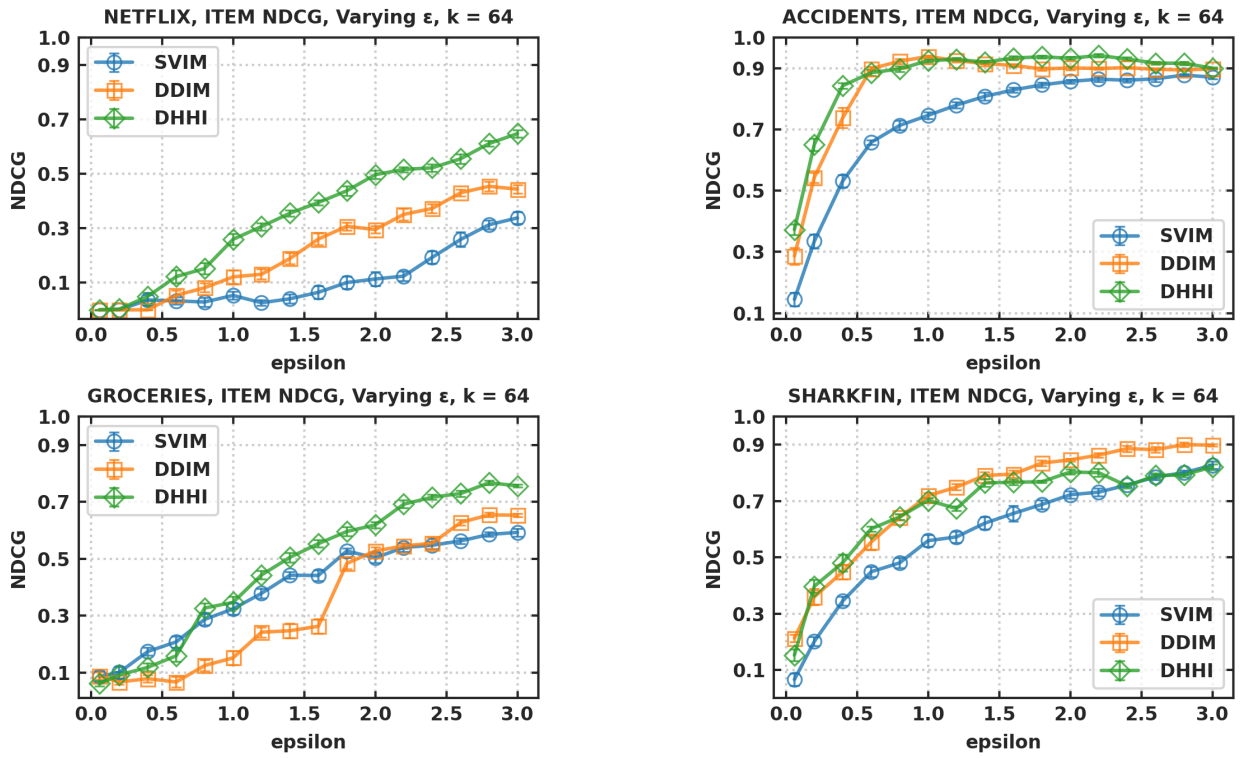

Figure 5. Item mining evaluation with $k = 64$ varying the privacy budget ϵ.

they estimate the top-k items and itemsets for each privacy budget.

7.1 Item Mining Results

Generally, as the privacy budget ϵ increases, utility tends to improve, given that less noise is added and the data becomes less privately assured. This effect can be seen in Figure 5, which shown the top-k item estimation NDCG.

However, each method responds differently to the noise introduced by LDP. On both the Netflix and Groceries datasets, DHHI exhibits a substantial increase in utility, measured by NDCG on the y-axis, as the privacy budget increases. Notably, on the Netflix dataset, DHHI achieves NDCG values close to 0.7, whereas DDIM and SVIM struggle to exceed 0.50 even at the most relaxed privacy budgets. Moreover, DHHI maintains an advantage of up to 0.40 NDCG over SVIM and up to 0.20 over DDIM for some values of ϵ. On the Groceries dataset, DHHI consistently outperforms SVIM across most evaluated privacy budgets, remaining more than 0.20 NDCG points above DDIM under strong privacy guarantees and achieving NDCG values above 0.70 for larger values of ϵ, a level of performance that DDIM did not achieve on this dataset.

By contrast, DHHI achieves performance comparable to that of DDIM on the Accidents and Sharkfin datasets. In the former, we observe only marginal gains in utility for certain privacy budgets, with no substantial improvement overall. In the latter, our method exhibits slight performance degradation for some privacy budgets when compared to our previous work, DDIM; however, it remains competitive with the SVIM baseline across all evaluated settings.

In the item mining task, whilst for some datasets DHHI presented results similar to or worse than DDIM, its overall performance remains robust across various settings. This suggests that the proposed group partitioning strategy serves the output's utility differently across data distributions. In particular, both the Netflix and Groceries datasets exhibit light-tailed, highly skewed distributions. Such a distributional imbalance likely hindered our previous fixed partitioning approach's ability to accurately estimate group counts.

For the transactional datasets considered in this problem, high standard deviation indicates heterogeneous transaction behaviors, making certain partitioning ratios more stable than others. Skewness, in contrast, captures the asymmetry of the transaction-size distribution. Highly skewed datasets often exhibit heavy or elongated tails, which are particularly sensitive to how users are partitioned into groups.

Our new approach accounts for these nuances by allocating the group partitions in a data-driven manner. When necessary, more data is allocated to candidate estimation steps that benefit from improved sampling balance. The Netflix dataset, in particular, benefits significantly from this strategy due to its large domain size, which leads to more heterogeneous transactions and greater imbalance during sampling.

These performance gains observed in the item mining task are significant, as in Apriori-based algorithms, the quality of item mining estimates directly impacts subsequent itemset mining performance. This relationship holds for all three algorithms analyzed, as they are all Apriori-based.

7.2 Itemset Mining Results

At the itemset mining task, illustrated in Figure 6, the impact of the preceding steps on performance becomes evident, as frequent itemset estimation heavily relies on the construction of candidate itemsets IC during Step 5 of DHHS. Consequently, Apriori-based techniques with poor performance in the item mining task are likely to exhibit poor performance in the itemset mining task, while strong item estimation per-

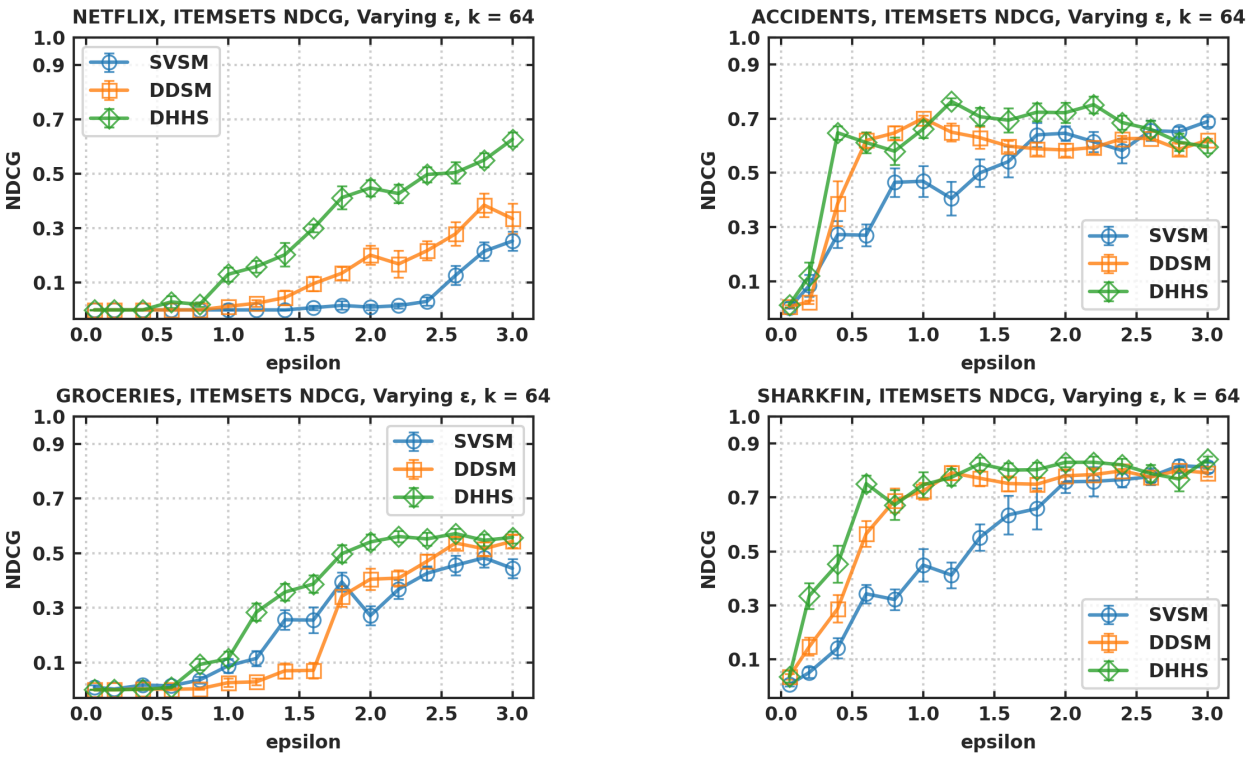

Figure 6. Itemset mining evaluation with $k = 64$ varying the privacy budget ϵ.

formance leads to strong itemset estimation performance.

Analyzing the results, DHHS performs notably well under strict privacy constraints, i.e., at lower values of ϵ. Specifically, on the Accidents dataset, it achieves a score of 0.65 at $\epsilon = 0.40$, whereas DDSM reaches the same performance at $\epsilon = 0.80$, and SVSM only attains a comparable result at $\epsilon = 2.0$. Similarly, on the SharkFin dataset, DHHS achieves an NDCG of 0.75 at $\epsilon = 0.60$, whilst for the same privacy budget DDSM scores 0.57 and SVSM scores 0.34.

These results show that even for datasets that did not present substantial improvements in the item mining task, our approach still achieved noticeably better performance than competitors in the itemset mining task. For the Accidents dataset, our approach leveraged the dataset's large minimum transaction length of 18 items per transaction and its features to optimize the utility of each differentially private query made by the aggregator. By computing an optimal ℓ for PSFO using Eq. 5 and predicting the group partitions with a trained model, DHHS enables more effective data estimation tailored to the underlying dataset profile.

Moreover, the results from the SharkFin dataset show that our technique performs especially well in scenarios where the smallest transaction size is one, but most transactions occur at larger sizes. In this context, DHHS and DDSM, unlike SVSM, do not use the unit size ℓ to estimate candidates with PSFO, and consistently outperform SVSM across most tested privacy budgets. This shows that our optimization function, along with the data-driven approach, adapts well to skewed transaction length distributions by dynamically computing optimal parameters through a distribution-aware optimization process, effectively avoiding the pitfalls of static parameter choices.

Results on the Netflix and Groceries datasets demonstrate that the proposed approach overcomes one of the main limitations present in DDSM and SVSM. This challenge stems from data profiles with high dimensionality, skewed distributions, and large variance. These characteristics contribute to sampling error and undermine the PSFO's ability to reliably estimate itemset patterns in the group partitions. As the domain grows, identifying frequent patterns becomes increasingly complex. This effect is captured by the FO variance expressions, which increase with domain and dataset sizes, as shown in Eqs. 3 and 4. Since PSFO relies on these FOs to generate reports, its accuracy is affected by the growth in this variance. The same phenomenon occurs when estimating groups with sparsely distributed patterns, as sampling error further reduces estimation accuracy.

DHHS shows substantial improvements over DDSM on both groceries and Netflix datasets. On Netflix, DHHS leads DDSM by more than 0.20 NDCG points across most privacy budgets, achieving a score of 0.62 at the most relaxed privacy budget of $\epsilon = 3.0$, whilst DDSM and SVSM remain around 0.30. On the Groceries dataset, DHHS consistently outperforms SVSM, whereas DDSM only surpasses it at $\epsilon = 2.0$.

Our data-driven group partitioning approach demonstrates that, for datasets of different profiles, analyzing dataset features and the impact of sampling error on each query enables reliable allocation of the user pool to improve the aggregator's estimates.

8 Conclusion

In comparison to prior work, DHHS demonstrates clear advantages in estimating patterns in high-dimensional domains, skewed transaction size distributions, and large variance. While DDSM introduced data-dependent optimization of oracle parameters, it continued to rely on a fixed user partitioning strategy, which, as shown in Section 7, limits its ability to perform well in datasets where sampling variance is uneven across the querying steps. Our work addresses this limitation

directly by modeling the relationship between dataset characteristics and the optimal group partition, enabling a more principled allocation of the user pool across the aggregator's queries. Against SVSM, the improvements are more pronounced and consistent. SVSM's reliance on static parameters across its methodology makes its performance fragile to shifts in data properties.

Nevertheless, the proposed approach carries its own limitations. The usage of a dedicated user group for metadata collection introduces its own trade-off, where the privacy budget and a fraction of the user pool are now consumed before any item estimation begins, which may reduce utility in very small datasets. Future work could investigate strategies to reduce this overhead in low-population settings.

Datasets differ in many ways, and this work demonstrates that privately modeling the relationship between dataset characteristics, privacy budget, and estimation utility enables data-driven user data allocation under LDP.

Acknowledgements

This work was partially supported by Lenovo, as part of its R&D investment under Brazil's Informatics Law, and by the LSBD/UFC.

References

Chan, T. H. H., Li, M., Shi, E., and Xu, W. (2012). Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies: 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings 12*, pages 140–159. Springer.

Chen, Z. and Wang, J. (2022). Ldp-fpminer: Fp-tree based frequent itemset mining with local differential privacy. *arXiv preprint arXiv:2209.01333*.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg.

Erlingsson, U., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA. Association for Computing Machinery.

Filho, J. S. C. and Machado, J. C. (2023). Felip: A local differentially private approach to frequency estimation on multidimensional datasets. In *International Conference on Extending Database Technology*.

Frenkel, S. and Isaac, M. (2018). Facebook security breach exposes accounts of 50 million users. `https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html`. Accessed: 2024-01-12.

Geurts, K., Wets, G., Brijs, T., and Vanhoof, K. (2003). Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16*, page 18pp.

Jha, S. (2019). Association rule mining apriori – data repository. `https://github.com/shubhamjha97/association-rule-mining-apriori/tree/master/data`. Accessed: 2025-06-12.

K, M. V., Immanuel Johnraja, J., Jeba Leelipushpam, G., Jebaveerasingh Jebadurai, J., and Santhosam, I. B. (2021). Security and privacy issues in the internet of things – a survey. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*.

Kairouz, P., Bonawitz, K., and Ramage, D. (2016). Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, pages 2436–2444. PMLR.

Li, Y., Huang, C., Cheng, M., Lv, T., Zhao, Y., Sun, Y., and Yuan, Y. (2025). Privminer: a similar-first approach to frequent itemset mining under local differential privacy. *World Wide Web*, 28(2):25.

Lin, F., Muzumdar, K., Laptev, N. P., Curelea, M.-V., Lee, S., and Sankar, S. (2020). Fast dimensional analysis for root cause investigation in a large-scale service environment. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(2).

Nasrullah, I. (2019). Groceries market basket dataset. `https://www.kaggle.com/datasets/irfanasrullah/groceries`. Kaggle dataset. Accessed: 2025-06-12.

Oliveira, R. (2026). Sharkfin. `https://github.com/renanlsbd/SharkFin`. Accessed: 2026-05-08.

Oliveira, R., Costa, J., Monteiro, J. M., and Machado, J. (2025). Data dependent itemset mining under local differential privacy. In *Anais do XL Simpósio Brasileiro de Bancos de Dados*, pages 521–534, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbbd.2025.247281.

Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., and Ren, K. (2016). Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 192–203, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2976749.2978409.

Satariano, A., Perlroth, N., and Tsang, A. (2018). Marriott hacking exposes data of up to 500 million guests. Accessed: 2024-01-12.

Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745.

Wang, T., Li, N., and Jha, S. (2018). Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 127–143. DOI: 10.1109/SP.2018.00035.

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

Zhu, E. (2018). Set similarity search benchmarks. `https://github.com/ekzhu/set-similarity-search-benchmarks/tree/master`. Accessed: 2025-04-06.

Zhu, T., Li, G., Zhou, W., and Yu, P. S. (2017). Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1619–1638.