On Differentially Private Frequent Itemset Mining*

Chen Zeng
zeng@cs.wisc.edu

Jeffrey F. Naughton
naughton@cs.wisc.edu

Jin-Yi Cai
jyc@cs.wisc.edu

Department of Computer Science, University of Wisconsin-Madison, Madison, WI, 53706

ABSTRACT

We consider differentially private frequent itemset mining. We begin by exploring the theoretical difficulty of simultaneously providing good utility and good privacy in this task. While our analysis proves that in general this is very difficult, it leaves a glimmer of hope in that our proof of difficulty relies on the existence of long transactions (that is, transactions containing many items). Accordingly, we investigate an approach that begins by truncating long transactions, trading off errors introduced by the truncation with those introduced by the noise added to guarantee privacy. Experimental results over standard benchmark databases show that truncating is indeed effective. Our algorithm solves the "classical" frequent itemset mining problem, in which the goal is to find all itemsets whose support exceeds a threshold. Related work has proposed differentially private algorithms for the top-k itemset mining problem ("find the k most frequent itemsets".) An experimental comparison with those algorithms show that our algorithm achieves better F-score unless k is small.

1. INTRODUCTION

Recently, concomitant with the increasing ability to collect personal data, privacy has become a major concern. In this paper, we focus on privacy issues that arise in the context of finding frequent itemsets in "transactional" data. Frequent itemset mining is widely used in many applications, perhaps the best known of which is market basket analysis. The goal of frequent itemset mining in market basket analysis is to find sets of items that are frequently bought together, which is helpful in applications ranging from product placement to marketing and beyond. Developing efficient algorithms for frequent itemset mining has been widely studied by our community [15]. However, with the exception of the recent work in [5, 16], a differentially private approach to frequent itemset mining has received little attention.

A frequent itemset mining algorithm takes as input a dataset consisting of the transactions by a group of individuals, and produces as output the frequent itemsets. This immediately creates a privacy concern — how can we be confident that publishing the frequent itemsets in the dataset does not reveal private information about the individuals whose data is being studied? This problem is compounded by the fact that we may not even know what data the individuals would like to protect nor what background information might be possessed by an adversary. These compounding factors are exactly the ones addressed by *differential privacy* [9], which intuitively guarantees that the presence of an individual's data in a dataset does not reveal much about that individual. Accordingly, in this paper we explore the possibility of developing differentially private frequent itemset mining algorithms. Our goal is to guarantee differential privacy without obliterating the utility of the algorithm.

We quantify the utility of a differentially private frequent itemset mining algorithm by its likelihood to produce a complete and sound result. Intuitively speaking, "completeness" requires an algorithm to include all the sufficiently "frequent" itemsets, and "soundness" requires an algorithm to exclude all the sufficiently "infrequent" ones. We start by a theoretical investigation of the tradeoff between privacy and utility in frequent itemset mining. Our result unfortunately indicates that the problem is very hard — that is, in general, one cannot simultaneously guarantee high utility and a high degree of privacy.

However, a closer investigation of this negative result reveals that it relies on the possibility of very long transactions (that is, transactions with many items). This raises the possibility of improving the utility-privacy tradeoff by limiting transactions' cardinality. Of course, one cannot in general impose such a limit — so instead, we explore enforcing the limit by truncating transactions. That is, if a transaction has more than a specified number of items, we delete items until the transaction is under the limit. Of course, this deletion must be done in a differentially private way; perhaps equally important, while it reduces the error due to the noise required to enforce privacy, it introduces a new source of error by discarding items from transactions. Exploring the impact of this tradeoff is one of the contributions of our work. Our experimental results with four datasets indicate that truncating has a large positive impact on quality.

For ease of exposition, we go from this observation to a differentially private frequent itemset mining algorithm by

*This work was supported by National Science Foundation grants IIS-0524671, CCF-0914969 and NIGMS grant R01LM011028-02.

Proceedings of the VLDB Endowment, Vol. 6, No. 1

a series of steps. First, we explore how to find frequent 1-itemsets (itemsets with only one item). Next, we generalize this to find frequent β-itemsets (itemsets with β items). Finally, we generalize this to solve the full problem: in a differentially private way, find all the itemsets whose support exceeds a given threshold. We found that each of these steps was non-trivial and had to be done with care to ensure privacy without destroying utility.

Finally, we compare our work with interesting recent work on differentially private top-k frequent itemset mining [5, 16]. This problem is somewhat different from the one we address ("find all itemsets whose support exceeds a threshold") but it is similar enough to allow a comparison. For example, one can use our algorithm with a sufficiently low threshold to guarantee that k itemsets will be found to solve the top-k problem; or one can use a top-k algorithm with k set large enough to find all the itemsets whose support exceeds a threshold. Our experimental study with four benchmark datasets shows that our algorithm produces a result with a higher F-score than either of the top-k algorithms unless k is very small (e.g., $k \sim 10$).

The rest of the paper is organized as follows: Section 2 briefly describes the problem of frequent itemset mining, and the notion of differential privacy. Section 3 explores the trade-off between utility and privacy in frequent itemset mining. Section 4 proposes our differentially private frequent 1-itemset mining algorithm. Section 5 generalizes the idea of truncating transactions to frequent β-itemset mining. Section 6 extends our β-itemset mining algorithm to frequent itemset mining. Section 7 evaluates our algorithm on benchmark datasets. Section 8 discusses related work, and Section 9 concludes our work. Most of the proofs and the pseudocode of some algorithms can be found in the long version of our paper available at [1].

2. PRELIMINARIES

In this section, we review the problem of frequent itemset mining [2], and the notion of differential privacy [9].

2.1 Frequent Itemset Mining

We model a database τ as a vector in D^m, where each entry represents the information contributed by an individual from the domain D. In our context, the database in frequent itemset mining is called a *transaction database*.

DEFINITION 1. *(Transaction database): A* transaction database *is a vector of transactions $\tau = \langle t_1, \ldots, t_m \rangle$ where each transaction t_i is a subset of the alphabet $\mathcal{I} = \{1, \ldots, n\}$.*

The domain of a single transaction is thus the power set of the alphabet $\mathcal{I}$. In this paper, we use "database" as a shorthand for "transaction database." Each subset of the alphabet $\mathcal{I}$ is called an *itemset*. If the number of transactions containing an itemset exceeds a predefined threshold, then that itemset is called a *frequent itemset*.

DEFINITION 2. *(Frequent itemset): For any itemset X, the support of X in a database is the number of transactions containing X. If that number exceeds a predefined threshold λ, then X is called a* frequent itemset *with respect to the threshold λ.*

In the rest of this paper, we assume the threshold λ is given, and use the term "frequent itemsets" as a shorthand for "frequent itemsets with respect to the threshold" when the threshold is clear from the context. For ease of presentation, we denote "the itemsets of cardinality (number of elements) β" by "β-itemsets", and the query that computes the support of a β-itemset by a "β-itemset query." A β-itemset query is a count query which computes the number of transactions containing the given itemset.

2.2 Differential Privacy

Intuitively, *differential privacy* guarantees that the presence or absence of an individual's information has little effect on the output of an algorithm, and thus, an adversary can learn limited information about any individual. In our context, the information contributed by an individual is her transaction. More precisely, for any database $\tau \in D^m$, let $nbrs(\tau)$ denote the set of *neighboring databases* of τ, each of which differs from τ by at most one *transaction*. *Differential privacy* requires that the probabilities of an algorithm to output the same result on any pair of neighboring databases are bounded by a constant ratio.

DEFINITION 3. *(ϵ-differential privacy [9]): For any input database τ, a randomized algorithm f is ϵ-differentially private iff for any $\mathcal{S} \subseteq Range(f)$, and any database $\tau' \in nbrs(\tau)$,*

$$\Pr(f(\tau) \in S) \leq e^{\epsilon} \times \Pr(f(\tau') \in \mathcal{S})$$

where Pr *is the probability taken over the coin tosses of the algorithm f.*

One way to guarantee differential privacy for a count query is to perturb the correct result. In particular, Ghosh et al. [12] propose the *geometric mechanism* to guarantee ϵ-differential privacy for a single count query. The geometric mechanism adds noise Δ drawn from the two-sided geometric distribution $G(\epsilon)$ with the following probability distribution: for any integer σ,

$$\Pr(\Delta = \sigma) \sim e^{-\epsilon|\sigma|} \tag{1}$$

The geometric mechanism is a discrete variant of the well-studied Laplacian mechanism [10], which adds random noise drawn from the Laplacian distribution. To ensure differential privacy for multiple count queries, we first compute the *sensitivity* of those queries, which is the largest difference between the output of those queries on any pair of neighboring databases.

DEFINITION 4. *(Sensitivity): Given d count queries, $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$, the* sensitivity *of $\mathbf{q}$ is:*

$$S_{\mathbf{q}} = \max_{\forall \tau, \tau' \in nbrs(\tau)} |\mathbf{q}(\tau) - \mathbf{q}(\tau')|_1$$

Notice that the output of $\mathbf{q}$ is a vector of dimension d, and we use $|\mathbf{x} - \mathbf{y}|_p$ to denote the L_p distance between two vectors $\mathbf{x}$ and $\mathbf{y}$. The following theorem is a straightforward extension of the Laplacian mechanism to the geometric mechanism.

THEOREM 1. *Given d count queries $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$, for any database τ, the database access mechanism: $A_{\mathbf{q}}(\tau) = \mathbf{q}(\tau) + \langle \Delta_1, \ldots, \Delta_d \rangle$ where Δ_i is drawn i.i.d from the geometric distribution $G(\epsilon/S_{\mathbf{q}})$ (1), guarantees ϵ-differential privacy for $\mathbf{q}$.*

As proved in [10], a sequence of differentially private computations also ensures differential privacy. This is called the *composition property* of differential privacy as shown in Theorem 2.

THEOREM 2. *[10] Given a sequence of computations, denoted as* $\mathbf{f} = f_1,\ldots,f_d$*, if each computation* f_i *guarantees* ϵ_i*-differential privacy, then* $\mathbf{f}$ *is* $(\sum_{i=1}^{i=d} \epsilon_i)$*-differentially private.*

3. A TRADE-OFF BETWEEN PRIVACY AND UTILITY

In this section, we present a theoretical study on the trade-off between utility and privacy in differentially private frequent itemset mining. First, we formally define the utility of a frequent itemset mining algorithm.

3.1 Our Utility Model

Our definition of utility follows that proposed in [5]. Intuitively, if the support of an itemset is much larger than the threshold, then the result should include that itemset; on the other hand, if the support of an itemset is much smaller than the threshold, then that itemset should be excluded from the output. We specify two criteria to capture that intuition in Definition 5.

DEFINITION 5. *(δ-approximation): Given a database τ and a threshold λ, let $\mathcal{S}$ be the output of a frequent itemset mining algorithm on the database τ. We say $\mathcal{S}$ is* δ-approximate *iff the following two properties are satisfied:*

1. *(Completeness) Every itemset with support exceeding $(1+\delta)\lambda$ is in $\mathcal{S}$.*
2. *(Soundness) No itemset with support less than $(1-\delta)\lambda$ is in $\mathcal{S}$.*

We quantify the utility of a frequent itemset mining algorithm by its likelihood to produce a good approximate result as shown in Definition 6.

DEFINITION 6. *($(\delta,\eta)-usefulness$): A frequent itemset mining algorithm f is* (δ,η)-useful *iff for any database τ, with probability at least $1-\eta$, the output of f on τ is δ-approximate.*

Both δ and η are within the range $(0,1)$ by definition. Next, we quantify the trade-off between privacy and utility for frequent itemset mining.

3.2 A Lower Bound on the Privacy Parameter

Let us first consider a constrained frequent itemset mining problem in which we are only interested in frequent 1-itemsets. We call that problem *frequent 1-itemset mining*. Lemma 1 shows the lower bound on the privacy parameter ϵ if a frequent 1-itemset mining algorithm must be both (δ,η)-useful and ϵ-differentially private.

LEMMA 1. *For any frequent 1-itemset mining algorithm that is both ϵ-differentially private and (δ,η)-useful,*

$$\epsilon \geq \frac{\ln[(2^n-1)\eta']}{2\delta\lambda+2}$$

where $\eta' = (1-\eta)/\eta$.

Note that a (δ,η)-useful frequent itemset mining algorithm is also a (δ,η)-useful frequent 1-itemset mining algorithm. Thus, Lemma 1 implies Theorem 3.

THEOREM 3. *For any frequent itemset mining algorithm that is both ϵ-differentially private and (δ,η)-useful,*

$$\epsilon \geq \frac{\ln[(2^n-1)\eta']}{2\delta\lambda+2}$$

where $\eta' = (1-\eta)/\eta$.

In a typical transaction database like BMS-WebView-2 shown in [22], $n = 3340$. Therefore, if we set $\lambda = 310$, which is 0.4% of the total number of transactions, $\delta = 0.2$ and $\eta = 0.5$, then $\epsilon \geq 18.4$. That lower bound shows that no matter how sophisticated a differentially private frequent itemset mining algorithm is, in order to guarantee $(0.2, 0.5)$-usefulness, the privacy parameter ϵ must exceed 18.4, which suggests a huge risk of privacy breach. This is a discouraging result. However, the proof of Lemma 1 requires the existence of very long transactions. This observation motivated us to study the approach of eliminating the possibility of these long transactions by truncating them. In the next few sections we use this observation to develop a differentially private algorithm that works on a truncated database, then explore its performance.

4. FREQUENT 1-ITEMSET MINING

In this section, as the first step in the development of our frequent itemset mining algorithm, we consider the simpler problem of frequent 1-itemset mining. We will show that by truncating transactions, we can significantly promote the utility of frequent 1-itemset mining while still guaranteeing differential privacy.

4.1 Intuition for Truncating Transactions

By fixing the threshold λ, and setting the utility parameters δ, η to constants, Lemma 1 suggests that we cannot set the privacy parameter for frequent 1-itemset mining to any value that is not $\Omega(n)$. An intuition for this is that the addition of a "long" transaction can drastically change the result of the mining algorithm. However, we will show, by limiting the maximal cardinality of transactions, there exists a frequent 1-itemset mining algorithm that is both (δ,η)-useful and ϵ-differentially private provided $\epsilon \geq \Omega(\log n)$. We construct that algorithm by utilizing the geometric mechanism.

We can formulate the problem of frequent 1-itemset mining by first computing n 1-itemset queries $\mathbf{q} = \langle q_1,\ldots,q_n\rangle$, where each q_i computes the support of the 1-itemset $\{i\}$, and then selecting those 1-itemsets whose support exceeds the threshold. We observe that as long as we compute $\mathbf{q}$ in a differentially private way, the frequent 1-itemset mining algorithm will be differentially private. Theorem 4 shows that the sensitivity of computing $\mathbf{q}$ is equal to the maximal cardinality of transactions.

THEOREM 4. *Let ℓ be the maximal cardinality of transactions. The sensitivity of computing n different 1-itemset queries is ℓ.*

For ease of presentation, we refer to the frequent 1-itemset mining algorithm, which first adds geometric noise to the support, and then checks the threshold, as the "geometric

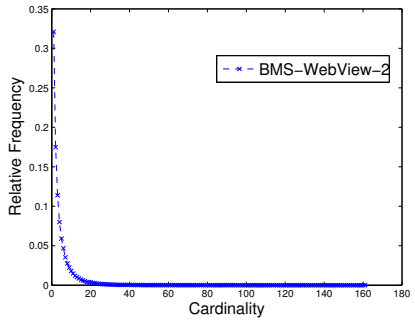


Figure 1: Frequencies of Transactions by Cardinality

noise algorithm." We will prove that by assuming the maximal cardinality $\ell = \mathrm{O}(1)$, the geometric noise algorithm guarantees both ϵ-differential privacy and (δ, η)-usefulness provided $\epsilon \geq \Omega(\log n)$. This is shown in Theorem 5.

THEOREM 5. *Given the maximal cardinality ℓ, where $\ell = \mathrm{O}(1)$, then the geometric noise algorithm is both (δ, η)-useful and ϵ-differentially private provided $\epsilon \geq \Omega(\log n)$.*

We can also prove that the geometric noise algorithm is optimal as shown in Theorem 6.

THEOREM 6. *For any frequent 1-itemset mining algorithm that is both (δ, η)-useful and ϵ-differentially private, ϵ must be $\Omega(\log n)$ provided the maximal cardinality $\ell = \mathrm{O}(1)$.*

Both Theorem 5 and Theorem 6 suggest that the constraint on the maximal cardinality of transactions has a significant impact on the utility of a differentially private frequent 1-itemset mining algorithm. This suggests that we can improve the utility/privacy tradeoff by enforcing a maximal cardinality on transactions. We do so by truncating transactions.

4.2 Rationale in Truncating Transactions

Our idea of limiting the maximal cardinality of transactions is simple — we truncate a transaction whose cardinality violates that constraint by only keeping a subset of that transaction. Of course, that truncating approach incurs certain information loss. However, if the cardinality of transactions in a dataset follows a distribution in which most are "short" and a few are "long", then these few "long" transactions, while having little impact on which itemsets are frequent, have a major effect on the sensitivity. If this is the case, perhaps we can reduce the sensitivity while still finding a relatively accurate set of frequent 1-itemset by truncating the long transactions. The three benchmark datasets [22] do follow such a distribution. Figure 1 illustrates the correlation between frequency and the cardinality of transactions in one of those datasets, BMS-WebView-2. As we can see, the short transactions, which contain fewer than 10 items, dominate the datasets. The other two datasets, BMS-WebView-1 and BMS-POS, also have a similar distribution. Therefore, we hope that the reduction in the magnitude of noise can offset the information loss incurred by truncating, and thus, we can get accurate results for frequent 1-itemsets.

However, the truncating approach raises a privacy concern: does it violate differential privacy? We will show that as long as that transformation is *local*, by which we mean that the output only depends on the input transaction, then applying any ϵ-differentially private algorithm to the truncated database also guarantees ϵ-differential privacy for the original database. The notion of *local transformation* is formulated in Definition 7.

DEFINITION 7. *(Local Transformation): A* local transformation *is a probabilistic function r: $2^{\mathcal{I}} \to 2^{\mathcal{I}}$ such that for any $t \subseteq \mathcal{I}$*

$$\sum_{t' \subseteq \mathcal{I}} \Pr(r(t) = t') = 1$$

In the rest of this paper we use "transformation" as a shorthand for "local transformation", and $r(\tau)$ to denote the computation that applies the transformation r to every transaction in the database τ. Theorem 7 proves that applying any differentially private algorithm to a transformed database also guarantees differential privacy.

THEOREM 7. *Let r be an arbitrary local transformation, and f be an ϵ-differentially private algorithm. Then for any pair of neighboring databases τ and τ', and any $\mathcal{S} \subseteq Range(f)$,*

$$\Pr(f(r(\tau)) \in \mathcal{S}) \leq e^{\epsilon} \Pr(f(r(\tau')) \in \mathcal{S})$$

Next, we will present our differentially private frequent 1-itemset mining algorithm.

4.3 Our Algorithm

Algorithm 1 TRUNCATEDATABASE

Input: input database τ; privacy parameter ϵ
Output: truncated database

1: $\langle z_1, \ldots, z_n \rangle$ = ESTIMATEDISTRIBUTION(τ, ϵ)
2: $\ell \leftarrow$ the smallest integer such that $\sum_{i=1}^{\ell} z_i \geq 0.85$
3: $\tau' = \emptyset$
4: **for** each transaction $t \in \tau$ **do**
5: add t' = TRUNCATE(ℓ, t) to τ'
6: **end for**
7: **return** τ'
8: **function** ESTIMATEDISTRIBUTION(τ, ϵ)
9: Let $\mathbf{z} = \langle z_1, \ldots, z_n \rangle$ where z_i is the # of transactions with cardinality i in τ
10: $\mathbf{z}' = \mathbf{z} + \langle \Delta_1, \ldots, \Delta_d \rangle$ where Δ_i is drawn i.i.d. from the geometric distribution $G(\epsilon)$ in (1).
11: **return** $\mathbf{z}'/m$ where m is the # of transactions in τ
12: **end function**
13: **function** TRUNCATE(ℓ, t)
14: $h = \min\{\ell, |t|\}$
15: $t' = \{$Randomly pick h items from $t.\}$
16: **return** t'
17: **end function**

We determine the maximal cardinality ℓ in a heuristic way in which we set ℓ to the value such that the percentage of the transactions with cardinality no greater than ℓ is at least 85%. That heuristic approach requires us to compute the percentage of transactions for each cardinality, which also has privacy implications, and thus, we add geometric noise to that computation. More precisely, let $\mathbf{z} = \langle z_1, \ldots, z_n \rangle$, where z_i is the number of transactions with cardinality i. It is not hard to show that the sensitivity of computing $\mathbf{z}$ is 1 since the addition or deletion of a single transaction can at

most increase or decrease a single z_i by 1. By Theorem 1, adding geometric noise $G(\epsilon)$ in computing $\mathbf{z}$ gurantees ϵ-differential privacy. Since that information is differentially private, it is safe to utilize that information. This is shown in the function "EstimateDistribution" in Algorithm 1.

We impose the cardinality constraint by randomly truncating transactions: if the cardinality of a transaction violates that constraint, then only a subset of its items is picked at random without replacement to generate a new transaction whose cardinality is equal to the enforced maximal cardinality. This is shown in the function "Truncate" in Algorithm 1.

Algorithm 2 F1M(τ, ϵ, λ)

Input: input database τ; privacy parameter ϵ; threshold λ
Output: frequent 1-itemsets

1: $\epsilon' = \min\{0.05, \epsilon/10\}$
2: $\tau' = \text{TruncateDatabase}(\tau, \epsilon')$
3: $\ell =$ the maximal cardinality of transactions in τ'
4: $R = \emptyset$
5: **for all** 1-itemset X in the alphabet $\mathcal{I}$ **do**
6: $\quad X.supp' = i$'s support in $\tau' + G((\epsilon - \epsilon')/\ell)$
7: $\quad$ **if** $X.supp' \geq \lambda$ **then**
8: $\quad\quad$ Add X to R
9: $\quad$ **end if**
10: **end for**
11: **return** R

Our differentially private *f*requent *1*-itemset *m*ining algorithm F1M is shown in Algorithm 2. Line 1 in Algorithm 2, which sets $\epsilon' = \min\{0.05, \epsilon/10\}$, is a configurable parameter that we set by trial and error for our datasets. We prove that F1M guarantees ϵ-differential privacy in Theorem 8.

Theorem 8. F1M *is ϵ-differentially private.*

5. FROM 1-ITEMSETS TO β-ITEMSETS

In this section, as the next step toward our complete algorithm, we generalize the 1-itemset mining algorithm to consider frequent β-itemset mining, in which we are interested in frequent itemsets with cardinality not exceeding β.

5.1 Challenges

There are two main challenges in frequent β-itemset mining that were not present in frequent 1-itemset mining:

1. Complexity: It is inefficient to compute the support of all itemsets as there are many.
2. Privacy: It is hard to precisely quantify the relationship between the maximal cardinality of transactions and the sensitivity of multiple i-itemset ($i = 2, \ldots \beta$) queries.

5.1.1 Complexity

Recall our frequent 1-itemset mining algorithm in which we first compute the support of all 1-itemsets, and then perturb their support by adding geometric noise. That algorithm is efficient since the maximal number of 1-itemsets is n — the size of the alphabet. However, it is not practical to extend that idea directly to frequent β-itemset mining since the number of itemsets grows combinatorially. More precisely, it is not hard to show that the number of all the itemsets is $\sum_{i=1}^{\beta} \binom{n}{i}$. In particular, $n = 1657$ in the dataset BMS-POS, and thus, when $\beta = 5$, the total number of all the itemsets is approximately 10^{15}!

In this paper, we attack the complexity problem by utilizing the well-known *a priori property* proposed in [2]. The a priori property states that a β-itemset is frequent only if all its subsets of cardinality $\beta - 1$ are frequent. For ease of presentation, we denote the "subset of cardinality $\beta - 1$" by "$(\beta - 1)$-subset." Our frequent β-itemset mining algorithm utilizes the a priori property by iteratively finding frequent itemsets in order of increasing cardinality. For the mining of frequent i-itemsets, we will only compute the support of the i-itemsets whose every $(i-1)$-subset is frequent. In accordance with [2], we denote those i-itemsets by *candidate i-itemsets*. The order of computation is well-defined since we have already presented our frequent 1-itemset mining algorithm in Algorithm 2. The generation of candidate i-itemsets is also safe since it relies on the noisy results of $(i-1)$-itemsets, which are already differentially private.

5.1.2 Privacy

By the a priori property, we only need to compute the support of the candidate i-itemsets ($i = 2, \ldots, \beta$). Similar to frequent 1-itemset mining, we can model that computation by d i-itemset queries $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$ where q_j computes the support of the candidate i-itemset C_j. To utilize the geometric mechanism, we need to compute the sensitivity of $\mathbf{q}$. However, we will show that given the maximal cardinality ℓ ($2 \leq \ell \leq n$), the precise computation of $\mathbf{q}$'s sensitivity is NP-hard. First, we formulate the problem of computing the sensitivity of multiple i-itemset queries in Problem 1.

Problem 1. *(i, ℓ)-sensitivity: Given a set of itemsets $\mathcal{C}$, each element of which is a subset of the alphabet $\mathcal{I}$, and is of the same cardinality i, find a set $\mathcal{T} \subseteq \mathcal{I}$ such that $|\mathcal{T}| \leq \ell$, and the number of itemsets in $\mathcal{C}$ contained by $\mathcal{T}$ is maximized.*

The number of itemsets in $\mathcal{C}$ contained in $\mathcal{T}$ is the sensitivity of $\mathbf{q}$. As indicated by Theorem 4, when $i = 1$, $(1, \ell)$-sensitivity can be trivially solved in polynomial time. However, this is not true for $i > 1$, as is shown in Theorem 9.

Theorem 9. *When $i \geq 2$, (i, ℓ)-sensitivity is NP-hard.*

In view of the hardness result for (i, ℓ)-sensitivity, in this paper we employ a *safe* approximate method by computing the upper bound of multiple i-itemset queries' sensitivity, and use that upper bound to perturb the support of the i-itemsets. The upper bound is shown in Theorem 10.

Theorem 10. *Given d i-itemset queries $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$, let ℓ be the maximal cardinality of transactions. Then the sensitivity of $\mathbf{q}$ is no greater than $\min(\binom{\ell}{i}, d)$.*

We want to emphasize that Theorem 10 only computes the upper bound of multiple i-itemset queries instead of the exact sensitivity. To show that, suppose the itemsets are {2,3}, {4,5}, and {6,7}, and $\ell = 3$. The sensitivity of those three 2-itemset queries is one instead of three since the addition or deletion of any transaction of cardinality three can only increase or decrease the support of one itemset by one. We use Theorem 10 to perturb the support of the candidate i-itemsets. Although our approach is not optimal with respect to the utility — we add more noise than required to guarantee differential privacy — our approach is safe because we use an upper bound on the sensitivity.

5.2 A Naïve Algorithm

By utilizing the a priori property and Theorem 10, we can extend our idea of truncating transactions to frequent β-itemset mining: we set the truncated database produced for frequent 1-itemset mining as the input database, and then by the a priori property, iteratively compute frequent 2, 3, $\ldots, \beta$-itemsets. The pseudocode of the naïve algorithm is shown in the long version of our paper [1]. However, we find that the performance of this naïve algorithm is poor for two reasons:

Random Truncating: Random truncating does not distinguish between frequent subsets and rare subsets of a transaction to be truncated. Yet frequent subsets are more likely to contribute to frequent item sets than rare ones.

Propagated Errors: If a frequent itemset is mistakenly labeled as infrequent, then any of its supersets is regarded infrequent without even computing its support.

To ameliorate these two problems, we propose two heuristic methods: *smart truncating*, and *double standards*. We discuss those two methods next.

5.3 Smart Truncating

Ideally, when truncating a transaction, we only need to keep the subsets that are frequent items since infrequent subsets will not contribute to frequent itemsets. However, our goal is to find those frequent itemsets in a differentially private way. We do so by providing a heuristic method to predict whether or not a candidate itemset is frequent. We observe that in practice, if all the subsets of a candidate itemset are "sufficiently" frequent, then that itemset is very likely to be frequent. To quantify that observation, we assign each candidate i-itemset ($i \geq 2$) a *frequency score* which is the summation over all its $(i-1)$-subsets' noisy support.

DEFINITION 8. *(Frequency Score): Given a set of $(i-1)$-itemsets $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$, the frequency score of an i-itemset X is:*

$$fs(X) = \sum_{Y_j \subset X \wedge Y_j \in \mathcal{Y}} Y_j.supp' \tag{2}$$

where $Y_j.supp'$ is the noisy support of the itemset Y_j.

When truncating transactions, we will keep the itemsets with high frequency scores. This is formulated in Problem 2.

PROBLEM 2. *Optimal (i,ℓ)-truncating: Given a set of i-itemsets $\mathcal{X} = \{X_1, \ldots, X_d\}$, the cover score of a set t' is defined as:*

$$cs(t') = \sum_{X_j \subseteq t' \wedge X_j \in \mathcal{X}} fs(X_j)$$

where fs is the frequency score defined in (2). Given a transaction t, find an ℓ-subset of t such that the cover score of that subset is maximized

Unfortunately, we can prove that there is no efficient algorithm to solve the optimal (i,ℓ)-truncating problem as shown in Theorem 11.

THEOREM 11. *Optimal (i,ℓ)-truncating is NP-hard.*

In view of the hardness result, we use a greedy algorithm to solve Problem 2.

5.3.1 Our Greedy Algorithm

The idea of our greedy algorithm is simple: for each transaction, we construct the truncated transaction by iteratively adding items in the candidate itemset that are both contained by the input transaction and have the highest frequency score until the truncated transaction's cardinality exceeds the maximal cardinality. However, an intuitive observation is that the frequency score of an itemset should not be static: it is possible that some items in an itemset have been added to the truncated transaction, and thus, the number of extra items to include that itemset is less than other itemsets. Therefore, the frequency score of an itemset should be updated with the addition of items to the truncated transaction. To avoid confusion with the static frequency score in Definition 8, we refer the "dynamic frequency score" by "weight."

Our greedy algorithm "SMARTTRUNCATING" accepts three input parameters: the original transaction t, the maximal cardinality ℓ, and the set of candidate i-itemsets $\mathcal{C} = \{C_1, \ldots, C_d\}$. For each candidate itemset C_j, its weight $C_j.weight$ is initialized by C_j's frequency score $fs(C_j)$ defined in (2). The algorithm "SMARTTRUNCATING" works as follows: we find the candidate i-itemsets contained by the input transaction t, and denote the set of those itemsets by $\mathcal{C}'$. Then, starting from an empty transaction t', we first pick the itemset C_j in $\mathcal{C}'$ with the highest weight, add the items in C_j to t', and delete C_j from $\mathcal{C}'$. Next, we update the weight of the remaining itemsets in $\mathcal{C}'$: for each remaining itemset C_h, we compute the average weight of a single item in C_h by $\alpha_h = fs(C_h)/i$. Suppose the number of items in C_h that has already been added to the truncated transaction t' is β_h. Then the weight of C_h is updated by $C_h.weight = C_h.weight + \alpha_h * \beta_h$. After updating the weight for every remaining itemset, we repeat those steps until the cardinality of t' exceeds ℓ. The pseudocode of SMARTTRUNCATING is shown in the long version of our paper. We show a running example of our greedy algorithm in Example 1.

EXAMPLE 1. *Given a transaction t = {1,2,3,4,5}, and three 2-itemsets {1,2}, {2,3} and {4,5} with weight 10, 8, and 9, respectively. Suppose the maximal cardinality of transactions is 3. The truncated transaction t' is initialized to be empty. First, we pick the itemset {1,2} which has the largest weight, and then t' is updated to {1,2}. Next, we update the weight of the remaining itemsets as follows: since the single item "2" has been added to t', the new weight of the itemset {2,3} is 8 + 1*8/2 = 12, and that for {4,5} remains the same. Hence, we add the itemset {2,3} to t', which is then updated to {1,2,3}. Since the cardinality of t' meets the constraint, the algorithm "SMARTTRUNCATING" stops, and the truncated transaction is t' = {1,2,3}.*

It is not hard to see that our smart truncating method is a *local transformation* which only relies on the noisy support, and so it is differentially private.

5.4 Double Standards

As discussed in Section 5.2, by the a priori property, if a frequent itemset is mistakenly labeled as infrequent, then any of its supersets is regarded infrequent without even computing its support. To alleviate this problem, we observe that a frequent itemset indeed serves two different roles — as a result of frequent itemset mining and as a "seed" to generate candidate itemsets. The errors are propagated due

to the second role. Therefore, instead of setting the same threshold for both roles for every itemset, we customize the thresholds for each itemset by setting a threshold to determine whether that itemset is frequent or not, and a possibly different one to decide whether or not to use that itemset to generate candidate itemsets. Speaking intuitively, the previous threshold is computed by measuring the "average" information loss in truncating while the latter one by the "maximal" information loss. By that approach, it is possible that some itemsets are labeled infrequent but are used to generate candidate itemsets, which helps to reduce the propagated errors by the a priori property.

5.4.1 Quantifying Information Loss in Truncating

So far, we have only considered the benefit of truncating transactions. Truncating also causes information loss, because the support of some itemsets decreases. For example, given a transaction {1,2,3}, by truncating that transaction to be {2,3}, the support of the itemset {1,2} changes from 1 to 0. To quantify this information loss, we formulate our problem as follows: given the noisy support θ' of an i-itemset X in the truncated database, estimate the support θ of X in the original database in a differentially private way. The major difficulty in solving this problem comes from the privacy requirement that whenever we need to utilize some information from the database, we need to do so in a differentially private way. Therefore, we must guarantee that our algorithm is either differentially private or only depends on differentially private information. In this paper, we take the latter approach, and our algorithm has two steps:

1. Given the noisy support of an itemset X, compute its support in the truncated database in a differentially private way.
2. Given the support of the itemset X in the truncated database, estimate its support in the original database in a differentially private way.

In the rest of this section, for ease of presentation, we use "original support" as a shorthand for "the support in the original database", and "truncated support" for "the support in the truncated database." Unless otherwise specified, we assume the i-itemset X is given, and use θ, $\hat{\theta}$ and θ' to denote X's original support, truncated support and noisy truncated support, respectively.

The first step is relatively straightforward where we use the Bayesian rule to compute the probability distribution of the truncated support as shown in Theorem 12

THEOREM 12.

$$\Pr(\hat{\theta}|\theta') \sim e^{-\epsilon|\theta'-\hat{\theta}|} \tag{3}$$

The second step quantifies the information loss in truncating. To give an intuition for our result, we start from an inverse problem: given the original support θ, what is the truncated support $\hat{\theta}$? We address that problem by using the analysis in random truncating to approximate the itemset X's truncated support. We want to emphasize that our smart truncating method is by no means equivalent to random truncating, and we *only* utilize the analysis in random truncating as a heuristic method to approximately quantify the information loss by our smart truncating method. We will discuss the rationale of that approximation at the end of this section.

We assume a uniform distribution among transactions with different cardinality containing the itemset X. More precisely, let z_j be the relative frequency of the transactions with cardinality j in the original database. Given an i-itemset X, we assume that the number of transactions with cardinality j containing X is $\theta * z_j / \sum_{h=i}^{n} z_h$.[1] Recall that we have already computed z_j $(j = 1 \dots n)$ in Algorithm 2 for frequent 1-itemset mining in a differentially private way, and thus, it is safe to utilize that information. For each transaction t, let t' be the truncated transaction of t. We observe that if a transaction does not contain X, then the truncated transaction does not contain X either. Thus, it suffices to only consider the transactions containing X to compute X's truncated support. We define a binary random variable M_t to quantify the effect of truncating a transaction on the support of the itemset X:

$$M_t = \begin{cases} 1 & \text{if } X \subseteq t' \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

(5) quantifies the probability that X remains in the truncated transaction.

$$\Pr(M_t = 1) = \frac{\binom{|t|-i}{\ell-i}}{\binom{|t|}{\ell}} \tag{5}$$

We also observe that the random variable M_t is identical for transactions of the same cardinality, and thus, we denote M_t by M_h, where $h = |t|$. Let f_h be the number of transactions with cardinality j containing the itemset X, and by our uniform assumption, $f_h = \theta * z_h / \sum_{j=i}^{n} z_j$. Thus, the truncated support of X is also a random variable M where:

$$M = \sum_{h=i}^{n} \sum_{j=1}^{f_h} M_h \tag{6}$$

The expectation of M is:

$$\begin{aligned} E(M) &= \sum_{h=i}^{n} f_i \Pr(M_h = 1) \\ &= \theta\left(\sum_{h=i}^{n} \frac{z_h}{\sum_{j=i}^{n} z_j} \frac{\binom{h-i}{\ell-i}}{\binom{h}{\ell}}\right) \end{aligned} \tag{7}$$

which quantifies the "average" truncated support of the itemset X. Next, we will compute the minimal truncated support, which quantifies the maximal information loss. It is not hard to see that a trivial lower bound for M is 0. However, by using 0 as the lower bound, it hardly provides us a way to estimate the original support given the truncated support. Therefore, we quantify that lower bound in a probabilistic way such that the truncated support is very unlikely to be smaller than that lower bound. Definition 9 formalizes this idea.

DEFINITION 9. *(ρ-lower bound): An integer σ is called a* ρ-lower bound *for the truncated support iff:*

$$\Pr(M \leq \sigma) \leq \rho$$

where M is the random variable defined in (6).

[1]Strictly speaking, it must be rounded to an integer.

We compute the ρ-lower bound by using the Chernoff bound [3]. Let $\mu = E(M)$, and for any $\gamma \geq 0$, by the multiplicative form of Chernoff bound,

$$\Pr(M \leq (1-\gamma)\mu) \leq exp(\frac{-\gamma^2\mu}{2})$$

Therefore, it suffices to solve the inequality $exp(-\gamma^2\mu/2) \leq \rho$, and the resulting $\lfloor(1-\gamma)\mu\rfloor$ serves as the ρ-lower bound. Thus, given the truncated support $\hat{\theta}$, let

$$exp\_ratio(i) = \sum_{h=i}^{n} \frac{z_h}{\sum_{j=i}^{n} z_j} \frac{\binom{h-i}{\ell-i}}{\binom{h}{\ell}}$$

By (7), the average original support is computed by:

$$avg\_os(\hat{\theta}) = \frac{\hat{\theta}}{exp\_ratio(i)} \tag{8}$$

Let μ be the expectation of the truncated support given the maximal original support. We compute μ by treating the truncated support $\hat{\theta}$ as the ρ-lower bound, which leads to solving the following two inequalities:

$$(1-\gamma)\mu \leq \hat{\theta} \leq \mu, \quad exp(-\frac{\gamma^2\mu}{2}) = \rho$$

It is not hard to show that

$$\mu \leq \mu^* = \hat{\theta} - \ln\rho + \sqrt{\ln^2\rho - 2\hat{\theta}\ln\rho} \tag{9}$$

provided $\ln\rho \leq 2\hat{\theta}$. Thus, the maximal original support is computed by

$$max\_os(\hat{\theta}) = \frac{\mu^*}{exp\_ratio(i)} \tag{10}$$

where μ^* is defined in (9), and if $\ln\rho > 2\hat{\theta}$, then we will set $max\_os(\hat{\theta})$ to be $avg\_os(\hat{\theta})$.

By combining our result of the first step, which estimates the truncated support given the noisy support, and our second step, which infers the original support given the truncated support, given θ', the noisy truncated support of the itemset X, we can compute the average original support by:

$$avg\_supp(\theta') = \sum_{j=0}^{n} \Pr(j|\theta') avg\_os(j) \tag{11}$$

and the maximal original support by:

$$max\_supp(\theta') = \sum_{j=0}^{n} \Pr(j|\theta') max\_os(j) \tag{12}$$

where $\Pr(j|\theta')$, $avg\_os(i)$ and $max\_os(i)$ are defined in (3), (8) and (10), respectively.

We use the average original support of an itemset to check whether or not it is frequent, and the maximal original support to determine whether or not to use it to generate candidate itemsets. Equivalently, given the noisy truncated support θ', we have changed the threshold to determine whether or not X is frequent from λ to $\lambda - avg\_supp(\theta') + \theta'$, and that for whether or not to use X to generate itemsets to $\lambda - max\_supp(\theta') + \theta'$. In that way, we have actually relaxed both thresholds.

5.4.2 Discussion

We want to emphasize that our approach of estimating the original support of an itemset is solely a heuristic method. As discussed, the major difficulty of estimating an itemset's original support comes from the privacy requirement, and thus, we must be quite careful of utilizing any information from the database. The reason why we use random truncating to approximate the information loss of smart truncating is because it relies on little information from the database. The exploration of other approaches to quantify the information loss by smart truncating is an interesting direction for future work.

Algorithm 3 FREQUENT_β_ITEMSET_MINING

Input: input database τ; itemsets' cardinality β; privacy parameter ϵ; threshold λ
Output: frequent itemsets of cardinality not exceeding β

1: $\epsilon' = \epsilon/\beta$
2: $\epsilon'' = \min\{0.05, \epsilon'/10\}$
3: $\tau' =$ TRUNCATEDATABASE(τ, ϵ'')
4: $\ell =$ the maximal cardinality of transactions in τ'
5: $\mathcal{S}_1 =$ I_ITEMSET_MINING$(\tau', \ell, \epsilon' - \epsilon'', \lambda, \emptyset)$
6: **for** i = 2 to β **do**
7: $\quad \ell = \psi(\lambda, i)$
8: $\quad \mathcal{S}_i =$ I_ITEMSET_MINING$(\tau, \ell, \epsilon', \lambda, \mathcal{S}_{i-1})$
9: **end for**
10: $R = \emptyset$
11: **for all** itemset X_j in $\cup_{i=1}^{\beta}\mathcal{S}$ **do**
12: $\quad$ **if** $avg\_supp(X_j.supp') \geq \lambda$ **then**
13: $\quad\quad$ Add X_j to R
14: $\quad$ **end if**
15: **end for**
16: **return** R

5.5 Our Algorithm

Our algorithm for frequent β-itemset mining improves over the naïve algorithm by using smart truncating to truncate transactions, and setting different thresholds for and itemset being frequent and for it generating candidate itemsets. This is shown in Algorithm 3. In particular, Algorithm 3 differs from the naïve algorithm in two places. First, instead of randomly truncating the transactions only once, we apply the method "SMARTTRUNCATING" to the original database for the mining of i-itemsets ($i = 2, \ldots, \beta$) by utilizing the results of the noisy $(i-1)$-itemsets as shown in line 10 in Algorithm 4. We initialize a candidate itemset's weight by its frequency score as shown in the code from line 7 to line 9 in Algorithm 4. Second, an itemset is used to generate candidate itemsets iff its estimated maximal original support exceeds the threshold. This is shown in line 18 in Algorithm 4. However, whether an itemset is frequent or not is determined by its estimated average original support. This is shown in line 12 in Algorithm 3.

The output of ψ in line 7 of Algorithm 3 is tunable given the maximal cardinality of transactions, the threshold and the current candidate itemsets' cardinality. The idea of tuning certain parameters in a differentially private algorithm by the data curator was first proposed by [8]. We consider how to automatically determine the maximal cardinality of transactions as an interesting direction for future work. We prove Algorithm 3 is differentially private in Theorem 13.

THEOREM 13. *Algorithm 3 is ϵ-differentially private.*

6. FREQUENT ITEMSET MINING

We are finally ready to present our complete algorithm. We first observe that Algorithm 3 is also a differentially pri-

Algorithm 4 i_itemset_Mining

Input: database τ; maximal cardinality ℓ; privacy parameter ϵ; threshold λ; noisy $(i-1)$-itemsets $\mathcal{S}$
Output: the i-itemsets whose estimated maximal original support exceeds the threshold

1: **if** i is 1 **then**
2: $\quad \mathcal{C} = \mathcal{I}$
3: **else**
4: $\quad \mathcal{C}$ = Generate candidate i-itemsets from $\mathcal{S}$
5: **end if**
6: **if** i is not 1 **then**
7: $\quad$ **for all** itemset $C_j \in \mathcal{C}$ **do**
8: $\quad\quad C_j.weight = fs(C_j)$
9: $\quad$ **end for**
10: $\quad \tau' \leftarrow$ apply SmartTruncating to τ's transactions
11: **else**
12: $\quad \tau' = \tau$
13: **end if**
14: $\kappa = \min\{\binom{\ell}{i}, |\mathcal{C}|\}$
15: $R = \emptyset$
16: **for all** itemset C_j in $\mathcal{C}$ **do**
17: $\quad C_j.supp' = C_j$'s support in $\tau' + G(\epsilon/\kappa)$
18: $\quad$ **if** $max\_supp(C_j.supp') \geq \lambda$ **then**
19: $\quad\quad$ Add C_j to R
20: $\quad$ **end if**
21: **end for**
22: **return** R

vate frequent itemset mining algorithm if β is at least maximal cardinality of any frequent itemset. However, this maximal cardinality is a property of the database so we cannot use it directly without privacy implications. Accordingly, we have the following problem:

Problem 3. *Let $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ where y_i is the maximal support of the i-itemsets. Given a threshold λ, find the index i^* such that y_{i^*} is the smallest integer that exceeds λ in a differentially private way.*

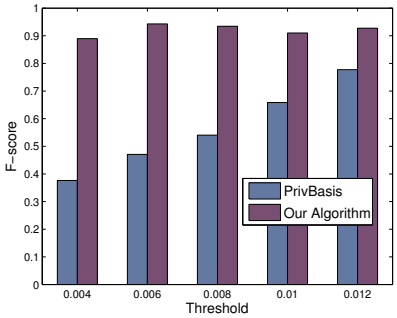
A naïve idea to solve Problem 3 is to first add geometric noise to each y_i, and then find the index i^*. It is not hard to show that the sensitivity of computing $\mathbf{y}$ is exactly n, and thus, in order to guarantee ϵ-differential privacy for Problem 3, we need to add the geometric noise $G(\epsilon/n)$ to each y_i.

In fact, we observe that $\mathbf{y}$ is non-increasing by the a priori property, and thus, we can reduce the required noise to $G(\epsilon/\lceil \log n \rceil)$ by using the same idea of binary search. Due to space limitations, we defer the details of this algorithm in the long version of our paper [1].

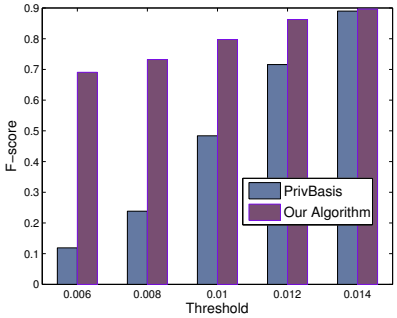
In practice, it is not practical to precisely compute $\mathbf{y}$. Therefore, we approximate $\mathbf{y}$ by setting the threshold to be $\lambda/20$, and run the original Apriori algorithm [2] to compute frequent itemsets. Suppose the maximal cardinality of the resulting frequent itemsets is j. Then for i from 1 to j, we can precisely compute the maximal support of i-itemsets y_i. For any i from $j+1$ to n, we set $y_i = y_j$. Our approxiamtion is safe in that the sensitivity of computing each y_i is still 1. We run our frequent itemset mining algorithm by first estimating the maximal cardinality of frequent itemsets, and then using that maximal cardinality as the input for Algorithm 3 to discover frequent itemsets. By the composition property of differential privacy, we conclude that our algorithm is differentially private.

dataset	m	$\lvert\mathcal{I}\rvert$	max $\lvert t\rvert$	avg $\lvert t\rvert$
BMS-POS (POS)	515,597	1,657	164	6.5
BMS-WebView-1 (WV1)	59,602	497	267	2.5
BMS-WebView-2 (WV2)	77,512	3,340	161	5.0
pumsb-star (PUMSB)	49,046	2,088	50.0	63

Table 1: Dataset characteristics



(a) WV1



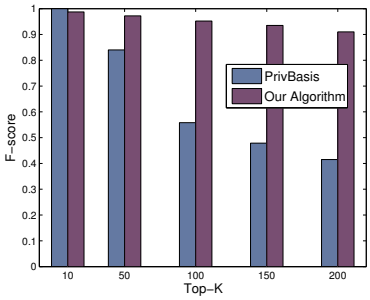
(b) WV2

Figure 2: Frequent Itemset Mining

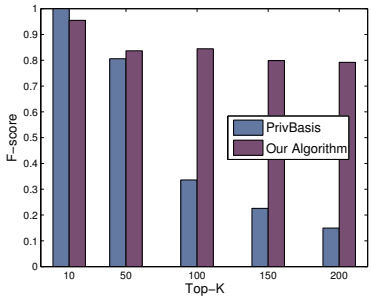
7. EXPERIMENTS

In this section, we experimentally evaluate our techniques on the benchmark datasets described in [22, 6]. A summary of those datasets is given in Table 1. For our experiments, instead of using the (δ, η)-usefulness, we employ a more intuitive metric F-score as shown in Definition 10. We implemented our algorithms in C++, and ran the experiments on an Intel Core 2 Duo 2.33GHZ machine with 1 GB RAM running Linux. Since our algorithms involve randomization, we ran each algorithm ten times to obtain its average performance. We use relative thresholds (percentage of transactions) in our experiments. Absolute thresholds can be easily derived by multiplying the relative threshold by the number of transactions in a database. In practice, we find that the overhead of our algorithm is less than 10% compared to the original Apriori algorithm [2][2], and thus, we do not present the running time of our algorithm. We compare our algorithm with the "PrivBasis" algorithm proposed in [16], which discovers top-k frequent itemsets, and the "TopK" frequent i-itemsets mining algorithm proposed in [5], which mines top-k frequent itemsets of the same cardinality i. We set the privacy parameter to be 1.0 for every algorithm in every set of experiments, and ρ to be 0.01 in

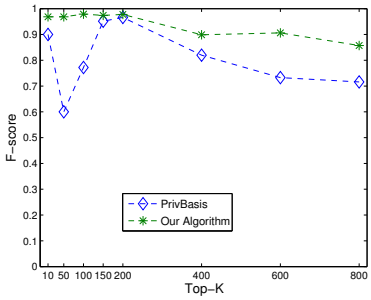
[2]We do not consider the overhead of computing frequent itemsets' maximal cardinality since we can pre-compute that value for each data set for a fixed threshold.



(a) WV1

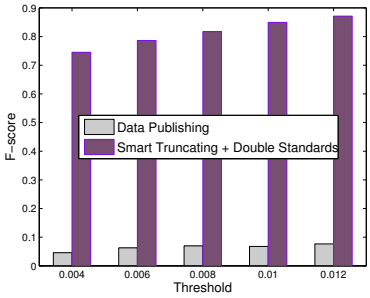


(b) WV2

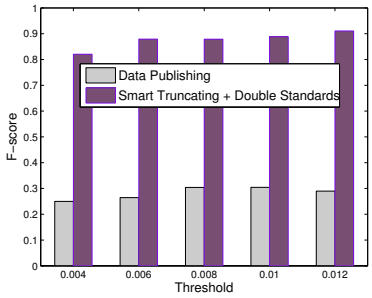


(c) PUMSB

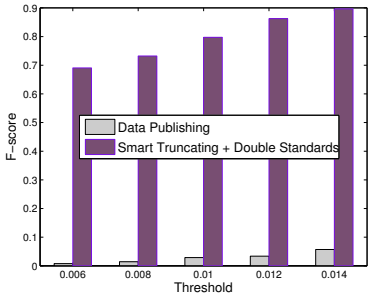
Figure 3: Top-K Frequent Itemset Mining



(a) POS



(b) WV1



(c) WV2

Figure 4: F-score of frequent itemsets

the double standards method.

DEFINITION 10. *(F-score): Let U_p be the set of frequent itemsets generated by a differentially **p**rivate frequent itemset mining algorithm, and U_c be the set of **c**orrect frequent itemsets, then*

$$precision = \frac{|U_p \cap U_c|}{|U_p|}, \quad recall = \frac{|U_p \cap U_c|}{|U_c|}$$

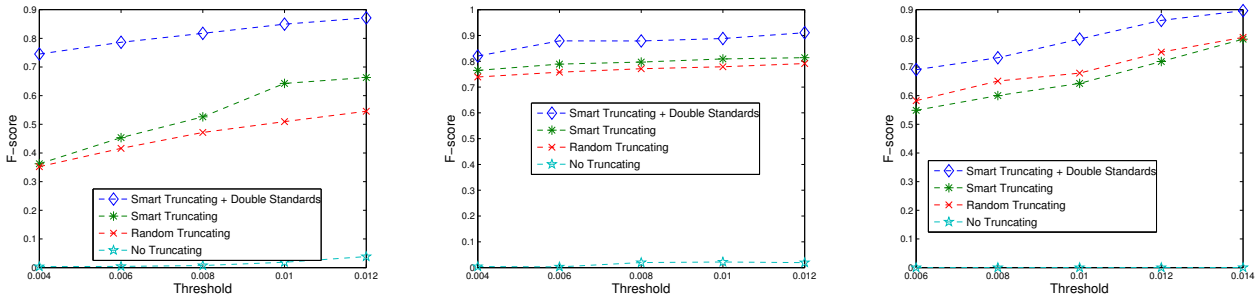
and the F-score is the harmonic mean of precision and recall:

$$F\text{-}score = 2 * \frac{precision * recall}{precision + recall}$$

We begin by noting that we do not present results for our algorithm without truncating because its performance was uniformly poor, at least an order of magnitude worse than the algorithm with truncating. Hence in our first experiment we move on to compare our algorithm with the top-k frequent itemset mining algorithm "PrivBasis" proposed in [16]. We adapt the "PrivBasis" algorithm to frequent itemset mining by setting k to be the number of frequent itemsets given a threshold. We want to emphasize that setting might have privacy implications. However, even with that relaxation in privacy, Figure 2 shows for the threshold frequent itemset computation, our algorithm outperforms the "PrivBasis" algorithm on two datasets WV1 and WV2. We were unable to compare the results on the dataset POS since the "PrivBasis" algorithm does not scale to handle larger k. In particular, when the threshold is 0.004, the total number of frequent itemsets in POS is more than 6000, and the "PrivBasis" algorithm can not efficiently discover the top-6000 frequent itemsets.

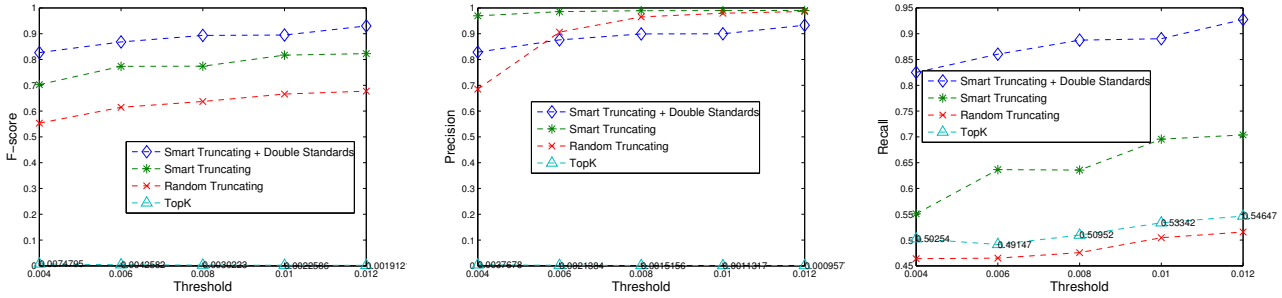
We are also interested in extending our algorithm to discover the top-k frequent itemsets. We can modify our algorithm to do so by setting the threshold to be the frequency of the k^{th} frequent itemsets. Of course, that computation of that frequency creates privacy concern. However, it is not hard to show that the sensitivity of that computation is one, and thus, we can add geometric noise to that computation. With that modification of our algorithm, we also compare our algorithm with the "PrivBasis" algorithm for top-k frequent itemset mining. Figure 3 shows that the quality of top-k frequent itemsets produced by our algorithm is better than that by the "PrivBasis" algorithm except the case when k is small (in our experiments, this means $k \sim 10$). We also evaluate our techniques on the dataset "PUMSB" in which "long" transactions dominate the datasets. Figure 3c shows that our algorithm is still superior. We also observe an interesting phenomena that the F-score of the "PrivBasis" algorithm drops to 0.6 when $k = 50$. One reason for that phenomena is because there is a 1-itemset whose frequency is very close to the frequency of the 50^{th} frequent itemset. In that way, the "PrivBasis" algorithm is very likely to label that 1-itemset as frequent, which also has side-effect on the result of 2-itemsets. As a result, it incurs a penalty on F-score since k is small.

Next, we compare our algorithm with the differentially private set-valued data publishing algorithm [8] on which we run the original Apriori algorithm over the anonymized data. Note that this is a generic approach — first anonymize the data, then run the (non-private) algorithm. We find that our algorithm outperforms the Apriori algorithm on the publishing anonymized data on all three datasets as shown in Figure 4. In particular, our algorithm increases the F-score



(a) POS (b) WV1 (c) WV2

Figure 5: Improvements of our heuristics



(a) F-score (b) Precision (c) Recall

Figure 6: Frequent 2-itemsets in POS

of frequent itemsets by an order of magnitude in both POS and WV2 as shown in Figure 4a and Figure 4c, respectively. The reason why the data publishing algorithm fails to generate accurate frequent itemsets is because the total number of transactions is greatly reduced after anonymization: for the dataset WV2, the average number of transactions after anonymizing is 6734.5, which is less than 10% of the original transactions. Thus, the anonymization incurs significant information loss.

We also want to know how our two heuristics — smart truncating and double standards — affect the performance of our algorithm. We show the results in Figure 5. The most significant improvement is the double standards approach. Although the smart truncating method improves the quality of frequent itemsets over random truncating in both POS and WV1, we find it does not work for WV2. This indicates that whether or not the frequency score is a good indicator of an itemset's frequency depends on the dataset. Whether there exists a truncating approach that outperforms the random truncating approach on any dataset is an interesting direction for future research.

To better understand why our heuristics work well on the dataset "POS", we show the results of 2-itemsets in Figure 6. We observe that the double standards approach significantly improves the recall of frequent 2-itemsets while it decreases the precision a little as shown in Figure 6c and Figure 6b, respectively. Furthermore, we also observe that by utilizing the a priori property and truncating transactions, our algorithm increases the F-score of frequent 2-itemsets by orders of magnitude comparing to the top-k frequent i-itemset mining algorithm proposed in [5] as shown in Figure 6a.

As discussed in Section 4, the success of our algorithm

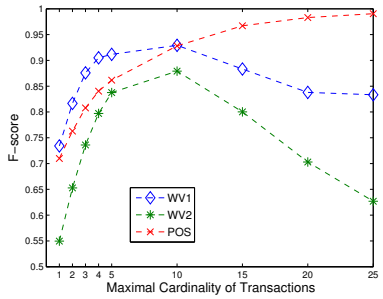


Figure 7: Frequent 1-itemsets

relies on the fact the information loss in truncating transactions is offset by the benefit of reducing noise. To show that, we set the threshold to be 0.004. We compare the F-score of frequent 1-itemsets by changing the constraint on transactions' maximal cardinality. Figure 7 shows the result. As we can see, when the maximal cardinality is very small, the information loss exceeds the benefit of reducing noise. On the other hand, when the maximal cardinality is large, the increase of the noise offsets the benefit of retaining the information.

8. RELATED WORK

In this section we discuss related work not explicitly mentioned elsewhere in our paper.

The notion of differential privacy was proposed by Dwork et al. in [9]. The same authors also propose the addition of Laplacian noise to guarantee differential privacy [10], and [12] propose adding geometric noise to achieve the same goal. The problem of frequent itemset mining has been ex-

tensively studied in literature [2, 13]. The data mining community has focused on hiding sensitive rules generated from transactional databases [4, 20]. In [4], the authors address this problem by altering the database to hide a given set of sensitive rules. However, how to define sensitive rules is unclear, and their approach does not satisfy differential privacy.

Although Evfimievski et al. have developed a privacy preserving frequent itemset mining algorithm in [11], their approach does not guarantee differential privacy. In [17], the authors propose a noisy a priori algorithm which is quite similar to our random truncating approach but they do not explicitly consider the problem of frequent itemset mining as we do.

Another way to develop a differentially private frequent itemsets mining is to use a basic, non-anonymizing frequent itemset mining algorithm, but to apply it to an anonymized version of the transactional data. This is an intriguing approach and it warrants exploration. Early work appears on anonymizing other types of data sets [21, 18] has shown a great success in this direction. In other related work, [14, 19, 7] propose ad hoc privacy criteria to resist certain kinds of attacks on publishing transaction databases. However, those ad hoc privacy criteria do not guarantee differential privacy. The latest effort to anonymize transactional data in a differentially private way is proposed by [8]. However, they only consider the workload of top-k frequent itemsets, and our experimental results indicate that our algorithm improves the F-score of frequent itemsets over [8] by an order of magnitude on two benchmark datasets.

9. CONCLUSION

In this paper, we have proposed a differentially private frequent itemset mining algorithm. We have precisely quantified the trade-off between privacy and utility in frequent itemset mining, and our results indicate that in order to satisfy a non-trivial utility requirement, a frequent itemset mining algorithm incurs a huge risk of privacy breach. However, we find that we can greatly promote the utility of a differentially private frequent itemset mining algorithm by limiting the maximal cardinality of transactions.

Motivated by that observation, we have proposed a new differentially private frequent itemset mining algorithm. Our results on benchmark datasets indicate that in comparison to the latest algorithm on publishing transactional data in a differentially private way [8], our algorithm improves the F-score of frequent itemsets by more than 200% in one dataset, and by an order of magnitude on the other two datasets. Our results also show that our algorithm significantly improves the quality of top-k frequent itemsets comparing to the differentially private top-k frequent itemset mining algorithm proposed in [16, 5] except when k is small.

There are many potential opportunities for future work. One such direction would be to explore other more sophisticated truncating algorithms. Another direction would be to explore alternative methods to limit the information loss due to truncating. Finally, the success of our algorithm relies on the assumption that the "short" transactions dominate the datasets. How to deal with datasets dominated by long transactions is an open problem, although with no constraints on the database our theoretical results on privacy/utility tradeoffs suggest that algorithms that simultaneously achieve good privacy and utility may prove elusive.

10. REFERENCES

[1] `http://pages.cs.wisc.edu/~zeng/dfim.pdf`.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[3] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *STOC*, 1977.

[4] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *The VLDB Journal*, 2008.

[5] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, 2010.

[6] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu. Mafia: A performance study of mining maximal frequent itemsets. In *FIMI*, 2003.

[7] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty: Inference proof transaction anonymization. In *VLDB*, 2010.

[8] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *VLDB*, 2011.

[9] C. Dwork. Differential privacy. In *ICALP*, 2006.

[10] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCS*, 2006.

[11] A. Evfimievski, R. Srikant, R. Agarwal, and J. Gehrke. Privacy preserving mining of association rules. 2004.

[12] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, 2009.

[13] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.

[14] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *VLDB*, 2009.

[15] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining: a general survey and comparison. *SIGKDD Explor. Newsl.*, 2000.

[16] N. Li, W. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemsets mining with differential privacy. In *VLDB*, 2012.

[17] F. McSherry and R. Mahajan. Differentially-private network trace analysis. *SIGCOMM Comput. Commun. Rev.*, 2010.

[18] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.

[19] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *VLDB*, 2008.

[20] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *TKDE*.

[21] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2009.

[22] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *KDD '01*, 2001.