Temporal Association Rule Using Without Candidate Generation

Keshri Verma　　　　O P Vyas

Abstract

Associationship is an important component of data mining. In real world, the knowledge used for mining rule is almost time varying. The items have the dynamic characteristic in terms of transaction, which have seasonal selling rate and it holds time-based associationship with another item. In database, some items which are infrequent in whole dataset may be frequent in a particular time period. If these items are ignored then associationship result will no longer be accurate. To restrict the time based associationship, calendar based pattern can be used [5]. Calendar units such as months and days, clock units, such as hours and seconds & specialized units , such as business days and academic years, play a major role in a wide range of information system applications.[11] Our focus is to find effective time sensitive algorithm for mining associationship by extending frequent pattern tree approach [3]. This algorithm reduces the time complexity of existing technique[5]. It also uses the advantages of divide & conquer method to decompose the mining task into a smaller tasks for database.

Keywords : Data Mining, Temporal Data Mining, Temporal Association Rule, Frequent Pattern Approach.

0. Introduction

The Associationship is an important component of data mining. It indicates the co-relationship of one item with another. For example Eggà coffee (support 3%, confidence 80%)

It means that 3% of all transaction contain both egg & coffee, and 80% of transaction that have egg also have coffee in them.

One important extension to association rule is to include a temporal dimension. For example egg and coffee may be ordered together primarily between 7 to 11 AM in this interval the support & confidence is 40% but at another interval, support is as low .005% in other transaction [5]. In market, various items have a seasonal selling rate.. It is also importance because many items are introduced or removed form the database, that is items lifespan[ZMTW] which means that item is valid on specific time interval. To discover such temporal intervals (with calendar information) together with the association rules that hold during the time interval may lead to useful knowledge. If calendar schema is applied it is called calendar based temporal association rule

A hierarchy of calendar concepts determines a calendar schema. A calendar unit such as months and days, clock units, such as hours and seconds & specialized units, such as business days and academic years, play a major role in a wide range of information system applications.[11]. A calendar schema defines a set of simple calendar – based patterns. Each calendar pattern defines a set of time intervals.

Our data mining problem is to discover all temporal association rules w.r.t. calendar schema from a set of time stamped transactions. This paper ,improve an existing frequent pattern tree approach to discover temporal association rule to increase the optimization over existing one[5].

The organization of the paper as follows :

we develop a data structure called Temporal FP-tree.

we introduce a novel algorithm for mining frequent pattern using Temporal FP-tree.

The rest of the paper is organized in five section In Section 2, we discuss some related works. In section 3 we define temporal association rule in term of calendar schema. In Section 4 elaborate the extended algorithm of frequent pattern approach ,section 5 shows conclusion & future works and section 6 provides application of above investigation.

1. Related Work

The concept of association rule was introduced as Apriori algorithm [1]. Its performance was improved by deploying frequent pattern tree approach [3]. In paper [7] the omission of the time dimension in association rule was very clearly mentioned. A temporal aspect of association rule was given by Juan [2]. The transaction in the database are time stamped and time interval is specified by the user to divide the data into disjoint segments, like month , days & years. Further The cyclic association rule was introduced by Ozden [7] with minimum support & high confidence. Using the definition of cyclic association rule, It may not have high support & confidence for the entire transactional database.

A nice bibliography of temporal data mining can be found in the Roddick literature [8].

Rainsford & Roddick presented extension to association rules to accommodate temporal semantics. According to his logic the technique first search the associationship than it is used to incorporate temporal semantics. It can be used in point based & interval based model of time simultaneously[10]. A Frequent pattern approach for mining the time sensitive data was introduced in[9] Here the pattern frequency history under a tilted-time window framework in order to answer time-sensitive queries. A collection of item patterns along with their frequency histories are compressed and stored using a tree structure similar to FP-tree and updated incrementally with incoming transactions [9].

2. Problem Definition

2.1 Association Rule

The concept of association rule, which was motivated by market basket analysis and originally presented by Agrawal. [1]. Given a set of T of transaction, an association rule of the form Xà Y is a relationship between the two disjoint itemsets X & Y. An association rule satisfies some user-given requirements. The support of an itemset by the set of transaction is the fraction of transaction that contain the itemset. An itemset is said to be large if its support exceeds a user-given threshold minimum support. The confidence Xà Y over T is a transaction containing X and also containing Y. Agrawal first introduced Apriori algorithm in1994 for mining associationship in market basket data [1]. Due to complex candidate generation in the data set Jiewai Han invented a new technique of FP-growth method for mining frequent pattern without candidate generation [3]. Efficiency of this mining technique is better than all mos all algorithm like Apriori, aprioriTid, Apriori Hybridm because (1). a large dataset is compressed into a condensed ,smaller data structure, which avoids costly & repeated data scan , (2). FP-tree-based mining adopts a pattern-fragment growth method too avoid the costly generation of a large number of candidate generation sets and,(3). A partitioning-based , divide- and-conquer method is used to decompose the mining task into a set of similar tasks for conditional database, which dramatically reduce the search space.

In our opinion this mining technique will be become more useful if we include the time factor in to it.

2.2 Temporal association rule

Definition 1 : The frequency of and itemset over a time period T is the number of transactions in which it occurs divided by total number, of transaction over a time period. In the same way , confidence of a item with another item is the transaction of both items over the period divided by first item of that period.

Support(A) = Frequency of occurrences of A in specified time interval / Total no of Tuples in specified time interval

Confidence(A => B[Ts,Te]) = Support_count(A Ç B) over Interval / occurrence of A in interval

2.3 Simple calendar based Pattern :

When temporal information is applied in terms of date, month , year & week form the term calendar schema. It is introduced in temporal data mining. A calendar schema is a relational schema (in the sense of relational databases) R = (f_n : D_n, F_{n-1} : D_{n-1},………F_1 :d_1) together with a valid constraint. A calendar schema (year : {1995,1996,1997…..} , month : {1,2,3,4,……12}, day : {1,2,3…..31} with the constraint is valid if that evaluates (yy, mm, dd) to True only if the combination gives a valid date. For example <195,1,3> is a valid date while ,<1996,2,31> is not.

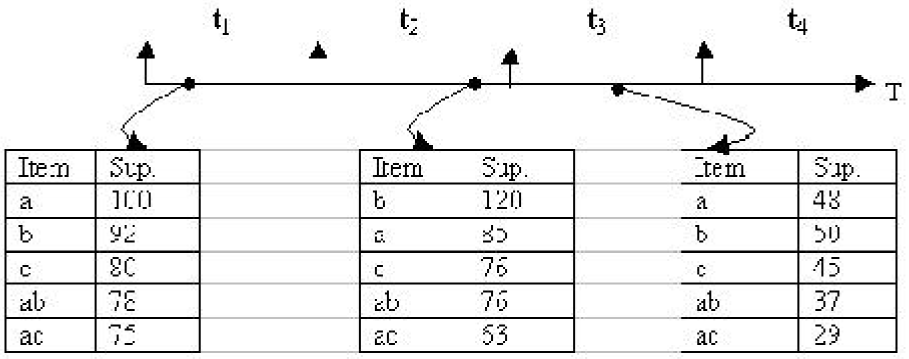
In calendar pattern , the branch e cover e' in the same calendar schema if the time interval e' is the subset of e and they all follow the same pattern. If a calendar pattern <d_n,d_{n-1},d_{n-2}……..d_1> covers another pattern <d'_n, $d'_{n-1;}$, d'_{n-2} ……..d_1> if and only if for each I, 1<=i<=n or d_i = d'_i.

Now Our task is to mine frequent pattern over arbitrary time interval in terms of calendar pattern schema.

3. Proposed Work

The support of dataset in the data warehouse can be maintained by dividing it in different intervals. The support of a item in interval t1 can not be the same in interval t2. A infrequent or less support item in interval t1 can be frequent item in interval t2.

The calendar schema is implemented by applying apriori algorithm [5]. It follows the candidate generation approach in order to mine the frequent item. We assist here that divide & conquer approach is more efficient than apriori approach. It construct a tree & each branch indicate the association ship of item. It reduces the size of dataset and increases the performance & efficiency of algorithm. It can solve following queries (1) What are the frequent set over the interval t_1 and t_2 ? (2) what are the period when (a,b) item are frequent ? (3) Item which are dramatically change from t4 to t1.



Item	Sup.
a	100
b	92
c	80
ab	78
ac	75

Item	Sup.
b	120
a	85
c	76
ab	76
ac	53

Item	Sup.
a	48
b	50
c	45
ab	37
ac	29

Figure 1 Frequent pattern in different interval

Lemma 1 : During transaction in database the association of a item over the support x can be obtained by projection of the branch of FP-tree.

Rationale : Based on the TFP-tree construction process its frequent item can be projected into a single branch.

For a path a_1, a_2 $a_{3 \ldots\ldots\ldots}a_k$ from the root to a node in a FP-tree. Suppose a_{ak} be the count at the node labeled $a_{k \text{ and}}$ c'_{ak} be the sum of the count of the branch of the node.

Table 1. A Transaction database in running example

Tid	Item bought	Date	Calendar pattern	Item in Desceing order of Frequency
100	f, a,c,d,g,i,m,p	01/01/2004	<*,01,04>	f,c,a,m
200	a,b,c,f,l,m,o	01/01/2004	<*,01,04>	f,c,a,b,m,o
300	b,f,h,j,o	02/01/2004	<*,01,04>	f,c,b,o
500	a,f,c,,e,l,p,m,n	03/06/2004	<*,06,04>	f,c,a,e,m,l
600	f,a,c,d,e	04/06/2004	<*,06,04>	f,c,a,e
700	b,f,h,j,m,l,o	06/06/2004	<*,06,04>	f,m,l,

Definition (Temporal FP-Tree) **–** A Temporal frequent pattern (FP) is tree structure defined below.

1. It consists of root labeled as “null”.
2. It consist of a set of item-prefix subtrees as the children of the root, and a frequent item header table.
3. Each node in the item prefix subtree consists of four fields : (a) Item name - Item name represents the name of item which is registers on that node (b) count - count registers the number of transactions represented by the portion of the path reaching this node (c) node link - node-link links to the next node of temporal FP-tree (d) calendar pattern time calendar pattern represent the time in which the item transaction appeared.
4. Each entry in the Frequent –item header table consists of two fields (1) Item name (2) Head of node link

Algorithm : (FP- Tree construction)

Input : A transaction database DB and a minimum support threshold

Output : FP Tree, Fp Tree, Frequent item

Method : The FP tree is constructed ad follows :

1. Scan the database DB once. Collect f, the set of frequent item and support of each item. Sort F from support in descending order as Flist, the list of frequent items
2. Create the root of Temporal FP tree and label it as “Null”. For each transaction in DB and do the following

   Select the frequent items in Trans and sort them in descending order of Flist. Let the sorted frequent –item list in the Trans be p[P] where p is first element and P is the remaining list. Cal insert_tree(p[P],T).

```
Procedure insert_tree(p[P],T).
{

Step(1)  If  T has a child N such that
Step(2)  if (N.time = P.time) then
Step(3) if (N.itemname = P. itemname) then
   Step(a) N.count = N.count +1  // Increment the count by 1
Step(4) else create a new node // Node created on the same branch
Step(5) Link to its parent P.count = 1  // Initialize the counter by 1.
Step(6)  else create a new Branch link from the root.
Call insert_tree(P,N)  recursively

} // End of Function
```

Temporal Frequent pattern Tree : Design & Construction

Let I = {a_1, $a_{2,}$ $a_{3..}$ a_m} be a set of items , and a transaction database DB {T_1, $T_{2,}$ $T_{3........}T_n$ } where T_i {i Î [1..n] } is a transaction which contains a set of items in I.

3.1 Temporal Frequent- Pattern Tree

To design the Temporal FP-tree for frequent pattern mining, let’s first examine example from table1.

1. Since time is the most important feature of real world data set, arrange the item in according to time and define the calendar pattern or interval in calendar unit form.
2. In calendar pattern <*> is used to define any day or month. For example <dd,mm,yy> calendar pattern <*,01,04> represents any day of Month January & year 2004.
3. Since only the frequent item will play a role in the frequent pattern mining, so first scan is used to identify the set of frequent items
4. If the set of frequent items of each transaction can be stored in some compact data structure , it may be possible to avoid repeatedly scanning the original transaction database.
5. If multiple transaction share a set of frequent items, it may be possible to merge the shared sets with the number of occurrences registered as count. It is easy to check whether two sets are identical, if the frequent items in all of the transaction are listed according to a fixed order.
6. If two transactions share a common prefix , according to some sorted order of frequent items, the shared part can be merged into one prefix structure as long as the count is registered properly.

With the above observation , a Temporal frequent pattern tree can be constructed as follows :

First , a scan of DB drives a frequent list of items in schema <*,01,04> are {(f:3),(C:2),(b:2), (a:2),(m:2} same for calendar pattern <*,06,04> frequent items are {(f:3),(c:2),(a:2),(e:2),(m:2)(l:2)} and remaining items are infrequent, so skip those item .

Second, the root of the tree is created and labeled with “null”. The FP- tree is constructed as follows by scanning the transaction database DB in second time.

1. The scan of the first transaction leads to construction of the first branch of the tree {(f:1),(c:1),(a:1),(m:1),(p:1)}& it follows the calendar pattern <*,01,04>.
2. For the second transaction, its temporal period is same, so it follows the same branch & the frequent item list {f,c,a,b,m,o}, shares a common prefix <f,c,a> and the count of each node along the prefix is incremented by 1. And a new node (b:1) is created and linked to child of (a:2), another new one (m:1) is created and linked to as the child of (b:1) and another new one (o:1) is created and linked to as the child of (m:1),
3. The third transaction, its time period being same as previous the transaction <f,c,b,o>, shares common prefix <f,c>. So f & c's count is incremented by 1, and a new node b is created although b is already existing but it does not go to that branch as it is not common prefix of the node. Node b is linked as a child of (c:2) and a new node o is created initializing the count because o is first time introduced on Temporal FP-tree and linked as a child of node <b:1>
4. The scan of forth transaction leader constructs another branch because its time period <*,06,04>does not match with existing branch's node time period. New nodes are created with <(f:1),(c:1),(a:1),(e:1),(m:1),(l:1)>
5. The scan of fifth transaction follows the time interval of forth transaction and it follows the same branch if the item prefix matches, It can share the common prefix <f,c,a,e> with existing path <f,c,a,e,m,l> and the count of each node is incremented by 1.
6. For the last transaction, <f,m,l> its time interval matches with second branch so it follow the second branch of FP-tree. Here it shares common prefix f, its count is incremented by 1 , a new node is created for item m , & it is linked to node f by initializing the counter value to1. For next item l again a new node will be created by initializing its counter value & it is linked as a child of node m.

3.3 Mining the frequent item from FP-tree

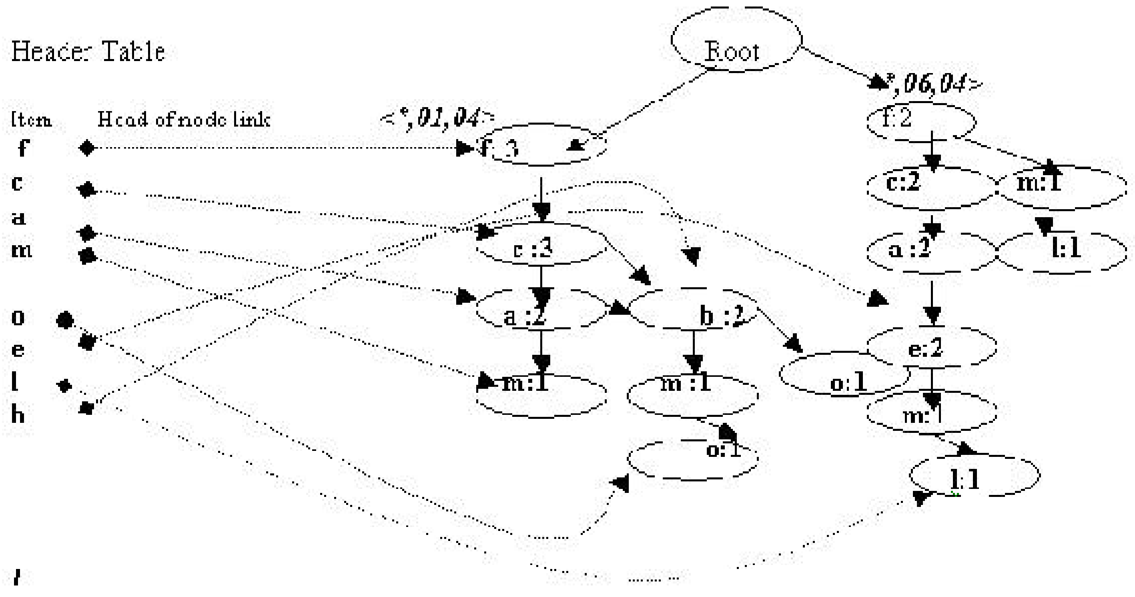


Figure 2 FP tree with different time interval

In Step(2) if (N.time = P.time) * then, it means that when a new node p appears in the FP-tree we check the time of transaction. If it is inside the time of transaction of N item, defined as per table below, then it follows the same branch otherwise a new branch will be created in FP-tree.

Table 2. Shows the path depend on the time of transaction in itemset.

Item name	***N.Time***	***P. Time***	***Branch***
F	<01,01,04>		First
C	<01,01,04>	<01,01,04>	First
A	<01,01,04>	<01,01,04>	First
E	<01,06,04>		Second
M	<01,06,04>	<01,06,04>	Second

Property [3] Node Link property . For any frequent item $a_{i,}$ all the possible patterns contining only frequent items and a_i can be obtained by following $a_{i'}$s node link's , starting from a_i 's head in the FP-tree header.

Example 2 : Mining process from the constructed Temporal FP-tree is shown in figure2. We examine the mining process by starting from the bottom of node-link header table.

For calendar pattern <*,01,04> for node m ,its intermediate frequent pattern is (m:3), and its path is <f:2,c:2,a:1,m:1,p:1> , <f:1,c:1,b:1,m:1> and <f:1, m:1>. Thus to study which appears together with m at time period <*,01,04> only m prefix {(fca:1) ,(fcb:1),(f:1) }, m's sub -pattern base , which is called m's pattern conditional pattern base.(which is called m's conditional Fp tree) leads to two different branches (fc:3) & (f:2).

For node a , its immediate frequent pattern is (a:4)and it is in two different path one for<*,01,04> & second for <*,06,04>. Calendar pattern <*,01,04> consist of <f:3,c:3 a:2> and <*,06,04> consists of <f:2,c:2>

Table 3 Mining Temporal frequent patterns by creating conditional (sub) pattern base

Item	**Time Interval**	**Conditional Pattern base**	**Conditional FP-tree**
m	<*,01,04>	{(fca:1) ,(fcb:1),(f:1) }	{f:2 c:2|m)
	<*,06,04>	{(fcae:1),(f:1)}	(f:2|m)
a	<*,01,04>	{(fc:2)}	(f:2|a)
	<*,06,04>	{(fc:2)}	(f:2|a)
0	<*,01,04>	{(fcabmo:1),(fcbo:1)}	(f:2,c:2,b:2|o)
	<*,06,04>	Nil	Nil
l	<01,*,04>	Nil	Nil
	<*,06,04>	{(fcaeml:1),(fml:1)}	(f:2|l)
e	<01,*,04>	Nil	
	<*,06,04>	{(fca:2)}	(f:2,c:2a:2|e)

c	<01,*,04>	{(fc:3)}	(f:2|c)
	<*,06,04>	{(fc:2)}	(f:2|c)
b	<01,*,04>	{(fcb:2)	(f:2,c:2|b)
	<*,06,04>	Nil	Nil
f	<01,*,04>	f	f
	<*,06,04>	f	f

From the Temporal FP-tree we get the conditional frequent pattern tree , which provides frequent pattern items by calling the procedure FP-growth[3]

4. Conclusion & Future work

We conclude that proposed algorithm gives an efficient time sensitive approach for mining frequent item in the dataset. Discovered rule is easier to understand. It uses divide & conquer technique for construction & traversing of tree which is used to decompose the mining task into a set of smaller task for mining confined pattern in conditional database which dramatically reduce the search space. is better than candidate generation. In fat Data mining concepts are applied where there are huge set of data available in data warehouse. It requires more scanning & processing time. Hence, after applying our logic of the scanning, this valid time & processing time can be decreased. It is very useful for retailer to create its own market strategy as per the requirement of time.

The work can be further extended by designing the tree that is hierarchy of time interval, if the retailer requires the associationship, it can be projected for that period of time.

5. Applications

- *Business Application :* This technique is most useful in Business mining. Most of the real world data have the time varying features. So the retails change their business policy with time to time for maximize the output, Time is here most important feature because some model of vehicle are not available from 1980s , suppose is currently appears in the market.Its history indicate no associationship but the fact is that product is not available on that period , so its associationship is started from the interval where it was valid.
- *Web Mining :* The concept can be applicable in web mining , In WWW the site which is no longer so its associationship also be no longer.
- *Clustering Problem :* This approach can be useful to solve the clustering problem, the cluster can be designed on the basis of period of data., that will reduce the size of data & processing time also.

6. References

1. R. Agrawal & R. Srikant (1994) “Fast algorithm for mining association rule.” In VLDB’94 Chile , Sept ,pp –487-499.

2. Juan M .Ale , Gustavo H. Rossi (2002) “ An approach to discovering temporal association rules”, ACM SIGDD March 1..21.

3. Jian Pei, Jiawei Han, Yiwen Yin and Running Mao (2002) "Mining Frequent Pattern without Candidate Generation", Kluwer online Academy.
4. Jiawei Han, Micheline Kamber , Book (2001) : "Data Mining Concept & Technique".
5. Yingjiu Li, Peng Ning, X. Sean Wang , Sushil Jajodia (2003) "Discovering calendar- based temporal association rules", Data & Knowledge Engineering volume 4 pp– 193-214 ,2003
6. Geraldo Zimbrao, Jano Moreeira de Souza, Victor Teieira de Almeida, Wanderson Araujo da Silva (2000) "An algorithm to Discover Calendar –based Temporal Association Rules with Item's Lifespan Restrictions"
7. Banu Ozden , Sridhar Ramaswamy , Avi Silberschatz (1998) "Cyclic Association Rule" ,In Proc. Of forteenth International conference on Data Engineering pp 412-425
8. John F. Roddick, Kathleen Hornsby, Myra Spiliopoulou (2000) "An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining" Research. TSDM 2000: pp147-164.
9. *Chris Giannella_, Jiawei Hany, Jian Peiz, Xifeng Yany, Philip S. Yu* (2003) "Mining Frequent Patterns in Data Streams at Multiple Time Granularities", pp 191 – 210, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining.
10. *Chris P. Rainsford, John F. Roddick (1999)* " Adding Temporal semantics to association rule", *3rd International conference KSS Springer 1999, pp 504-509*
11. *Claudio Bettini, X. Sean Wang (2000)* " Time Granularies in databases , Data Mining , and Temporal reasoning*" , pp 230, ISBN 3-540-66997-3.*

About Authors

Ms. Keshri Verma is a lecturer in School of Studies in Computer Science, Pt. Ravishankar Shukla University, Raipur, Chhattisgarh.
E-mail : seema2000_2001@yahoo.com

Mr. O P Vyas is working in School of Studies in Computer Science, Pt. Ravishankar Shukla University, Raipur, Chhattisgarh.
E-mail : opvyas@rediffmail.com