Towards Association Rule Hiding Heuristics vs Border-based Approaches

Afrah Farea[1], Ali KARCI[2]

[1]Computer Engineering Department, Inonu University, 44280, Malatya Turkey
Othello2009@hotmail.com,
[2] Computer Engineering Department, Inonu University, 44280, Malatya Turkey
ali.karci@ inonu.edu.tr

Abstract

Abstract— Association rule hiding has been playing a very important rule in Privacy Preserving Data Mining and has been gaining the attention of many researchers. Many algorithms are developed with authors claiming the superiority of their algorithms. In this respect, we selectively choose some state of the art approaches in the context of association rule hiding heuristics and border-based approaches and examine their results on Mushroom and Retail datasets.

1. Introduction

Applications of data mining are increasing every day. However, one of the side effects of data mining is privacy; the more we want accurate data mining results, the more we need to breach privacies of individuals and organizations. There are three directions related privacy in this regard; namely data privacy, knowledge privacy and owner privacy [1]. The first direction concerns with raw data that may reveal confidential information like medical records, criminal records, records related to national or international security, etc. The second direction concerns with results of data mining such as classification, clustering and segmentation, sequence analysis, etc. Association rule hiding comes under this category. The third direction deals with sharing data between parties for mining purposes without letting the other party know the content of these data. This problem is commonly known as Secure Multiparty Computation (SMC) [2].

Our contribution here includes the following: First, we present the most common heuristic approaches for solving ARH problem along with the border-based approaches. Second, we evaluate their behavior on Mushroom and Retail datasets. Finally, a number of suggestions are provided to improve these algorithms.

The rest of this paper is organized as follows. In section 2, we state out the problem and provide the necessary background. In section 3, we present the heuristic and border-based approaches. Experimental evaluation is shown in section 4. Section 5 concludes this work.

2. Problem Formulation

Let £ = {i1 ,i2 ,i3 ….im} be a set of items in a dataset D. Any subset I ϵ T is called an itemset. An Itemset is called frequent if its frequency in the dataset is greater than (or equal to certain threshold value called support threshold. Association rule is an implication of the form $X \rightarrow Y$ where X is a frequent itemset called rule antecedent and Y is a frequent itemset called rule consequent and X and Y are disjoint i.e. $X \cap Y = \emptyset$ [3]. Association rule is called strong (or interesting) association rule if its confidence is greater than a certain threshold value called confidence threshold.

Support and confidence are mathematical measures and they are calculated as follows:

$$\text{Support, } s\,(X \rightarrow Y) = (\sigma(X \cup Y))/(|D|) \quad (1)$$
$$\text{Confidence, } c\,(X \rightarrow Y) = (\sigma(X \cup Y))/(\sigma(X)) \quad (2)$$

where X is the rule antecedent , Y is the rule consequent, $\sigma(X \cup Y)$ is number of transactions in which both X and Y occur , $\sigma(X)$ number of transactions in which X occur, $|D|$ is the size of the database. These thresholds are usually defined by the user or the database owner.

Support and confidence have many interesting applications and they can reveal patterns unknown even to the database owner. However, these patterns could sometimes be sensitive due to legal constraints or because of the competition between companies [4]. Thus, association rule hiding concerns with finding efficient algorithms able to hide sensitive patterns with minimal impact on the database. Actually, Atallah et.al. [5]proves that the optimal solution i.e. removing sensitive patterns from frequent itemset list without affecting non-sensitive ones , is NP-hard.

In general, Association rule hiding algorithms should achieve at least one of the following goals under the same support and confidence thresholds or higher [6]:

- No sensitive pattern determined by the database owner is revealed after the application of ARH algorithm.
- Non-sensitive pattern that can be mined from the original database should also be mined from the sanitized database. Such original pattern that could not be mined from the sanitized database is known as missing pattern.
- No new pattern generates from the sanitized database. Such patterns are known as false or ghost patterns.
- Database distortion is minimum.

In order to achieve these goals, association rule hiding algorithms carefully select one transaction at a time, called victim transaction, to modify and an item in this transaction, called victim item, such that the deletion of this item reduces the support of the sensitive pattern with as minimum impact on the non-sensitive patterns as possible. This process continues until sensitive pattern support equals minimum support -1.

3. Sanitation Algorithms

In this section, we select some popular heuristic approaches that have been suggested by their authors and compare them with three border-based approaches. We emphasize here that the core of each algorithm is the candidate selection process, which is to effectively find the victim items with as less impact on the original dataset as possible. We should also mention that the border-based algorithms discussed here are not free from heuristics.

3.1 Heuristic Approaches

As we will see, these approaches carefully select a set of transactions and sanitize them in order to hide sensitive knowledge. These algorithms are simple, fast, efficient, and scalable. However, they do not guarantee the best solution and usually suffer from the local optima problem [1].

3.1.1. Algorithm 2.b.

This algorithm was proposed by Verykios et.al [7], which is considered the first to address the problem by reducing either support or confidence of the sensitive pattern. Here, sensitive itemsets are sorted by their size then support in a decreasing order. Supporting transactions are sorted in an ascending order of transaction size. The item in the sensitive itemset with the highest support is the victim item. In the next step, starting with the first sensitive itemset in order, we remove the victim item from the first sensitive transaction in the list and propagate the results. In propagation, we reduce the support of any other sensitive pattern affected by this deletion and remove the corresponding sensitive transaction from its supporting transaction list. The process continues until the support of the current itemset goes below support threshold**.**

3.1.2 Item Grouping Algorithm (IGA)

This approach is based on clustering the sensitive patterns we need to hide into common patterns called restrictive group. Each restrictive group has a label. The label is an item that belongs to the restrictive group and has the smallest support among other items in the group. Another term is the conflicting sensitive transactions, which means transactions supporting more than one sensitive pattern. These transactions have the priority to modify according to the degree of conflict. Thus when removing the label from the conflicting sensitive transactions, we take care of more than one restrictive pattern at a time. Using this approach, we achieve the purpose of the hiding goal and the minimal impact on the database at the same time.

However, when grouping restrictive patterns, there can be an overlapping between groups because clustering is done in a pair-wise basis and is not transitive. The way of removing the overlapping is defined in [8]. Sensitive transactions have a number associated with them used for counting number of sensitive patterns the transaction supports. This number is called the degree of conflict. Transactions are sorted in a decreasing order according to degree of conflict. Finally, for each sensitive pattern, remove the label of the restrictive group that sensitive pattern belongs to from the first sensitive transaction in order [8].

3.1.3. Hybrid Approach

The advantage of this algorithms is that it considers non-sensitive itemsets during sanitizations. This approach is a combination of two approaches. The first approach is called Aggregate approach in which we find the sensitive transactions and sort them according to supporting sensitive/non-sensitive ratio. The transaction with the maximum ratio is selected as the victim transaction. In the second step Disaggregate approach is used to find the victim item within the transaction found in the previous step using the same criteria i.e. item with maximum sensitive/non-sensitive ratio[9]. This makes the Hybrid approach require many calculations. Actually, the worst case computational complexity of this algorithm is $O(|F|^2|I||D^C|)$ [9]. In our implementation, we reduced the search space by first taking the minimum set of the sensitive frequent itemsets and then continuously updating the search space. This is done by continuously deleting any transaction that turns to be non-sensitive from the sensitive transaction list and any item that does not support any sensitive itemset within the sensitive transaction.

3.1.4. Sliding Window Algorithm (SWA)

In this algorithm, for each K transaction in the database (K is the window size), we sort sensitive transactions ascendingly of transaction size. Then for each sensitive transaction, we count the frequency of each item that belongs to the sensitive pattern. Items inside each transaction are then sorted in a descending order of their frequencies. Item with the highest frequency is the victim item [10]. Finally, we remove the victim item defined in the previous step from the first transaction in order.

3.2. Border-based Approaches

This and the following algorithms are based in on the concept of the border theory [11]. For convenience, we introduce this concept here. For any set of itemsets U, an itemset X $\in$ upper border of U (also called positive border) Bd^+(U), if 1) Bd^+(U) is an anti-chain collection of sets. 2) $\exists$ Y $\in$ U such that Y $\subseteq$ X. Similarly, an itemset X $\in$ lower border of U (also called negative border) Bd^- (U), if 1) Bd^- (U) is an anti-chain collection of sets. 2) $\exists$ Y $\in$ U such that Y $\supseteq$ X [12]. In other words, the upper border is the maximum frequent itemsets from which all frequent itemsets are generated and the lower border is the minimum set of infrequent itemsets and the border is the union between them. These two concepts are direct consequences from the Apriori property since any frequent itemset implies that all its subsets are also frequent and no frequent superset could be generated from a non-frequent one. Thus, these two concepts are very useful since they allow us to track the impact of deleting victim items on the database during the hiding process. In all these algorithms, we try move the sensitive itemsets to be on the negative border side while keeping other elements in the positive border intact. That is why Bd^+ and Bd^- are usually used to denote maximum frequent non-sensitive itemsets and minimum set of sensitive itemsets respectively.

3.2.1. Border-based algorithm(BBA)

The key idea of this algorithm is to keep track of the positive border elements by assigning them weights showing their vulnerability of being affected by item deletion. These weights are continuously updated according to the current support of their corresponding itemsets. The authors suggest the following equation to track these weights:

$$W(I \in Bd^{+}) = \begin{cases} \frac{sup(I,D_0) - sup(I,D') + 1}{sup(I,D_0) - msup}, & sup(I,D') \geq msup + 1 \\ \lambda + msup - sup(I,D'), & 0 \leq sup(I,D') \leq msup \end{cases}$$

where λ is an integer larger than number of revised positive border itemsets, D' represents database during sanitization.

In order to find the victim item, the algorithm associates an interval, called impact interval, with each item belonging to the sensitive itemset where the left boundary of the interval represents the summation of the weights of the direct positive border elements. An itemset X is called direct positive border of sensitive itemset Y if X is a positive border element and $X \subset Y$. The right boundary is the summation of the weights of all relevant positive border elements including the direct positive ones. Upon finding the impact interval of each item on the border elements, a partial order relation $\succeq$ is used to find the item with minimal impact on the border [12].

In order to reduce the search space for finding the victim transaction, the algorithm associates a vector map with each supporting sensitive transaction. The length of this vector equals to $|Bd^{+}|_{x_i}|$ i.e. the positive border elements supported by this transaction and will be affected when hiding sensitive itemset X and choosing item i as victim item. Then the victim transaction is the one with the least weight summation [12].

3.2.2. MaxMin1 Algorithm

This and the next algorithm are based on the max-min principle used in decision theory to maximize the minimum profit. They hide sensitive itemsets according to a set of theories devised by the authors in [13, 14]. First, we find Bd^{+} and Bd^{-} and sort elements in Bd^{-} in increasing order of support and decreasing order of length. Then, for each element X in Bd^{-}, we find $Bd^{+}|_{X}$. For each item, called tentative victim item, in X, we construct a list called affinity list defining the possibly affected positive border elements. From each list, we choose the itemsets with the minimum support and from the result; we take the elements with the maximum support. This final list is called max-min list from which we randomly select an itemset and its corresponding tentative victim item in the affinity list is the victim item. Finally, we remove this item from the first transaction supporting the sensitive itemset.

3.2.3. MaxMin2 Algorithm

This algorithm improves over the previous algorithm. It distinguishes between three cases the first case scenario is when the max-min itemsets $Bd^{+}|_{j}$ belong to only one tentative victim item j. Here we delete this item from transactions supporting the current sensitive itemset but not supporting any of the itemsets in the max-min list. Otherwise, it chooses a transaction at random. The second case scenario is when the max-min itemsets are all derived from different tentative victim items. In this case, the algorithm iterates over each max-min itemset relevant to each tentative victim item to find the transaction supporting current sensitive itemset but not supporting any of the itemsets in the max-min list relevant to that item. If there are any, it removes the victim item from a transaction in the resultant list. When all cases fail, the algorithm iterates over all possible pair itemsets in the max-min list to find transactions affecting only one list, if there are any, it removes the corresponding victim item from one of them. Otherwise, it removes victim item from a random selected transaction supporting the first list [14, 15].

4. Experimental Results

In this section, we present the software used to implement those algorithms as well as the measurements and the datasets used for evaluation.

4.1. Software Description

All the algorithms were coded in Visual Studio 2013 with C#. For Frequent itemset generation, we used FPgrowth algorithm implemented in java using SPMF library [17]. We also used IKVM.openJDK.Core library [18] to link java codes in .Net environment.

We run the algorithms on an Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz (4 CPUs), ~2.6GHz with 8084MB RAM available memory.

4.2. Description of the real datasets

For algorithms evaluation, we used two publically available datasets, namely, Mushroom from UCI Machine Learning Repository and Retail dataset from an anonymous Belgian retail store. The two datasets can also be found here [13]. Description of these datasets is shown in Table 1. Threshold values and number of randomly selected itemsets to be sensitive are shown in Table 2.

Table 1.Datasets descriptions

Dataset Name	# Transactions	# Items	Average Transaction Length
Mushroom	8124	119	23.0
Retail	88162	16470	11.0

For algorithms BBA, MaxMin1 and MaxMin2, we compare our implementation results with results in [14, 15] using a sample dataset mentioned in [12, 16]. For this particular example, we use the same notion used in [14, 15]. As suggested by the authors, the notion m/n is used to measure the performance of each algorithm where m indicates the number of raw database changes and n indicates number of non-sensitive frequent itemsets accidently hidden. The winning algorithm is underlined [14, 15].

Because both MaxMin1 and MaxMin2 choose the victim item and/or transactions at random in some cases, we run these algorithms twice. By comparing results in table (3), we see that our implementation for both MaxMin1 and MaxMin2 are approximately similar to the results in [14, 15]. However, our implementation of BBA algorithm shows better results.

Table 2. Datasets and parameters used

Dataset Name	Relative Min Support (%)	Absolute Support	Non Singleton Frequent İtemsets	# Sensitive Itemsets(with supersets)
Mushroom	25	2031	5510	5(208)
				13(803)
				25(1208)
				50(1670)
				100(2894)
				150(3292)
				200(3374)
				300(3964)
Retail	0.14	123.43	3231	5(7)
				13(15)
				25(37)
				50(83)
				100(134)
				150(360)

Table 3 Comparison between our results and results in [14, 15] for evaluating Border-based, MaxMin1 and MaxMin2 algorithms (our implementations are shown with background color).

Sensitive Itemsets	BBA	BBA (our imp)	Max Min1	MaxMin1 (our imp)	MaxMin2	MaxMin2 (our imp)
Ab	2/0	2/0	2/0	2/0 – 2/0	2/0	2/0 – 2/0
Ad	4/1	4/1	4/1	4/0 – 4/0	4/0	4/1 – 4/1
Cd	4/2	4/1	4/3	4/3 - 4/3	4/3	4/1 – 4/1
Abd	1/0	1/0	1/0	1/0 – 1/0	1/0	1/0 – 1/0
Cde	1/1	1/1	1 /2	1/3 – 1/2	1/1	1/1 – 1/1
ab, Acd	4/1	3/0	3/0	3/0 – 3/0	3/0	3/0 – 3/0
ac, abd	4/1	3/0	3/1	3/0 – 3/1	3/0	3/0 – 4/1
ad, bcd	5/1	5/0	5/1	5/1 – 5/1	5/0	5/1 – 5/1
bc, cde	2/1	2/1	2/1	2/1 – 3/3	2/1	3/1 – 3/1
ce, abd	2/0	2/1	2/1	2/1 – 2/1	2/0	2/1 – 2/0
ac, abd, cde	4/1	3/1	4/2	4/3 – 4/3	3/1	4/1 – 3/1
ab, de, acd	5/2	4/1	4/1	4/1 – 4/1	3/0	4/0 – 4/0
ac, ad, bcd	5/0	5/0	6/1	6/2 – 6/2	5/0	5/0 – 5/0
abd, acd, cde	4/2	3/1	3/2	3/2 – 3/2	3/2	3/2 – 3/2
abd, acd, bcd	4/0	4/0	3/0	3/0 – 3/2	3/0	4/0 – 4/0
ab, bc, cd, de	9/2	7/0	8/2	8/2 – 8/2	7/0	7/0 – 7/0

4.3. Effectiveness Measures

Here we evaluate the algorithms using two measurements namely; Misses Costs and database difference size suggested by authors in [8]. We should also mention that in all these algorithms no hiding failures or artificial patterns found. We set disclosure threshold at 0.0 in IGA and SWA. For SWA, we set K (window size) at 100.

- Misses Cost (MC): measures the amount of legitimate patterns accidently deleted due to sanitization. Misses cost is calculated by:

$$MC = \frac{\#\sim R_R(D) - \#\sim R_R(D')}{\#\sim R_R(D)}$$

  where $\sim R_R(X)$ is the non-restrictive patterns mined from database X.
- Difference between the original (D) and sanitized datasets (D'), is measured as:

$$\mathrm{Diff}(D,D') = \frac{1}{\sum_{i=1}^{n} f_D(i)} * \sum_{i=1}^{n}[f_D(i) - f_{D'}(i)]$$

where n is the number of distinct items in the original dataset and $f_X(i)$ is the frequency of the i_{th} Item in dataset X. This later measurement is also known as item-wise accuracy, as opposed to transaction-wise accuracy, which means number of transactions that remain intact after sanitization [9].

5. Evaluation

To evaluate the algorithms, we choose two publically published datasets with different characteristics in terms of number of transactions, number of items and average transaction length as shown in table (1). From the frequent itemsets generated in each dataset, we randomly selected six sets and eight sets of non-singleton frequent itemsets to be sensitive in Retail and Mushroom respectively. The results of these selections are shown in table (2). Thus for the sake of comparison, we used the same selected sensitive itemsets in each algorithm. In case of Sliding Window Algorithm, we set disclosure threshold value at minimum support value to hide sensitive itemsets completely. We also set K=|D|, i.e. size of the window equals size of the dataset at hand.

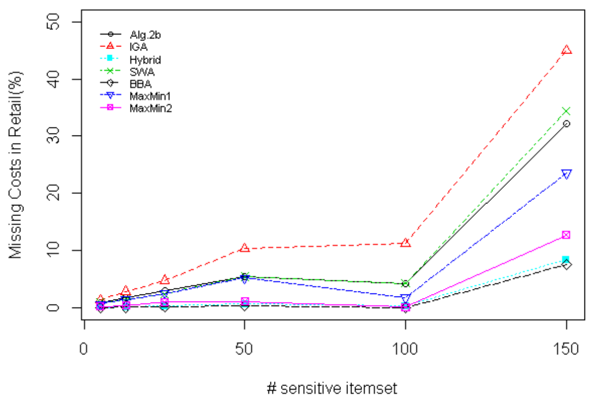

Figure (1) effect of hiding sensitive patterns on Retail dataset using Missing-Costs measurement.

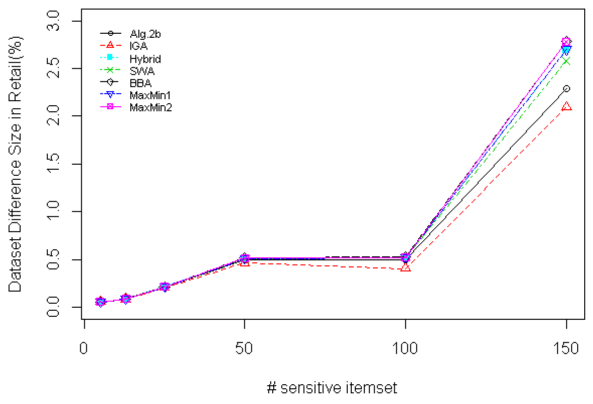

Figure (2) effect of hiding sensitive patters on Retail dataset using Dataset difference size measurement.

Figures (1, 2) show the effect of hiding selected itemsets on Retail datasets. From the figures, we see that the algorithms show approximately the same performance in terms of dataset difference size. However, the missing costs are different. This is because the items chosen for deletion are different with the different algorithms. We also see that BBA, MaxMin2 and Hybrid approaches show the best missing costs performance.

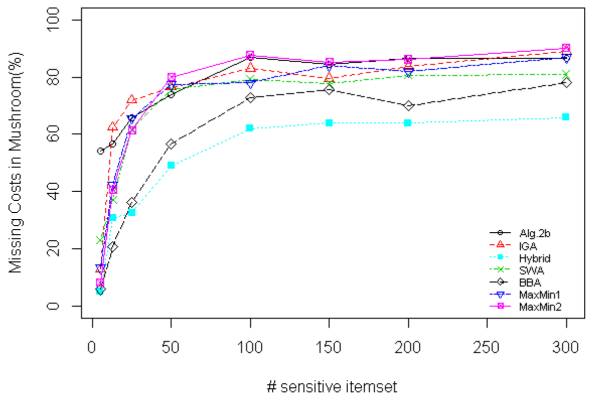

Figure (3) effect of hiding sensitive patterns on Mushroom dataset using Missing-Costs measurement.

Using Mushroom dataset, we see in figure (3) that Hybrid and BBA show the best missing costs performance while MaxMin1 and MaxMin2 performances degenerate with the increasing number of sensitive itemsets.

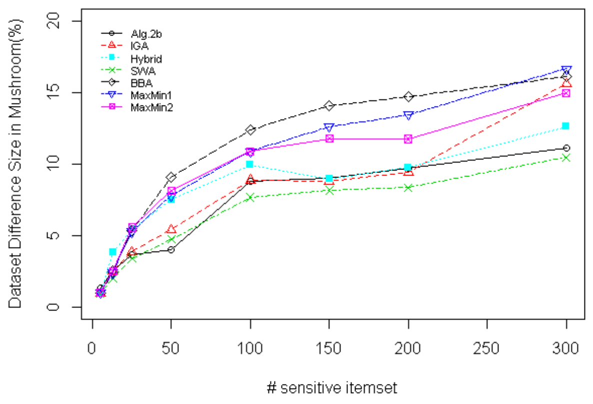

Figure (4) effect of hiding sensitive patterns on Mushroom dataset using DB-difference-size measurement.

In terms of dataset difference size, algorithm alg.2b and SWA shows the best performance in this particular experiment. From the figures, we see that the behavior of the algorithms turns to be more stable in Mushroom and turns to increase in Retail dataset. The reason is the nature of these datasets. Items are more correlated in Mushroom than in Retail; choosing five random different itemsets to be sensitive results in 7 and 208 sensitive itemsets in Retail and Mushroom respectively (see table 2) as an example.

6. Conclusion

Hiding sensitive itemsets depends on the difference between the support of the sensitive itemset and the minimum support; the larger this difference, the more we need to modify the original dataset. Many algorithms are developed to efficiently hide sensitive itemsets with minimal impact on the original datasets. In this paper, we discussed a number of heuristic and border-based approaches to solve association rules hiding problem and applied them on Mushroom and Retail datasets. Experimental results show that BBA, Hybrid and MaxMin2 algorithms have the best performance. Despite its simplicity, alg.2b algorithm shows good performance. Hybrid Algorithm shows good missing costs and database-difference size performances. However, it is slow and requires many computations. This algorithm could be improved dramatically by reducing the search space since many irrelevant transactions and items are examined. We can also reduce the number of sensitive itemsets by hiding only the minimum set of sensitive itemsets according to the Apriori principle as in BBA algorithm.

7.References

[1] C.C. Aggarwal, J.Han, Frequent Pattern Mining, Springer, 2014.

[2] C.Clifton,J.Vaidya. Privacy-Preserving Data Mining: Why, How, and When, IEEE, 2004.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Conference on Management of Data,1993

[4] H.Q.Le, S.Arch-int,N.Arch-int,Association Rule Hiding Based on Intersection Lattice, Hindawi Publishing Corporation,Volume 2013, Article ID 210405,2013.

[5] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure limitation of sensitive rules, 1999.

[6] A.G.Divanis,V.S.Verykios.,Association Rule Hiding for Data Mining, vol 41,Springer,2010.

[7] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding Association Rules by Using Confidence and Support. In Proc. of the 4th Information Hiding Workshop, April 2001.

[8] S. Oliveira, O. Zaiane, Privacy preserving frequent itemset mining, Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining, December 2002.

[9] Amiri. Dare to share: Protecting sensitive knowledge with data sanitization. Decision

[10] S. R. M. Oliveira and O. R. Zaiane. An Efficient One-Scan Sanitization For Improving The Balance Between Privacy And Knowledge Discovery, June 2003.

[11] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery, 1(3):241–258, 1997.

[12] X. Sun and P. S. Yu. A border–based approach for hiding sensitive frequent itemsets. In Proceedings of the 5th IEEE International Conference on Data Mining (ICDM), pages 426– 433, 2005.

[13] http://fimi.ua.ac.be/

[14] G. V. Moustakides and V. S. Verykios. A max–min approach for hiding frequent itemsets. In Workshops Proceedings of the 6th IEEE International Conference on Data Mining (ICDM), pages 502–506, 2006.

[15] G. V. Moustakides and V. S. Verykios. A maxmin approach for hiding frequent itemsets. Data and Knowledge Engineering, 65(1):75–89, 2008.

[16] X. Sun and P. S. Yu. Hiding sensitive frequent itemsets by a border–based approach. Computing science and engineering , 1 (1):74–94, 2007.

[17] http://www.philippe-fournier-viger.com/spmf/http://www.ikvm.net/devguide/net2java.html