ISSN 2047-3338

Mining Temporal Association Rules by Automatic Partitioning and Cope Up With Missing Values

Mitula H. Pandya[1] and Mahesh H. Panchal[2]
[1,2]Department of Computer Engineering, Kalol Institute of Technology & Research Center, Kalol, Gujarat, India
[1]mitula.pandya.88@gmail.com, [2]mkhpanchal@gmail.com

Abstract– **Discovering hidden knowledge from large amount of data in form of association rules have become very popular in growing field of data mining. Several algorithms have been developed for mining association rules. All those algorithms can be effectively applied on dataset where data has not any time granularity means non-temporal dataset. Temporal dataset is one where every term has time dimension. In any dataset there is much possibility of having missing values. If association rules are discovered for such dataset the rules are not be perfect, imprecise. This paper mainly focus on the temporal association rules and present a technique satisfying two main aims of temporal association rules are all the possible partitions and filled missing values. In this work, a novel approach is used to fill the most associative values at the place of missing values by temporal association rules after generating all the possible partitions of the dataset automatically.**

Index Terms– **Data Mining, Temporal Data Mining, Temporal Association Rule Mining, Temporal Frequent Pattern and Missing Value**

I. INTRODUCTION

OVER the last decade, explosion of growth in stored data has generated urgent need of a technique called data mining. So ultimate intent of data mining is to deal with the massive collection of data to utilize it in an understandable way so that the unidentified patterns can be discovered for the decision making and to achieve the competitive benefits. The traditional data mining miss the ability to analyze variation in data over time. To overcome this lack of traditional data mining, temporal data mining is used which consider the time attribute analysis.

Association rules are at the heart of data mining. Association rule is produced in two steps. First, all frequent item sets in a database are extracted utilizing a minimum support. Then in the second step, various rules are created using frequent item sets and minimum confidence constraints. For example, examining the dataset of a supermarket we may get an association rule like:

Milk$\rightarrow$Cold Drinks (support=0.01%, confidence=5%)

This means that 0.01% of all transactions contain both milk and cold drinks, and 5% of all transactions which contain milk also contain cold drinks. It can be said that above rule is not a prominent rule as the support of the rule is very low [1], [5]. It means that milk and cold drinks seldom coexist in the entire dataset. As well here confidence of the rule is also very low means a very few transactions which contain milk contain cold drinks. However by considering the transactions in a specific time interval, it can be found that a large number of transactions contain both milk and cold drinks. So the rule can have higher support and higher confidence in a specific interval.

So it conclude from above example is that an itemset may not be frequent for the entire dataset but may be frequent in some specific intervals. In traditional association rule mining if the minimum support value is kept very low then it can be found many trivial rules, and if the minimum support value is kept very high then it may lose the valuable rules.

The temporal association rule contains time attribute so it has little change in finding temporal frequent patterns. The temporal frequent patterns mined for the entire database same as in traditional association rule mining are called as global frequent patterns. But sometimes only global frequent patterns may lead to the wrong decisions[1,5]. For example, if the shopkeeper mines the above rule and promote a sale package for 20% discount for buying cold drinks and milk together in the winter, then it may be failed promotion. As most sales of cold drinks are in summer, so the above rule may suitable for summer not winter. So it can be said that lots of meaningful frequent patterns can found in some specific interval. So it is important to generate the association rule which contains specific interval to be applied for the best results, are known as the local frequent patterns.

Temporal association rule mining is mainly useful for the market basket data analysis. But real world market basket data tend to be incomplete having the missing values in the dataset are known as the null values. To uncover these kind of null values sometimes software routines are used such as "?", "none", "don't know", "not applicable". Such missing values sometimes may cause the error in the data, which can be a strong reason for the misleading temporal market basket analysis.

There are different methods for finding the temporal association market basket data mining and for filling the missing values. One of the best techniques is used for our work presented further in the paper. The next section describes the background and the related work of temporal association rule mining. In section III, the proposed best algorithm is described. Section IV provides the experimental results of the algorithm. Section V concludes the work and gives the overview for the future work.

II. BACKGROUND AND RELATED WORK

Temporal association rule mining is an extension of association rule generation. In past decades, various algorithms have been proposed to efficiently discover the temporal association rule mining. Mainly algorithm methods for temporal association rule mining are differing from each other for the partitioning and for finding frequent itemsets. This section introduces the related studies by describing some of them.

The [2] explored the problem of temporal data mining and suggest the solution in a directed graph based approach. It partitions the dataset by common starting or common ending time and then finds the candidate 2-itemsets for each partition. In the last combining all the candidate 2-itemsets it makes a directed graph tree form which frequent itemsets are easily known and then the association rules are generated.

The [3] proposed the algorithm which partitions the dataset based on the calendar based patterns. It having the time granularity like year, month, week, day etc. After the partitioning for finding the frequent patterns it needs another utility table from which it finds the direct frequent temporal utility 2-itemsets and then further finds the association rules.

The [4] addressed the algorithm in which the partitions are done by the calendar based approach. The frequent itemsets are found based on the apriori algorithm for the each partition. Then the association rules are generated. After the rules generation, the strong rules are taken by the correlation calculation known as the interest measure. So here negative temporal association rules are discarded.

The [6] addressed the up-to-date association rule. The main idea comes from that an itemset may not be frequent for an entire database but may be large up-to-date since the items seldom occurring early may often occur later. An up-to-date pattern is composed of an itemset and its up-to-date lifetime, in which the user-defined minimum support threshold must be satisfied.

Every item has a period of lifetime or lifespan in the database, which explicitly represents the temporal duration of the item information, i.e. the time in which the item is relevant to the user. There are more other methods for the partitions are like weighted sliding window method[7], exhibition period[8],time-stamped[9,10]. Some algorithms may not have the assumptions for the partitions.

In our algorithm, we have adopted GLF Miner [5]. In which we don't assume the transaction is time stamped and use the transaction sequence as the temporal factor to generate all the intervals of itemsets. Our main contribution is that it can not only find out global frequent patterns in the whole database, but also discover local frequent patterns within the specific intervals without any domain knowledge. And after finding the association rules the missing values in the input dataset are filled with the best associated itemset.

III. ALGORITHM DESCRIPTION

This section is conducted for mining temporal association rules by automatic partitioning. It also shows the best associated method for filling the missing value for the work. Input market basket dataset containing missing values has been generated through the DTM data generator tool. The sample dataset is shown in Table 1.

Tid	Transaction Time (YY-MM-DD)	Items
1	2001/1/1	A B C
2	2001/1/11	? C
3	2001/2/23	A B D
4	2001/3/5	A B C
5	2001/3/10	C D
6	2001/4/2	? C

Table 1: Sample temporal dataset

The steps of the algorithm are as under.

Step 1: Find All Possible Frequent Itemsets and Partitions [5]:

1) The Dataset Is Transferred to the Bitmap Representation

The bitmap representation is illustrated as under. For the containing value for particular transaction it takes the value '1' or 'true' else '0' or 'false'.

bitmap representation=α[begin,end](bitmap):count

e.g., A[1,6](101100):3

AB[1,1](1):1 , [3,4] (11):2

2) Finds Global Frequent Item Set

To find the global frequent itemset the support count is calculated for the each item. Then if the support count value be greater than or equal to the predefined min_support count value then the itemset is known as global frequent itemset. If the itemset is not the global frequent then those itemsets are further processed for the localization. Suppose here the min_support = 0.5, then for eg.

- count of itemset 'A' = 3 → support of 'A' = 3/6 = 0.5 ≥ 0.5 → so global frequent
- count of itemset 'D' = 2 → support of 'D' = 2/6 = 0.33 < 0.5 → so not global frequent (so localize it)

3) Perform Localization

The itemsets which are not global are processed further for the localization. Localization is done using the three operations, which are described below.

Partitioning

Suppose in our example 'D(001010):2' is not the global frequent itemset. So for the localization we first partition the itemset 'D'. The procedure of partitioning for the itemset 'D' is shown in the Table 2.

No.	Bit	Interval		Action
1	0			skip
2	0			skip
3	1	D[3,3]:1	$1/0.5=2 \geq (3-3+1)$	
4	0	D[3,3]:1	$1/0.5=2 \geq (4-3+1)$	Add a new interval
5	1	D[5,5]:1	$1/0.5=2 \geq (5-5+1)$	
6	0	D[5,5]:1	$1/0.5=2 \geq (6-5+1)$	Add a new interval

Table 2: Partitioning the interval of itemset 'D'

Merging

After the partitioning, two consecutive intervals are merged from the back to the front in order to form a larger frequent interval. To merge the itemsets there is a condition as below.

- The two consecutive intervals 'αn-1[begin,end]:count' & 'αn[begin,end]:count' of an itemset α are mergeable if , (αn.count + αn-1.count) / (αn.end - αn-1.begin+1) ≥ min_support

  For eg. Two consecutive intervals of itemset 'D' = D1[3,3](1):1 & D2[5,5](1):1 then, (D2.count + D1.count) / (D2.end - D1.begin+1) = (1+1) / (5-3+1) = 0.66 ≥ 0.5 min_support

Merge is defined as:

- Merge(αn-1, αn) = (αn-1)[αn-1.begin, αn.end](bitmap):(αn-1.count + αn.count)

  For eg. Merge(D1, D2) = (D1)[D1.begin, D2.end](bitmap of D1 & D2):(D1.count + D2.count) = D1[3,5](101):2

Pruning

Prune the intervals whose length is less than Min_length in order to remove trivial intervals.

If the length of an interval < Min_length then interval is not prominent enough so the interval is deleted.

- Eg. If the length of an interval = 1, then support & confidence of this interval becomes 100%. So it is meaningless and that should be regarded as a noise and should delete it.

In our dataset at the 1st level, no items are pruned. But taking the example of the second level's itemset in our sample dataset,

- Eg. Suppose Min_length = 0.2 ,

  the length of interval AD[3,3] = 1 → Min_length of 'AD' = 1/6 = 0.16 < 0.2, so remove 'AD'

4) Recursively Join the Itemsets in Depth-First Order from the Root

The nodes of itemset α and itemset β are called joinable if and only if both the nodes have the same parent node in the interval tree. The node of length (k-1) joins all the right siblings to generate the combined itemset of length k. To join the itemset the condition is as under.

- γ[begin,end](bitmap)=join(α[begin,end](bitmap), β[begin,end](bitmap))

  = (α U β) [Max (α.begin, β.begin), Min (α.end, β.end)] {AND (bitmap of the both itemset for new interval after joining)}

  Eg. Suppose joining two nodes 'A[1,6](101100):3' and 'D[3,5](101):2' then, γ[begin,end](bitmap) = join(A[1,6](101100):3, D[3,5](101):2) = AD[3,3](1):1

This process is repeated until there is no more partition for the frequent pattern are possible or there are no more frequent patterns to partition.

Now suppose for the sample dataset Table 1, if the frequent global and local frequent items are generated as in the Table 3.

Step 2: Generate the Association Rules:

Using the above generated frequent itemsets and the partitions from all the levels, temporal association rules are generated. From those generated rules, the required valid association rules based on the minimum confidence constraint are considered further.

Here the different possible partitions are generated so from that the required time period mining can be done from the generated required strong valid association rules. For the sample dataset of table 1, generated association rules are shown in Table 4 using the frequent items of Table 3.

Possible association rules with possible partitions A→B[1,6],A→C[1,4],B→C[1,4],A→B[1,4],AB→C[1,4],AC →B[1,4],BC→A[1,4]
Strong valid association rules with possible partitions satisfying min_conf constraint A→B[1,6],A→B[1,4],AC→B[1,4],BC→A[1,4]

Table 4: Association rules considering Min_confidence=70%

Bitmap representation of every item A(101100):3, B(101100):3, C(110111):5, D(001010):2			
Level1	Global frequent itemsets		A[1,6](101100):3, B[1,6](101100):3, C[1,6](110111):5
	Localization	Partition	D[3,3](1):1, D[5,5](1):1
		Merge	D[3,5](101):2
		Prune	-
	Frequent items at Level1 A[1,6](101100):3, B[1,6](101100):3, C[1,6](110111):5, D[3,5](101):2		
Level2 join(itemset n,*)	Global frequent itemsets		AB[1,6](101100):3
	Localization	Partition	AC[1,1](1):1,AC[4,4](1):1, AD[3,3](1):1, BC[1,1](1):1,BC[4,4](1):1, BD[3,3](1):1, CD[5,5](1):1
		Merge	AC[1,4](1001):2, BC[1,4](1001):2,
		Prune	AD[3,3](1):1, BD[3,3](1):1, CD[5,5](1):1
	Frequent items at Level2 AB[1,6](101100):3, AC[1,4](1001):2, BC[1,4](1001):2		
Level3 join(itemsets nm,*)	Global frequent itemsets		-
	Localization	Partition	ABC[1,1](1):1, ABC[4,4](1):1
		Merge	ABC[1,4](1001):2
		Prune	-
	Frequent items at Level3 ABC[1,4](1001):2		
Level4	-		

Table 3: Frequent itemsets for the sample dataset table 1[5]

TID	Itemsets	Strong valid association rules with partitions	Possible suitable items to fill missing values	Possible rules having suitable item in the rule as consequent with partitions having the particular transaction for filling the missing values	Best valid associated rules by confidence constraint	Possible best associated items to be filled
1	A B C	A→B[1,6] A→B[1,4] AC→B[1,4] BC→A[1,4]	-			
2	? C		A B	A→B[1,6] A→B[1,4] AC→B[1,4] BC→A[1,4]	A→B[1,4] AC→B[1,4] BC→A[1,4]	A B
3	A B D		-			
4	A B C		-			
5	C D		-			
6	? C		A B	A→B[1,6]	A→B[1,6]	B

Table 5: An example to fill missing value with Min_support = 50% , Min_length = 20% , Min_confidence = 70%

Step 3: Fill the Missing Values:

After generating the strong valid required association rules, the missing categorical item would be chosen after searching the highest valued associated ruled item. For that first select all the suitable items which are suited for filling the missing value. Then select all the possible rules which have the suitable items as consequent of the rule. Also consider those rules only, which must have that particular transaction in its partition length. After selecting the rules for that particular transaction, select the best associated rule which satisfies the highest minimum confidence constraint. Now from best associated rules, select the best associated items to fill the missing value, which are consequent of the rule. For our sample dataset, the example for missing value is shown in Table 5.

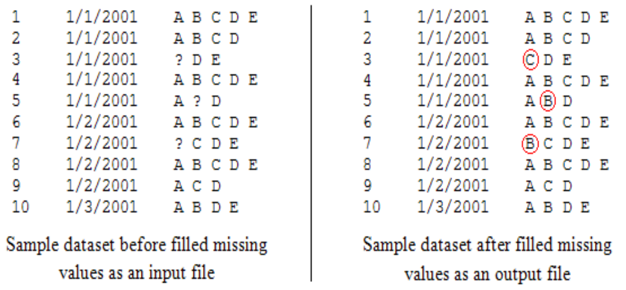
IV. EXPERIMAENTAL RESULTS

We implemented our algorithm in JAVA on an Intel Pentium Dual-Core Processor personal computer with 2 GB RAM on Windows XP service pack 2 OS having 120 GB Hard disk.

One thing is noted here that with the more efficient configuration, the execution time may decrease. For this configuration also it works so effectively, so it can be said that it is an efficient algorithm that consumes the less time and gives the accurate mining.

The input dataset are generated using the DTM data generator tool. The size of itemsets in a transaction is 5, the size of potential maximal frequent itemsets is 5, and the number of transactions is 1,000.

The first experimental result is a comparison done between the text files where the given input dataset file which contains the missing values, with the generated output file which has filled the missing values with the best associated item. As shown in the fig 1, the dataset with missing value as input file and the dataset by filled missing values after execution can be compared. By this comparison one can get better idea about the filled missing values. Here the missing values are filled by the most associated item. So this filled missing values output can be used for the more accurate decision making in the future.



1	1/1/2001	A B C D E
2	1/1/2001	A B C D
3	1/1/2001	? D E
4	1/1/2001	A B C D E
5	1/1/2001	A ? D
6	1/2/2001	A B C D E
7	1/2/2001	? C D E
8	1/2/2001	A B C D E
9	1/2/2001	A C D
10	1/3/2001	A B D E

Sample dataset before filled missing values as an input file

1	1/1/2001	A B C D E
2	1/1/2001	A B C D
3	1/1/2001	C D E
4	1/1/2001	A B C D E
5	1/1/2001	A B D
6	1/2/2001	A B C D E
7	1/2/2001	B C D E
8	1/2/2001	A B C D E
9	1/2/2001	A C D
10	1/3/2001	A B D E

Sample dataset after filled missing values as an output file

Fig. 1: Comparison between input and output dataset for missing values

The second experimental result is below graph which shows the results between various values for the total transactions, support and the time for the execution in ms.

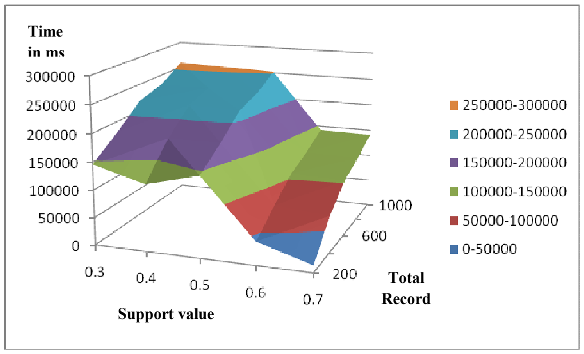


Fig. 2: Comparison between different values for total records, support and the time

From the graph we can get the idea that as the support values are increased the execution time is decreased. As well as the total no of records increases the time of execution is also increased. Also one can get the idea that how the time it takes to mine the different values for records and the support for the different configuration.

V. CONCLUSION AND FUTURE WORK

From the study, it is concluded that the temporal data mining is the one of the best growing and important branch of data mining for the decision making and the market basket analysis. For that the interested rules for particular time interval can be generated through the temporal association rule mining. Different algorithms are developed for temporal association rule mining, but the algorithm which is chosen here is most suitable as it satisfies the main aim of temporal data mining by generating all the possible partitions. It is an efficient algorithm as it finds all the possible frequent itemsets from all the intervals for the temporal association rules as well generate the all possible intervals for the frequent items. The proposed algorithm generates the automatic partitions without need of any domain knowledge. It also fills the missing values by the most associated value so that the missing items can be known and further the accurate mining does not mislead for the decision making.

In the future work, one can use the resulting filled missing values output file of our algorithm, as an input file for the further mining, and then the most accurate decision can be taken as filled missing values and the all possible partitions are the main aspects for the temporal data mining. This algorithm can be extended further for online temporal continuous string mining with some modifications in the implemented algorithm. This algorithm uses the association rule mining method, but if any other way is possible which gives the better mining than by association rule mining, then also it is possible to generate the same by another efficient manner.

REFERENCES

[1] Mitula H. Pandya, Mahesh H. Panchal, "*Survey on Temporal Mining Techniques from Time Series Database***"**, ATAST 2012: 978-81-923514-0-7, pg 214-219.

[2] A. V. Senthil Kumar, Adnan Alrabea, Pedamallu Chandra Sekhar, "*Temporal Association Rule Mining in Large Databases*", IGI Global, ch003, pg 48-65.

[3] Maragatham. G, Lakshmi M., "*A Strategy for Mining Utility based Temporal Association Rules*", 2010 IEEE, pg 38-41.

[4] Chen Zhuo, Luan jiahui, Lu chen, "*A Fuzzy Calendar-based Algorithm for Mining Temporal Association Rules and Its Application*", 2009 Sixth International Conference on Fuzzy System and Knowledge Discovery, IEEE Computer Society, pg 28-33.

[5] Kuo-Cheng Yin, Yuh-Long Hsieh, Don-Lin Yang, "*GLFMiner-Global and Local Frequent Pattern Mining with Temporal Intervals*", 2010 5th IEEE conference on Industrial Electronics and Applicationsis, pg 2248-2253.

[6] Tzung-Pei Hong, Yi-Ying Wu, Shyue-Liang Wang, "*An effective mining approach for up-to-date patterns*", Expert systems with applications, 36,2009, pp. 9747-9752.

[7] Pauray S. M. Tsai, "*Mining frequent itemsets in data streams using the weighted sliding window model,*" Expert Systems with Applications, Volume 36, Issue 9, November 2009, pp. 11617-1162.

[8] C. H. Lee, C. R. Lin and M. S. Chen, "*On mining general temporal association rules in a publication database,*" Proceedings of the 2001, IEEE International Conference on Data Mining, pp. 337-344.

[9] Yingjiu Li, Peng Ning, X. Sean Wang and Sushil Jajodia, "*Discovering calendar-based temporal association rules,*" Data and Knowledge Engineering, Volume 44, Issue 2, Feb. 2003, pp. 193-218.

[10] K. Verma, O. P. Vyas, and R. Vyas, "*Temporal approach to association rule mining using T-tree and P-tree*" , Lecture Notes in Computer Science, 3587, 2005, 651-659.