An incremental mining algorithm using pre-large sequences

[1]Radhika Kailas Jagzap, [2] Nilesh G Pardeshi

[1]ME Computers, [2] Asst. Professor,

[1]Computer Engineering,
[1]Sanjivani College of Engineering, Kopargaon, India

***Abstract* - Mining useful information and helpful knowledge from large databases has evolved into an important research area in recent years. Among the classes of knowledge derived, finding sequential patterns in temporal transaction databases is very important since it can help model customer behavior. In the past, researchers usually assumed databases were static to simplify data-mining problems. In real-world applications, new transactions may be added into databases frequently. Designing an efficient and effective mining algorithm that can maintain sequential patterns as a database grows is thus important. In this paper, we propose a novel incremental mining algorithm for maintaining sequential patterns based on the concept of pre-large sequences to reduce the need for rescanning original databases.**

***Index Terms* - Data mining, Sequential pattern, Large sequence, Pre-large sequence, Incremental mining.**

I. INTRODUCTION

What is sequential pattern mining? Sequential pattern mining is nothing but mining of frequently occurring ordered events or subsequences as patterns. For example if a customer purchases Digital camera he may purchase the color printer. The areas where sequential mining can be used are customer shopping sequences, web access patterns analysis, weather prediction, production, biological sequences and network intrusion detection [2].

The rapid development of computer technology, especially increased capacities and decreased costs of storage media has led businesses to store huge amounts of external and internal information in large databases at low cost. Mining useful information and helpful knowledge from these large databases has thus evolved into an important research area [3]. Years of effort in data mining has produced a variety of efficient techniques. Among them, finding sequential patterns in temporal transaction databases is important since it allows modeling of customer behavior [2] [1]. Mining sequential patterns was first proposed by [2], and is a non-trivial task. It attempts to find customer purchase sequences and to predict whether there is a high probability that when customers buy some products, they will buy some other products in later transactions. For example, a sequential pattern for a video shop may be formed when a customer buys a television in one transaction, he then buys a video recorder in a later transaction. Note that the transaction sequences need not be consecutive. Although customer behavior models can be efficiently extracted by the mining algorithm in [2] to assist managers in making correct and effective decisions, the sequential patterns discovered may become invalid when new customer sequences occur.

II. LITERATURE SURVEY

Recently, some researchers have strived to develop incremental mining algorithms for maintaining association rules such as.

FUP algorithm proposed by [3] in this paper Weighted Frequent Pattern Mining (WFPM) has brought the notion of the weight of the items into the Frequent Pattern mining algorithms. WFPM is practically much efficient than the frequent pattern mining. Several Weighted Frequent Pattern Mining methods have been used. However, they do not deal with the interactive and incremental database. An IWFPTWU algorithm has been proposed to allow the users to decide the level of interest and provides the direction for mining the interesting patterns. The Incremental Weighted Frequent Patterns based on Transaction Weighted Utility (IWFPTWU) considers both the weight and the frequency of the item. The IWFPTWU arranges the items in a decreasing order of the Transaction Weighted Utilization (weighted Support). This makes uses of a single scan of the database for the construction of the IWFPTWU tree.

The incremental mining algorithm with pre-large itemsets proposed [4] [5]. In this paper, an existing incremental algorithm, Probability-based incremental association rule discovery is used. The previous algorithm, probability-based incremental association rule discovery algorithm uses principle of Bernoulli trials to find frequent and expected frequent k-itemsets. The set of frequent and expected frequent k-itemsets are determined from a candidate k-itemsets. Generating and testing the set of candidate is a time-consuming step in the algorithm. To reduce the number of candidates 2-itemset that need to repeatedly scan the database and check a large set of candidate, our paper is utilizing a hash technique for the generation of the candidate 2-itemset, especially for the frequent and expected frequent 2-itemsets, to improve the performance of probability-based algorithm. Thus, the algorithm can reduce not only a number of times to scan an original database but also the number of candidate itemsets to generate frequent and expected frequent 2 itemsets. As a result, the algorithm has execution time faster

than the previous methods. This paper also conducts simulation experiments to show the performance of the proposed algorithm. The simulation results show that the proposed algorithm has a good performance.

The common idea in these approaches is that previously mined information should be utilized as much as possible to reduce maintenance costs. Intermediate results, such as large itemsets, are kept and checked against newly added transactions, thus saving much computation time for maintenance, although original databases may still need to be rescanned. Studies on maintaining sequential patterns are relatively rare compared to those on maintaining association rules. Lin and Lee proposed the FASTUP algorithm to maintain sequential patterns by extending the FUP algorithm [1]. Their approach works well except when newly coming candidate sequences are not large in the original database. If this occurs frequently, the performance of the FASTUP algorithm will correspondingly decrease.

In this paper, thus an attempt to develop a novel and efficient incremental mining algorithm capable of updating sequential patterns based on the concept of pre-large sequences is done. A pre-large sequence is not truly large, but nearly large. A lower support threshold and an upper support threshold are used to realize this concept. Pre-large sequences act like buffers and are used to reduce the movement of sequences directly from large to small and vice versa during the incremental mining process. A safety bound for newly added customer sequences is derived within which rescanning the original database can be efficiently reduced and maintenance costs can also be greatly reduced. The safety bound also increases monotonically along with increases in database size. Thus, the proposed algorithm becomes increasingly efficient as the database grows. This characteristic is especially useful for real-world applications.

III. PROPOSED SYSTEM

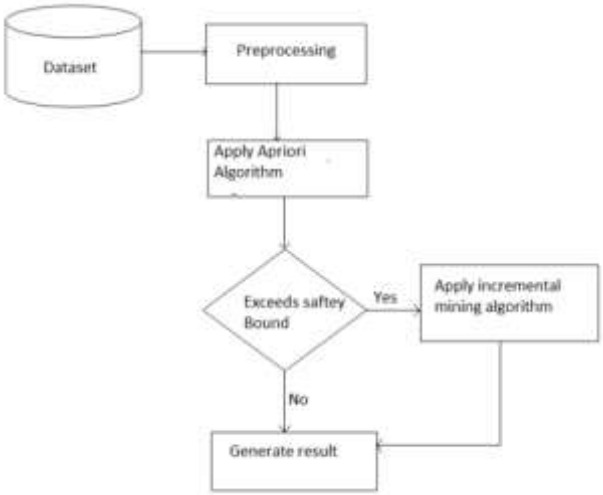

Figure 1: Basic Architecture of Incremental Mining Process

The system works as follows:

I. The Datasets are taken as input.

II. Preprocessing is the first step in data mining in which data is cleaned.

III. Apply the apriori algorithm to generate the large and pre-large item set. Apriori takes upper and lower support as input.

IV. Where support is probability of getting item set from given dataset and confidence is percentage of one item with respect to another in given dataset.

V. Check for the safety bounds if it is within the bound then it will generate the result directly else the incremental mining algorithm is being applied and then the result is generated.

III-1. EXTENDING THE CONCEPT OF PRE-LARGE ITEM- SETS TO SEQUENTIAL PATTERNS

Maintaining sequential patterns is much harder than maintaining association rules since the former must consider both itemsets and sequences. In this paper, an attempt to extend the concept of pre-large itemsets to maintenance of sequential patterns is done. The prelarge concept is used here to postpone original small sequences directly becoming large and vice versa when new transactions are added. A safety bound derived from the lower and upper thresholds determines when rescanning the original database is needed. When new transactions are added to a database, they can be divided into two classes:

Class 1: new transactions by old customers already in the original database;

Class 2: new transactions by new customers not already in the original database.

Considering the old customer sequences in terms of the two support thresholds, the newly merged customer sequences may fall into the following three cases illustrated in figure 2.

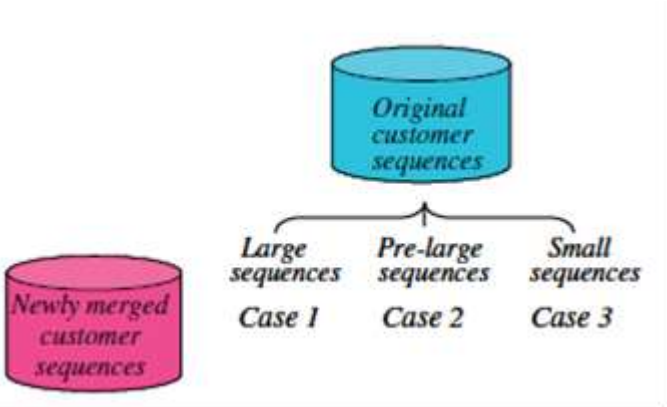

Figure 2: Three cases arising from adding new transaction to existing databases.

IV. IMPLEMENTATION DETAILS

Before applying the incremental mining algorithm the apriori algorithm is applied in order to determine whether the given data exceeds the safety bound. If it exceeds the safety bound then the incremental mining algorithm is applied. Following are the steps of apriori algorithm:

- THERE ARE TOTAL FIVE PHASES INCLUDED IN THIS APPROACH.

1. In the first phase, transactions are sorted first using customer ID as the major key and then using transaction time as the minor key. This phase thus converts the original transactions into customer sequences.
2. In the second phase, large itemsets are found in customer sequences by comparing their counts with the predefined support parameter α. This phase is similar to the process of mining association rules. Note that when an itemset occurs more than once in a customer sequence, it is counted once for this customer sequence.
3. In the third phase, large itemsets are mapped to contiguous integers and the original customer sequences are transferred to the mapped integer sequences.
4. In the fourth phase, the integer sequences are examined for finding large sequences.
5. In the fifth phase, maximally large sequences are then derived and output to users.

- INCREMENTAL MINING ALGORITHM

The proposed an incremental mining algorithm is based on the concept of pre-large itemsets to reduce the amount of rescanning of original databases required whenever new transactions are added [4][5]. A pre-large itemset is not truly large, but promises to be large in the future. A lower support threshold and an upper support threshold are used to realize this concept. The upper support threshold is the same as the minimum support used in conventional mining algorithms. The support ratio of an itemset must be larger than the upper support threshold in order to be considered large. On the other hand, the lower support threshold defines the lowest support ratio for an itemset to be treated as pre-large. An itemset with a support ratio below the lower threshold is thought of as a small itemset. Pre-large itemsets act like buffers and are used to reduce the movement of itemsets directly from large to small and vice versa in the incremental mining process. Therefore, when few new transactions are added, the original small itemsets will at most become pre-large and cannot become large, thus reducing the amount of rescanning necessary. A safety bound for new transactions is derived from the upper and lower thresholds and from the size of the database. This algorithm is described as follows:

Step 1: Retain all previously discovered large and pre-large itemsets with their counts.
Step 2: Scan newly inserted transactions to generate candidate 1- itemsets with counts.
Step 3: Set k = 1, where k is used to record the number of items currently being processed.
Step 4: Partition all candidate k-itemsets as follows.
Case 1: A candidate k-itemset is among the previous large 1- itemsets.
Case 2: A candidate k-itemset is among the previous pre-large itemsets.
Case 3: A candidate k-itemset is among the original small itemsets.
Step 5: Calculate a new count for each itemset in cases 1 and 2 by adding its current count and previous count together; prune the itemsets with new support ratios smaller than the lower support threshold.
Step 6: Rescan the original database if the accumulative amount of new transactions exceeds the safety threshold.
Step 7: Generate candidate k + 1-itemsets from updated large and pre-large k-itemsets, and then go to step 3 until they are null.

The above algorithm, like the FUP algorithm, retains previously mined information, focuses on newly added transactions, and further reduces the computation time required to maintain large itemsets in the entire database. The algorithm can further reduce the number of rescans of the original database as long as the accumulative amount of new transactions does not exceed the safety bound.

V. EXAMPLE

This section describes how actually the incremental mining algorithm for maintain sequential pattern work.

Table 1:
Sixteen transactions sorted according to Cust_id and Trans_time.

Cust_id	Trans_time	Trans_content
1	1998/01/01	A
1	1998/01/20	B
2	1998/01/11	C,D
2	1998/02/02	A
2	1998/02/11	E,F,G
3	1998/01/07	A,H,G
4	1998/02/09	A
4	1998/02/19	E,G
4	1998/02/23	B
5	1998/01/05	B
5	1998/01/12	C
6	1998/01/05	A
6	1998/01/13	B,C
7	1998/01/01	A
7	1998/01/17	B,C,D
8	1998/01/23	E,G

Table 2:
The customer sequences transformed from the transactions in Table 1.

Cust_id	Customer sequence
1	((A)(B))
2	((C,D)(A)(E,F,G))
3	((A,H,G))
4	((A)(E,G)(B))
5	((B)(C))
6	((A)(B,C))
7	((A)(B,C,D))
8	((E,G))

Table 3:
All large sequences generated for the customer sequences in Table 2.
Large Sequences

1-sequence	Count	2-sequence	Count
((A))	6	((A)(B))	4
((B))	5		
((C))	4		
((G))	4		

Table 4:
Two new transactions according to Cust_id and Trans_time.

Cust_id	Trans_time	Trans_content
5	1998/02/01	E,G
9	1998/02/05	E,F,G

Table 5:
Two newly added customer sequences.

Cust_id	Customer sequence
5	((E,G))
9	((E,F,G))

Table 6:
The two merger customer sequences.

Cust_id	Customer sequence
5	((B)(C)(E,G))
9	((E,F,G))

Table 7:
The candidate 1-sequences with their counts.
Candidate 1-sequences

1-sequence	Count
((B))	0
((C))	0
((E))	2
((F))	1
((G))	2
((E,F))	1
((E,G))	2
((F,G))	1
((E,F,G))	1

Table 8:
The large sequences for the customer sequences Table 2.

1-sequence	Count	2-sequence	Count
((A))	6	((A)(B))	4
((B))	5		
((C))	4		
((G))	4		

Table 9:
The pre-large sequences for the customer sequences in Table 2.

1-sequence	Count	2-sequence	Count
((E))	3		
((E,G))	3		

Table 10:
Two newly added customer sequences.

Cust_id	Customer sequence
5	((E,G))
9	((A)(B,C))

Table 11:
The merged customer sequences.

Cust_id	Customer sequences
5	((B)(C)(E,G))
9	((A)(B,C))

Table 12:
All candidate 1-sequences from the two merged customer sequences

Candidate 1 sequences	Count
((A))	1
((B))	1
((C))	1
((E))	1
((G))	1
((B,C))	1
((E,G))	1

Table 13:
Three partitions of all the candidate 1-itemsets in Table 12

Originally large 1-sequences		Originally pre-large 1-sequences		Originally small 1-sequences	
1-sequence	Count	1-sequence	Count	1-Sequence	Count
((A))	1	((E))	1	((B,C))	1
((B))	1	((E,G))	1		
((C))	1				
((G))	1				

Table 14:
The total counts of ((A)), ((B)), ((C)) and ((G)).

1-sequence	Count
((A))	7
((B))	6
((C))	5
((G))	5

Table 15:
The total counts of ((E)) and ((E, G)).

1-sequence	Count
((E))	4
((E,G))	4

Table 16
All candidate 2-itemsets appearing in newly merged customer sequences.

Candidate 2 Itemsets				
((B)(C))	((B)(E))	((B)(G))	((B)(E,G))	((C)(E))
((C)(G))	((C)(E,G))	((A)(B))	((A)(C))	

Table 17
All large and pre-large 2sequences for entire updated database.

Large 2-sequences		Pre-large 2-sequences	
sequences	Count	sequences	Count
((A)(B))	5		

VI. Results Discussion

Table 18
Execution time required as the size of coming dataset increases

Dataset Name	No. of transaction	Execution time
Data12	77	0sec
Data123	470	17ec
Data1234	50	1sec
Data1	520	1sec

Table 19
Number of large and prelarge itemset

Dataset name	No. of transaction	No. of large items	No. of prelarge items
Data12	77	30	96
Data123	470	402	8529
Data1234	50	402	8527
Data1	520	465	8506

The Table 18 and Table 19 shows as the number of transaction of coming dataset increases the execution time reduces because it process only prelarge item set. The Algorithm avoids recomputing large sequences that have already been discovered. It focuses on newly added transactions, thus greatly reducing the number of candidate sequences. As a result, the new method is faster than the previous approach.

VII. Conclusion

In this paper, we have proposed the concept of pre-large itemsets, and designed a novel, efficient, incremental mining algorithm based on it. Using two user-specified upper and lower support thresholds, the pre-large itemsets act as a gap to avoid small itemsets becoming large in the updated database when transactions are inserted. Our proposed algorithm also retains the following features of the FUP algorithm.

1. It avoids re-computing large itemsets that have already been discovered.
2. It focuses on newly inserted transactions, thus greatly reducing the number of candidate itemsets.
3. It uses a simple check to further filter the candidate itemsets in inserted transactions.

Moreover, the proposed algorithm can effectively handle cases, in which itemsets are small in an original database but large in newly inserted transactions, although it does need additional storage space to record the pre-large itemsets. Note that the FUP algorithm needs to rescan databases to handle such cases. The proposed algorithm does not require rescanning of the

original databases until a number of new transactions determined from the two support thresholds and the size of the database have been processed. If the size of the database grows larger, then the number of new transactions allowed before rescanning will be larger too. Therefore, as the database grows, our proposed approach becomes increasingly efficient. This characteristic is especially useful for real-world applications.

VIII. ACKNOWLEDGMENT

I take this opportunity to express my profound gratitude to my guide Prof.N.G.Pardeshi for his personal involvement and constructive criticism provided beyond technical guidance during the course of the preparation of the paper. He has been keen enough for providing me with the invaluable suggestions from time to time. Above all, his keen interest in the seminar helped me to come out with the best .I would like to thank Prof.D.B.Kshirsagar, Head of department for his guidance and support. I would also like to thanks Prof. P. N. Kalvadekar, ME Coordinator for providing necessary lab facilities during the period of paper work. I would like to thank my parents and my friends who have constantly bolstered my confidence and without whose moral support and encouragement, this seminar would have been impossible. This paper is the outcome of the help of these. Any error that might have crept in is solely mine.

REFERENCES

[1] Lin, M. Y., Lee, S. Y., "Incremental update on sequential patterns in large databases", In The 10th IEEE international conference on tools with artificial intelligence,(1998) pp. 24-31.
[2] Agrawal, R., Imielinksi, T. Swami, A.," Mining association rules between sets of items in large database", In the ACM SIGMOD conference. Washington DC, USA, (1993) pp. 207-216.
[3] Hong, T. P., Wang, C. Y., Tao, Y. H., "Incremental data mining based on two support thresholds", In The fourth international conference on knowledge based intelligent engineering systems and allied technologies.(2000)
[4] Ratchadaporn Amornchewin,"Probability-based incremental association rules discovery algorithm with hashing technique"', International Journal of Machine Learning and Computing, Vol. 1, No. 1, (2011)
[5] Tzung-Pei Hong, Ching-Yao Wangc, Shian-Shyong Tseng.,"An incremental mining algorithm for maintaining sequential patterns using prelarge sequences", Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan(2010).