A Set-Checking Algorithm for Mining Maximal Frequent Itemsets from Data Streams

Ye-In Chang*, Meng-Hsuan Tsai, Chia-En Li, and Pei-Ying Lin

Dept. of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, R.O.C.

Abstract. Online mining the maximal frequent itemsets over data streams is an important problem in data mining. In order to solve mining maximal frequent itemsets from data streams using the Landmark Window model, Mao *et al*. propose the INSTANT algorithm. The structure of the INSTANT algorithm is simple and it can save much memory space. But it takes long time in mining the maximal frequent itemsets. When the new transaction comes, the number of comparisons between the old transactions of the INSATNT algorithm is too much. Therefore, in this paper, we propose the Set-Checking algorithm to mine frequent itemsets from data streams using the Landmark Window model. We use the structure of the lattice to store our information. The structure of the lattice records the subset relationship between the child node and the parent node. From our simulation results, we show that the process time of our Set-Checking algorithm is faster than that of the INSTANT algorithm.

Keywords: Data Stream, Itemset, Landmark Window Model, Lattice, Maximal Frequent Itemset.

1 Introduction

Traditional database management systems expect all data to be managed within some forms of persistent data sets. For many recent applications, the concept of a data stream, possibly infinite, is more appropriate than a persistent data set. A data stream is an order sequence of transactions that arrives in a timely order. Different from data in traditional static databases, data streams have the following characteristics [1-6]. First, they are continuous, unbounded, and usually come with high speed. Second, the volume of data streams is large and usually with an open end. Third, the data distribution in streams usually changes with time. A maximal frequent itemset is the itemset which is not the subset of other frequent itemset and the support is large or equal to the mining support. Therefore, the result of maximal frequent itemsets is more compact than the result of frequent itemsets. The INSTANT algorithm [5] uses a compact data structure to mine maximal frequent itemsets from data streams based on the Landmark Window. In the Landmark Window model, knowledge discovery is performed based on the values between the beginning time and the present. The

* Corresponding author: Ye-In Chang (E-mail: changyi@mail.cse.nsysu.edu.tw).

advantage of using the Landmark Window model is that the results are correct as compared to the other models. The structure of the INSTANT algorithm is simple so that it can save the memory space. However, in the part of the itemset comparison, the INSTANT algorithm will take long time. Therefore, in the paper, we propose the Set-Checking algorithm. In our structure, we add the link between the set and its subsets to show the relation between itemsets. When the new transaction comes, we will check the relation between the new transaction and the old transaction. Because of this links between the set and the subset, there are two advantages. First, we can decrease the number of comparisons between the new transaction and the old transactions. Second, when the support increases, the lattice structure does not change. From our simulation results, we show that the process time of our Set-Checking algorithm is faster than that of the INSTANT algorithm. The rest of the paper is organized as follows. Section 2 gives a survey for mining association rules related problem. Section 3 presents the proposed Set-Checking algorithm. In Section 4, we study the performance. Finally, Section 5 gives the conclusion.

2 The Related Work

In this section, we describe some well-known data mining algorithms for association rules related problems in the traditional database and data streams [1-6].

Maximal frequent itemsets mining is one of the most important research issues in data mining. Take Transaction Database *T* in the Fig. 1-(a) as an example, Fig. 1-(b) shows the frequent itemsets of Transaction Database *T*, where the value of the minimal support is 2. In Fig. 1-(c), large itemset "*CD*" and "*ABC*" is a maximal frequent itemset, because there is no superset which contains it. Itemset "*AB*" is not a maximal frequent itemset, because the frequent itemset "*ABC*" contains it.

TID	Items
1	ABC
2	CD
3	ABC
4	ACD

(a)

L1	A：3 B：2 C：3 D：2
L2	AB：2 AC：3 BC：2 CD：2
L3	ABC：2

(b)

L2	CD：2
L3	ABC：2

(c)

Fig. 1. An example: (a) Transaction Database *T*; (b) Frequent itemsets of Transaction Database *T* with the minimal support = 2; (c) Maximal frequent itemsets of Transaction Database *T* with the minimal support = 2.

The INSTANT algorithm [5] is a single-phase algorithm for mining maximal frequent itemsets from data streams. Given a continuous data stream, a user-specified minimum support count δ, and a shedding condition, the INSTANT algorithm consists of four steps: Step 1: generate a sorted itemset α from the current transaction of the data stream. Step 2: form new frequent itemsets and update the set of frequent itemsets. Step 3: modify information about infrequent elements and their supports based on α. Step 4: execute a shedding plan to maintain search efficiency and memory usage.

3 The Set-Checking Algorithm

One of the well-known models for data streams is the Landmark Window model. In this section, we present our algorithm based on the Landmark Window model.

3.1 The Proposed Algorithm

We first define some basic definitions and notations and then describe how to use them to find the maximal frequent itemsets and organize the maximal frequent itemsets in a lattice. According to the characteristic of the maximal frequent itemsets, we propose the Set-Checking algorithm. We can check the relation between the new transaction and old transactions, and then build the lattice. The variables used are defined as follows. *minsup* is the minimal support. *Sup* is a transaction's support. *f_list* is the maximal frequent itemsets.

Our algorithm has three steps. Step 1: transform the itemset to the bit-pattern. Step 2: check the relation between new transaction and old transaction. There are five cases in the relations. Case 1: equivalent. This case only updates the support of the old transaction. Case 2: superset. This case will update the support of the old transaction and the old transaction becomes the child of the new transaction. The new transaction will be inserted into the lattice. Case 3: subset. The new transaction will become the child of old transaction. Case 4: intersection. There are two situations. Case 4-1: the node is the intersection of the old transaction and the new transaction is not in the lattice. This case will insert the new transaction and intersected node. Case 4-2: the node which is the intersection of the old transaction and the new transaction is in the lattice. This case will insert the new transaction and update the support of intersected node. Case 5: empty set. The new transaction will be inserted into the lattice. Step 3: examine which transaction is the maximal frequent itemsets. If the frequent node is not the subset or equal to the *f_list*, we can add the frequent node to *f_list* and delete the node which is the subset of the frequent node from *f_list*.

3.2 Data Structure

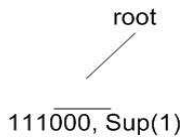
In our algorithm, we propose a lattice structure. The lattice structure has two advantages. First, we can know the relation between the new transaction and the present transactions easily. Second, we can update the support of the transaction efficiency. We also build *f_list* to store the maximal frequent itemsets. The lattice structure contains the root, nodes, and child-link. The root is the start point. It has no information. When a new transaction is coming, we start to search them from the root. The nodes are infrequent itemsets. Each node has some information: (1) *Bit-Pattern* represents itemset. (2) *Sup* represents the support of the itemset. The child-link is the node link to subset nodes. Building the child-link has many advantages. First, we can check the node's relation and insert the node into the tree easily. Second, we can increase the node's support easily. Our lattice structure has two characteristics. First, the high level nodes are the subset of low level nodes. Second, the supports of high level nodes are larger than that of the low level nodes.

3.3 Data Insertion

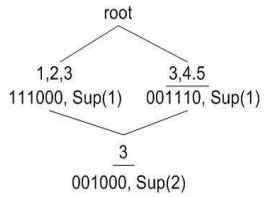
We use an example to illustrate our algorithm. Fig. 2-(a) shows an example of the data stream. The maximal length of bit-length is 5 and the *minsup* is 6. Transaction Tid_1 {1, 2, 3} is the first transaction of the data stream. The root does not have a child, so the itemset is inserted into the root's child directly as shown in Fig. 2-(b). When transaction Tid_2 {3, 4, 5} comes, we will check whether the itemset is in the lattice or not. Obviously, transaction Tid_2 is not in the lattice. Then, we will check the relation between transaction Tid_1 and transaction Tid_2. We find that Tid_2 intersects with Tid_1 and intersected node {3} is in the lattice. So node {3} becomes the child of node {1, 2, 3} and node {3, 4, 5}. The support of node {1, 2, 3} which is in the lattice is increased. When we process the last child, we insert Tid_2 {3, 4, 5} into the lattice as shown in Fig. 2-(c). When transaction Tid_3 {3, 6} comes, we find that the intersected node {3} is in the lattice. Node {3} becomes the child of node {3, 6} and the support of node {3} is increased. Consider when the new data stream comes, the support of each node in the lattice is increased at once. Because transaction Tid_3 {3, 6} is not in the lattice. When we process the last child, we insert transaction Tid_3 {3, 6} into the lattice as shown in Fig. 3-(a). Finally, the lattice will be built as shown in Fig. 3-(b).

Tid	Itemset	Bit-Patten
1	1, 2, 3	111000
2	3, 4, 5	001110
3	3, 6	001001
4	1, 2, 3, 4, 5	111110
5	2	010000
6	1, 2, 3, 4, 5	111110

(a)

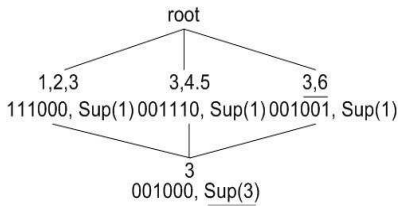


(b)

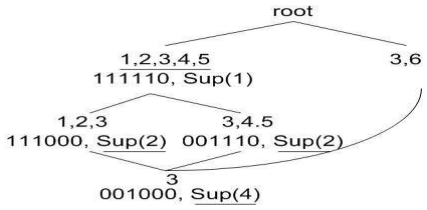


(c)

Fig. 2. An example: (a) an example of the data stream; (b) insertion of transaction Tid_1 into the lattice; (c) insertion of transaction Tid_2 into the lattice.



(a)



(b)

Fig. 3. Data insertion: (a) insertion of transaction Tid_3 into the lattice; (b) the final lattice.

3.4 The Maximal Frequent Itemsets

When the support of the transaction is larger than or equal to the minimal support, we can add this transaction to *f_list*. Fig. 2-(a) shows an example of the data stream. Let the minimal support = 3. When transaction Tid_1 {1, 2, 3} and Tid_2 {3, 4, 5} comes, the lattice will be built as shown in Fig. 2-(c). When transaction Tid_3 {3, 6} comes,

the support of item {3} becomes 3. It equals the minimal support. So we can add item {3} to *f_list* as *f_list* -> 3(001000).

4 Performance

In this section, we describe how to generate the synthetic data which will be used in the simulation. Then, we study the performance of the INSTANT algorithm and our proposal Set-Checking algorithm.

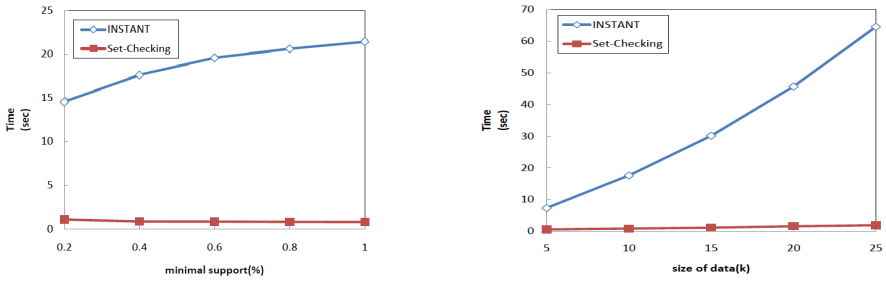
4.1 The Simulation Model

We describe the way to generate synthetic transaction data from the IBM synthetic data developed by Agrawal *et al.* [1]. The parameters used in the generation of the synthetic data are follows. |*T*| is the average size of transactions. |*MT*| is the maximum size of transactions. |*I*| is the average size of maximal potentially frequent itemsets. |*D*| is the number of transactions and |*MI*| is the maximum size of potentially frequent itemsets. |*SD*| is the size of added data and *corr* is the number of the correlation level.

First, the length of a transaction is determined by the Poisson Distribution with a mean which is μ equal to |*T*|. The size of a transaction is between 1 and |*MT*|. The transaction is repeatedly assigned items from a set of potentially maximal frequent itemsets *F*, while the length of the transaction does not exceed the generated length. Then, the length of an itemset in *F* is determined according to the Poisson Distribution with a mean μ which is equal to |*I*|. The size of each potentially frequent itemset is between 1 and |*MI*|. To model the phenomenon that frequent itemsets often have common items, some fraction of items in subsequent itemsets are chosen from the previous itemset generated. We use an exponentially distributed random variable with a mean which is equal to the *correlation level*, to decide this fraction for each itemset. The *correlation level* was set to 0.5. The remaining items are chosen randomly. Each itemset in *F* has an associated weight that determines the probability that this itemset will be chosen. The weight is chosen from an exponential distribution with a mean equal to 1.

4.2 Experiment Results

In this section, we show the experiment results. Fig. 4-(a) shows the comparison of the processing time under different minimum supports. The synthetic data is T3.MT10.I10.MI20.D10k. The` number of item is 1000 and the minimum support threshold is changed from 0.2% to 1%. We can build 9k data then add 1k data to the data which we build. From the figure, we can find that the Set-Checking algorithm is faster than the INSTANT algorithm. Because the number of the itemset comparisons of the Set-Checking algorithm is less than that of the INSTANT algorithm. Fig. 4-(b) shows the comparison of the processing time under different size of data. The minimum support threshold is 0.4% and the number of items is 1000.



(a) (b)

Fig. 4. A comparison: (a) a comparison of the processing time under different minimum supports; (b) a comparison of the processing time under different size of the data.

5 Conclusion

In this paper, we have proposed the Set-Checking algorithm that could efficiently mining maximal frequent itemsets in data streams. When updating the transaction data streams occur, how to maintain these rules efficiently is the key point of stream mining. The simulation results have shown that the proposed Set-Checking algorithm outperforms the INSTANT algorithm.

Acknowledgments. The research was supported in part by the National Science Council of Republic of China under Grant No. NSC-101-2221-E-110-091-MY2.

References

1. Agrawal, R., Srikant, R.: Fast Algorithm for Mining Association Rules in Large Databases. In: 20th International Conference on Very Large Data Bases, pp. 487--499. Morgan Kaufmann Publishers, San Francisco (1994)
2. Li, H., Zhang, N.: Mining Maximal Frequent Itemsets Over a Stream Sliding Window. In: IEEE Youth Conference on Information Computing and Telecommunications, pp. 110--113. IEEE Press, New York (2010)
3. Li, J.W., Lee, G.Q.: Mining Frequent Itemsets over Data Streams Using Efficient Window Sliding Techniques. The International Journal of Expert Systems with Applications, vol. 36, no. 2, pp. 1466--1477. Pergamon Press, New York (2009)
4. Lin, K.C., Liao, I.E., Chen, Z.S.: An Improved Frequent Pattern Growth Method for Mining Association Rules. The International Journal of Expert Systems with Applications, vol. 38, no. 5, pp. 5154--5161. Pergamon Press, New York (2011)
5. Mao, G., Wu, X., Zhu, X., Chen, G., Liu, C.: Mining Maximal Frequent Itemsets from Data Streams. Journal of Information Science, vol. 33, no. 3, pp. 251--262. Sage Publication, CA (2007)
6. Xin, J.W., Yang, G.Q., Sun, J.Z., Zhang, Y.P.: A New Algorithm for Discovery Maximal Frequent Itemsets Based on Binary Vector Sets. In: 5th International Conference on Machine Learning and Cybernetics, pp. 1120--1124. IEEE Press, New York (2006)