IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 5, Issue 1 (Sep-Oct. 2012), PP 06-11
Www.iosrjournals.org

An Effective Heuristic Approach for Hiding Sensitive Patterns in Databases

Mrs. P.Cynthia Selvi[1], Dr. A.R.Mohamed Shanavas[2]
[1]Associate Professor, Dept. of Computer Science, KNGA College(W), Thanjavur 613007 /Affiliated to Bharathidasan University, Tiruchirapalli, TamilNadu, India.
[2]Associate Professor, Dept. of Comuter Science, Jamal Mohamed College, Tiruchirapalli 620 020/ Affiliated to Bharathidasan University, Tiruchirapalli, TamilNadu, India.

Abstract: *Privacy has been identified as a vital requirement in designing and implementing Data Mining (DM) systems. This motivated Privacy Preservation in Data Mining (PPDM) as a rising field of research and various approaches are being introduced by the researchers. One of the approaches is a sanitization process, that transforms the source database into a modified one that the adversaries cannot extract the sensitive patterns from. This study address this concept and proposes an effective heuristic-based algorithm which is aimed at minimizing the number of removal of items in the source database possibly with no hiding failure.*

Keywords: *Privacy Preserving Data Mining, Restrictive Patterns, Sanitized database, Sensitive Transactions.*

I. Introduction

PPDM is a novel research direction in DM, where DM algorithms are analyzed for the side-effects they incur in data privacy. The main objective of PPDM is to develop algorithms for modifying the original data in some way, so that the private data and private knowledge remain private even after the mining process[1]. In DM, the users are provided with the data and not the association rules and are free to use their own tools; So, the restriction for privacy has to be applied on the data itself before the mining phase.

For this reason, we need to develop mechanisms that can lead to new privacy control systems to convert a given database into a new one in such a way to preserve the general rules mined from the original database. The procedure of transforming the source database into a new database that hides some sensitive patterns or rules is called the *sanitization process*[2]. To do so, a small number of transactions have to be modified by deleting one or more items from them or even adding noise to the data by turning some items from 0 to 1 in some transactions. The released database is called the *sanitized database.* On one hand, this approach slightly modifies some data, but this is perfectly acceptable in some real applications[3, 4]. On the other hand, such an approach must hold the following restrictions:

- The impact on the source database has to be minimal
- An appropriate balance between the need for privacy and knowledge has to be guaranteed.

This study mainly focus on the task of minimizing the impact on the source database by reducing the number of removed items from the source database with only one scan of the database. Section-2 briefly summarizes the previous work done by various researchers; In Section-3 preliminaries are given. Section-4 states some basic definitions and of which definition 5 is framed by us which is used in the proposed heuristic-based algorithm. In Section-5 the proposed algorithm is presented with illustration and example. As the detailed analysis of the experimental results on large databases is under process, only the basic measures of effectiveness is presented in this paper, after testing the algorithm for a sample generated database.

II. Related Work

Many researchers have paid attention to address the problem of privacy preservation in association rule mining in recent years. The class of solutions for this problem has been restricted basically to randomization, data partition, and data sanitization.

The idea behind data sanitization to reduce the support values of restrictive itemsets was first introduced by Atallah et.al[1] and they have proved that the optimal sanitization process is NP-hard problem. In [4], the authors generalized the problem in the sense that they considered the hiding of both sensitive frequent itemsets and sensitive rules. Although these algorithms ensure privacy preservation, they are CPU-intensive since they require multiple scans over a transactional database. In the same direction, Saygin [5] introduced a method for selectively removing individual values from a database to prevent the discovery of a set of rules, while preserving the data for other applications. They proposed some algorithms to obscure a given set of sensitive rules by replacing known values with unknowns, while minimizing the side effects on non-sensitive rules. These algorithms also require various scans to sanitize a database depending on the number of association rules to be hidden.

Oliveira introduced many algorithms of which IGA[6] & SWA[7] aims at multiple rule hiding. However, IGA has low misses cost; It groups restrictive itemsets and assigns a victim item to each group. This clustering leads to the overlap between groups and it is not an efficient method to optimally cluster the itemsets. It can be improved further by reducing the number of deleted items. Whereas, SWA improves the balance between protection of sensitive knowledge and pattern discovery but it incurs an extra cost because some rules are removed inadvertently. In our work, we focus on the heuristic based data sanitization approach.

III. Preliminaries

Transactional Database. A transactional database is a relation consisting of transactions in which each transaction *t* is characterized by an ordered pair, defined as *t* = <*Tid, list-of-elements*>, where *Tid* is a unique transaction identifier number and *list-of-elements* represents a list of items making up the transactions. For instance, in market basket data, a transactional database is composed of business transactions in which the *list-of-elements* represents items purchased in a store.

Basics of Association Rules . One of the most studied problems in data mining is the process of discovering association rules from large databases. Most of the existing algorithms for association rules rely on the support-confidence framework introduced in [8].

Formally, association rules are defined as follows: Let $I = \{i_1,...,i_n\}$ be a set of literals, called items. Let *D* be a database of transactions, where each transaction *t* is an itemset such that $t \subseteq I$. A unique identifier, called *Tid*, is associated with each transaction. A transaction *t* supports *X*, a set of items in *I*, if $X \subset t$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \phi$. Thus, we say that a rule $X \Rightarrow Y$ holds in the database *D* with *support* (σ) if $\frac{|X \cup Y|}{N} \geq \sigma$, where *N* is the number of transactions in *D*. Similarly, we say that a rule $X \Rightarrow Y$ holds in the database *D* with *confidence* (φ) if $\frac{|X \cup Y|}{|X|} \geq \varphi$, where $|A|$ is the number of occurrences of the set of items *A* in the set of transactions *D*. While the support is a measure of the frequency of a rule, the confidence is a measure of the strength of the relation between sets of items.

Association rule mining algorithms rely on the two attributes, *minimum Support(minSup σ)* and *minimum Confidence(minConf φ)*. The problem of mining association rules have been first proposed in 1993[8].

Frequent Pattern. A pattern X is called a frequent pattern if *Sup(X) ≥ minSup* or if the *absolute support* of X satisfies the corresponding *minimum support count* threshold. [pattern is an itemset; in this article, both terms are used synonymously]. All association rules can directly be derived from the set of frequent patterns[8, 9]. The conventions followed here are

- *Apriori property[10]: all non empty subsets of a frequent itemsets(patterns) must also be frequent.*
- *Antimonotone property: if a set cannot pass a test, then all of its supersets will fail the same test as well.*

Privacy Preservation in Frequent Patterns. The most basic model of privacy preserving data processing is one in which we erase the sensitive entries in the data. These erased entries are usually particular patterns which are decided by the user, who may either be the owner or the contributor of the data.

IV. Problem Definition

In this approach, the goal is to hide a group of frequent patterns which contains highly sensitive knowledge. Such sensitive patterns that should be hidden are called *restrictive patterns*. Restrictive patterns can always be generated from frequent patterns.

Definition 1. Let D be a source database, containing a set of all transactions. T denotes a set of transactions, each transaction containing itemset $X \in D$. In addition, each k-itemset $X \subseteq I$ has an associated set of transactions $T \subseteq D$, where $X \subseteq t$ and $t \in T$.

Definition 2 : *Restrictive Patterns* : Let D be a source database, P be a set of all frequent patterns that can be mined from D, and $Rules_H$ be a set of decision support rules that need to be hidden according to some security policies. A set of patterns, denoted by R_P is said to be *restrictive*, if $R_P \subset P$ and if and only if R_P would derive the set $Rules_H$. $\sim R_P$ is the set of *non-restrictive patterns* such that $\sim R_P \cup R_P = P$.

Definition 3 : *Sensitive Transactions* : Let T be a set of all transactions in a source database D, and R_P be a set of restrictive patterns mined from D. A set of transactions is said to be *sensitive*, denoted by S_T, if every $t \in S_T$ contain atleast one restrictive pattern, ie $S_T = \{ t \in T \mid \exists X \in R_P, X \subseteq t \}$. Moreover, if $S_T \subset T$ then all restrictive patterns can be mined one and only from S_T.

Definition 4 : *Transaction Degree* : Let D be a source database and S_T be a set of all sensitive transactions in D. The *degree of a sensitive transaction t*, denoted as *deg(t)*, such that $t \in S_T$ is defined as the number of restrictive patterns that *t* contains.

Definition 5: *Cover* : The *Cover* of an item A_k can be defined as, $C_{Ak} = \{ rp_i \mid A_k \in rp_i \subset R_P, 1 \leq i \leq |R_P| \}$ *i.e.*, set of all restrictive patterns(rp_i's) which contain A_k. The item that is included in a maximum number of rp_i's is the one with *maximal cover or maxCover; i.e., maxCover = max*($|C_{A1}|, |C_{A2}|, \ldots |C_{An}|$) such that $A_k \in rp_i \subset R_P$.

Based on the above definitions, the main strategy addressed in this work can be stated as given below:
If D is the source database of transactions and P is the set of relevant patterns that would be mined from D, the goal is to transform D into a sanitized database D', so that the most frequent patterns in P can still be mined from D' while others will be hidden. In this case, D' becomes the released database.

V. Sanitization Algorithm

The optimal sanitization has been proved to be an NP-hard problem. To alleviate the complexity of the optimal sanitization, some heuristics could be used. A heuristic does not guarantee the optimal solution but usually finds a solution close to the best one in a faster response time. In this section, the proposed sanitizing algorithm and the heuristics to sanitize a source database are introduced.

Given the source *database (D)*, and the *restrictive patterns(R_P)*, the goal of the sanitization process is to protect R_P against the mining techniques used to disclose them. The sanitization process decreases the support values of restrictive patterns by removing items from sensitive transactions. This process mainly includes four sub-problems:

1. identifying the set of sensitive transactions for each restrictive pattern;
2. selecting the partial sensitive transactions to sanitize;
3. identify the candidate item(*victim item*) to be removed;
4. rewriting the modified database after removing the *victim items*.

Basically, all sanitizing algorithms differs only in subproblems 2 & 3.

5.1. Heuristic approach

In this work, the proposed algorithm is based on the following heuristics:

Heuristic-1. To solve subproblem-2 stated above, S_T is sorted in decreasing order of (*deg + size)*, thereby the sensitive transactions that contains more number of patterns can be selected; this enable multiple patterns to be sanitized in a single iteration.

Heuristic-2. To solve subproblem-3, the following heuristic is used in the algorithm :

for every item $A_k \in R_P$, find *cover* and starting from *maximal Cover*, find T = $\bigcap_{i=1}^{|rpi-list(Ak)|} t\_list(rpi)$;
for every $t \in$ T, mark A_k as the *victim item* and remove.

The rationale behind these two heuristics is to minimize the *sanitization rate* and thereby reducing the impact on the source database.

Note : In this work, no sensitive transaction is completely removed. (ie, the number of transactions in the source database is not altered).

5.2. Algorithm

Input : (i) D – Source Database (ii) R_P – Set of all Restrictive Patterns
Output : D' – Sanitized Database
Pre-requisites :
(i) Find Frequent Patterns(Itemsets) using *Matrix Apriori* Algorithm;
(ii) Form Look-up Table-1 : $\forall A_k \in R_P$, $LT1(A_k) \leftarrow t\text{-}list \,/\, t \in D$
(iii) Form Look-up Table-2 : $\forall rp_i \in R_P$, $LT2(rp_i) \leftarrow t\text{-}list \,/\, t \in D$
(iv) Form Look-up Table-3 : $\forall A_k \in R_P$, $LT3(A_k) \leftarrow rp_i\text{-}list \,/\, rp_i \in R_P$

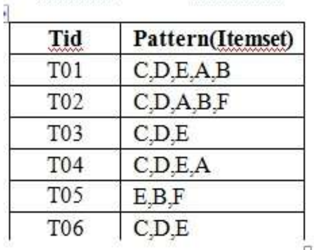
***Algorithm maxCover1*:** // based on Heuristics 1 & 2 //

```
Step 1 : calculate $supCount(rp_i)$, $\forall rp_i \in R_P$ and sort in decreasing order ;
Step 2 : find Sensitive Transactions($S_T$) w.r.t. $R_P$ ;
        a) calculate $deg(t), size(t)\ \forall\ t \in S_T$;
        b) sort $t \in S_T$ in decreasing order of deg & size ;
Step 3 : find $\sim S_T \leftarrow D - S_T$;            // $\sim S_T$ - non sensitive transactions //
Step 4 : // Find $S_T$' //
        find cover for every item $A_k \in R_P$ and sort in decreasing order of cover;
        for each item $A_k \in R_P$ do
        {
         repeat
        find T = $\bigcap_{i=1}^{|rpi-list|} t$
        for each $t \in T$ do
        {
        delete item $A_k$ in non VictimTransactions such that $A_k \in rp_i \subset rp_i$-list ; // $A_k$ – victimItem //
         // initially all t are nonvictim //
        decrease supCount of $rp_i$'s for which t is nonVictim;
        mark t as victimTransaction in each t-list of $rp_i \subset rp_i$ _list($A_k$) ;
        }
        until (supCount = 0) for all $rp_i \in R_P$
        }
        Step 5 : D' $\leftarrow$ ~ $S_T \cup S_T$'
```

5.3. Illustration

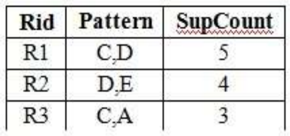
The following examples help understand how the proposed Heuristics work. Refer the *Source Database(D)* in Table-1. The set of all *Restrictive Patterns* to be hidden are given in Table-2. *The sensitive transactions-* $\boldsymbol{S_T}$ (transactions which include atleast one *Restrictive Pattern*) are identified from D and are extracted. They are sorted in decreasing order of their *deg* and *size*(Table-3). *Non sensitive transactions*(~$\boldsymbol{S_T}$) are also filtered and stored separately(Table-5).

Table-1. Source database(D)



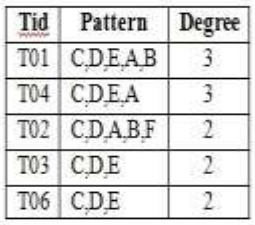
Tid	Pattern(Itemset)
T01	C,D,E,A,B
T02	C,D,A,B,F
T03	C,D,E
T04	C,D,E,A
T05	E,B,F
T06	C,D,E

Table-2. Restrictive patterns(R_P)



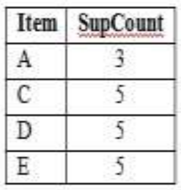
Rid	Pattern	SupCount
R1	C,D	5
R2	D,E	4
R3	C,A	3

Table-3. Sensitive transactions(S_T) Dec.order of deg & size



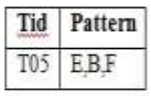
Tid	Pattern	Degree
T01	C,D,E,A,B	3
T04	C,D,E,A	3
T02	C,D,A,B,F	2
T03	C,D,E	2
T06	C,D,E	2

Table-4. SupCount(A_k) in Asc.order



Item	SupCount
A	3
C	5
D	5
E	5

Table-5. NonSensitive transactions (~S_T)



Tid	Pattern
T05	E,B,F

The proposed algorithm refer the *Look-up Tables*(listed below) to speed up the process.

LookUp Table-1 [*item* ← *t-list*]

Item	Transactions	No.of Trans.
C	T01,T04,T02,T03,T06	5
D	T01,T04,T02,T03,T06	5
E	T01,T04,T03,T06	4
A	T01,T04,T02	3

LookUp Table-2 [rp_i ← *t-list*]

Item	Transactions	SupCount
R1	T01,T04,T02,T03,T06	5
R2	T01,T04,T03,T06	4
R3	T01,T04,T02	3

LookUp Table-3 [*item* ← *rp*$_i$*-list*]

Item	Rules	Cover
C	R1, R3	2
D	R1, R2	2
E	R2	1
A	R3	1

Table.6. Sanitized Database(D')

Tid	Pattern(Itemset)
T01	E,A,B
T02	D,A,B,F
T03	C,E
T04	E,A
T05	E,B,F
T06	C,E

The proposed algorithm is based on the Heuristics 1 and 2. Here, for every item A_k starting from *maxCover* (refer LookUp Table-3), find the sensitive transactions which are common for all rules associated with A_k. *i.e.*, [T = $\bigcap_{i=1}^{|rpi-list|} t$]. In every transaction *t* in T, remove A_k and decrease the *supCount* of all *restrictive patterns(rp*$_i$*)* associated with *t* and which contain A_k.

Example : Item C (being the first item with *maxCover*), R1 & R3 form the *rp*$_i$*-list(C);* whose common transactions are [T01, T04, T02]. Remove C from these transactions, which reduce the *supCount* of both R1 and R3 by 3. Mark these *t* as victim transactions for R1 and R3. When we consider the next item D, the ***t****-list* of *rp*$_i$*-list(D)* are [T01, T04, T03, T06]. Removing D from T01 and T04 would not reduce the *supCount* of R1(because they are already considered in the previous iteration); but would reduce the *supCount* of R2. Hence, remove it and decrease only the *supCount* of R2. Whereas removing D from T03 and T06 would reduce the *supCount* of both R1 and R2. This process is repeated until the *supCount* of all *rp*$_i$'s are reduced to 0. The modified form of sensitive transactions are denoted as S_T*'*.

Then the Sanitized database ***D'***(refer table-6)is formed bycombining $\sim S_T$ (Table-5) and S_T**'**.

VI. Implementation

This algorithm is tested in the environment of Intel core 2 duo Processor with 2.5 GHz speed and 4 GB RAM, running Windows XP. We have used NetBean 6.9.1 to code the algorithm using Java(JDK 1.7) with SQL Server 2005. It has been tested for a sample database to verify the main objectives (minimal removal of items and no hiding failure) and it requires only one scan of the source database. However, the testing process for very large real databases is in progress for a detailed analysis. Before the hiding process, the *frequent patterns* are obtained using ***Matrix Apriori Algorithm***[11], which is faster and use simpler data structure than the Apriori algorithm[9,10]. Moreover, it scans the database only twice and works without candidate generation.

VII. Effectiveness Measures

(i) Sanitization Rate(SR) : It is defined as the ratio of removed items(*victim items*) to the total support value of restrictive patterns(*rp*$_i$) in the source database D.

$$\text{SR} = \frac{|victim\ items|}{total\ supCount(rpi)}$$

With the sample database tested, it is found that the SR is less than 50%.

(ii) Hiding Failure(HF) : It is assessed by the restrictive patterns that were failed to be hidden. In other words, if a hidden restrictive pattern cannot be extracted from the released database D' with an arbitrary *minSup*, the hidden pattern has no hiding failure occurrence.

$$\text{HF} = \frac{|RP(D')|}{|RP(D)|}$$

As far as the algorithm *maxcover1* is concerned, the restrictive patterns are modified till their respective *supcount* becomes zero. Moreover it ensures that no transaction is completely removed. Hence this algorithm is 100% HF free.

(iii) Misses Cost(MC) : This measure deals with the legitimate patterns(non restrictive patterns) that were accidently missed.

$$\text{MC} = \frac{|\sim RP(D)| - |\sim RP(D')|}{|\sim RP(D)|}$$

Note: There is a compromise between the MC and HF; ie, the more patterns we hide, the more legitimate patterns we miss. But with the sample database tested, this algorithm has 0% MC.

(iv) Artifactual Pattern(AP) : AP occurs when some artificial patterns are generated form D' as an outcome of the sanitization process.

$$\text{AP} = \frac{|P'| - |P \cap P'|}{|P'|}$$

As this algorithm hides restrictive patterns by selectively removing items (instead of swapping, replacement, etc.,) from the source database(D), it does not generate any artifactual pattern.

VIII. Conclusion

In this competitive but cooperative business environment, companies need to share information with others, while at the same time, have to protect their own confidential knowledge. To facilitate this kind of data sharing with privacy protection, the algorithm based on *maxCove*r is proposed. This algorithm ensure that no counterpart or adversary can mine the restrictive patterns even with an arbitrarily very small *supCount.*

This algorithm is based on the strategy to simultaneously decrease the support count of maximum number of sensitive patterns (itemsets), with possibly minimum number of removal of items and it reduce the impact on the source database. The proposed algorithm has minimal sanitization rate possibly with no hiding failure and low misses cost. Above all this algorithm scans the original database only once. It is important to note that the proposed algorithm is robust in the sense that there is no desanitization possible. The alterations to the original database are not saved anywhere, since the owner of the database still keeps an original copy of the database intact, while disturbing the sanitized database. Moreover, There is no possible way to reproduce the original database from the sanitized one, as there is no encryption involved.

As already mentioned, the work on the time complexity analysis and the dissimilarity study between the original and sanitized databases in an elaborate manner using very large databases is in progress for which the publicly available real databases are being used.

References

[1]. Verykios,V.S, Bertino.E, Fovino.I.N, ProvenzaL.P, Saygin.Y and Theodoridis.Y, “State-of-the-art in Privacy Preservation Data Mining”, New York,ACM SIGMOD Record, vol.33, no.2, pp.50-57,2004.

[2]. Atallah.M, Bertino.E, Elmagarmid.A, Ibrahim.M and Verykios.V.S, “Disclosure Limitation of Sensitive Rules”, In *Proc. of IEEE Knowledge and Data Engineering Workshop*, pages 45–52, Chicago, Illinois, November 1999.

[3]. Clifton.C and Marks.D, “ Security and Privacy Implications of Data Mining”, In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, February 1996.

[4]. Dasseni.E, Verykios.V.S, Elmagarmid.A.K and Bertino.E, “ Hiding Association Rules by Using Confidence and Support”, In *Proc. of the 4th Information Hiding Workshop*, pages 369– 383, Pittsburg, PA, April 2001.

[5]. Saygin.Y, Verykios.V.S, and Clifton.C, “Using Unknowns to Prevent Discovery of Association Rules”, *SIGMOD Record*, 30(4):45–54, December 2001.

[6]. Oliveira.S.R.M and Zaiane.O.R, “Privacy preserving Frequent Itemset Mining”, in the Proc. of the IEEE ICDM Workshop on Privacy, Security, and Data Mining, Pages 43-54, Maebashi City, Japan, December 2002.

[7]. Oliveira.S.R.M and Zaiane.O.R, “An Efficient One-Scan Sanitization for Improving the Balance between Privacy and Knowledge Discovery”, Technical Report TR 03-15, June 2003.

[8]. Agrawal R, Imielinski T, and Swami.A, “Mining association rules between sets of items in large databases”, Proceedings of 1993 ACM SIGMOD international conference on management of data, Washington, DC; 1993. p. 207-16.

[9]. Agrawal R, and Srikant R. “Fast algorithms for mining association rules”, Proceedings of 20th international conference on very large data bases, Santiago, Chile; 1994. p. 487-99.

[10]. Han J, and Kamber M, “Data Mining Concepts and Techniques”, Oxford University Press, 2009.

[11]. Pavon.J, Viana.S, and Gomez.S, “Matrix Apriori: speeding up the search for frequent patterns,” Proc. 24th IASTED International Conference on Databases and Applications, 2006, pp. 75-82.