Responses to Review Comments

We are very thankful to the reviewers for their constructive comments, based on which we have revised the manuscript significantly. Our main changes are summarized below.

Meta Reviews

R1: Include experimental comparison with prior LDP-based item mining mechansms (Reviewer 1: O1), e.g., label-DP (Reviewer 3: O1).
R2: Provide more details on the rationale behind the choices of key parameters (Reviewer 1: O2).
R3: Provide a comparison with deep learning-based approaches for item mining (Reviewer 1: O3).
R4: Provide an analysis of time/memory complexity of the solution (Reviewer 2: O1).
R5: Provide additional experimental results around variance analysis (Reviewer 2: O2).

Response: Our main changes are summarized as below.

(1) We have addressed R1.O2 by explaining the rationale for the choices of key parameters, including the privacy budget allocation in the first paragraph of Section VII-F, and the selections of parameters a and b in the second paragraph of Section VII-F.
(2) We have addressed R2.O2 by providing the theoretical and experimental analysis of variance in the second paragraph of Section V-C and Section VII-C, respectively.
(3) We have addressed R2.O1 by providing the complexity analysis in the second paragraph of Section VI-A and the fourth paragraph in Section VI-B, respectively.
(4) We have addressed R1.O1 by modifying our experiments in Section VII to emphasize that our mechanisms are compared with state-of-the-art mechanisms.
(5) We have responded to R1.O3 and R3.O1 by explaining the infeasibility of comparing our approach with deep learning-based methods and Label-DP.

In what follows, we describe in detail the changes we have made in response to each comment in the order of occurrences. For ease of reference, we have attached the updated manuscript with highlighted changes at the end of this response letter. In the interest of space, the proofs are included in our technical report [23].

Reviewer 1

R1.O1: *The experiments lack a comparison with existing LDP-based item mining mechanisms, making it difficult to evaluate the relative performance of the proposed approach. A direct comparison with other state-of-the-art methods would strengthen the paper.*

Response: Thank you for the suggestion. We apologize for the vague description, as we mentioned "state-of-the-art (SOTA)" only in the methodology sections but omitted an explanation in the experiments. In fact, our manuscript includes a comparison between SOTA LDP mechanisms and our proposed mechanisms.

The existing SOTA LDP mechanisms are designed for item mining problems. The **adaptive mechanism** [9], which adaptively runs OUE or GRR, represents the SOTA approach for frequency estimation, while the **prefix-extending mechanism (PEM)** [24] is the SOTA method for top-k item mining.

Notably, there is no existing work for the problem of multi-class item mining in LDP. Therefore, we design our framework, coupled with extended SOTA LDP mechanisms, to tackle this problem. Specifically, we compare the extended mechanisms (HEC, PTJ, and PTS) with our improved mechanism — PTS-CP, PTJ-Shuffling+VP, and PTS-Shuffling+VP+CP — which leverage validity perturbation (VP) and correlated perturbation (CP) as substitutes for the SOTA mechanisms.

R1.O2: *The paper does not provide a sufficient explanation for the choices of key parameters, such as the privacy budget allocation and values for parameters a and b. More details on the rationale behind these choices would help readers better understand their impact on the results.*

Response: Thank you for your comment. The rationale behind the parameter settings is as follows.

(1) Privacy budget allocation. In our paper, the privacy budget ϵ is divided into two parts: ϵ_1 and ϵ_2. The first part, ϵ_1, is used for label perturbation, while ϵ_2 is allocated for item perturbation. As both item and label perturbation affect the aggregation of class-specific items, the perturbations for both are crucial. To investigate the impact of privacy budget allocation, we conduct experiments on the synthetic dataset SNY4 with 5, 10, and 20 classes, respectively. The proportion of ϵ_1, denoted by p, varies from 0.1 to 0.9. The results are shown in Figs. 1, 2, and 3. We observe that the results increase and then decrease with p, which is consistent with our analysis above. The best p lies between 0.4 and 0.6, and does not influence the results significantly. Therefore, we empirically set the proportion to 0.5 in our paper, i.e., $\epsilon_1 = \epsilon_2 = \epsilon/2$. We have now added Section VII-F in the experiment for a clear explanation of privacy budget allocation.

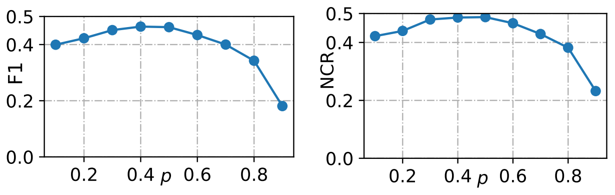


Fig. 1 Results for 5 Classes

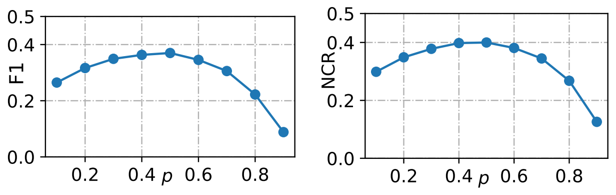


Fig. 2 Results for 10 Classes

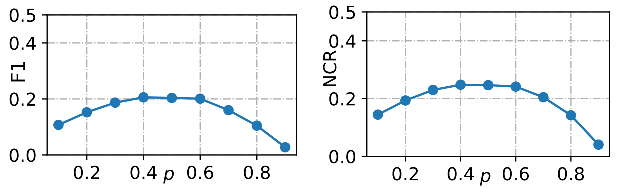


Fig. 3 Results for 20 Classes

(2) Parameter a. Parameter a (Algorithm 1 Line 2) controls the sampled group size used to identify item candidates, including the globally frequent items during the initial iterations. Given a large value of a, there may be insufficient data for top-k item mining, while for a small value of a, it may fail to generate reasonable candidates. To figure out the impact of parameter a, we conducted experiments using varying values of a on the real-world datasets JD and Anime. The results are shown in Figs. 4 and 5. Clearly, the impact of parameter a depends on the distribution of the dataset. For simplicity, we chose $a = 0.2$ in our experiments, meaning that one-fifth of the data is sampled to generate the global candidates, while the remaining data is used for top-k item mining. To clarify the setting of parameter a, we have updated the description in Section VII-D and added an experiment for explanation in Section VII-F.

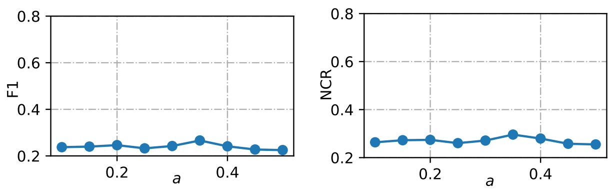


Fig. 4 Anime Dataset

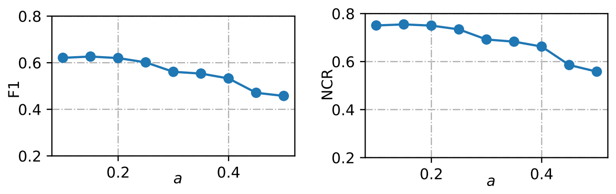


Fig. 5 JD Dataset

(3) Parameter b. The amount of data within different classes varies a lot. After applying DP perturbations, classes with less data are injected with a significant amount of noise (i.e., items) from other classes. To estimate the noise level for each class, we use the labels perturbed during the global candidates aggregation. If the injected noise is too large (or specifically, the collected data amount exceeds b times the estimated class data amount), the noise level becomes excessive. In this case, the amount of valid data may be insufficient for accurate estimation, rendering the correlated perturbation mechanism infeasible. To study the impact of parameter b, we conducted experiments with varying b on the real-world datasets JD and Anime. The results are presented in Figs. 6 and 7. Although the results are dependent on the dataset, they do not fluctuate significantly. In our experiments, we chose $b = 2$ as the default value. Additionally, we have updated Section VII-E in the revised manuscript for clarity, and added an experiment in Section VII-F.

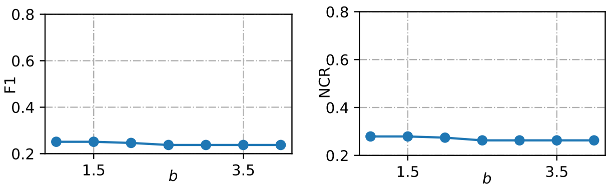


Fig. 6 Anime Dataset

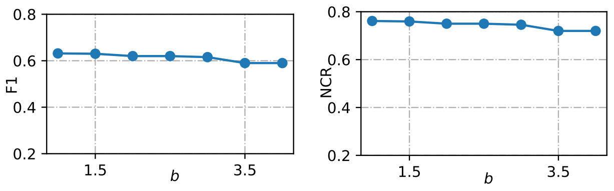


Fig. 7 JD Dataset

R1.O3: *The paper does not provide a theoretical or experimental comparison with deep learning-based approaches for item mining, which could be an important benchmark for assessing the effectiveness of the proposed methods.*

Response: Thanks for your suggestion. We would like to clarify that, because of the following two reasons, a direct comparison with deep learning approaches is unfortunately infeasible. Firstly, our work tackles a new problem without any existing baselines. The problem of differentially private multi-class item mining has not been systematically studied in prior literature, let alone deep learning approaches. Secondly, while DP-based deep learning is a paradigm for learning tasks, these methods address distinct tasks (e.g., model training via DP-SGD for prediction) rather than multi-class item mining — they focus on feature representations rather than identifying multi-class item statistics.

Therefore, we conducted comparisons across three multi-class item mining frameworks (i.e., HEC, PTS, and PTJ in Section III) integrated with LDP mechanisms. For multi-class frequency estimation, we incorporated the SOTA mechanism, adaptive mechanism [9], as a baseline within these frameworks. To evaluate performance, we employed Root Mean Square Error (RMSE) as the primary metric. Our improved correlated perturbation mechanism demonstrates superior performance compared to this baseline, particularly in addressing the critical challenge of invalid perturbed labels. Regarding multi-class top-k item mining, we developed a novel mining scheme that significantly enhances the SOTA mechanism, PEM [24], making it more compatible with multi-class top-k item mining scenarios.

Reviewer 2

R2.O1: *The paper understates critical scalability challenges. There was no analysis of time/memory complexity of the solution and its practicality (a very short communication cost paragraph). In fact, communication costs scale poorly with domain size, limiting practicality.*

Response: Thank you for the comment. We have now added the time and space complexity to Sections VI-A and VI-B in the revised manuscript. As for the communication cost, it is because we integrate the optimized unary encoding (OUE) mechanism into the multi-class item mining framework. OUE is a state-of-the-art LDP mechanism for frequency estimation and has been widely used in previous research [20]-[22]. While its communication cost $\mathcal{O}(d)$ (d is the domain size) increases with domain size, it benefits from the enhanced accuracy, especially for a large domain.

(1) Complexity analysis for multi-class frequency estimation. Given the class domain size c, item domain size d, data amount N, we use the GRR mechanism for label perturbation and the OUE mechanism for item perturbation [9]. The time complexity is analyzed as follows:

- **HEC.** We analyze the time complexity of HEC from both the user side and the server side. On the user side, the mechanism begins by randomly selecting a class for a user, which takes $\mathcal{O}(1)$ time. If the selected class does not match the user's actual label, the user then randomly selects an item, which also takes $\mathcal{O}(1)$ time. Afterward, the item is perturbed using the OUE mechanism, which requires $\mathcal{O}(d)$ time for each bit flip. Therefore, the overall time complexity of the mechanism on the user side is $\mathcal{O}(d)$, and the time complexity on the server side is $\mathcal{O}(Nd)$ for statistic aggregation.
- **PTJ.** The perturbation domain consists of $c \cdot d$ elements. The mechanism requires $\mathcal{O}(cd)$ time for each bit flip on the user side, and the time complexity on the server side is $\mathcal{O}(Ncd)$.
- **PTS.** On the user side, the mechanism first perturbs the label using GRR, which takes $\mathcal{O}(1)$ time. And then, the item will be perturbed using OUE with d bits flipping, which takes $\mathcal{O}(d)$ time. Therefore, the overall time complexity of the mechanism on the user side is $\mathcal{O}(d)$, and the time complexity on the server side is $\mathcal{O}(Nd)$.
- **PTS-CP.** The mechanism is similar to PTS but with an extra validity flag. Therefore, the overall time complexity of the mechanism on the user side is $\mathcal{O}(d)$, and the time complexity on the server side is $\mathcal{O}(Nd)$.

The space complexity is analyzed as follows:

- **HEC.** On the user side, the mechanism begins by randomly selecting a class for a user, which requires $\mathcal{O}(1)$ space. If the selected class does not match the user's actual label, the user then randomly selects an item, which also takes $\mathcal{O}(1)$ space. Afterward, the item is perturbed using the OUE mechanism, which requires $\mathcal{O}(d)$ space to store perturbed bits. Therefore, the overall space complexity of the mechanism on the user side is $\mathcal{O}(d)$, while the server side requires $\mathcal{O}(cd)$ space.
- **PTJ.** The perturbation domain consists of $c \cdot d$ elements, where c represents the class number and d is the item number. The mechanism requires $\mathcal{O}(cd)$ space for element storage on both the user and server sides.
- **PTS.** The mechanism first perturbs the label using GRR, which takes $\mathcal{O}(1)$ space. And then, the item will be perturbed using OUE with d bits flipping, which takes $\mathcal{O}(d)$ space. Therefore, the overall space complexity of the mechanism on the user side is $\mathcal{O}(d)$, while it is $\mathcal{O}(cd)$ on the server side.
- **PTS-CP.** The mechanism is similar to PTS but with an extra validity flag. Therefore, the overall space complexity of the mechanism on the user side is $\mathcal{O}(d)$, while the server side requires $\mathcal{O}(cd)$ space.

In summary, given the class domain size c, item domain size d, and data amount N, we consider the GRR and OUE as the LDP mechanisms [9]. The communication cost for each user of HEC, PTS, and PTS-CP is $\mathcal{O}(d)$; while for PTJ, it is $\mathcal{O}(cd)$ with an enlarged perturbation domain including combinations of both items and labels. The time complexities of the HEC, PTS, and PTS-CP mechanisms are $\mathcal{O}(d)$ on the user side, and $\mathcal{O}(Nd)$ on the server side. While the time complexity of PTJ is $\mathcal{O}(cd)$ on the user side and $\mathcal{O}(Ncd)$ on the server side. As for the space complexity, the HEC, PTS, and PTS-CP require $\mathcal{O}(d)$ space on the user, while the server side requires $\mathcal{O}(cd)$ space. On the other hand, the PTJ requires $\mathcal{O}(cd)$ space on both the user and server sides.

(2) Complexity analysis for multi-class top-k item mining. Given the class domain size c, item domain size d, and data amount N, we use the GRR mechanism for label perturbation and the OUE mechanism for item perturbation [9]. The mining scheme for the fundamental frameworks is PEM, and the extending prefix length for each iteration is m [24].

The time complexity is analyzed as follows:

- **HEC.** We analyze the time complexity of HEC from both the user side and the server side. On the user side, each user is first assigned to a class for aggregation. After receiving the prefix sent by the server, the user checks her label. If the user's label does not match the assigned class, she randomly selects a prefix for perturbation. Finally, the user perturbs the prefix according to the held item. In terms of the server side with $\mathcal{O}(\frac{\log d}{m})$ iterations, the server generates the top-k prefixes from the $2^m \cdot 2k$ candidates for each class, and runs $2^m \cdot 2k \cdot \mathcal{O}(N\frac{m}{\log d})$ statistic aggregation during each iteration. Specifically, the candidate prefixes from last iteration will extend m bits, and finally the top-k frequent prefixes are retained for next iteration. Therefore, the overall time complexity on the user side is $\mathcal{O}(2^m k)$ for bit flips. The time complexity on the server side for this process is $\mathcal{O}(2^m k[c(m+\log k)\frac{\log d}{m} + N])$.
- **PTJ.** Since the new perturbation domain is the Cartesian product of labels and items, the remaining top amount is set to $c \cdot k$ to ensure coverage of the top-k items for each class. On the server side, during each iteration, the server extends $2^m \cdot 2ck$ candidates and thus requires $\mathcal{O}(2^m ckN\frac{m}{\log cd})$ time to aggregate statistics. Finally, the top-ck candidates are retained for next iteration. The time complexity on the server side is $\mathcal{O}(2^m ck[(m + \log ck)\frac{\log cd}{m} + N])$. On the user side, a user perturbs the prefix based on their label and item, resulting in a time complexity $\mathcal{O}(2^m ck)$.
- **PTJ with optimized methods.** On the server side, the server extends $4 \cdot ck$ candidates, and finds the top-ck prefixes during each iteration. The time complexity of the server is $\mathcal{O}(ck(\log ck \log \frac{d}{k} + N))$. And the time complexity of a user is $\mathcal{O}(ck)$.
- **PTS.** On the server side, the server extends $2^m \cdot 2k$ candidates for each class and finds the top-k prefixes during each iteration. The time complexity of the server is $\mathcal{O}(2^m k[c(m+\log k)\frac{\log d}{m} + N])$. On the user side, each user perturbs the prefixes, resulting in the time complexity $\mathcal{O}(2^m k)$.
- **PTS with optimized methods.** Under the framework of PTS, a user may be assigned to either global candidate generation or top-k item mining. For global candidate generation, the server extends $4 \cdot c \cdot k$ candidates. While for top-k item mining, the server extends $4 \cdot k$ candidates. Therefore, the time complexity for a user on the global candidate generation is $\mathcal{O}(ck)$, while it is $\mathcal{O}(k)$ for top-k item mining. On the server side, the time complexity for global candidate generation is $\mathcal{O}(ck(\log ck \log \frac{d}{k} + N))$, while for top-$k$ item mining is $\mathcal{O}(k(c \log k \log \frac{d}{k} + N))$.

The space complexity is analyzed as follows:

- **HEC and PTS.** As described in the time complexity analysis, the space complexity for a user is $\mathcal{O}(2^m k \log d)$ to store the prefixes, while for the server, it is $\mathcal{O}(2^m ck \log d)$.
- **PTJ.** The space complexity for a user is $\mathcal{O}(2^m ck \log cd)$, while for the server, it is $\mathcal{O}(2^m ck \log cd)$.
- **PTJ with optimized methods.** The space complexities on the user and server sides are both $\mathcal{O}(cd)$ to store the permuted items.
- **PTS with optimized methods.** Since the mechanism needs to store the permutation of the items, the space complexity the user side is $\mathcal{O}(d)$, while for the server side is $\mathcal{O}(cd)$.

Given the class domain size c, item domain size d, and data amount N, we consider GRR and OUE as the LDP mechanisms [9], and communication cost is measured for each user. The complexity analysis is summarized in Table 1.

Table 1 Complexity Analysis

Mechanism	Communication	Time	Space
HEC	$\mathcal{O}(2^m k \log d)$	$\mathcal{O}(2^m k)$	$\mathcal{O}(2^m k \log d)$
PTS		$\mathcal{O}(2^m k[c(m+\log k)\frac{\log d}{m}+N])$	$\mathcal{O}(2^m ck \log d)$
PTJ	$\mathcal{O}(2^m ck \log cd)$	$\mathcal{O}(2^m ck)$	$\mathcal{O}(2^m ck \log cd)$
		$\mathcal{O}(2^m ck[(m+\log ck)\frac{\log cd}{m}+N])$	$\mathcal{O}(2^m ck \log cd)$
PTJ*	$\mathcal{O}(ck)$	$\mathcal{O}(ck)$	$\mathcal{O}(cd)$
		$\mathcal{O}(ck(\log ck \log \frac{d}{k}+N))$	$\mathcal{O}(cd)$
PTS*	$\mathcal{O}(ck)$	$\mathcal{O}(ck)$	$\mathcal{O}(d)$
		$\mathcal{O}(ck(\log ck \log \frac{d}{k}+N))$	$\mathcal{O}(cd)$

* represent the optimized methods, with the first line of each row showing the user-side results and the second line showing the server-side results.

R2.O2: *The paper presents variance analysis in Section V.C with Theorem 8, which derives the variance of estimated frequency $\hat{f}(C, I)$ for correlated perturbation However, the analysis only considers ideal conditions without examining how variance behaves under:*

i) Different privacy budget splits (beyond the simple $\epsilon_1 = \epsilon_2 = \epsilon/2$) (What is the motivation behind this?),
ii) Varying correlation strengths between labels and items, and
iii) Different class distributions.

The variance comparison with GRR+OUE in Section V.C only shows correlated perturbation has smaller variance but doesn't quantify how much smaller

Response: Thank you for your comments. We have added additional sections in the experimental part to elaborate on the privacy budget allocation and present the experimental results for variance analysis.

i) Privacy Budget Allocation. In our paper, the privacy budget ϵ is divided into two parts: ϵ_1 and ϵ_2. The first part, ϵ_1, is used for label perturbation, while ϵ_2 is allocated for item perturbation. As both item and label perturbation affects the aggregation of class-specific items, the perturbations for the both are crucial. To investigate the impact of privacy budget allocation, we conduct experiments on the synthetic dataset SNY4 with 5, 10, and 20 classes, respectively. The proportion of ϵ_1, denoted by p, varies from 0.1 to 0.9. The results are shown in Figs. 1, 2, and 3, as above. We observe that the results increase and then decrease with p, which is consistent with our analysis above. The best p lies between 0.4 and 0.6, and does not influence the results significantly. Therefore, we empirically set the proportion to 0.5 in our paper, i.e., $\epsilon_1 = \epsilon_2 = \epsilon/2$. We have now added Section VII-F in the experiment for a clear explanation on privacy budget allocation.

ii) Varying correlation strengths between labels and items. We have added the second paragraph in Section V-C to theoretically analyze the variance under varying correlation strength between labels and items. Given the perturbation probabilities of LDP mechanisms, p_1, p_2, q_1, and q_2, the class amount n, and the data amount N, the variance of the estimated frequency $\hat{f}(C, I)$ is

$$\begin{aligned} Var[\hat{f}(C,I)] = & \frac{f(C,I)p_1(1-q_2)p_2\left[1-p_1(1-q_2)p_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2} \\ & + \frac{(n-f(C,I))p_1(1-q_2)q_2\left[1-p_1(1-q_2)q_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2} \\ & + \frac{(N-n)q_1(1-p_2)q_2\left[1-q_1(1-p_2)q_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2} \\ & + \left[\frac{q_2\left[p_1(1-q_2)-q_1(1-p_2)\right]}{p_1(1-q_2)(p_2-q_2)}\right]^2 \\ & \times \frac{n\left[p_1(1-p_1)-q_1(1-q_1)\right]+Nq_1(1-q_1)}{(p_1-q_1)^2}. \end{aligned} \tag{1}$$

To measure the correlation strength between items and labels, we use pointwise mutual information (PMI) [28] as the metric. Given the marginal probabilities of label C and item I, i.e., $p(C)$ and $p(I)$, the joint probability of the label-item pair $p(C, I)$, PMI is denoted as $\mathrm{PMI(C;I)} = \log_2 \frac{p(\mathrm{C,I})}{p(\mathrm{C})p(\mathrm{I})}$. A larger PMI indicates a stronger correlation between the label and the item. When $p(C)$ and $p(I)$ are fixed, we have $\mathrm{PMI(C;I)} \propto \mathrm{f(C,I)}$. Since the variance in Eq. (1) is proportional to $f(C, I)$, it follows that $Var[\hat{f}(C, I)] \propto \mathrm{PMI(C;I)}$. Note that the variance equation exhibits complex structure. We numerically estimate the variable coefficients, which depend on the dataset, under varying epsilon values to analyze variance dynamics. The coefficients are in Table 2. Crucially, $f(C, I)$ is always much smaller than class amount n and data amount N, so that the coefficients of $f(C, I)$ in Table 2 cannot offset orders of magnitude differences. This magnitude disparity explains why correlation variations are concealed in variance analysis.

Table 2 Coefficients of variables in $Var(\hat{f}(C, I))$

ϵ	0.5	1	1.5	2	2.5	3	3.5	4
$f(C, I)$	87.4	32.9	17.1	10.3	6.8	4.9	3.7	2.9
n	213.8	58.9	22.8	10.5	5.4	3.0	1.8	1.1
N	441.8	53.3	12.0	3.6	1.3	0.5	0.2	0.1

In addition to the theoretical analysis, we also include an experimental analysis in Section VII-C. To control n and N, we generate the synthetic dataset SYN1 with 4 classes and 4 items, and vary the label-item pair amounts at 1,000, 10,000, 100,000, and 1,000,000. We fix item amount and class amount ($f(I) = n = 1.111 \times 10^6$), and vary label-item frequencies ($f(C, I) \in \{10^3, 10^4, 10^5, 10^6\}$) in each class. The variance is computed as $Var(\hat{f}(C, I)) = \frac{1}{t}(\hat{f}(C, I) - f(C, I))^2$ [9] with $\epsilon = 1$ and experimental time $t = 1000$. And correlation strength PMI [28] is calculated according to $f(C, I)$ within one class. PTS refers to the framework PTS with GRR and OUE, while PTS-CP denotes the PTS framework with our improved mechanism, correlated perturbation (CP). As shown in Fig. 8, despite the increase in PMI, the observed variance exhibits negligible variation, empirically confirming that changes in correlation strength are concealed in variance due to the dominance of class amount n and data amount N.

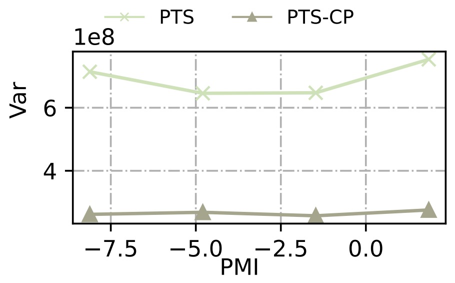


Fig. 8 Varying correlation strength PMI.

iii) Different class distributions. We have added the second paragraph in Section V-C to analyze the variance under different class distribution. According to Eq. (1), the variance is proportional to class amount n, i.e., $Var(\hat{f}(C, I)) \propto n$. Based on the coefficients derived in Table 2, an increase in class amount directly amplifies variance. This theoretical prediction is empirically validated through experiments.

Similarly to SYN1, we generate the SYN2 dataset with fixed label-item frequency $f(C, I) = 10^4$ for an item across the classes, and varying class amount $n \in \{1.3 \times 10^4, 2.11 \times 10^5, 1.21 \times 10^6, 3.01 \times 10^6\}$. As shown in Fig. 9, the results reveal a strong positive correlation between class amount n and variance magnitude.

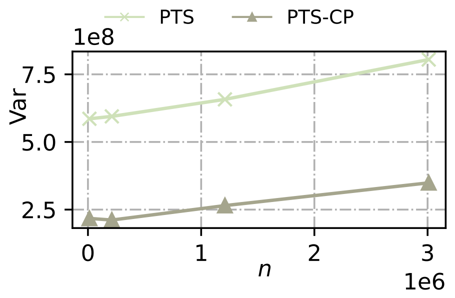


Fig. 9 Varying class amount n.

Quantify the improvement on variance. We have derived how much smaller the variance of correlated perturbation is in Section V-C. Specifically, given the dataset D with data amount N, item domain $\mathcal{I}$ and class domain $\mathcal{C}$, the label-item pair frequency $f(C,I)$, the class amount n, the difference of their variance can be derived as

$$
\begin{aligned}
&Var[\hat{f}(C,I)]_{GRR+OUE} - Var[\hat{f}(C,I)]_{CP} > \\
&\quad \frac{1}{[p_1(1-q_2)(p_2-q_2)]^2}\{Var[\tilde{f}(C,I)]_{GRR+OUE} \\
&\qquad\quad - Var[\tilde{f}(C,I)]_{CP}\} \\
&\qquad + [(\frac{q_2}{p_2-q_2})^2 - (\frac{q_2[p_1(1-q_2)-q_1(1-p_2)]}{p_1(1-q_2)(p_2-q_2)})^2]Var[\hat{n}] \\
&\qquad + \left[\frac{q_1}{(p_1-q_1)}\right]^2 Var(\sum_{C_i\in\mathcal{C}} \hat{f}(C_i,I)) \\
&\quad > \frac{(n-f(C,I))p_1^2q_2^2(1-q_2)^2 + (N-n)q_1q_2p_2(1-q_1q_2)^2}{[p_1(1-q_2)(p_2-q_2)]^2} \\
&\quad + [\frac{q_1q_2(1-p_2)}{p_1(1-q_2)(p_2-q_2)}]^2 \frac{np_1(1-p_1)+(N-n)q_1(1-q_1)}{(p_1-q_1)^2} \\
&\quad + [\frac{q_1}{(p_1-q_1)(p_2-q_2)}]^2[\sum_{C_i\in\mathcal{C}} f(C_i,I)p_2(1-p_2) + (N-\sum_{C_i\in\mathcal{C}} f(C_i,I))q_2(1-q_2)].
\end{aligned}
$$

Reviewer 3

R3.1: *The evaluation only focused on the performance of the frameworks proposed. there is no comparative evaluation with other approaches e.g Label-DP that has been cited in the related work section*

Response: Thank you for the comment. We sincerely apologize for the insufficient clarification regarding Label-DP in related-work section and appreciate the opportunity to address this concern. A comparison with label-DP is unfortunately inapplicable to our problem. Label-DP is defined as follows:

Definition (Label Differential Privacy [1]). A randomized training algorithm $\mathcal{A}$ satisfies (ϵ, δ)-label differential privacy if for any two neighboring training datasets D and D' which differ in the label of one single record, and for any subset S of the algorithm $\mathcal{A}$ outputs, there is $Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(D') \in S] + \delta$, where $0 < \delta < 1$ and $\epsilon > 0$.
[1] Ghazi, B., et al. "Deep learning with label differential privacy," in Advances in neural information processing systems, vol. 34, pp. 27131–27145, 2021.

Clearly, the privacy models of Label-DP and the DP definition used in our work are different. Specifically, Label-DP is designed to protect only the labels in supervised learning tasks (e.g., classification) while assuming other attribute data are non-private. However, our work requires privacy protection for both items and labels. The noise levels required to ensure Label-DP and the DP definition in our work are totally different. A direct comparison on the result accuracy could be unfair.

To ensure clarity, we have now revised the description of Label-DP in the related work section to explicitly differentiate their differences in privacy model.

Multi-class Item Mining under Local Differential Privacy

Yulian Mao[1,2], Qingqing Ye[2], Rong Du[2], Qi Wang[1], Kai Huang[3], Haibo Hu[2]

[1]Department of Computer Science and Engineering, Southern University of Science and Technology
[2]Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University
[3]School of Computer Science and Engineering, Macau University of Science and Technology
{yulian.mao, roong.du}@connect.polyu.hk, {qqing.ye, haibo.hu}@polyu.edu.hk, wangqi@sustech.edu.cn, kylehuangk@gmail.com

***Abstract*—Item mining, a fundamental task for collecting statistical data from users, has raised increasing privacy concerns. To address these concerns, local differential privacy (LDP) was proposed as a privacy-preserving technique. Existing LDP item mining mechanisms primarily concentrate on global statistics, i.e., those from the entire dataset. Nevertheless, they fall short of user-tailored tasks such as personalized recommendations, whereas classwise statistics can improve task accuracy with fine-grained information. Meanwhile, the introduction of class labels brings new challenges. Label perturbation may result in invalid items for aggregation. To this end, we propose frameworks for multi-class item mining, along with two mechanisms: validity perturbation to reduce the impact of invalid data, and correlated perturbation to preserve the relationship between labels and items. We also apply these optimized methods to two multi-class item mining queries: frequency estimation and top-k item mining. Through theoretical analysis and extensive experiments, we verify the effectiveness and superiority of these methods.**

***Index Terms*—Local differential privacy, Multi-class item mining, Frequency estimation, Top-k item mining**

I. Introduction

With the rise of big data analytics, item mining has become essential for collecting valuable statistics to enhance model performance [1]. During the data collection process, users' private information may be at risk of exposure [2]. To address this privacy concern, differential privacy [3] has been widely employed to ensure rigorous privacy guarantees. Its variant, namely local differential privacy (LDP), eliminates the requirement of a trusted third party for distributed data aggregation [4], [5], where local noise is added to users' data before publication. The advent of LDP has enabled rigorous privacy-preserving statistic estimation such as mean and frequency [6]–[8], based on which item mining becomes prevalent [9]–[12]. For instance, Google integrates RAPPOR into Chrome browser to collect web data [6], and Apple uses HCMS mechanism to gather emoji usage statistics from keyboard inputs [7].

Existing item mining mechanisms under LDP primarily focus on global statistics derived from the entire dataset. However, in many real-life applications in recommendation and machine learning systems, statistics from specific "classes" can provide fine-grained information and thus enhance task accuracy. The following two examples show the potential applications of multi-class item mining.

- **Shopping preference across user groups.** Studies have shown that personal attributes, such as gender and age, reflect the shopping preference [13]–[16]. Consequently, for recommendation systems, item statistics "grouped by" these attributes are more precise than global statistics derived from the entire dataset.
- **Patients' Data for Disease Diagnosis.** Machine learning plays a critical role in early-stage disease diagnosis [17], [18]. To identify disease symptoms, statistics from medical test data need to be collected for model training. Consequently, the collected data should be associated with labels indicating whether the data belongs to a healthy individual or a patient.

Therefore, accurately estimating the statistics of multi-class items is crucial under LDP. In this paper, we formulate such a problem, where each user holds a label-item pair that indicates the item belongs to the class label. Our goal is to mine classwise statistics from these label-item pairs under LDP. In particular, we aim to gather the item statistics within each class by aggregating label-item pairs from users using LDP mechanisms.

An intuitive solution framework is to treat multi-class item mining as a series of item mining tasks across different classes by user partition [6], [11], [19]. Specifically, users are grouped according to the number of classes, with each group mining the items for a specific class. However, this framework has a notable limitation: most users in each group may not possess the target label, leading to a significant proportion of invalid users.

In essence, the key challenge in multi-class item mining lies in identifying the relationship between items and their corresponding labels. To preserve such a relationship, we propose two new frameworks to perturb the label-item pairs. The first framework perturbs the label-item pairs *jointly* through expanding the perturbation domain to the Cartesian product of the item and label domains. Then each label-item pair is perturbed as a whole within this enlarged domain. In contrast, the second framework perturbs labels and items *separately*. However, using existing perturbation mechanisms on these frameworks may still incur invalid data. For instance, the item becomes meaningless if the label is perturbed to other values.

To address this, we propose two perturbation mechanisms to process invalid data and efficiently preserve the relationship between items and labels. To mitigate the impact of invalid data, we introduce *validity perturbation mechanism*, which reduces the noise injected by invalid data. Based on this, we propose *correlated perturbation mechanism* to effectively preserve the relationship between labels and items by checking the perturbed labels before item perturbation. We then apply these optimized methods to multi-class item mining queries, including frequency estimation and top-k item mining. Our contributions are as follows:

- To the best of our knowledge, this is the first study to address item mining in the multi-class context. To preserve the relationship between labels and items, we propose two foundational frameworks for aggregating label-item statistics.
- To enhance the utility of the frameworks, we introduce two optimized mechanisms as the perturbation module: validity perturbation mechanism which reduces noise injected from invalid data, and correlated perturbation mechanism that maintains the label-item relationship during perturbation.
- Leveraging on the proposed frameworks and mechanisms, we explore two typical types of item mining queries. We derive unbiased frequency estimations and propose optimized methods specifically tailored for top-k item mining. Theoretical analysis and extensive experiments on both real-world and synthetic datasets show the effectiveness of our approach.

The remainder of this paper is organized as follows. Section II provides the preliminary on LDP and formulates the problem. Section III presents two frameworks, followed by the optimized mechanisms in Section IV. Section V provides a theoretical comparison between our proposed mechanisms and existing approaches. Section VI discusses two item mining queries using the proposed methods. Section VII presents the experimental results. Section VIII reviews the related literature, and Section IX concludes this paper.

II. Preliminaries and Problem Formulation

In this section, we first introduce the fundamental concepts of local differential privacy and the mechanisms used for frequency estimation. Following this, we present a formal problem definition.

A. Local Differential Privacy

Differential privacy (DP) was introduced as a privacy measure technique with theoretical guarantee. The formal definition is as follows.

Definition 1 (Differential Privacy [3]). *Given any two neighboring datasets D and D' with one record difference, a randomized mechanism $\mathcal{A}$ satisfies ϵ-differential privacy if for all possible outputs $S \subseteq Range(\mathcal{A})$, $\Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \times \Pr[\mathcal{A}(D') \in S]$, where ϵ is the privacy budget.*

Differential privacy ensures that an adversary cannot distinguish between records from two neighboring datasets, effectively concealing individual records within the overall dataset. However, a trusted third party is required to aggregate user data under DP, which may not be feasible in many real-world scenarios. To this end, local differential privacy (LDP) [4], [5] was introduced. In LDP, each user's data is perturbed locally before uploading to an untrusted third party, ensuring that an adversary cannot distinguish between any two individual reports. The formal definition of LDP is as follows:

Definition 2 (Local Differential Privacy [4], [5]). *Given any two inputs v and v' from the domain, a randomized mechanism $\mathcal{A}$ satisfies ϵ-local differential privacy if for all possible outputs $V \subseteq Range(\mathcal{A})$, $\Pr[\mathcal{A}(v) \in V] \leq e^{\epsilon} \times \Pr[\mathcal{A}(v') \in V]$, where ϵ is the privacy budget.*

B. Statistical Estimations under Local Differential Privacy

Statistical estimations are fundamental tasks in the context of LDP, including mean and frequency estimations [8], [9], [12]. Specifically, the frequency of a value v in a given dataset D is denoted as $f(v) = \sum_{v_i \in D} \mathbb{1}_{v_i}(v)$, where $\mathbb{1}_{v_i}(v)$ is the indicator function, which returns 1 if $v_i = v$, and returns 0 otherwise.

One of the state-of-the-art mechanisms for frequency estimation under LDP, known as Optimal Unary Encoding (OUE) [9], [20]–[22], consists of two steps, namely encoding and perturbation.

- **Encoding.** Given an item v and the item domain $\{1, 2, \cdots, d\}$ with size d, the item is encoded into a d-bit vector $B = [b_1, b_2, \cdots, b_d]$, where only $b_v = 1$ and all other bits are zero.
- **Perturbation.** Given a privacy budget ϵ, each bit $B[i]$ at position i will be perturbed to $B'[i]$ as

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

Another commonly used mechanism is Generalized Random Response (GRR) [9]. Given an item $v \in \mathcal{I}$ with domain size d and the privacy budget ϵ, v is perturbed to v' as

$$\Pr[\text{GRR}(v) = v'] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+d-1} & v' = v \\ q = \frac{1}{e^{\epsilon}+d-1} & v' \in \{\mathcal{I} \setminus v\}. \end{cases}$$

Wang *et al.* [9] proposed an adaptive mechanism that selects either OUE or GRR based on item domain size d to minimize variance: if $d < 3e^{\epsilon} + 2$, GRR is chosen; otherwise, OUE is chosen.

C. Problem Definition

Given a datast D consisting of N users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, c classes $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$, and d items $\mathcal{I} = \{I_1, I_2, \ldots, I_d\}$, each user u_i possesses a label-item pair (C, I) where $C \in \mathcal{C}$ and $I \in \mathcal{I}$. Note that the item domain of each class is initially unknown. Given the dataset D, our goal is to perform two specific item mining tasks in a multi-class setting: frequency estimation and top-k item mining.

Definition 3 (Multi-class Frequency Estimation). *Given a dataset D composed of label-item pairs from N users, i.e.,*

$D = \{(C_{u_1}, I_{u_1}), (C_{u_2}, I_{u_2}), \cdots, (C_{u_N}, I_{u_N})\}$, *where each user* u_i *holds a label-item pair* (C_{u_i}, I_{u_i}). *The frequency of an item* $I \in \mathcal{I}$ *within a class* $C \in \mathcal{C}$ *is denoted as* $f(C, I) = \sum_{u_i \in \mathcal{U}} \mathbb{1}_{u_i}(C, I)$, *where* $\mathbb{1}_{u_i}(C, I)$ *is an indicator function defined by*

$$\mathbb{1}_{u_i}(C, I) = \begin{cases} 1, & \textit{if } C_{u_i} = C \wedge I_{u_i} = I, \\ 0, & \textit{if } C_{u_i} \neq C \vee I_{u_i} \neq I. \end{cases}$$

The estimated frequency of a label-item pair (C, I) *from a mechanism* $\mathcal{A}$ *is said to be unbiased if* $\mathbb{E}[\mathcal{A}(C, I)] = f(C, I)$, *where* $\mathbb{E}(\cdot)$ *denotes the expectation.*

Definition 4 (Multi-class Top-k Item Mining)**.** *Given a dataset* D *from* N *users holding label-item pairs, i.e.,* $D = \{(C_{u_1}, I_{u_1}), (C_{u_2}, I_{u_2}), \cdots, (C_{u_N}, I_{u_N})\}$, *an item* $I \in \mathcal{I}$ *is a top-*k *item within a class* $C \in \mathcal{C}$ *if it is among the* k *most frequent items within that class.*

D. HEC: A Strawman Solution

In many item mining tasks, to avoid privacy budget allocation, user partition is commonly used, where users are divided into groups, with each group focusing on mining a particular item or set of items [6], [11], [19]. In our multi-class item mining problem, a straightforward approach is to divide users by class and aggregate item information within each class independently.

Handling each class independently (HEC). The users are divided into c groups corresponding to the number of classes. Within each group, item statistics of the assigned class are collected using an LDP mechanism with privacy budget ϵ. If a user's label does match the assigned class, her item is considered invalid for that class. To comply with the LDP guarantee, she needs to randomly select an item from the item domain to ensure deniability.

Although HEC enables multi-class item collection, its effectiveness is limited because only users with items in the assigned class can contribute to item mining. This is especially true when the domain size $|\mathcal{I}'| = d$ is large, leading to a substantial amount of invalid data.

III. Frameworks for Multi-class Item Mining

In multi-class item mining, it is crucial to consider class information. With the introduction of class labels, it becomes essential to maintain the relationship between a label and the corresponding item in a label-item pair. In this section, we present two frameworks specifically designed to preserve this relationship.

A. Overview

We propose two foundational frameworks designed to maintain the intrinsic relationship between labels and items. The first framework jointly perturbs the label and item (PTJ), maintaining their inherent connection. The second framework perturbs the label and item separately (PTS), followed by consolidated aggregation. Figure 1 illustrates both frameworks. Initially, a label-item pair is fed into one of the frameworks—either PTJ or PTS. Perturbation is then executed using a mechanism including existing LDP mechanisms, validity perturbation mechanism, and correlated perturbation mechanism. After perturbation, the perturbed pair is transmitted to the server for aggregation. Once all user data is aggregated, the server compiles the classwise statistics.

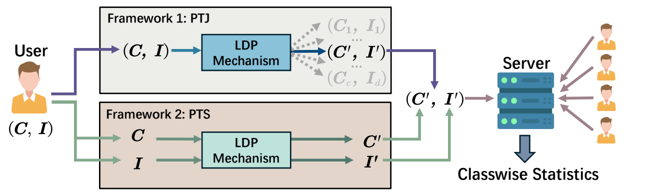


Fig. 1. The overview illustrates two frameworks for multi-class item mining. The first framework, referred to as PTJ, treats a label-item pair as a whole and perturbs it to another pair. The second framework PTS perturbs each element separately.

B. Details of Frameworks

As previously discussed, the HEC framework neglects the class information, leading to a significant amount of invalid users due to mismatched labels. To this end, an alternative approach treats the label-item pair as a cohesive unit, and then jointly perturbs the entire pair.

Perturbing the pair jointly (PTJ). Given the label domain $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$, and the item domain $\mathcal{I} = \{I_1, I_2, \ldots, I_d\}$, the perturbation domain is defined as the Cartesian product $\mathcal{P} = \{(C_1, I_1), (C_1, I_2), \cdots\}$, with size $|\mathcal{P}| = c \times d$. Each user, holding a label-item pair, perturbs her pair using an LDP mechanism with privacy budget ϵ over the perturbation domain $\mathcal{P}$. After aggregating all the perturbed pairs, the item information within each class can be inferred.

When either the label domain or the item domain is large, the size of the perturbation domain increases significantly, leading to high communication or computation cost [9]. To mitigate this, an alternative approach is to perturb the label and item separately.

Perturbing the pair separately (PTS). For a user u_i holding a label-item tuple (C, I), the label is first perturbed via an LDP mechanism with part of the privacy budget ϵ_1. Similarly, the item is perturbed using an LDP mechanism with the remaining privacy budget ϵ_2.

Within these frameworks, existing perturbation mechanisms still suffer from invalid data in item mining tasks. For instance, the class label may be perturbed to other classes, resulting in an invalid item for that class. To enhance the utility of multi-class item mining, it is imperative to optimize perturbation mechanisms to process invalid information and account for the label-item relationship throughout the perturbation process.

IV. Optimized Perturbations

In this section, we introduce the validity perturbation to process data including invalid items. Building on validity perturbation, we propose correlated perturbation to further preserve the label-item relationship during perturbation process. In the interest of space, the proofs in this section are included in our technical report [23].

A. Validity Perturbation Mechanism

In this subsection, we propose validity perturbation mechanism based on the Unary Encoding (UE) [9] mechanism. To process invalid items, an intuitive approach is to randomly select a valid item to replace the invalid one for perturbation and aggregation [24], [25]. However, such random noise can distort the aggregated statistics of valid items, resulting in utility degradation. To address this, we should exclude those invalid items in the aggregation process. Specifically, we first privately publicize item validity, and then omit the invalid items. To avoid consuming extra privacy budget for item validity, we integrate a validity flag into the unary encoding (UE) mechanism to process invalid items. Both validity flag and items are perturbed simultaneously. The validity perturbation mechanism consists of encoding and perturbation.

Encoding. As shown in Fig. 2, given an item v with domain size d, if the item is valid, $\mathrm{Encode}(v) = [0, \cdots, 0, 1, 0, \cdots, 0]$ is a $(d+1)$-length binary vector with the v-th position set to "1". Conversely, if the item is invalid, $\mathrm{Encode}(v) = [0, \cdots, 0, 0, 0, \cdots, 1]$ is a $(d+1)$-length binary vector with the last position set to "1".

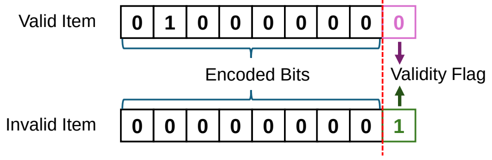


Fig. 2. An illustration for encoding scheme. For a valid item, the encoded bits from UE are padded with a validity flag "0". Conversely, for an invalid item, all encoded bits are set to "0", and the validity flag is set to "1".

Perturbing. Given the encoded binary vector $B = \mathrm{Encode}(v)$, the output B' from the perturbation $\mathrm{Perturb}(B)$ follows the same process as the UE mechanism [9]. For any position i in B,

$$\Pr[B'[i] = 1] = \begin{cases} p, & \text{if } B[i] = 1 \\ q, & \text{if } B[i] = 0. \end{cases} \tag{1}$$

Theorem 1. *(Privacy of Validity Perturbation Mechanism) The validity perturbation mechanism satisfies ϵ-LDP for $\epsilon = \ln \frac{p(1-q)}{(1-p)q}$.*

In this paper, the perturbation probabilities in Eq. (1) are set the same as Optimized Unary Encoding (OUE) mechanism for a convenient comparison with other LDP mechanisms, i.e., $p = \frac{1}{2}$, $q = \frac{1}{e^\epsilon + 1}$, where ϵ represents the privacy budget [9]. A utility analysis of the validity perturbation mechanism is presented in Section V. We use the noise injected by invalid users as the utility metric [26]. The theoretical results demonstrate the superiority of the validity perturbation mechanism.

With validity perturbation, we can effectively manage invalid data and reduce its impact. Based on this, we introduce correlated perturbation, which preserves the relationship between labels and items during the perturbation process.

B. Correlated Perturbation Mechanism

In this subsection, we present the correlated perturbation mechanism. In multi-class item mining, the label and item are paired, exhibiting their inherent correlation. When the label is perturbed, the item should be changed accordingly, or labeled as noise to avoid unexpected influence. Therefore, the labels and the items should be perturbed in a correlated manner. Specifically, we can perturb the label first, and then perturb the item according to the label's perturbation result. If the label differs from its original value, the item becomes invalid and must be excluded from the aggregation result. For the purpose of LDP guarantee, extra privacy budget is required to learn the the condition of the perturbed label — whether the perturbed label matches its original value. To save privacy budget, we adopt validity perturbation mechanism, using the label condition as the validity flag. Based on this, we propose the correlated perturbation mechanism as follows.

Given a label-item pair from a user with the class label $C \in \mathcal{C}$ ($\mathcal{C} = \{C_1, C_2, \cdots, C_c\}$) and the item $I \in \mathcal{I}$ ($\mathcal{I} = \{I_1, I_2, \cdots, I_d\}$), the perturbation process consists of label perturbation and item perturbation phases, where the total privacy budget ϵ is divided into ϵ_1 for label perturbation and ϵ_2 for item perturbation, such that $\epsilon = \epsilon_1 + \epsilon_2$. In this paper, we set $\epsilon_1 = \epsilon_2 = \frac{\epsilon}{2}$.

Label Perturbation. For a given class label C and privacy budget ϵ_1, the label is perturbed by Φ with privacy budget ϵ_1 such that

$$\Pr[\Phi(C) = C'] = \begin{cases} p_1 & \text{if } C' = C \\ q_1 & \text{if } C' \in \mathcal{C} \backslash \{C\}, \end{cases} \tag{2}$$

where Φ is an LDP mechanism, such as the Generalized Random Response (GRR) mechanism, and the ratio is $p_1/q_1 \leq \epsilon_1$ for LDP guarantee.

Item Perturbation. If the perturbed label differs from its original value, the item is marked as invalid; otherwise, the item is valid and the validity perturbation mechanism is applied to perturb the item with the privacy budget ϵ_2. For a given item I from the item domain $\mathcal{I}$ with size d, and privacy budget ϵ_2, the item will be first encoded to a bit vector B with length $d+1$ according to the validity. Any position j of B after encoding is denoted as

$$B[j] = \begin{cases} 1 & \text{if } (j = I \wedge C' = C) \vee (j = d+1 \wedge C' \neq C) \\ 0 & \text{Otherwise.} \end{cases}$$

Given the encoded bit vector B, any bit $B[j] \in B$ will be flipped via the OUE perturbation mechanism Ψ with privacy budget ϵ_2 [9],

$$\Pr[\Psi(B[j]) = 1] = \begin{cases} p_2 & \text{if } B[j] = 1 \\ q_2 & \text{if } B[j] = 0, \end{cases} \tag{3}$$

where $p_2 = \frac{1}{2}$ and $q_2 = \frac{1}{e^{\epsilon_2} + 1}$.

Theorem 2. *(Privacy of Correlated Perturbation Mechanism) The correlated perturbation mechanism satisfies ϵ-LDP.*

In certain item mining tasks, an unbiased frequency estimation is required [9]. In the following, we present the calibration on correlated perturbation mechanism to obtain the unbiased result. Given the collected count $\tilde{f}(C, I)$ of a label-item pair (C, I) aggregated from N users, the calibrated result is

$$\begin{aligned}\hat{f}(C, I) =& \frac{\tilde{f}(C, I) - Nq_1q_2(1-p_2)}{p_1(1-q_2)(p_2-q_2)} \\ &- \frac{\hat{n}q_2[p_1(1-q_2) - q_1(1-p_2)]}{p_1(1-q_2)(p_2-q_2)},\end{aligned} \tag{4}$$

where p_1, q_1, p_2, and q_2 are the perturbation probabilities in Eqs. (2) and (3). Additionally, $\hat{n}$, representing the unbiased frequency estimation of users with label C, is derived by $\hat{n} = \frac{\tilde{n} - Nq_1}{p_1 - q_1}$, where $\tilde{n}$ is the collected count of users with label C.

Theorem 3. *The calibrated frequency $\hat{f}(C, I)$ is unbiased.*

V. Utility Analysis

In this section, we theoretically analyze the utility of the proposed mechanisms, validity perturbation and correlated perturbation, to demonstrate their superiority. In the interest of space, the proofs in this section are included in our technical report [23].

A. Utility Analysis for Validity Perturbation Mechanism

As aforementioned, the validity perturbation mechanism is to mitigate the impact of invalid data. We use the noise injected by invalid data as the utility metric [26]. The uploaded item from an invalid user can be regarded as a random injection. For instance, the item domain is narrowed down during the mining process, and infrequent items may be pruned. An invalid user is the one who possesses an infrequent item that has been pruned. To comply with LDP, the invalid user randomly selects a valid candidate for deniability [24], [25]. The injected noise can be derived as follows.

Theorem 4. *Given the valid item domain size d and the number of invalid users m, the noise injected into a valid item from an LDP mechanism is $\mathbb{E}_{noise} = mq + \frac{1}{d}m(p-q)$, and the variance of the injected noise is $Var_{noise} = mq(1-q) + \frac{m}{d}(p-q)(1-p-q)$, where p and q are the perturbation probabilities set by an LDP mechanism.*

Similarly, the injected noise of the validity perturbation can be expressed as follows.

Theorem 5. *Given the valid item domain size d and m invalid users, the noise injected on a valid item via the validity perturbation mechanism is $\mathbb{E}_{noise} = mq$, and the variance of the injected noise is $Var_{noise} = mq(1-q)$, where q is the probability in Eq. (1).*

Clearly, the validity perturbation mechanism reduces the injected noise on valid items from invalid users. The utility of the validity perturbation mechanism is superior to that of the OUE mechanism as a comparison [9], and by extension, it outperforms the other LDP mechanisms in processing invalid data.

B. Effectiveness of Validity Perturbation

Since the validity perturbation mechanism affects the perturbation results on both valid and invalid users, its effectiveness still needs to be evaluated. We first derive the expectation and variance of the collected counts using LDP mechanisms, and then compare these with the corresponding values derived from the validity perturbation mechanism.

Theorem 6. *Given that N_1 users hold the target item, N_2 users hold the other items in the valid item domain with size d, and m users hold the invalid items. For an LDP mechanism, the count expectation of the target item is*

$$\begin{aligned}\mathbb{E}(count) &= N_1p + N_2q + \frac{m}{d}p + m(1-\frac{1}{d})q \\ &= N_1p + N_2q + mq + \frac{m}{d}(p-q).\end{aligned}$$

Correspondingly, the variance is

$$\begin{aligned}Var(count) =& N_1(p-p^2) + N_2(q-q^2) + m(q-q^2) \\ &+ \frac{m}{d}(p-q)(1-p-q).\end{aligned}$$

Similarly, the expectation and the variance of the validity perturbation mechanism can be derived as follows.

Theorem 7. *Suppose N_1 users hold the target item, N_2 users hold the other items in the valid item domain with the size d, and m users hold the invalid items. For the validity perturbation mechanism, the count expectation of the target item is*

$$\begin{aligned}\mathbb{E}(count) &= N_1p \cdot (1-q) + N_2q \cdot (1-q) + mq \cdot (1-p) \\ &= (1-q)(N_1p + N_2q + mq - mq\frac{p-q}{1-q}).\end{aligned}$$

And the variance is

$$\begin{aligned}Var(count) =& N_1(p - p^2 + 2p^2q - pq - p^2q^2) \\ &+ N_2(q - 2q^2 + 2q^3 - q^4) \\ &+ m(q - q^2 + 2pq^2 - pq - p^2q^2).\end{aligned}$$

To compare the validity perturbation mechanism with LDP mechanisms with invalid data, we use the Optimized Unary Encoding (OUE) mechanism as a comparison [9]. In terms of expectations, the validity perturbation mechanism is preferable because it introduces less noise. Although the expectation is scaled, the counts of all items are scaled consistently, preserving the rank orders. Regarding variance, the difference between the validity perturbation mechanism and the OUE mechanism can be expressed as follows:

$$\begin{aligned}&N_1(2p^2q - pq - p^2q^2) + N_2(2q^3 - q^2 - q^4) \\ &+ m(2pq^2 - pq - p^2q^2) - \frac{m}{d}(p-q)(1-p-q) \\ =&N_1pq(2p - 1 - pq) + N_2q^2(2q - 1 - q^2) \\ &+ mpq(2q - 1 - pq) - \frac{m}{d}(p-q)(1-p-q),\end{aligned}$$

which is always smaller than 0. Namely, the validity perturbation mechanism is better than the OUE mechanism to process data with invalid ones, thereby it is also better than the other LDP mechanisms.

C. Utility Analysis for Correlated Perturbation

In this section, we analyze the utility of the correlated perturbation mechanism with unbiased calibration for frequency estimation. To demonstrate the superiority of the correlated perturbation mechanism, we compare its variance with other LDP mechanisms to perturb the label-item pair. We first derive the variance of the correlated perturbation mechanism as follows.

Theorem 8. *The variance of the estimated frequency $\hat{f}(C,I)$ in Eq. (4) is*

$$\begin{aligned}Var[\hat{f}(C,I)] = &\frac{f(C,I)p_1(1-q_2)p_2\left[1-p_1(1-q_2)p_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2}\\&+\frac{(n-f(C,I))p_1(1-q_2)q_2\left[1-p_1(1-q_2)q_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2}\\&+\frac{(N-n)q_1(1-p_2)q_2\left[1-q_1(1-p_2)q_2\right]}{\left[p_1(1-q_2)(p_2-q_2)\right]^2}\\&+\left[\frac{q_2\left[p_1(1-q_2)-q_1(1-p_2)\right]}{p_1(1-q_2)(p_2-q_2)}\right]^2\\&\times\frac{n\left[p_1(1-p_1)-q_1(1-q_1)\right]+Nq_1(1-q_1)}{(p_1-q_1)^2},\end{aligned} \tag{5}$$

where p_1, q_1, p_2 and q_2 denote the perturbation probabilities in Eqs. (2) and (3). Additionally, n represents the true number of users with label C, and N refers to the total number of users.

Variance analysis. Our investigation focuses on two critical aspects: the correlation strengths between labels and items, and the class distribution. To quantify label-item correlations, we employ pointwise mutual information (PMI) [28], defined as $\mathrm{PMI(C;I)} = \log_2\frac{p(\mathrm{C,I})}{p(\mathrm{C})p(\mathrm{I})}$, where $p(C,I)$ denotes the joint probability of the label-item pair (C,I), while $p(C)$ and $p(I)$ represent marginal probabilities of labels and items, respectively. A larger PMI indicates a stronger correlation between the label and the item. When $p(C)$ and $p(I)$ are fixed, we have $\mathrm{PMI(C;I)} \propto \mathrm{f(C,I)}$. Since the variance in Eq. (5) is proportional to $f(C,I)$, it follows that $Var[\hat{f}(C,I)] \propto \mathrm{PMI(C;I)}$. Note that the variance equation exhibits complex structure. We numerically estimate the variable coefficients, which depend on the dataset, under varying epsilon values to analyze variance dynamics. The coefficients are in Table I. Crucially, $f(C,I)$ is always much smaller than class amount n and data amount N, so that the coefficients of $f(C,I)$ in Table I cannot offset orders of magnitude differences. This magnitude disparity explains why correlation variations are concealed in variance analysis. The impact of class amount n on variance can be analyzed in a similar manner. With fixed label-item pair frequency $f(C,I)$ and data amount N, we establish $Var(\hat{f}(C,I)) \propto n$, revealing that class diversity expansion directly amplifies variance. These theoretical predictions are empirically validated in Section VII.

Similarly, the unbiased frequency estimation of a label-item pair and its corresponding variance under LDP mechanisms can be derived as follows.

TABLE I
COEFFICIENTS OF VARIABLES IN $Var(\hat{f}(C,I))$

ϵ	0.5	1	1.5	2	2.5	3	3.5	4
$f(C,I)$	87.4	32.9	17.1	10.3	6.8	4.9	3.7	2.9
n	213.8	58.9	22.8	10.5	5.4	3.0	1.8	1.1
N	441.8	53.3	12.0	3.6	1.3	0.5	0.2	0.1

Theorem 9. *Given a dataset D consisting of N users holding label-item pairs, using an LDP mechanism for label perturbation with probabilities p_1 and q_1, and an LDP mechanism for item perturbation with probabilities p_2 and q_2, the estimated frequency $\hat{f}(C,I)$ of an item I within a class C is given by:*

$$\begin{aligned}\hat{f}(C,I) = &\frac{\tilde{f}(C,I)-\hat{n}q_2(p_1-q_1)}{(p_1-q_1)(p_2-q_2)}\\&-\frac{\sum_{C_i\in\mathcal{C}}\hat{f}(C_i,I)q_1(p_2-q_2)+Nq_1q_2}{(p_1-q_1)(p_2-q_2)},\end{aligned} \tag{6}$$

where $\tilde{f}(C,I)$ represents the collected count of the target label-item pair (C,I), and

$$\sum_{C_i\in\mathcal{C}}\hat{f}(C_i,I) = \frac{\sum_{C_i\in\mathcal{C}}\tilde{f}(C_i,I)-Nq_2}{p_2-q_2}$$

is the unbiased frequency estimation of the item I, where $\sum_{C_i\in\mathcal{C}}\tilde{f}(C_i,I)$ is the collected count of the item I. Additionally, $\hat{n}$, the unbiased frequency estimation of users with label C, is given by: $\hat{n} = \frac{\tilde{n}-Nq_1}{p_1-q_1}$, where $\tilde{n}$ is the collected count the label C.

To compare the variance between the correlated perturbation and state-of-the-art LDP mechanisms, we use the OUE mechanism as a representative LDP mechanism for item perturbation and GRR for label perturbation [9].

Theorem 10. *Given the dataset D with data amount N, item domain $\mathcal{I}$ and class domain $\mathcal{C}$, the label-item pair frequency $f(C,I)$, the class amount n, the difference of the variances of the estimated frequency $\hat{f}(C,I)$ in Eq. (6) and Eq. (4) can be derived as*

$$\begin{aligned}&Var[\hat{f}(C,I)]_{GRR+OUE} - Var[\hat{f}(C,I)]_{CP} >\\&\frac{(n-f(C,I))p_1^2q_2^2(1-q_2)^2+(N-n)q_1q_2p_2(1-q_1q_2)^2}{[p_1(1-q_2)(p_2-q_2)]^2}\\&+[\frac{q_1q_2(1-p_2)}{p_1(1-q_2)(p_2-q_2)}]^2\frac{np_1(1-p_1)+(N-n)q_1(1-q_1)}{(p_1-q_1)^2}\\&+[\frac{q_1}{(p_1-q_1)(p_2-q_2)}]^2[\sum_{C_i\in\mathcal{C}}f(C_i,I)p_2(1-p_2)\\&\quad+(N-\sum_{C_i\in\mathcal{C}}f(C_i,I))q_2(1-q_2)].\end{aligned}$$

Finally, we discuss the utility of the PTS framework with the correlated perturbation and the PTJ framework with an LDP mechanism. Since the PTJ framework consumes the whole privacy budget, integrating OUE into the PTJ framework improves the utility compared with the PTS framework with the correlated perturbation. However, the communication cost in the PTJ framework increases significantly due to the enlarged perturbation domain, which combines all labels and items. Consequently, the PTJ framework is not suitable for scenarios with a large label domain or item domain due to its high communication cost [9].

VI. Queries for Multi-class Item Mining

To demonstrate the usability of our framework for multi-class item mining, in this section, we implement it for multi-class frequency estimation in Section VI-A and multi-class top-k item mining in Section VI-B, respectively.

A. Multi-class Frequency Estimation

Many machine learning models, such as decision tree, rely on frequency information to build models [29], [30]. To protect users' private information when building the model, a privacy-preserving approach for multi-class frequency estimation is essential. Building upon the foundational frameworks — HEC, PTJ, and PTS, we derive the corresponding unbiased frequency estimations within the multi-class context under LDP.

- **Frequency estimation under HEC.** Given a dataset D consisting of N users, with c classes $\mathcal{C} = \{C_1, C_2, \cdots, C_c\}$ and d items $\mathcal{I} = \{I_1, I_2, \cdots, I_d\}$. The users are divided into c groups, each group is responsible for aggregating one class. For each user holding a label-item pair, an LDP mechanism is employed to perturb the item with a privacy budget ϵ. If a user's label does not match the assigned class, she randomly selects an item from the item domain for perturbation to ensure deniability. The unbiased frequency of a label-item pair (C, I) is then derived as follows: $\hat{f}(C, I) = \frac{c\tilde{f}(C,I)-Nq}{p-q}$, where p and q are the corresponding perturbation probabilities of the LDP mechanism.
- **Frequency estimation under PTJ.** Given the same dataset illustrated above. Each user perturbs her label-item pair using an LDP mechanism with privacy budget ϵ over the perturbation domain $\mathcal{P} = \{(C, I) \mid C \in \mathcal{C}, I \in \mathcal{I}\}$. After aggregating all the perturbed pairs, the unbiased frequency for a pair (C, I) can be inferred as: $\hat{f}(C, I) = \frac{\tilde{f}(C,I)-Nq}{p-q}$, where p and q are the corresponding perturbation probabilities of the LDP mechanism.
- **Frequency estimation under PTS.** For a label-item pair (C, I) from a user, the label is first perturbed via an LDP mechanism with the privacy budget ϵ_1, and the item is perturbed with the remaining privacy budget ϵ_2. The unbiased estimated frequency is derived in Eq. (6).
- **Frequency estimation under PTS with the correlated perturbation.** For a user with a label-item pair (C, I), the pair is perturbed via the correlated perturbation with the privacy budget ϵ, which perturbs the item based on the perturbed label. The unbiased frequency is derived in Eq. (4).

Complexity analysis. Given the class domain size c, item domain size d, and data amount N, we consider the OUE as the LDP mechanism [9]. The communication cost for each user of HEC, PTS, and PTS-CP is $\mathcal{O}(d)$; while for PTJ, it is $\mathcal{O}(cd)$ with an enlarged perturbation domain including combinations of both items and labels. The time complexities of the HEC, PTS, and PTS-CP mechanisms are $\mathcal{O}(d)$ on the user side, and $\mathcal{O}(Nd)$ on the server side. While the time complexity of PTJ is $\mathcal{O}(cd)$ on the user side and $\mathcal{O}(Ncd)$ on the server side. As for the space complexity, the HEC, PTS, and PTS-CP require $\mathcal{O}(d)$ space on the user side, while the server side requires $\mathcal{O}(cd)$ space. On the other hand, the PTJ requires $\mathcal{O}(cd)$ space on both the user and server sides.

B. Multi-class Top-k Item Mining

In this subsection, we first elaborate on two issues in the existing top-k item mining schemes and then propose corresponding solutions. Finally, we present our scheme for multi-class top-k item mining.

Existing issues and our solutions. The top-k item mining is an essential task to identify top frequent items with a large item domain. Many top-k item mining schemes under LDP use a prefix trie to collect item frequencies [24], [25], [31]. We take the state-of-the-art method, PEM [24], as an example to demonstrate the prefix trie-based schemes under LDP. Under PEM, items are encoded into bits, converting the top-k item mining problem into a frequent sequence mining task. The trie progressively expands from shallow to deeper levels, as the server collects prefix frequencies and prunes infrequent ones to identify longer frequent prefixes. However, this mining scheme can produce false positive prefixes, and also introduce invalid data during pruning.

- **False positive prefixes**. While prefix expansion is based on plausible insights, it is not always reliable, as frequent prefixes do not necessarily correspond to frequent sequences [32]. The prefix trie may yield false positives, identifying prefixes that appear frequent but are not truly among the top sequences, which potentially causes genuine top items to be overlooked. An example without LDP noise is illustrated in Fig. 3, where the most frequent item '000' is eventually missed as its prefix '0' is less frequent than '1' in the upper layer. To mitigate the

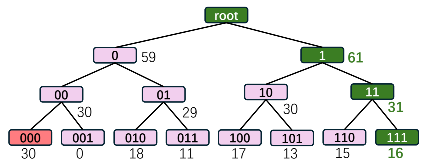


Fig. 3. The goal is to mine the top-1 item among eight items encoded from '000' to '111', with each node showing the corresponding frequency. With the prefix expansion scheme, the true top item is missed even without LDP noise. Green nodes represent the expansion path of the trie, the red node is the actual top-1 item, and pink nodes are pruned during trie expansion. However, integrating the mining scheme with our shuffling can achieve the top-1 item with probability $(C_8^2 \cdot C_6^2 \cdot C_4^2/4! - C_6^2 \cdot C_4^2/3!)/(C_8^2 \cdot C_6^2 \cdot C_4^2/4!) = 0.857$.

generation of such false prefixes, a viable approach is to decouple prefix groups to reduce specific combinations, for which shuffling is particularly effective. To reduce communication cost, only the random seeds and the bucket states for pruning are sent to users rather than

Algorithm 1 Candidate generation.

```
Input: Dataset $D$, $\epsilon$, $k$, item domain size $d$, class number $|\mathcal{C}|$, a constant $a$
1:  Obtain the iteration number $IT = \log_2 \frac{d}{4k} + 1$
2:  # Use $a \cdot |D|$ data to conduct the first $IT_f$ iterations
3:  for Iteration $it \leq IT_f$ do
4:      Shuffling and split the item candidates into $4 \cdot k \cdot |\mathcal{C}|$ buckets using the given random seeds and bucket states
5:      Process each user's item: it is valid if the item is in the pruned candidate set; otherwise, it is invalid.
6:      Perturb the item with $\epsilon_2$ via the validity perturbation and perturb the label via an LDP mechanism using $\epsilon_1$
7:      Remain top $2 \cdot k \cdot |\mathcal{C}|$ buckets as candidates for next iteration
8:  # Assess the injected noise for each class
9:  Estimate user amount $|D'_C|$ for each class from perturbed labels to indicate injected noise levels
10: Return the candidates and noise level for each class
```

Algorithm 2 Top-k item mining within each class.

```
Input: Dataset $D_C$, $\epsilon_2$, $k$, the estimated user amount $|D'_C|$, the remaining iteration number $IT_r$, a constant $b$, the candidates from Algorithm 1
1:  # Process the first $IT_r - 1$ iterations
2:  for Iteration $it \leq IT_r - 1$ do
3:      Shuffling and split the item candidates into $4 \cdot k$ buckets with a given random seed
4:      The same steps as Algorithm 1 Line 5
5:      Perturb item with $\epsilon_2$ via the validity perturbation mechanism
6:      Remain top $2 \cdot k \cdot$ buckets as pruned items for next iteration
7:  # Process the last iteration
8:  If the collected user amount $|D_C| > b \cdot |D'_C|$, only apply validity perturbation mechanism in the last iteration
9:  Choose the mechanism from the validity perturbation mechanism and correlated perturbation mechanism according to the noise level
10: Return the top-$k$ items according to the aggregation
```

the entire set of the shuffled candidates. The shuffling process is illustrated in Fig. 4.

- **Invalid data during pruning.** Besides introducing false positive prefixes, existing schemes also neglect the impact of invalid data. As the pruning process progresses, more and more items are excluded from the candidate set, resulting in invalid data. When encountering invalid data, PEM substitutes it with a randomly selected item [24]. To reduce the impact from invalid data, we apply validity perturbation.

Multi-class top-k item mining scheme. For multi-class context, in addition to shuffling method and validity perturbation, we leverage globally frequent items across classes to enhance utility. We present the details of the PTS-based scheme below with an illustration in Fig. 4. Note that PTJ-based scheme can be implemented in a similar way, using shuffling method and validity perturbation mechanism.

- **Step 1: Candidate generation**. In general, classwise top-k items are also frequent items among the entire dataset. That is because those top frequent items within a class are typically shared among different classes, such as popular goods common to all age groups. Consequently, we can use items from a small group of users sampled from the entire dataset (with parameter a in Algorithm 1 Line 2 controlling the sample proportion) to mine item candidates including these globally frequent items during the initial iterations. Meanwhile, the perturbed labels can be used to assess the noise level in each class. If the injected noise is too large — specifically, if it exceeds b times the estimated class size (Algorithm 2 Line 8) — the amount of valid data may be insufficient for accurate estimation, rendering the correlated perturbation mechanism infeasible. In such class, only validity perturbation mechanism is applied. Note that only the PTS framework can benefit from the globally frequent items. Details are in Algorithm 1.
- **Step 2: Top-k item mining within each class.** After obtaining the candidates and noise level, the remaining users are assigned to each class for classwise top-k item mining based on their perturbed labels. As iterations progress, the proportion of valid data decreases. In the later stages, most of the data becomes invalid due to candidate pruning and label perturbation. To accommodate the decrease of valid data, correlated perturbation is applied only in the final iteration to identify classwise top frequent items, while the validity perturbation mechanism is adopted in other iterations. The details of this step can be found in Algorithm 2.

Complexity analysis. Given the class domain size c, item domain size d, and data amount N, we use the GRR mechanism for label perturbation and the OUE mechanism for item perturbation [9], the communication cost is measured for each user. The mining scheme for the fundamental frameworks is PEM, and the extending length in each iteration is m [9]. The costs are summarized in Table II.

TABLE II
COMPLEXITY ANALYSIS

	Communication	**Time**	**Space**
HEC **PTS**	$\mathcal{O}(2^m k \log d)$	$\mathcal{O}(2^m k)$ $\mathcal{O}(2^m k[c(m + \log k)\frac{\log d}{m} + N])$	$\mathcal{O}(2^m k \log d)$ $\mathcal{O}(2^m ck \log d)$
PTJ	$\mathcal{O}(2^m ck \log cd)$	$\mathcal{O}(2^m ck)$ $\mathcal{O}(2^m ck[(m+\log ck)\frac{\log cd}{m}+N])$	$\mathcal{O}(2^m ck \log cd)$ $\mathcal{O}(2^m ck \log cd)$
PTJ*	$\mathcal{O}(ck)$	$\mathcal{O}(ck)$ $\mathcal{O}(ck(\log ck \log \frac{d}{k} + N))$	$\mathcal{O}(cd)$ $\mathcal{O}(cd)$
PTS*	$\mathcal{O}(ck)$	$\mathcal{O}(ck)$ $\mathcal{O}(ck(\log ck \log \frac{d}{k} + N))$	$\mathcal{O}(d)$ $\mathcal{O}(cd)$

* represent the optimized methods, with the first line of each row showing the user-side results and the second line showing the server-side results.

VII. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the performance of the three frameworks, HEC, PTJ, and PTS, and the optimized methods for item mining tasks. All methods are implemented in Python 3, and each experiment is averaged from 20 trials.

A. Datasets

We conduct the experiments over six datasets including both real-world datasets and synthetic datasets.

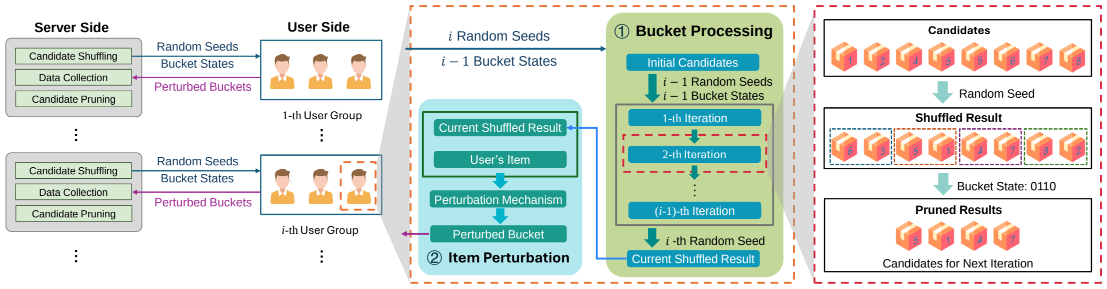


Fig. 4. Each user receives random seeds and bucket states to generate a current shuffled result before perturbing her label-item pair. The choice of perturbation mechanism depends on the aggregation goal described in Algorithms 1 and 2.

- **Comprehensive Diabetes Clinical Dataset.**[1] It is collected from 100,000 individuals for diabetes research, including eight features. And continuous values are rounded to one decimal place, with the largest feature domain containing about 600 items.
- **Heart Disease Health Indicator Dataset.**[2] This dataset includes 253,680 cleaned survey responses from BRFSS 2015, primarily for binary heart disease classification. It contains 21 categorical features, with the largest item domain being 84.
- **MyAnimeList Dataset.**[3] This dataset contains information on anime viewing habits, comprising around 116,000 users, 14,000 anime titles, and 35 million records. We treat gender and watched anime as label-item pairs, mining the top anime titles across different gender groups. We sample 20% data for experiments.
- **JD Contest Dataset.**[4] This dataset from JD.COM contains sale records about 28,000 items. We utilize the five age groups (below 25, 26-35, 36-45, 46-55, and above 56) and treat the age group as the label, resulting in 45 million valid item-label pairs. A 20% sample is used for experiments.
- **Synthetic Datasets.** We generate four datasets: SYN1 and SYN2 are used for variance analysis, while SYN3 and SYN4 are employed to study the effect of varying class numbers, ranging from 10 to 50. To control $f(C, I)$, n, and N, SYN1 and SYN2 are generated with 4 classes, 4 items, and label-item pair amounts of 1,000, 10,000, 100,000, and 1,000,000. SYN1 fixes the class amount n to investigate correlation strength varying, while SYN2 fixes one label-item pair frequency $f(C, I)$ across the classes to examine the impact of the class distribution. SYN3 and SYN4 contain 20,000 items and five million instances, the data size of each class satisfies the normal distribution. Within each class, the items are drawn from the exponential distribution with scale from 0.01 to 0.1. SYN3 is simulated based on real-world datasets and includes globally frequent items, with an average of eight overlapping items among the top-20 items between any two classes. In contrast, SYN4 is generated using the same method but excludes globally frequent items.

[1]https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset
[2]https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset
[3]https://www.kaggle.com/datasets/azathoth42/myanimelist
[4]https://www.kaggle.com/datasets/pwang001/jjingdong-contest-dataset

We use the Diabetes and Heart Disease datasets for the frequency estimation task. To align with our problem setting where each user holds a label-item pair, we calculate label-item frequency within each feature by dividing users into groups, with each group focusing on a single feature to mine corresponding label-item pairs. Moreover, we assess the performance of top-k item mining using the Anime and JD datasets, while the two synthetic datasets are used to analyze the impact of varying class numbers.

B. Metrics

For frequency estimation, we apply root mean square error (RMSE) to measure the difference between the estimated frequency $\hat{f}(C, I)$ of label-item pairs and the ground truth $f(C, I)$,

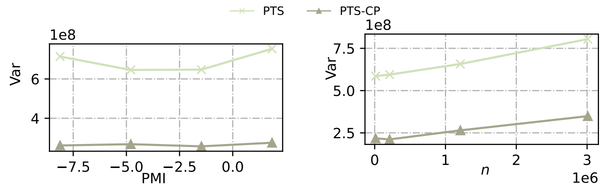
$$RMSE = \sqrt{\frac{1}{|\mathcal{C}| \cdot |\mathcal{I}|} \sum_{C \in \mathcal{C}} \sum_{I \in \mathcal{I}} (\hat{f}(C, I) - f(C, I))^2},$$

where $\mathcal{C}$ is the class domain, $\mathcal{I}$ is the item domain, and $|\cdot|$ denotes the set size.

To compare the mined top-k items $\mathcal{I}^m$ with the ground truth $\mathcal{I}^g = \{I_1^g, I_2^g, \cdots, I_k^g\}$ within each class, we use the same metrics as in PEM [24]: F1 Score [33] and Normalized Cumulative Rank (NCR) [34]. Notably, since precision equals recall in this context, the F1 Score evaluates the ratio of mined true positive items. The NCR measures the quality of the mined items and is defined as $NCR = \frac{2 \sum_{I_i \in \mathcal{I}^m} q(I_i)}{k(k+1)}$, where $q(I_1^g) = k, q(I_2^g) = k-1, \cdots, q(I_k^g) = 1$. To obtain an overall assessment, we average the F1 Score and NCR across the classes.

C. Variance Analysis

To investigate the impact of label-item correlations on variance, we conduct experiments on the SYN1 dataset. We fix item amount and class amount ($f(I) = n = 1.111 \times 10^6$), and vary label-item frequencies ($f(C, I) \in \{10^3, 10^4, 10^5, 10^6\}$) in each class. The variance is computed as $Var(\hat{f}(C, I)) = \frac{1}{t}(\hat{f}(C, I) - f(C, I))^2$ [9] with $\epsilon = 1$ and experimental time $t = 1000$. And correlation strength measure PMI [28] is calculated according to $f(C, I)$ within one class. PTS refers to the framework PTS with GRR and OUE, while PTS-CP denotes the PTS framework with our improved mechanism, correlated perturbation (CP). As shown in Fig. 5(a), despite the increase in PMI, the observed variance exhibits negligible variation, empirically confirming that changes in correlation strength are concealed in variance due to the dominance of class amount n and data amount N. To examine class distribution effects, we utilize the SYN2 dataset with fixed label-item frequency $f(C, I) = 10^4$ for an item across the classes, and varying class amount $n \in \{1.3\times10^4, 2.11\times10^5, 1.21\times10^6, 3.01\times10^6\}$. As shown in Fig. 5(b), the results demonstrate a strong positive correlation between class amount n and variance magnitude. These quantitative relationship directly validates the theoretical analysis derived in Section V-C.

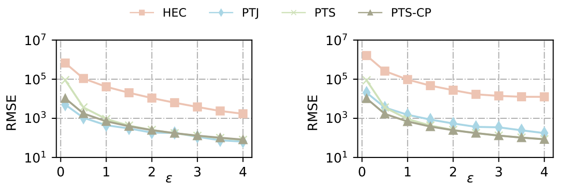


(a) Varying correlation strength. (b) Varying class amount n.

Fig. 5. Empirical variance analysis.

D. Results of Frequency Estimation

For frequency estimation, the multi-class item mining frameworks incorporate the state-of-the-art mechanism, the adaptive mechanism, which adaptively runs OUE and GRR according to the item domain size d [9]. Specifically, if $d > 3e^{\epsilon} + 2$, OUE is employed to decrease the variance. The HEC and PTJ frameworks use the adaptive mechanism. In the PTS framework, where the label domain is small and the item domain is large, GRR is applied for label perturbation, while OUE is used for item perturbation. Since PTJ does not produce any invalid data in this task, correlated perturbation can only be integrated with the PTS framework (designated as PTS-CP), where the privacy budgets are set as $\epsilon_1 = \epsilon_2 = \epsilon/2$. The details have been derived in Section VI-A. Comparative results are presented in Fig. 6.



(a) Diabetes Dataset. (b) Heart Disease Dataset.

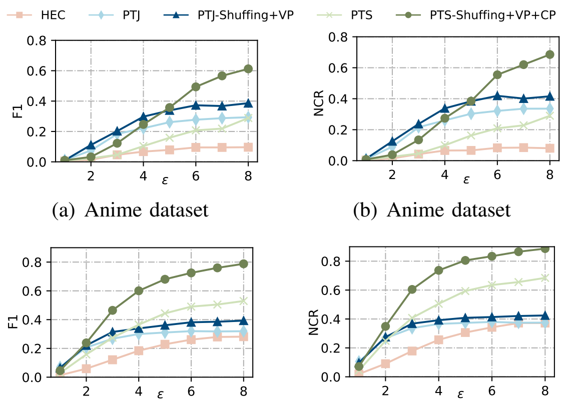
Fig. 6. RMSE from two real-world datasets with varying privacy budget ϵ.

As shown in Fig. 6, PTJ and PTS significantly outperform HEC. Correlated perturbation (PTS-CP) notably enhances utility compared with the PTS framework, particularly when the privacy budget ϵ is small. As the privacy budget increases, label perturbation results in fewer instances of invalid data, further improving the utility of PTS. PTJ, in particular, performs the best on the Diabetes dataset, benefiting from the utilization of the whole privacy budget and the OUE mechanism [9]. However, despite this advantage, PTJ incurs the highest communication load due to its expanded perturbation domain that combines both label and item domains.

E. Results of Top-k Item Mining

In this section, we present the results of top-k item mining. For comparison, we employ the state-of-the-art top-k item mining scheme, PEM [24], on multi-class item mining frameworks, compared with our optimized methods, including validity perturbation (VP), correlated perturbation with the globally frequent items (CP), and the shuffling method tailored for top-k mining scheme (Shuffling). As for the parameters, we empirically set $\epsilon_1 = \epsilon_2 = \epsilon/2$, and $IT_f = int(IT/2)$ in Algorithms 1 and 2. We choose $a = 0.2$, meaning that one-fifth of the data will be used for generating global candidates, while the remaining data is used for top-k item mining. The parameter b is set to 2, meaning that correlated perturbation is only applied when the collected class amount is less than twice the estimated class amount. We examine the impact of varying the privacy budgets, k values, and class numbers, along with an ablation study of the optimized methods.

The results of epsilon varying are shown in Fig. 7, with k fixed as 20. All methods improve as the privacy budget increases, with PTS-based methods demonstrating a more significant enhancement. Moreover, the optimized methods consistently outperform the initial frameworks. For the Anime dataset, the optimized methods on the PTS framework achieve average improvements of 116.6% in F1 Score and 134.3% in NCR, while the optimized methods on the PTJ framework yield average improvements of 25.6% in F1 Score and 25.0% in NCR Score. In the JD dataset, the optimized methods on the PTS framework achieve average improvements of 55.6% in F1 Score and 40.8% in NCR Score, whereas the optimized methods on the PTJ framework enhance the F1 Score by 13.6% and the NCR by 6.9% on average.



(a) Anime dataset (b) Anime dataset

(c) JD dataset (d) JD dataset

Fig. 7. Results from real-world datasets with $k = 20$ and varying ϵ.

We also investigate the performance in each class. For sake of brevity, we only present the F1 Score on JD dataset with $\epsilon = 8$ and $k = 20$ in Fig. 8, as the NCR Score follows a similar trend. The results match the data size in each class, with instance counts of 850k, 4m, 3m, 314k, and 170k, respectively. Classes 2 and 3 contain significantly more data than the others, and the data volumes in classes 4 and 5 are insufficient to yield reliable results. The PTS framework can benefit from the globally frequent items, even those items with falsely perturbed labels from other classes. Conversely, PTJ can not utilize the global information, and thus fails to produce results in classes 4 and 5.

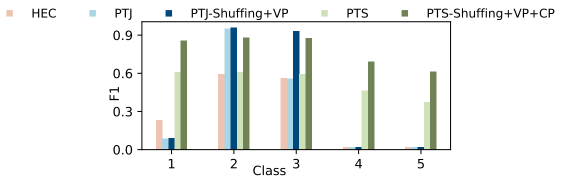


Fig. 8. Results of each class on JD dataset with $\epsilon = 8$ and $k = 20$.

In addition to examining the impact of varying privacy budgets, we also explore the influence of the top-k setting, with k ranging from 10 to 50, and the privacy budget is set as $\epsilon = 4$. For brevity, we present the results of JD dataset in Fig. 9. As k increases, the utility of PTS decreases, as less frequent items become harder to detect. In contrast, the utility of PTJ improves since a larger k results in a much larger candidate set, allowing more candidate label-item pairs to be investigated.

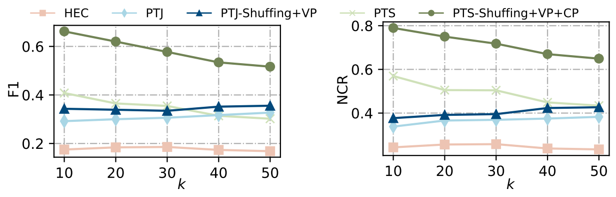


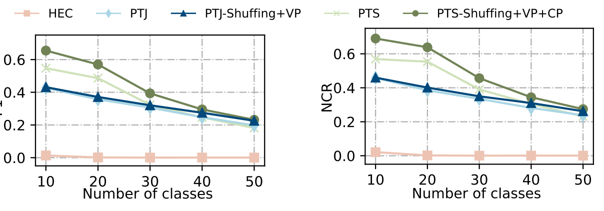
Fig. 9. F1 Score and NCR on JD dataset with $\epsilon = 4$ and varying k.

Furthermore, we examine the impact of varying class numbers using two synthetic datasets, SYN3 and SYN4. SYN3 contains globally frequent items, whereas SYN4 is generated using the same method but excludes these items. The task is to identify the top-20 items with a privacy budget of $\epsilon = 4$. The results are shown in Fig. 10. Notably, the utility of all methods declines as the class number increases, and all optimized methods perform better than the original frameworks. Without access to globally frequent items, the utility of PTS degrades significantly in SYN4. In contrast, the results of PTJ on both synthetic datasets are similar, indicating that the PTJ framework does not benefit from the inclusion of globally frequent items. Moreover, the PTJ framework suffers from higher communication costs due to the expanded domain size.

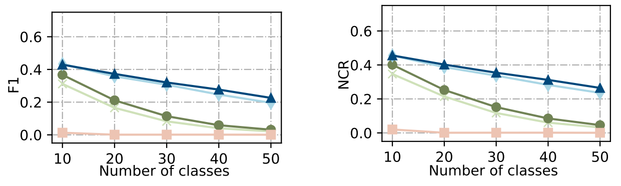
To further analyze the impact of the optimized methods, we conduct an ablation study, fixing $k = 20$ and $\epsilon = 5$. For brevity, we present only the results from the Anime dataset. As shown in Table III, all optimized methods demonstrate effectiveness, with improvements being particularly pronounced in the PTS framework. This enhancement in PTS stems from its ability to leverage global information across classes with correlated perturbation. Although HEC can also utilize global information, it lacks the ability to differentiate class-specific details at the final stage. We also observe that the shuffling method and validity perturbation enhance both the PTS and PTJ frameworks effectively.

F. *Impacts of Parameters*

In this section, we investigate the impact of parameters, including the settings of privacy budget allocation, and the parameters a and b in Algorithms 1 and 2. In the interest of space, we only present the results of the F1 score, as the NCR follows a similar trend. As aforementioned, the privacy budget is divided into two parts: ϵ_1 for label perturbation and ϵ_2 for item perturbation. As both item and label perturbation affects the aggregation of class-specific items, the perturbations for the both are crucial. To investigate the impact of privacy budget allocation, we conduct experiments on the synthetic dataset SNY4 with 5, 10, and 20 classes, respectively. The proportion of ϵ_1, denoted by p, varies from 0.1 to 0.9. The results are shown in Fig. 11. We observe that F1 Score increases and then decreases with p, which is consistent with our analysis as above. The best p lies between 0.4 and 0.6, and does not influence the results significantly. Therefore, we empirically set the proportion to 0.5 in our paper, i.e., $\epsilon_1 = \epsilon_2 = \epsilon/2$.



(a) SYN3 dataset (Global) (b) SYN3 dataset (Global)



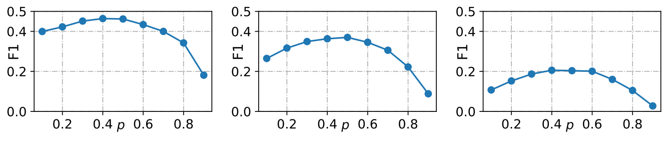
(c) SYN4 dataset (d) SYN4 dataset

Fig. 10. Results on synthetic datasets with $\epsilon = 4$, $k = 20$, and varying class numbers.

TABLE III
ABLATION STUDY ON PTJ AND PTS

PTJ+	(Baseline)	VP	Shuffling	All optimizations
F1	0.261	0.280	0.316	0.340
NCR	0.303	0.326	0.360	0.387

PTS+	(Baseline)	Global	VP	Shuffling	All optimizations
F1	0.159	0.165	0.214	0.241	0.358
NCR	0.163	0.180	0.229	0.270	0.385



(a) 5 Classes (b) 10 Classes (c) 20 Classes

Fig. 11. Varying the proportion of privacy budget allocation.

Parameter a (Algorithm 1 Line 2) controls the sampled group size used to identify item candidates, including the globally frequent items during the initial iterations. Given a large value of a, there may be insufficient data for top-k item mining, while for a small value of a, it may fail to generate reasonable candidates. To figure out the impact of parameter a, we conduct experiments using varying values of a on the real-world datasets JD and Anime. The results are shown in Figs. 12(a) and 12(b). Clearly, the impact of parameter a depends on the distribution of the dataset. For simplicity, we choose $a = 0.2$ in our experiments, meaning that one-fifth of the data is sampled to generate the global candidates, while the remaining data is used for top-k item mining. We investigate the parameter b (Algorithm 2 Line 8) in a similar manner. The data amount within different classes varies a lot. After applying DP perturbations, classes with less data are injected with a significant amount of noise (i.e., items) from other classes. To estimate the noise level for each class, we use the labels perturbed during the global candidates aggregation. If the injected noise is too large (or specifically, the collected data amount exceeds b times the estimated class data amount), the noise level becomes excessive. In this case, the amount of valid data may be insufficient for accurate estimation, rendering the correlated perturbation mechanism infeasible. To study the impact of parameter b, we conduct experiments with varying b on the real-world datasets JD and Anime. The results are shown in Figs.12(c) and 12(d). Although the results are dependent on the dataset, they do not fluctuate significantly. In our experiments, we choose $b = 2$ as the default value.

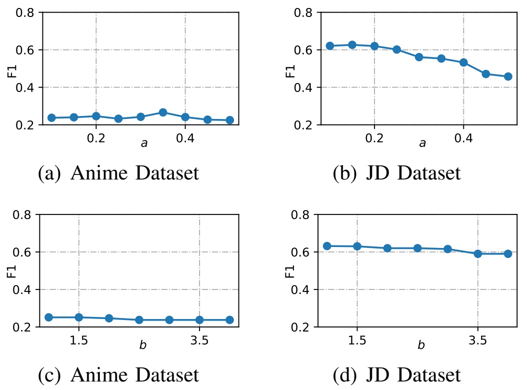


Fig. 12. Varying the parameters a and b.

VIII. Related Works

In this section, we review existing works on local differential privacy, with a focus on frequency estimation and top-k item mining.

Local Differential Privacy and Its Applications. Differential privacy was introduced as a privacy-preserving technique with theoretical guarantee for data collection [3]. However, differential privacy requires a trusted third party to aggregate data from users which is not feasible in many real-world scenarios. To this end, local differential privacy (LDP) is proposed [4], [5]. In the context of LDP, users will perturb their data locally before uploading to the untrusted server. Since the advent of LDP, it has been widely applied to various statistical collection tasks, including mean and frequency estimations [6], [8], [9], [12].

Local Differential Privacy Item Mining. To meet the growing demand for item mining, researchers have proposed various mechanisms for frequency estimation and top-k item mining. RAPPOR was developed to aggregate frequencies of categorical input items [6]. Subsequently, Optimal Unary Encoding (OUE) and Optimal Local Hashing (OLH) were introduced to achieve unbiased frequency estimation with optimal variance [9]. These two mechanisms remain state-of-the-art for frequency estimation, with OUE being more commonly adopted due to its ease of implementation, finding application in various downstream tasks [20]–[22]. In terms of top-k item mining, tree-based data structures are often preferred due to their efficiency in performing multi-iteration pruning and estimation [24], [25], [31]. The PEM algorithm [24], recognized as a state-of-the-art method, is designed to mine top-k items from large categorical datasets. Building on PEM, Zhu et al. [35] proposed additional mechanisms to enhance top-k mining in set-valued datasets. Recent advancements include Du et al.'s [36] adaptive sampling technique for set-valued data to enhance utility, and Li et al.'s [37] application of the HeavyGuardian data structure for mining top-k items in bounded-memory data streams.

The aforementioned item mining mechanisms rely exclusively on global statistics from the entire dataset, without incorporating class information. While integrating class information can benefit real-world applications such as recommendation systems, it requires a higher privacy budget to ensure privacy guarantees. To date, only label differential privacy (label-DP) has been proposed to protect labels [38], but it does not extend protection to other sensitive information. Label-DP assumes that feature information is non-sensitive, focusing solely on label protection — an assumption that limits its applicability to specific scenarios [39]. In multi-class item mining settings, where both items and labels are sensitive, label-DP fails to meet the necessary privacy requirements.

To the best of our knowledge, we are the first to investigate multi-class item mining under LDP, proposing foundational frameworks and corresponding optimized mechanisms to enhance utility.

IX. Conclusion and Future Work

In this paper, we propose two frameworks, PTJ and PTS, for multi-class item mining, accompanied by two optimized mechanisms as the perturbation module: validity perturbation mechanism and correlated perturbation mechanism. These optimized methods are applied to two types of item mining tasks: frequency estimation and top-k item mining in the multi-class setting. Our core idea is to handle invalid data while preserving the relationship between labels and items. Additionally, we derive unbiased frequency estimation and optimize the top-k item mining scheme. Theoretical analysis and experimental results validate the effectiveness and superiority of the proposed mechanisms.

As for future work, we aim to study multi-class item mining on more data types, such as numerical items. Additionally, we plan to extend this study to real-world mining applications, including gradient descent optimization and k-means clustering.

References

[1] Turning big data into big money: How amazon is leveraging big data. *Linkedin*, May 2, 2017

[2] A. Narayanan, and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, pages 111–125. IEEE, 2008.

[3] C. Dwork. Differential privacy. In *ICALP*, pages 1–12. Springer, 2006.

[4] S. Kasiviswanathan, H. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal On Computing*, 40(3):793–826, 2011.

[5] Q. Ye and H. Hu. Local differential privacy: Tools, challenges, and opportunities. In *WISE*, pages 13–23. Springer, 2020.

[6] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067. ACM, 2014.

[7] Differential Privacy Team, Apple. Learning with Privacy at Scale. https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf, 2017.

[8] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. Hui, H. Shin, J. Shin, and G. Yu. Collecting and analyzing multidimensional data with local differential privacy. In *ICDE*, pages 638–649. IEEE, 2019.

[9] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *USENIX Security Symposium*, pages 729–745, 2017.

[10] Q. Ye, H. Hu, X. Meng, and H. Zheng. PrivKV: Key-value data collection with local differential privacy. In *S&P*, pages 317–331. IEEE, 2019.

[11] T. Wang, N. Li, and S. Jha. Locally differentially private frequent itemset mining. In *S&P*, pages 127–143. IEEE, 2018.

[12] Z. Li, T. Wang, M. Lopuhaä-Zwakenberg, N. Li, and B. Škorickoric. Estimating numerical distributions under local differential privacy. In *SIGMOD*, pages 621–635. ACM, 2020.

[13] C. Cole, G. Laurent, A. Drolet, J. Ebert, A. Gutchess, R. Lambert-Pandraud, E. Mullet, M. Norton, and E. Peters. Decision making and brand choice by older consumers. *Marketing Letters*, 19(3):355–365, 2008.

[14] Z. Zhou, X. Jin, and Y. Fang. Moderating role of gender in the relationships between perceived benefits and satisfaction in social virtual world continuance. *Decision Support Systems*, 65:69–79, 2014.

[15] J. Lian and D. Yen. Online shopping drivers and barriers for older adults: Age and gender differences. *Computers In Human Behavior*, 37:133–143, 2014.

[16] J. Fang, C. Wen, B. George, and V. Prybutok. Consumer heterogeneity, perceived value, and repurchase decision-making in online shopping: The role of gender, age, and shopping motives. *Journal Of Electronic Commerce Research*, 17(2), 2016.

[17] I. Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence In Medicine*, 23(1):89–109, 2001.

[18] M. Ahsan and Z. Siddique. Machine learning-based heart disease diagnosis: A systematic literature review. *Artificial Intelligence In Medicine*, 128:102289, 2022.

[19] X. Ren, L. Shi, W. Yu, S. Yang, C. Zhao, and Z. Xu. LDP-IDS: Local differential privacy for infinite data streams. In *SIGMOD*, pages 1064–1077. ACM, 2022.

[20] Y. Du, Y. Hu, Z. Zhang, Z. Fang, L. Chen, B. Zheng, and Y. Gao. LDPTrace: Locally differentially private trajectory synthesis. *PVLDB*, 16(8):1897–1909, 2023.

[21] K. Huang, G. Ouyang, Q. Ye, H. Hu, B. Zheng, X. Zhao, R. Zhang, R and X. Zhou. LDPGuard: Defenses against data poisoning attacks to local differential privacy protocols. *IEEE Transactions On Knowledge And Data Engineering*, 36(7):3195–3209, 2024.

[22] Y. Hu, Y., Du, Z. Zhang, Z. Fang, L. Chen, K. Zheng, and Y. Gao. Real-time trajectory synthesis with local differential privacy. In *ICDE*, pages 1685–1698. IEEE, 2024.

[23] Tech. Rep., https://github.com/AddPaper/TechReport [Backup link: https://list.pa.ci/Code/TechReport.pdf]

[24] T. Wang, N. Li, and S. Jha. Locally differentially private heavy hitter identification. *IEEE Transactions On Dependable And Secure Computing*, 18(2):982–993, 2021.

[25] N. Wang, X. Xiao, Y. Yang, T. Hoang, H. Shin, J. Shin, and G. Yu. PrivTrie: Effective frequent term discovery under local differential privacy. In *ICDE*, pages 821–832. IEEE, 2018.

[26] X. Cao, J. Jia, and N. Gong. Data poisoning attacks to local differential privacy protocols. In *USENIX Security Symposium*, pages 947–964, 2021.

[27] N. Li, M. Lyu, D. Su, and W. Yang. Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, & Trust*, 8(4):1–138, 2016.

[28] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[29] J. Li, K. Cheng, S. Wang, F. Morstatter, R. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Computing Surveys*, 50(6):94, 2017.

[30] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[31] R. Bassily, K. Nissim, U. Stemmer, and A. Guha Thakurta. Practical locally private heavy hitters. In *NeurIPS*, pages 2285–2293, 2017.

[32] R. Agrawal and R. Ramakrishnan. Fast algorithms for mining association rules in large databases. *PVLDB*, pages 487–499, 1994.

[33] C. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. 2008.

[34] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions On Information Systems*. 20(4): 422–446, 2002.

[35] Y. Zhu, Y. Cao, Q. Xue, Q. Wu, and Y. Zhang. Heavy hitter edentification over large-domain set-valued data with local differential privacy. *IEEE Transactions On Information Forensics And Security*, 19:414–426, 2023.

[36] R. Du, Q. Ye, Y. Fu, H. Hu, and K. Huang. Top-k discovery under local differential privacy: An adaptive sampling approach. *IEEE Transactions On Dependable And Secure Computing*, 2024.

[37] X. Li, W. Liu, J. Lou, Y. Hong, L. Zhang, Z. Qin, and K. Ren. Local differentially private heavy hitter detection in data streams with bounded memory. In *SIGMOD*, 2(1):30, 2024.

[38] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, and C. Zhang. Deep learning with label differential privacy. In *NeurIPS*, pages 27131–27145, 2021.

[39] M. Malek Esmaeili, I. Mironov, K. Prasad, I. Shilov, and F. Tramer. Antipodes of label differential privacy: Pate and alibi. In *NeurIPS*, pages 6934–6945, 2021.