Mining Recent Frequent Itemsets in Data Streams by Radioactively Attenuating Strategy*

Lifeng Jia, Zhe Wang, Chunguang Zhou, and Xiujuan Xu

College of Computer Science, Jilin University,
Key Laboratory of Symbol Computation and Knowledge Engineering
of the Ministry of Education, Changchun 130012, China
jia_lifeng@hotmail.com cgzhou@jlu.edu.cn

Abstract. We propose a novel approach for mining recent frequent itemsets. The approach has three key contributions. First, it is a single-scan algorithm which utilizes the special property of suffix-trees to guarantee that all frequent itemsets are mined. During the phase of itemset growth it is unnecessary to traverse the suffix-trees which are the data structure for storing the summary information of data. Second, our algorithm adopts a novel method for itemset growth which includes two special kinds of itemset growth operations to avoid generating any candidate itemset. Third, we devise a new regressive strategy from the attenuating phenomenon of radioelement in nature, and apply it into the algorithm to distinguish the influence of latest transactions from that of obsolete transactions. We conduct detailed experiments to evaluate the algorithm. It confirms that the new method has an excellent scalability and the performance illustrates better quality and efficiency.

1 Introduction

Mining frequent itemsets is an essential data mining operation in many data mining problems such as mining association rules, sequential patterns, closed patterns, and maximal pattern. It has been widely studied and applied since the last decade. The problem of mining frequent itemsets in large databases was first proposed by Agrawal *et al*. [1] in 1993. When it comes to the environment of data streams, mining frequent itemsets becomes a challenging problem, because the information in the streaming data is huge and rapidly changing. Consequently, infrequent items and itemsets can become frequent later on and hence cannot be ignored.

A data stream is a continuous, huge, fast changing, infinite sequence of data elements. The nature of streaming data shapes the algorithm which only requires scanning the whole dataset when it is devised to support aggregation queries. In addition, this kind of algorithms usually owns a data structure far smaller than the size of the whole dataset. The first algorithm to on mining frequent itemsets in data streams using

* This work was supported by the Natural Science Foundation of China (Grant No. 60433020) and the Key Science-Technology Project of the National Education Ministry of China (Grant No. 02090).

X. Li, S. Wang, and Z.Y. Dong (Eds.): ADMA 2005, LNAI 3584, pp. 804 – 811, 2005.

the estimation mechanism is the Lossy Counting proposed by Manku and Motwani [2]. It is a single-pass algorithm based on the well-known Apriori property: if any length k pattern is not frequent in the database, its length (k+1) super-patterns can never be frequent. Han et al. [3] developed a FP-tree-based algorithm, called FP-stream, to mine frequent itemsets at multiple time granularities by a novel titled-time windows technique. Ruoming and Agrawal [4] developed a single-scan algorithm to mine frequent itemsets by viewing the streaming data as an indivisible model, instead of breaking it into batches of transactions. All of the above algorithms capture the situation in which the entire data steam is divided into many batches of transactions. All above algorithms only focus on the frequent itemsets over the entire data stream, and overlook the importance and practicability of mining recent frequent itemsets. To find recent frequent itemsets in the data stream, the algorithm not only needs to scan the dataset once, but also must differentiate the information of recently generated transactions from the useless or invalid information of obsolete transactions. Teng *et al*. [5] proposed a regression-based algorithm, called FTP-DS, to mine recent frequent itemsets in the sliding window. Chang *et al*. [6] developed estDec for recent frequent itemsets in the data stream where each transaction has a weight and it decreases with age. Moreover, Chang *et al*. [7] also proposed a single-scan algorithm for recent frequent itemsets based on the estimation mechanism of Lossy Counting algorithm.

The rest of the paper is organized as follows. In Section 2, we present our algorithm by and large, and then we will introduce the suffix-forest for storing the summary information of streaming data, the lattice for storing frequent itemset in the data stream, and the counter sequence and depth first itemset growth method. The introduction and analysis of radioactively attenuating strategy is established in detail in Section 3. In Section 4, we turn to the experiment evaluation. Eventually, we draw the final conclusion of our algorithm in Section 5.

2 MRFISF Algorithm

MRFISF is newly-devised algorithm based on the estimation mechanism [2] and is also batch-processed. To make it understood, we illustrate each main operation of MRFISF algorithm with a simple example. Note that, it is assumed that all the items in data streams were ordered beforehand.

2.1 MRFISF Algorithm

Input: A data stream D, minimal support threshold θ**,** and maximal estimated support error threshold ε**,** decaying factor ϕ .

Output: recent frequent itemsets in lattice.
For each batch of transactions in the data stream D**:**

1. Construct the suffix-forest to store the summary information**.**
2. Generate frequent itemsets from the suffix-forest by counter sequence and depth first itemset growth method**.**
3. According to the estimation mechanism, insert new frequent itemsets into lattice or update the frequencies of old frequent itemsets already in lattice.
4. Pruning the infrequent itemsets by now from the lattice**.**

5. According to the attenuating points, reduce frequencies of itemsets which need attenuating by decaying factor ϕ and Pruning infrequent itemsets due to the attenuation from the lattice..

2.2 Suffix-Forest and Lattice

To mine frequent itemsets in data streams, MRFISF chooses the suffix-forest to store the summary information of data streams. Now, let us turn to describing the steps of constructing a suffix-forest.

For the problem of mining frequent itemsets, the data element in data streams is the transaction composed of items. First, when a new transaction $t=\{I_1,I_2,\ldots,I_n\}$arrives, it is projected into its suffix-sets:$\{\{I_1,I_2,\ldots I_n\},\{I_2,I_3,\ldots I_n\},\ldots,\{I_{n-1},I_n\},\{I_n\}\}$. Second, to store them, suffix-trees are established according to these suffix-sets which are viewed as many different individuals. Third, all of the identical nodes of each suffix-tree are connected by a node link team (NLT for short). We use "suffix-tree(I_i)"and "node(I_k)" to denote the suffix-tree whose root is item I_i and a node I_k of suffix-tree respectively.

Lemma 1: In the suffix-tree(I_i), its sub-tree whose root is I_k (k>i) must be a sub-tree of suffix-tree(I_k).

Proof: During the construction of suffix-tree(I_i), when constructing a certain branch of sub-tree whose root is item I_k, MRFISF algorithm must deal with a certain suffix-set $\{I_i,\ldots,I_k,\ldots,I_n\}$.Meanwhile, the suffix-set$\{I_k,\ldots,I_n\}$must appear together with this suffix-set$\{I_i,\ldots,I_k,\ldots,I_n\}$. Thus, MRFISF either constructs a new branch of suffix-tree(I_k) to store the suffix-set $\{I_k,\ldots,I_n\}$or inserts it into an already existed branch in suffix-tree(I_k) by updating the frequency of nodes in that branch. Consequently, the lemma 1 is proofed

Based on lemma 1, we can draw a conclusion: if a certain node(I_m) of suffix-tree(I_i) has a corresponding node(I_m) in the suffix-tree(I_k), such a node(I_m) must have an ancestor node(I_k) in the suffix-tree(I_i). In order to recognize the corresponding identical nodes in different suffix-trees, we need to code nodes of suffix-tree. Suppose that the nodes of complete suffix-tree are coded by the depth first. MRFISF algorithm endows nodes of the actual (complete or incomplete) suffix-tree the same serial numbers as those they own in the corresponding complete suffix-tree. To make this coding method clear, let us illustrate it with a simple example, a batch of transactions, {a,c,d}, {a,b,d},{b,d},{b,c,d}. The serial numbers of nodes of suffix-tree(a) of above example are endowed on the basis that the complete itemset includes 5 items {a,b,c,d,e}.

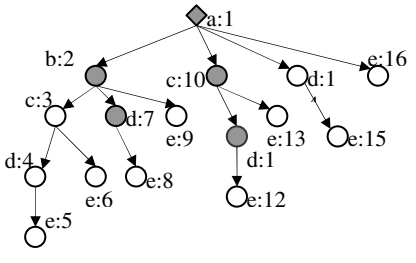

Fig. 1. the complete suffix-tree(a)

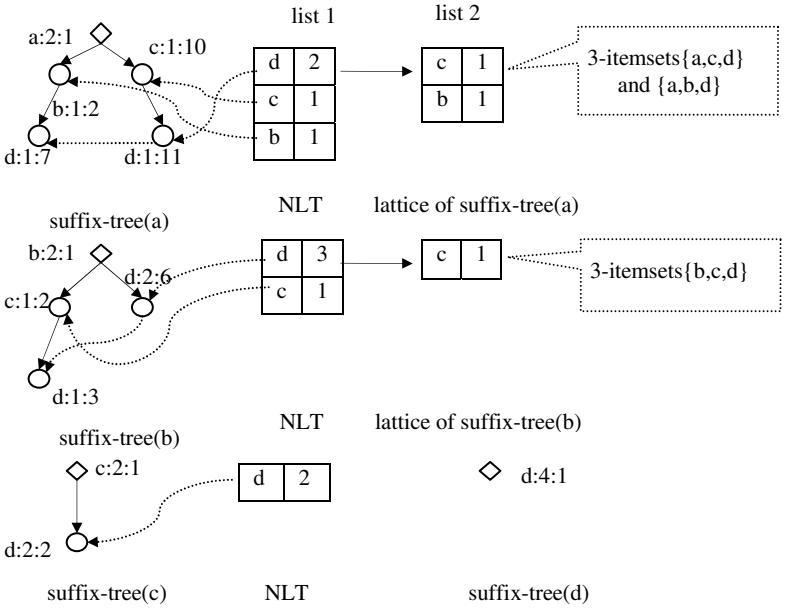

Fig. 2. The suffix-forest and the lattice of frequent itemsets

Fig 1 shows the serial number of each node in the complete suffix-tree(a) by means of depth first. Fig 2 shows the suffix-forest and lattice generated by MRFISF algorithm with above simple example. Note that the instance is so simple that MRFISF do not execute the radioactively attenuating strategy. Every node of the suffix-tree shown in Fig 2 contains three aspects of information: nodename, frequency, and serial number. In addition, the shaded nodes of the complete suffix-tree(a) in Fig 1 are the corresponding nodes of actual suffix-tree(a) in Fig 2. Therefore, same nodes in complete and actual suffix-tree(a) have the identical serial numbers.

2.3 Itemset Growth

According to the method of coding the nodes of suffix-forest, there is a special relationship between the corresponding identical nodes in different suffix-trees: In the suffix-tree(I_i), the difference of the serial number of node(I_n) and that of its ancestor node(I_m) is equal to the serial number of its corresponding node(I_n) in the suffix-tree(I_m)**.** Consequently, we can use this relationship to identify whether a node in a suffix-tree has a corresponding identical node in another suffix-tree. To explain this relationship clearly, let us illustrate it with the suffix-tree(a) in Fig 2. For the node(d) whose serial number is 7, and its ancestor node(b) whose serial number is 2, the difference of serial numbers of the two nodes is (7-2)+1=6. The serial number of its corresponding node(d) in the suffix-tree(b) is 6, which is equal to the difference value. All above phenomena are based on the special relationship.

Lemma 2: In an assumed suffix-tree(I_1) with i kinds of child nodes$\{I_2,\ldots,I_{i+1}\}$, let $f(I_{i+1})$ and $c(I_{i+1})$ denote the frequency and the serial number of node(I_{i+1}) respectively. For each node(I_{i+1}) of suffix-tree(I_1), if it has a corresponding node(I_{i+1}) in suffix-tree(I_k), the serial number of this node(I_{i+1}) is stored in the serial number set C_k ($1<k<i+1$). Let the total serial number set (TSN-SET for short) $C = C_2 \hbar \ldots \hbar C_i$, if $C \neq \varnothing$, the frequency of (i+1)-itemset$\{I_1,\ldots,I_{i+1}\}=\sum f(I_{i+1})$, such that $c(I_{i+1}) \in C$; Otherwise, the frequency of (i+1)-itemset$\{I_1,\ldots,I_{i+1}\}=0$.

Proof: In order to compute the frequency of (i+1)-itemset$\{I_1,\ldots, I_{i+1}\}$, we need to know all nodes(I_{i+1}) which are the common child nodes of node(I_1),…, node(I_i) in the suffix-tree(I_1). According to the conclusion of lemma 1, if a certain node(I_m) of suffix-tree(I_i) has a corresponding node(I_m) in the suffix-tree(I_k), such a node(I_m) must have an ancestor node(I_k) in the suffix-tree(I_i), and the statement of lemma 2, the set C_k stores serial numbers of nodes(I_{i+1}) which have nodes(I_k) as ancestor nodes and the TSN-SET $C = C_2 \hbar \ldots \hbar C_i$. So, if there is the occurrence of nodes(I_{i+1}) which are the common child nodes of node(I_1),…, node(I_i), the serial numbers of these nodes(I_{i+1}) must appear in the TSN-SET C($C \neq \varnothing$). Subsequently, the frequency of (i+1)-itemset$\{I_1,\ldots, I_{i+1}\}$ is equal to the sum of frequencies of these nodes(I_{i+1}), $\sum f(I_{i+1})$, such that $c(I_{i+1}) \in C$. Otherwise, if the TSN-SET C is empty($C = \varnothing$), the frequency of (i+1)-itemset$\{I_1,\ldots, I_{i+1}\}$must be equal to zero.

Based on the lemma 2, we turn to introducing the new itmeset growth method: counter sequence and depth first itemset growth. However, before detailing it, we first define two core operations as follow:

Definition 1 (Insert Itemset Growth(IIG)): When MRFISF has already computed the frequency of i-itemset$\{I_1,\ldots,I_m,\ldots,I_k\}$(k>m>i), through the operation of insert itemset growth, MRFISF will compute the frequency of (i+1)-itemset$\{I_1,..,I_p,I_m,..,I_k\}$, such that I_p is newly inserted item(i<p<m<k). For example, the IIG operation makes 3-itemset{c,e,f} grow to 4-itemset{c,d,e,f}

Definition 2 (Replace Itemset Growth(RIG)): When MRFISF has already computed the frequency of i-itemset$\{I_1,\ldots,I_m,\ldots,I_k\}$(k>m>i), through the operation of replace itemset growth, MRFISF will compute the frequency of i-itemset$\{I_1,\ldots,I_p,\ldots,I_k\}$, such that I_p is newly inserted item to replace the item I_m (i<p<m<k). For example, the RIG operation makes 3-itemset {c,e,f}grow to 3-itemset {c,d,f}.

We stipulate that the priority of IIG operation is higher than that of RIG operation. We also stipulate that the sequence of newly-inserted item by both IIG and RIG operations is according to the counter item sequence. Based on the discussion so far, the counter sequence of inserting items of IIG and RIG operations not only guarantees that all frequent itemsets will be generated by MRFISF algorithm, but also makes sure that all redundant computation will be eliminated during the phase of itemset growth. Now, let us turn to the simple example in Fig 2 again, the c unit in the list 2 of lattice of suffix-tree(a) denotes 3-itemset {a,c,d}, because c unit grows from the d unit which denotes itemset {a,d} by IIG operation. The b unit in the list 2 of lattice of suffix-tree(a) denotes 3-itemset{a,b,d}, because b unit grows from the c unit which denotes itemset {a,c,d} by RIG operation.

3 Radioactively Attenuating Strategy

In nature, the attenuating rule of radioelement is $N = N_0 \cdot 2^{-t/T}$ .In this formula, N denotes the number of existing radioelement atoms without attenuating, N_0 denotes the number of radioelement atoms before the attenuation, t denotes the attenuating time, and T denotes the half life of the radioelement which is the time the quantitative radioelement atoms need to attenuate to the half number of them.

3.1 Radioactively Attenuating Strategy

Before clearing the details of new strategy, let us introduce two definitions first.

Definition 3 (Lifecycle of a Frequent Itemset): Lifecycle of a frequent itemset is the phase of time which begins when it appears as a frequent one and ends when it is turning to be infrequent.

Definition 4 (The Attenuating Point): The attenuating point of a frequent itemset is the time when $t > 2^p \cdot t_0$ and $t \geq 2^{P+1} \cdot t_0$, such that t is the lifecycle of frequent itemset, p is an integer, and t_0 is a unit of time.

Whenever a certain attenuating point of itemset arrives, the frequency of itemset is reduced by the decaying factor ϕ (ϕ <1). The new regressive strategy guarantees that the older a transaction is, the less its influence is put to the recent frequent itemsets it contains. The new strategy also secures that the attenuating rate of influence of transactions is slowing down with the lapse of time, which is generally consistent with the attenuating rate of radioelement reasoned from the formula above. Consequently, the regressive method is called radioactively attenuating strategy.

3.2 The Analysis of Accuracy

Now, we analyze the accuracy of radioactively attenuating strategy by comparing it with other two regressive strategies adopted by FTP-DS algorithm and escDet algorithm respectively.

The FTP-DS algorithm only focuses on the transactions in the sliding window, and overlooks the influence of transactions outside the sliding window to the recent frequent itemsets. If the size of sliding window is too small, the result will fail to reflect the recent change of a data stream. In other words, the result will be inaccurate. If the size of sliding window is big enough, FTP-DS fails to differentiate the different influence of transactions in the big sliding window. However, the MRFISF algorithm with radioactively attenuating strategy takes all transactions in data streams into account, and gradually reduces the influence of old transactions to the recent itemsets. Obviously, the ability of monitoring the continuous variation of a data stream of radioactively attenuating strategy is better than that of FTP-DS algorithm.

The estDec algorithm mines the recent frequent itemsets in the data stream in which each transaction has a weight and decreases with age. However, there is a situation that some useful transactions might be faded away so fast that lessen the accuracy of result, because the influence of an old transaction is decayed whenever a new transaction arrives. Nevertheless, such an improper situation will not appear in

the radioactively attenuating strategy, because the regressive rate of transaction is slowing down as the lapse of time, which guarantees that whenever a transaction is faded away, it already is useless or invalid for the recent frequent itemsets.

4 Experimental Evaluation

We evaluate our algorithm MRFISF by using a synthetic database generated by the IBM Quest Synthetic Data Generator. We use MFISF and MRFISF to denote the novel algorithms for mining frequent itemsets in data streams and for mining recent frequent itemsets by adopting the radioactively attenuating strategy respectively. The minimal support threshold θ is 0.1% and the maximal estimated support error threshold ε is a tenth of θ.

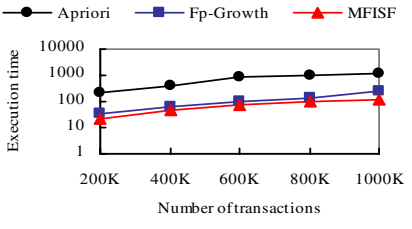

Fig. 3. The performance of algorithms (a)

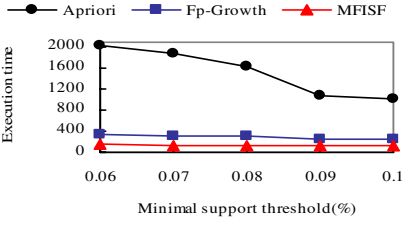

Fig. 4. The performance of algorithms (b)

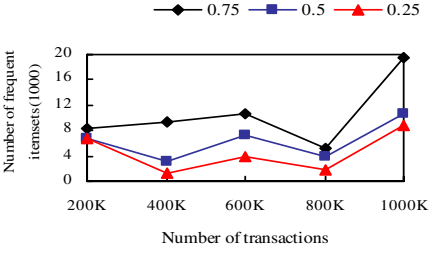

Fig. 5. The number of frequent item sets of different sizes of datasets

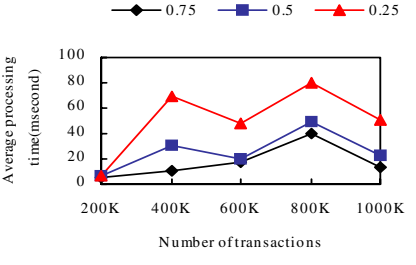

Fig. 6. Average provessing time

First, we compare the performance of Apriori and FP-Growth algorithms which can be downloaded in the web: *http://www.cs.umb.edu/~laur/ARtool/* with that of MFISF algorithm in the environment of T5.I4.D1000K. Fig 3 and Fig 4 show execution times of three different algorithms with the increasing dataset size and with the increasing minimal support thresholds. Since the test dataset is very dense, i.e. the frequency of each item is very high, the execution time of MFISF algorithm do not vary a great extent when θ varies. However, MFISF still outperforms both Apriori and Fp-Growth algorithms greatly. Second, Fig 5 and Fig6 provide the information

concerning the number of frequent itemsets mined by the MRFISF algorithm with different decaying factors: 0.75, 0.5, 0.25 and the average processing time with the increasing size of dataset. The evaluation manifests the outstanding ability of MRFISF algorithm to mine recent frequent itemsets.

5 Conclusion

We proposed a new algorithm MRFISF to find out all recent frequent or frequent itemsets over the whole data streams. MRFISF not only takes full advantage of the special property between suffix-trees to avoid traversing every suffix-tree and generating candidate frequent itemsets, but also utilizes a newly-proposed itemset growth method to optimize the algorithm. Moreover, we also bring forward a novel regressive strategy for differentiating the information of recently generated transactions from that of obsolete transactions. Experiment results show that MRFISF algorithm has an excellent ability to mine recent frequent itemsets in data streams.

References

1. R. Agrawal, T. Imielinski, and A. Swami.: Mining Association Rules between Sets of Items in Large Databases. In ACM SIGMOD Conf. Management of Data (1993) 207-216
2. G. S. Manku and R. Motwani.: Approximate Frequency Counts Over Data Streams. In Proceeding of the International Conference on Very Large Data Bases, Hong Kong, China (2002) 346-357
3. C.Giannella, J. Han, and J. Pei, X. Yan and P. S. Yu.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. Next Generation Data Mining, Chapter 3 (2002) 191-211.
4. Ruoming Jin, Gafan Agrawal. An Algorithm for In-Core Frequent Itemset Mining on Streaming Data, [online] Available: http://www.cse.ohio-state.edu/~agrawal/ (2004)
5. W. G. Teng, M. S. Chen, and P. S. Yu.: A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In Proceeding of the 29th VLDB Conference (2003) 93-104
6. J. Chang and W. Lee.: Finding Recent Frequent Itemsets Adaptively over Online Data Streams. In Proceeding of the ACM International Conference on Knowledge Discovery and Data Mining, Washington, DC (2003) 487-492
7. J. Chang and W. Lee.: A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams. Journal of Information Science and Engineering, (2004) 753-762