Computers, Materials & Continua CMC, vol.65, no.1, pp.369-385, 2020

Frequent Itemset Mining of User's Multi-Attribute under Local Differential Privacy

Haijiang Liu[1], Lianwei Cui[2], Xuebin Ma[1,*] and Celimuge Wu[3]

Abstract: Frequent itemset mining is an essential problem in data mining and plays a key role in many data mining applications. However, users' personal privacy will be leaked in the mining process. In recent years, application of local differential privacy protection models to mine frequent itemsets is a relatively reliable and secure protection method. Local differential privacy means that users first perturb the original data and then send these data to the aggregator, preventing the aggregator from revealing the user's private information. We propose a novel framework that implements frequent itemset mining under local differential privacy and is applicable to user's multi-attribute. The main technique has bitmap encoding for converting the user's original data into a binary string. It also includes how to choose the best perturbation algorithm for varying user attributes, and uses the frequent pattern tree (FP-tree) algorithm to mine frequent itemsets. Finally, we incorporate the threshold random response (TRR) algorithm in the framework and compare it with the existing algorithms, and demonstrate that the TRR algorithm has higher accuracy for mining frequent itemsets.

Keywords: Local differential privacy, frequent itemset mining, user's multi-attribute.

1 Introduction

Mining frequent itemsets is very useful in many cases, such as in the record of someone purchasing cold medicine; whether they would buy antipyretics or cough medicines. By collecting data from multiple users, the degree of association among these drugs can be obtained, so a doctor can select the appropriate drug purchase ratio based on these records. However, users do not want to disclose sensitive personal information during the purchase process. In this study, we aim to mine frequent itemset information while ensuring that the user's personal information is not revealed.

There are many ways to protect privacy, such as data scrambling, data encryption, data anonymity, and other privacy protection technologies. Existing anonymity-based privacy

[1] College of Computer Science, Inner Mongolia University, Hohhot, 010020, China.

[2] Inner Mongolia Big Data Development Authority, Hohhot, 010020, China.

[3] Graduate School of Informatics and Engineering, the University of Electro-Communication, Tokyo, 182-8585, Japan.

[*] Corresponding Author: Xuebin Ma. Email: csmaxuebin@imu.edu.cn.

Received: 13 April 2020; Accepted: 11 May 2020.

protection models require some background knowledge and special attack assumptions. Moreover, they cannot quantify the intensity of privacy protection, so significant limitations exist in the actual. To address the shortcomings of the anonymous privacy protection model, Dwork [Dwork (2006)] proposed a differential privacy model that randomly perturbed the published data. In a statistical sense, regardless of background knowledge that the attacker possesses, identifying whether a record exists in the original data table is impossible. The user sends the original data to a trusted third-party aggregator; then the aggregator perturbs the user data and publishes it. However, when the aggregator is untrustworthy or malicious third party attacks them, the user's personal privacy information can be leaked. To solve the shortcomings of this mechanism, Kasiviswanathan et al. [Kasiviswanathan, Lee and Nissim (2011)] proposed local differential privacy, which advances the steps of the disturbance. Each user first perturbs the data on the user side and then sends it to the third-party aggregator. This process aggregator can be untrusted because it only analyzes the perturbed data and publishes it.

Randomized aggregatable privacy-preserving ordinal response (RAPPOR) is the most convincing application of the local differential privacy model, and Erlingsson et al. [Erlingsson, Pihur and Korolova (2014)] has deployed RAPPOR to Google Chrome. In addition to Google, Thakurta et al. [Thakurta, Vyrros and Vaishampayan (2017)] applied for a patent to protect users' privacy and deploy to the ISO in 2017. Other electronic information companies such as Samsung have also proposed a similar privacy protection system. Companies are beginning to value the importance of protecting user privacy data.

Currently, most papers focus on privacy protection for a single frequent item, and each user has only one attribute. However, this situation does not apply in practical applications. For example, the user's personal information may not only include the name or age, but also the education and income, and these attributes may be related to each other. For such complex user data, we propose a novel framework to implement frequent itemset mining under local differential privacy.

Our contribution:

1) For the multi-attribute of users, we propose a complete framework to mine frequent itemsets under local differential privacy.

2) We also consider that the different attributes have different possible value ranges, such as gender and income. For this case, we chose different perturbation algorithms to achieve the best data availability.

3) Finally, the aggregator uses the FP-tree algorithm to mine frequent itemsets and compares them with the existing methods.

The rest of our paper is organized as follows. Section 2 introduces related work. Section 3 describes the problem and defines local differential privacy. Section 4 introduces two mechanisms to satisfy the local differential privacy. Section 5 proposes the TRR algorithm to satisfy the user's multi-attribute. Section 6 introduces the aggregator using the FP-tree algorithm to mine frequent itemsets and compares the TRR algorithm with the existing methods. Section 7 concludes the paper.

2 Related work

In recent years, with the explosive growth of data and the rapid development of information technology, various industries have accumulated large amounts of data through various channels. However, various industries are beginning to value the importance of protecting user privacy data. Aiming at the security and privacy issues in cloud computing, for instance, Min et al. [Min, Yang, Wang et al. (2019)] proposed a homomorphic encryption algorithm, which utilized the characteristics of multi-nodes and matrix multiplication for parallel encryption. As for the problem of privacy leakage in the smart grid, He et al. [He, Zeng, Xie et al. (2017)] proposed a random linear network coding scheme to protect user privacy and effectively organize traffic analysis. Furthermore, for the leakage of location privacy information, Gu et al. [Gu, Yang and Yin (2018)] proposed a multi-level query tree structure to publish location data on database, and added an exponential mechanism to the query results. In addition, for the privacy leakage of personalized recommendation service system, Yin et al. [Yin, Shi, Sun et al. (2019)] proposed an efficient privacy protection collaborative filtering algorithm based on differential privacy protection and time factor.

To ensure privacy of data mining, Wong et al. [Wong, Li and Fu (2006)] proposed a traditional method based on k-anonymity and Li [Li (2007)] proposed its extended models. These methods require certain assumptions and it is difficult to protect privacy when the assumptions are violated. The insufficiency of k-anonymity and its extended models is that there is no strict definition of the attack model, and that the knowledge of the attacker cannot be quantitatively defined. To pursue strict privacy analysis, Dwork [Dwork (2006)] proposed a strong privacy protection model called central differential privacy. It features independence of the background knowledge of the attacker and proves to be very useful.

For centralized datasets, Wong et al. [Wong, Cheung and Hung (2007)] proposed using a 1-to-n encryption method to change original itemsets to protect data privacy when outsourcing frequent itemset mining. Qiu et al. [Qiu, Li and Wu (2006)] proposed an algorithm that transforms business information into a very long binary vectors and a series of random mapping functions based on bloom filters. Tai et al. [Tai, Yu and Chen (2010)] proposed a k-support anonymity-based frequent itemset mining algorithm. All these methods sacrifice the precision of the mining result.

Because traditional approaches are based on heuristics, a solid privacy guarantee is missing. Therefore, researchers began to investigate frequent itemset mining with differential privacy. Bhaskar et al. [Bhaskar, Laxman and Smith (2010)] presented two mining algorithms, which are representatives of frequent itemset mining with differential privacy. Cheng et al. [Cheng, Su and Xu (2015)] implemented differential privacy protection to the apriori mining algorithm. Xiong et al. [Xiong, Chen and Huang (2018)] implemented differential privacy protection to the FP-tree mining algorithm, which can reduce the number of database traversals. These are frequent itemset mining protection methods based on central differential privacy.

Many previous studies used local differential privacy to solve heavy hitter problem, rather than frequent itemset. Heavy hitters are simply the frequency of occurrence and do not consider the relationship between frequent items. Examples include papers [Bassily

and Smith (2015); Bassily, Nissim and Stemmer (2017); Bun, Nelson and Stemmer (2018)], which solve heavy hitter problem under local differential privacy and only have a single attribute for the user. Wang et al. [Wang and Li (2018)] considered that there may be differences in the thresholds of attributes, but they still only considered a single attribute and heavy hitter. Herein, we study local differential privacy protection in frequent itemset mining and consider user multi-attribute values and associations.

We use two important local differential privacy mechanisms, the RAPPOR [Erlingsson, Pihur and Korolova (2014)] mechanism and randomized response (RR) [Kairouz, Oh and Viswanath (2014)] mechanism. Zhang et al. [Zhang, Huang and Fang (2017)] proposed that the Multiple Randomized Response (MRR) algorithm applies personalized differential privacy to mine frequent itemsets. The RR algorithm is selected when the attribute is protected to a low degree, and the RAPPOR algorithm is selected when the attribute is protected to a high degree. Although the MRR algorithm has multiple attribute values, it does not consider that the thresholds of different attributes may have large differences. They have some understanding flaws in the RR and RAPPOR algorithms because their theoretical basis is derived from data distribution estimates. Choosing the RR and RAPPOR algorithms in frequent itemset mining is closely related to the attribute threshold. Our propose TRR algorithm solves the shortcomings of the above problems. Finally, to reduce the number of database traversals, the aggregator uses an FP-tree to mine frequent itemsets.

3 Preliminaries

3.1 Problem definition

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items in the transaction database, transaction T be a set of some items ($T \subseteq I$), and a database $D = \{T_1, T_2, \ldots T_n\}$ be the set of transactions. Each P where $P \subseteq I$ is called an "itemset" and P is also called a k-itemset, where $|P| = k$. Transaction T contains an itemset P if and only if $P \subseteq T$; the support of P, which is denoted as support (P), is defined as the percentage of transactions in D containing P. Let $\min -$ support be the user-defined minimum support threshold. There P is a frequent itemset if and only if $min - support \leq support(P)$. Given database D and the min-support threshold, the frequent itemset mining task is defined as "discovering all frequent itemsets with their supports."

3.2 Local differential privacy

Based on the untrustworthy third-party aggregator, we used local differential privacy to perturb the data on the user side, and the user can also choose different perturbation methods according to the protection degree of the original data. The aggregator is only responsible for collecting and analyzing data and publishing the overall model.

Definition 1 (Local Differential Privacy [Kasiviswanathan, Lee and Nissim (2011)]). For any inputs x_1 and x_2, the output y is obtained by an algorithm ψ. If the following inequalities are satisfied, we say that algorithm ψ that satisfies ε-local differential privacy.

$$P[\psi(x_1) = y] \leq e^{\varepsilon} \cdot P[\psi(x_2) = y] \quad (1)$$

Here, P denotes the output y probability by the algorithm ψ and ε represents the privacy

budget, which is inversely proportional to the privacy protection degree. Specifically, a smaller ε means higher privacy protection of the user data, and a larger ε means lower privacy protection.

3.3 Sequence combination

Local differential privacy inherits the sequence combination of central differential privacy. When a single user satisfies local differential privacy and different users may select different perturbation algorithms, multiple users can still satisfy local differential privacy, as follows:

Given a dataset D and n privacy algorithms $\{\psi_1, \psi_2 \ldots \psi_n\}$ and ψ_i $(1 \le i \le n)$ that satisfy ε_i -local differential privacy, the sequence combination of $\psi_i(D)$ satisfies $(\sum_{i=0}^{n} \varepsilon_i)$-differential privacy.

4 Random disturbance mechanism

4.1 k-RR random perturbation algorithm

The previously proposed the w-RR perturbation algorithm, which is mainly applied for the two values. For example, we want to count the number of people with AIDS, and users respond to whether they have AIDS. If a user has AIDS, the probability of answering the suffering is p,and the not probability is q. If a user does not have AIDS, the probability of the unaffected person answering is p, and the probability of suffering is q.

However, the w-RR algorithm can only be applied to a relatively simple user attribute. If the user attribute is more complex than a binary attribute, the w-RR would not be satisfied. The k-RR perturbation algorithm is proposed to improve the w-RR algorithm. When $k = 2$, w-RR is only a special case of k-RR. We use ψ to represent the random perturbation algorithm. The specific perturbation probability of k-RR is as follows:

$$\Psi(y) = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+k-1}, & y = x \\ q = \frac{1}{e^{\varepsilon}+k-1}, & y \neq x \end{cases} \tag{2}$$

In general, we need to map the user's data to a binary string and perturb with probability p or q. In the above formula, x and y are the input and output, respectively; ε represents the privacy budget; and k is the possible value of the user data.

Theorem 1: The algorithm k-RR satisfies ε-local differential privacy.

Proof: We use ψ_{k-RR} to denote the k-RR perturbation algorithm. To achieve data availability, we usually set that no disturbance probability is greater than the perturbation probability. The probability of p should be greater than q and satisfy the following inequalities.

$$\frac{P[\psi_{k-RR}(x_1)=y]}{P[\psi_{k-RR}(x_2)=y]} \le \frac{p}{q} = \frac{e^{\varepsilon}/(e^{\varepsilon}+k-1)}{1/(e^{\varepsilon}+k-1)} = e^{\varepsilon} \tag{3}$$

4.2 k-RAPPOR random perturbation algorithm

RAPPOR was proposed by Erlingsson et al. [Erlingsson, Pihur and Korolova (2014)] and has been successfully applied to Google Chrome. It is one of the few perturbation

algorithms deployed in a real environment. RAPPOR is roughly divided into three stages. The most complete RAPPOR must first hash each value with a bloom filter and then perform two random perturbations, namely, permanent and temporary perturbations. The k-RAPPOR we introduced is only a relatively simple case, called a one-time RAPPOR, used only for permanent perturbations.

We use ψ to represent the perturbation algorithm. The probability of not perturbing a bit of a binary string is p, and the probability of perturbation is q. The specific probability formula is as follows:

$$\Psi(z) = \begin{cases} p = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}, v_i = z \\ q = \frac{1}{e^{\varepsilon/2}+1}, v_i \neq z \end{cases} \tag{4}$$

Here, v and z are the user's input and output, respectively, and v_i is the disturbance to the i-th bit.

Theorem 2: The algorithm k-RAPPOR satisfies ε-local differential privacy.

Proof: We use v_1 and v_2 to represent the input, and the output is z as follows:

$$\frac{P[z|v_1]}{P[z|v_2]} = \frac{\prod_{i\in[d]} P[z[i]|v_1]}{\prod_{i\in[d]} P[z[i]|v_2]} \leq \frac{P\left[z[v_1]=1\middle|v_1\right]\cdot P\left[z[v_2]=1\middle|v_1\right]}{P\left[z[v_1]=1\middle|v_2\right]\cdot P\left[z[v_2]=1\middle|v_2\right]} = \frac{p}{q}\cdot\frac{1-q}{1-p} = e^{\varepsilon} \tag{5}$$

5 TRR disturbance algorithm

The TRR algorithm selects the k-RR when the threshold is small, and selects the k-RAPPOR algorithm when the threshold is large. Next, we use some theory to explain its correctness.

5.1 Pure local differential privacy protocol

The biggest difference between local differential privacy is that the input v will obtain a certain output value, and Wang et al. [Wang and Li (2018)] proposed the pure differential privacy protocol that defines this output value as a set.

Suppose there are two different probabilities, p^* and q^*. The non-disturbing probability p^*should be greater than the perturbation probability q^* and satisfy the following inequality:

$$P[\psi(v_1) \in \{y|v_1 \in add(y)\}] \leq e^{\varepsilon} \cdot P[\psi(v_2) \in y|v_2 \in add(y)] \tag{6}$$

In the above inequality, ψ is the perturbation algorithm, v_1 and v_2 are the user's input, and y is the output. Add is a function that maps each possible output value y to an input value set.

If y_j represents the perturbation data sent to the aggregator by user j, the aggregator needs to estimate the number of per bit and then perform statistics:

$$\tilde{n}(\mathrm{i}) = \frac{\sum_j S_{add(y_j)}(i) - nq^*}{p^* - q^*} \tag{7}$$

In the above formula, $\tilde{n}$ is used to indicate the number of times each bit is used. It should be noted that there is a flag function, and the specific calculation formula is as follows:

$$S_X(i) = \begin{cases} 1, & i \in X \\ 0, & i \notin X \end{cases} \tag{8}$$

From (6), X is the user's binary string, and i is one of the strings. It can be obtained after

encoding that the user has a certain attribute coded to 1 or coded to 0.

Theorem 3: In order to verify the correctness of our estimate for each bit, we need to calculate its variance:

$$Var[\tilde{n}(i)] = \frac{nq^*(1-q^*)}{(p^*-q^*)^2} + \frac{nf_i(1-p^*-q^*)}{p^*-q^*} \tag{9}$$

Proof.

$$Var[\tilde{n}(i)] = Var\left[\frac{\sum_j S_{add(y_j)}(i) - nq^*}{p^* - q^*}\right] = \frac{\sum_j Var\left[S_{add(y_j)}(i)\right]}{(p^* - q^*)^2}$$

$$= \frac{nf_i p^*(1-p^*) + n(1-f_i)q^*(1-q^*)}{(p^* - q^*)^2}$$

$$= \frac{nq^*(1-q^*)}{(p^*-q^*)^2} + \frac{nf_i(1-p^*-q^*)}{p^*-q^*} \tag{10}$$

However, most values have fewer occurrences and are determined by frequent values in most applications. Avoiding a large number of false positives allows one to obtain lower estimated variances between infrequent values. In the above formula, when the frequency f_i is small, the variance is mainly determined by the first term, and then an approximate variance Var^* is obtained:

$$Var^*[\tilde{n}(\mathrm{i})] = \frac{nq^*(1-q^*)}{(p^*-q^*)^2} \tag{11}$$

5.2 k-RR and k-RAPPOR variance

The additional function of the k-RR perturbation algorithm is $add_{k-RR}(i) = \{i\}$, which satisfies the pure local differential privacy protocol. Furthermore, $p^* = p$ and $q^* = q$ can be obtained, and then the k-RR approximate variance is obtained by combining Eqs. (7) and (11):

$$Var^*[\tilde{n}_{k-RR}(i)] = n \cdot \frac{k-2+e^{\varepsilon}}{(e^{\varepsilon}-1)^2} \tag{12}$$

We can see that as the threshold k increases, the k-RR variance also increases. It can be said that the accuracy of the k-RR result decreases as the threshold increases. We, therefore, use the k-RR algorithm when the threshold is small.

Similarly, the additional function of the k-RAPPOR is $add_{k-RAPPOR}(z) = \{i|z[i] = 1\}$, which satisfies the pure local differential privacy protocol. We can obtain $p^* = p$ and $q^* = q$, and then combine the above Eqs. (7) and (11) to obtain the approximate variance:

$$Var^*[\tilde{n}_{k-RAPPOR}(i)] = n \cdot \frac{e^{\varepsilon/2}}{(e^{\varepsilon}-1)^2} \tag{13}$$

We can see that the approximate variance of the k-RAPPOR algorithm is irrelevant to the threshold k. We use the k-RAPPOR algorithm better when the threshold is larger.

5.3 Encoding

The situation we have to consider is more complicated. Each user may have multiple attribute values, and the possible values of each attribute are different. How should we choose the encoding method and the perturbation algorithm for this complicated situation?

We use bitmap encoding, first considering a universal set U with cardinalityn. We can represent each subset of U by a bitmap of size n. Each element of U is assigned to one of the bits in the bitmap. If an element is a member of a subset S ($S \subseteq U$), then its corresponding bit is 1; otherwise it is 0. Consider the following example: let there be a universal set $U = \{a_3, a_2, a_1, a_0\}$, and subsets $A = \{a_3, a_2\}$ and $B = \{a_3, a_0\}$. With two bitmaps of size four, in which each $a_i (0 \leq i \leq 3)$ is assigned to their ith bit, these subsets are represented as $A = 1100$ and $B = 1001$. With this representation of sets, some common set operators can be implemented faster using bitwise operators. For example, to calculate the intersection (union) of two given sets, we can use the bitwise operator AND (OR) on their corresponding bitmaps. Bitwise operators are implemented efficiently in CPUs and performed in one CPU cycle.

Specific to the user's multi-attribute, assuming that each user has h attributes, and the value of each attribute is represented by k. Then, k_1 is the possible value of the first attribute, and k_h is the possible value of the hth attribute. We mapped each attribute to a binary string. For example, the first attribute may have k_1 values that are mapped to k_1 bits. If the user has a value of this attribute mapped to 1, otherwise is mapped to 0, so we can get a binary string of length $m = k_1 + k_2 \ldots + k_h$.

5.4 TRR algorithm

We select k-RR when the attribute threshold is small, and select k-RAPPOR when the attribute threshold is large in the TRR algorithm. When $k_i < 3e^{\varepsilon} + 2$ is a smaller threshold, and $k_i > 3e^{\varepsilon} + 2$ is a larger threshold [Wang and Li (2018)].

The TRR algorithm is suitable because the user data are more complicated. In particular, the user has multiple attributes, and the thresholds between the attributes are significantly different. The TRR algorithm is more practical and has a wider range of applications. The specific process of the algorithm is as follows:

Algorithm 1. TRR perturbation algorithm

Input: binary string $m \in \{0,1\}^n$

Input: User's possible attribute value h, threshold value $k_1, k_2 \ldots k_h$ for each attribute

Input: Privacy budget ε

Output: binary string $\widetilde{m}$ after disturbance

1:

et a variable $t = 0$ that controls the movement of each bit.

2:

or each item k_i in h do

3:

if $k_i < 3e^{\varepsilon} + 2$

4:

for $l = 0$ in k_i

5:

$$\widetilde{m}(t) = \begin{cases} m(t), & p = \frac{e^{\varepsilon}}{e^{\varepsilon}+k_i-1} \\ 1-m(t), & q = \frac{1}{e^{\varepsilon}+k_i-1} \end{cases}$$

6:

$$t = t + 1$$

7:

end for

8:

else if $k_i > 3e^{\varepsilon} + 2$

9:

for $l = 0$ in k_i

10:

$$\widetilde{m}(t) = \begin{cases} m(t), & p = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} \\ 1-m(t), & q = \frac{1}{e^{\varepsilon/2}+1} \end{cases}$$

11:

$$t = t + 1$$

12:

end for

13:

end if

14:

nd for

15:

eturn $\widetilde{m}$

5.5 Decoding

Next, we introduce some unbiased estimation content, mainly considering that the original data will have some impact after the perturbation algorithm. If unbiased estimation is not performed, the i-th bit of the binary string of all users is directly added to obtain:

$$n(i) = \sum_j S_{add(y_j)}(i) \tag{14}$$

This method will cause some errors, so we will estimate it unbiasedly:

$$\tilde{n}(i) = \frac{\sum_j S_{add(y_j)}(i) - nq^*}{p^* - q^*} \tag{15}$$

Theorem 4: By calculating the number $\tilde{n}(i)$ of a certain bit, the expectation can $E[\tilde{n}(i)] = nf_i$ be proved to be unbiased.

Proof.

$$E[\tilde{n}(i)] = E\left(\frac{\sum_j S_{add(y_j)}(i) - nq^*}{p^* - q^*}\right) = \frac{nf_i p^* + n(1 - f_i)q^* - nq^*}{p^* - q^*}$$

$$= n \cdot \frac{f_i p^* + q^* - f_i q^* - q^*}{p^* - q^*} = nf_i \tag{16}$$

6 Experiment

6.1 FP-tree frequent itemset mining

After the server receives the user's binary string, it constructs an FP-tree for frequent itemset mining.

First, building an FP-tree requires creating an item header table, so we scan the database for the first time and obtain all 1-itemset $\tilde{n}(i)$. Then, we delete the items whose support is lower than $\min - \text{support}$, obtain frequent 1-itemsets into the header table, and sort them in descending order of support. Next, the second scan database deletes the infrequent 1-itemsets of the data and sorts them in descending order of support.

After acquiring the item header table and sorted datasets, we can start building the FP-tree. The FP-tree has no data at the beginning. When building an FP-tree, we need to insert the sorted datasets one by one. The node that ranks first is the ancestor node, and the next one is the descendant node. If they have a shared ancestor, the ancestor node count is incremented by one. After the insertion, if a new node appears, the node corresponding to the item header table is linked to the new node through the node list. The creation of the FP-tree is complete when all the data has been inserted.

Next, we mine frequent itemsets from the item at the bottom of the item header table. For each item in the item header table that corresponds to the FP-tree, we need to find its conditional pattern base. The conditional pattern base is the FP-subtree corresponding to the leaf node that we want to mine. We set the count of each node in the FP subtree to the count of the leaf nodes and delete the nodes whose count is lower than the support. From this conditional model base, we can recursively mine frequent itemsets. In the experiment, we set, $min - support = 0.5$ and obtain frequent 2-itemsets and frequent 3-itemsets. These settings are not fixed.

6.2 Algorithm experiment comparison

Our experimental metric is F-score, which is a commonly used measure in data mining, and it is the harmonic mean of the correct rate and recall rate.

$$F - score = 2 * \frac{precision * recall}{precision + recall} \tag{17}$$

Here, precision denotes the accuracy rate and recall is the recall rate. The higher F-score, the more effective the experimental method.

Additionally, to measure the error with the actual support of itemsets in the dataset, we calculated the relative error (RE) of the support of released itemsets.

$$RE = median_X \frac{|sup'(x) - \sup(x)|}{\sup(x)} \quad (18)$$

Here, X is the set of all frequent itemsets generated by a private algorithm, $sup(x)$ is the actual support of itemset x, and $sup'(x)$ is its noisy support. It should be noted that the smaller the RE is, the smaller the error; it also indicates that the utility of the algorithm is higher.

Our experimental environment was implemented in Python 3.7. The experimental dataset generates a normal distribution, and an exponential distribution by its definition, and then verifies the TRR algorithm. If the dataset obeys a uniform distribution, our algorithm is not applicable. It does not make much sense to mine frequent itemsets.

We conducted two sets of experiments under the normal distribution to mine frequent 2-itemsets and frequent 3-itemsets. Fig. 1 shows the impact of four different local differential privacy algorithms on frequent mining results under different privacy budgets. The abscissa indicates that the privacy budget has a value from 1.0 to 6.0, with an increase of 0.5 each time. As shown in Fig. 1, as the privacy budget increase, the overall F-scores of the four algorithms show an upward trend. Further, the F-score of the TRR algorithm is larger than that of the other three algorithms, which indicates that the TRR algorithm has a higher accuracy for mining frequent 2-itemsets. As seen in Fig. 2, similar to Fig. 1, the F-score of the TRR algorithm in mining frequent 3-itemsets is larger than that of the other three algorithms. This also shows that the TRR algorithm has a better effect than the RR, RAPPOR and MRR algorithms in mining frequent itemsets.

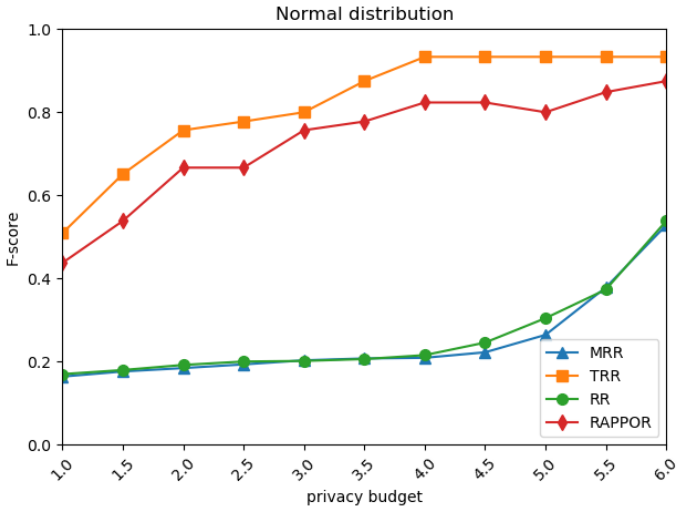

Figure 1: Frequent 2-itemsets of the F-score under normal distribution

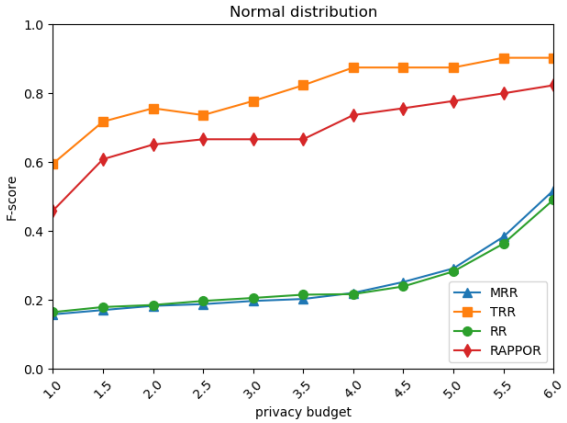

Figure 2: Frequent 3-itemsets of the F-score under normal distribution

Figs. 3 and 4 show the MRR, TRR, RR, and RAPPOR algorithms mining frequent 2-itemsets and frequent 3-itemsets in an exponential distribution. As shown in Fig. 3, as the privacy budget increases, the F-score of the four algorithms gradually increase. As shown in Fig. 3, the F-score change curves of the MRR and RR algorithms almost coincide, mainly because the MRR algorithm selects the RR algorithm when the privacy budget is low. Moreover, as the privacy budget gradually increases from 4.0 to 6.0, the F-score of the RAPPOR and TRR algorithms do not change significantly. This is because as the privacy budget increases, the degree of perturbation of the original data approaches 0, and the dataset produced by the perturbation closer to the real dataset. Therefore, the TRR algorithm is superior to the RR, RAPPOR and MRR algorithms because the TRR algorithm has the lowest probability of disturbance under the same privacy budget, resulting in a higher F-score. Similarly, Fig. 4 shows that TRR and the other three algorithm mine frequent 3-itemsets. The F-score obtained by the TRR algorithm is larger than that of the other three algorithms, indicating that the TRR algorithm is also suitable for an exponential distribution.

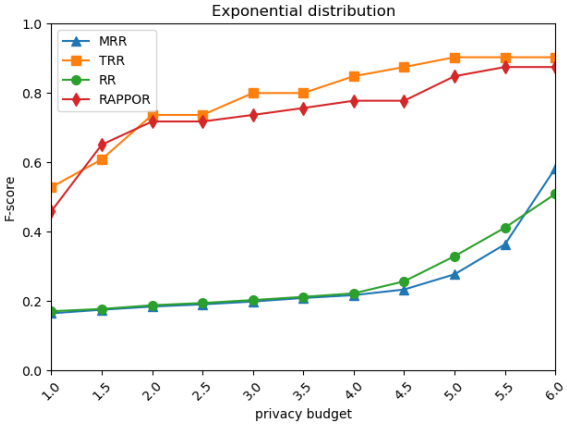

Figure 3: Frequent 2-itemsets of the F-score under exponential distribution

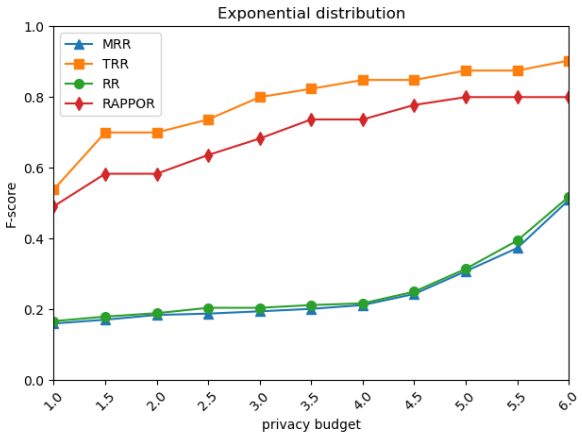

Figure 4: Frequent 3-itemsets of the F-score under exponential distribution

The F-score experimental standard only measures the accuracy and recall of the mining results, but it does not evaluate the experimental error. To analyze the error of the experimental results, we used the relative error measure. For each element in the frequent itemset, we find its initial support for the original data and then find the perturbation support for the perturbed data. Subsequently, we calculate these two support values and then find the median of this set to determine the relative error.

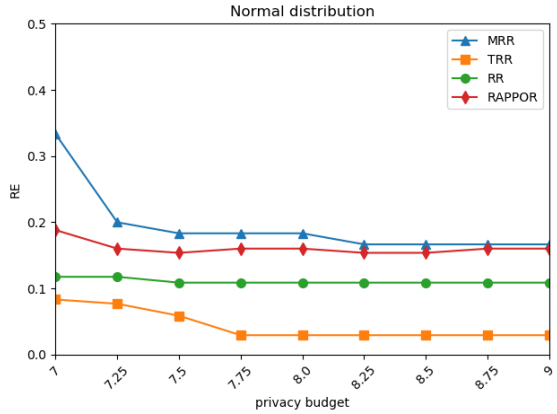

Figure 5: Frequent 2-itemsets of the RE under normal distribution

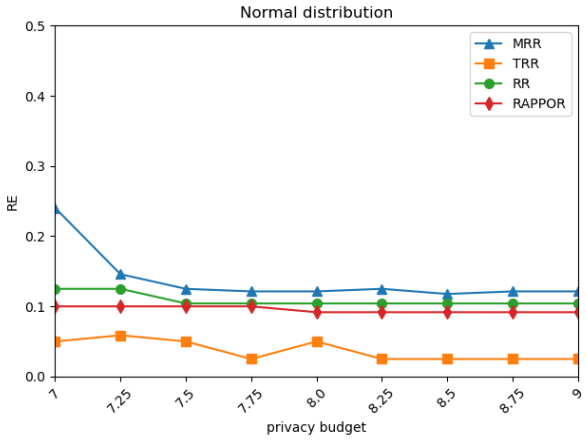

Figure 6: Frequent 3-itemsets of the RE under normal distribution

Figs. 5 and 6 show the results of relative errors of the TRR, MRR, RR, and RAPPOR algorithms for frequent itemset under a normal distribution. Fig. 5 shows that in the process of mining frequent 2-itemsets, as the privacy budget becomes larger, the relative errors of the four algorithms generally show a downward trend. Further, the TRR algorithm has higher accuracy in data mining than the MRR, RR and RAPPOR algorithms. The relative error of the MRR algorithm is the largest, and the relative error varies significantly and is unstable when the privacy budget is 7.0-7.5. This is mainly because the disturbance probability is more sensitive to these privacy budgets and the uncertainty of random disturbances. Similarly, Fig. 6 similar characteristics to Fig. 5. With the increase in the privacy budget, the relative error of the TRR algorithm is smaller than the relative errors the other three algorithms, which also shows that the TRR algorithm has better utility than the MRR, RR and RAPPOR algorithms in mining frequent itemsets.

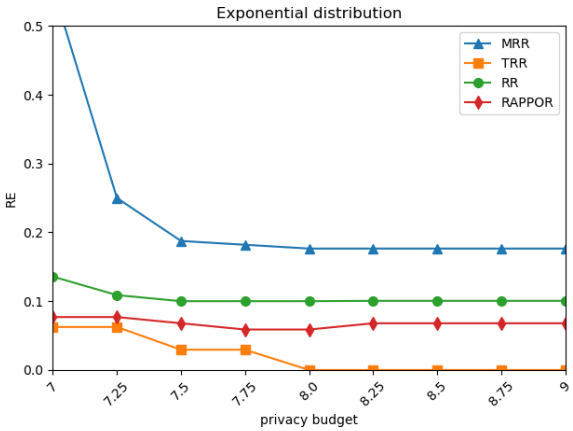

Figure 7: Frequent 2-itemsets of the RE under exponential distribution

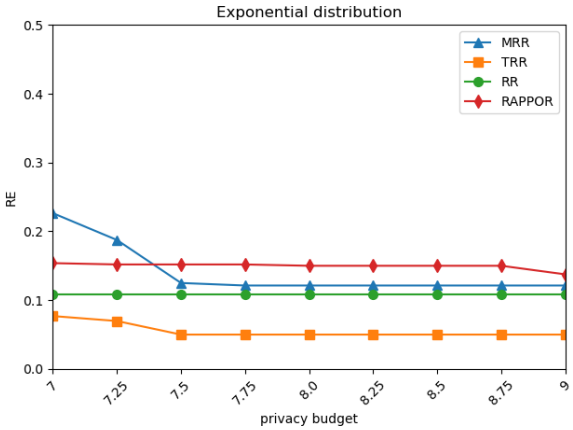

Figure 8: Frequent 3-itemsets of the RE under exponential distribution

Fig. 7 and 8 show the influence of the privacy budget on the relative errors of the TRR, MRR, RR and RAPPOR algorithms under exponential distribution. Fig. 7 shows that the relative error of TRR algorithm is smaller than that of the MRR, RR and RAPPOR algorithms, and the relative error of the MRR algorithm is high. The relative error of the TRR algorithm is less because the TRR disturbance probability is lower than that of the other three algorithms under the same privacy budget. Fig. 8 shows frequent 3-itemset ming, demonstrating that the relative error of the TRR algorithm is still less than that of the RR and RAPPOR algorithms.

In summary, F-score of the TRR algorithm is larger, and the relative error is the smallest. When the user has multiple attributes, the RR and RAPPOR algorithms are determined by the attribute threshold. The disadvantage of the MRR algorithm is that the data distribution estimation is not suitable for frequent itemset mining. In short, if the user has multiple attributes, choosing the TRR algorithm will be a better choice.

7 Conclusions

We propose a complete implementation framework to mine frequent itemsets under local differential privacy protection. The general process is that the original data is first bitmap encoded, and the encoded data implement local differential privacy perturbation; the aggregator performs unbiased estimation after perturbing data, and then uses FP-tree to mine frequent itemsets. We propose a new TRR algorithm, mainly to satisfy the complex data types under multiple attributes of users, and the thresholds of multiple attributes can vary greatly. The TRR algorithm satisfies the requirements of complex data and ensures that the user's personal information dose not leak. Finally, the FP-tree algorithm is used to mine frequent itemsets and the TRR, RR, RAPPOR and MRR algorithms are compared experimentally. We found that the TRR algorithm is better than the existing algorithms.

Funding Statement: This paper is supported by the Inner Mongolia Natural Science Foundation (Grant Number: 2018MS06026, Sponsored Authors: Liu, H. and Ma, X., Sponsors' Websites: http://kjt.nmg.gov.cn/) and the Science and Technology Program of

Inner Mongolia Autonomous Region (Grant Number: 2019GG116, Sponsored Authors: Liu, H. and Ma, X., Sponsors' Websites: http://kjt.nmg.gov.cn/).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Bassily, R.; Nissim, K.; Stemmer, U. (2017): Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, pp. 2288-2296.

Bhaskar, R.; Laxman, S.; Smith, A. (2010): Discovering frequent patterns in sensitive data. *Proceedings of the 16th International Conference on Knowledge Discovery and Data mining*, pp. 503-512.

Bun, M.; Nelson, J.; Stemmer, U. (2018): Heavy hitters and the structure of local privacy. *Proceedings of the 37th Symposium on Principles of Database Systems*, pp. 435-447.

Cheng, X.; Su, S.; Xu, S. (2015): Dp-apriori: a differentially private frequent itemset mining algorithm based on transaction splitting. *Computers & Security*, vol. 50, no. 1, pp. 74-90.

Dwork, C. (2006): Differential privacy. *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming,* pp. 1-12.

Erlingsson, Ú.; Pihur, V.; Korolova, A. (2014): Rappor: Randomized aggregatable privacy-preserving ordinal response. *Proceedings of the Conference on Computer and Communications Security*, pp. 1054-1067.

Gu, K.; Yang, L. H.; Yin, B. (2018): Location data record privacy protection based on differential privacy mechanism. *Information Technology and Control*, vol. 47, no. 4, pp. 639-654.

He, S. M.; Zeng, W. N.; Xie, K.; Yang, H. M.; Lai, M. Y. et al. (2017): PPNC: privacy preserving scheme for random linear network coding in smart grid. *KSII Transactions on Internet & Information Systems*, vol. 11, no. 3, pp. 1510-1532.

Kairouz, P.; Oh, S.; Viswanath, P. (2014): Extremal mechanisms for local differential privacy. *Advances in Neural Information Processing Systems*, pp. 2879-2887.

Kasiviswanathan, S. P.; Lee, H. K.; Nissim, K. (2011): What can we learn privately? *SIAM Journal of Computing*, vol. 40, no. 3, pp. 793-826.

Li, N.; Li, T.; Venkatasubramanian, S. (2007): t-closeness: Privacy beyond k-anonymity and l-diversity. *Proceedings of the 23rd International Conference on Data Engineering*, pp. 106-115.

Min, Z.; Yang, G.; Wang, J.; Kim, G. J. (2019): A privacy-preserving bgn-type parallel homomorphic encryption algorithm based on LWE. *Journal of Internet Technology*, vol. 20, no. 7, pp. 2189-2200.

Qiu, L.; Li, Y.; Wu, X. (2006): An approach to outsourcing data mining tasks while protecting business intelligence and customer privacy. *Proceedings of the International Conference on Data Mining*, pp. 551-558.

Tai, C. H.; Yu, P. S.; Chen, M. S. (2010): k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. *Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining*, pp. 473-482.

Thakurta, A. G.; Vyrros, A. H.; Vaishampayan, U. S. (2017): Learning new words. US Patent 9,594,741.

Wang, T.; Li, N.; Jha, S. (2019): Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, pp. 1-12.

Wong, R. C. W.; Li, J.; Fu, A. W. C. (2006): (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. *Proceedings of the 12th International Conference on Knowledge Discovery and Data mining*, pp. 754-759.

Wong, W. K.; Cheung, D. W.; Hung, E. (2007): Security in outsourcing of association rule mining. *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 111-122.

Xiong, X.; Chen, F.; Huang, P. (2018): Frequent itemsets mining with differential privacy over large-scale data. *IEEE Access*, vol. 6, no. 1, pp. 28877-28889.

Yin, C. Y.; Shi, L. F.; Sun, R. X; Wang, J. (2019): Improved collaborative filtering recommendation algorithm based on differential privacy protection. *Journal of Supercomputing*, https://doi.org/10.1007/s11227-019-02751-7.

Zhang, X.; Huang, L.; Fang, P. (2017): Differentially private frequent itemset mining from smart devices in local setting. *Wireless Algorithms, Systems, and Applications*, pp. 433-444.