Efficient Algorithm for Mining Temporal Association Rule

JUNHENG-HUANG[1] *WANG-WEI*[2]

[1]*Department of Computer Science & Technology, Harbin Institute of Technology at Weihai,, China, 264209*

Summary
It presents an SPFA(Standing for Segmented Progressive Filter Algorithm).The basic idea behind SPFA is to first segment the database into sub-databases in such a way that item in each sub-database will have either the common starting time or the common ending time. Then, for each sub-databse, SPFA progressively filters candidate 2-itemsets with cumulative filtering thresholds either forward or backward in time. This feature allows SPFA of adopting the scan reduction technique by generating all candidate k-itemsets from candidate 2-itemsets directly. The experimental results show that SPFA significantly outperforms other schemes which are extended from prior methods in terms of the execution time and scalability. The advantage of SPFA becomes even more prominent as the size of the database increases.

Keywords: A*ssociation rule; Itemset; Data mining*

1 INTRODUCTION

Date Mining was first recognized as a field in its own right ten years ago, when researchers from a number of different field started to explore the possibilities of extracting information from the large quantities of data held in database. A significant amount of research effort has been elaborated upon deriving data mining techniques to discover useful but unknown knowledge from large databases.

The problem of mining association rules was first explored by [1].Many variants of mining association rules are studied to explore more mining capabilities, such as incremental updating[2,3],mining of generalized and multi-level rules[4],mining associations among correlated or infrequent items[5],and temporal association rule discovery[6,7].

While these are important results toward enabling the integration of association mining and fast searching algorithms, their mining methods, however cannot be effectively applied to the transaction database where the exhibition periods of the items are different from one to another. As a matter of fact, it is a common phenomenon that the items in a real transaction databases have different exhibition periods.

2 Problem Description

We present an efficient standing for segmented progressive filter algorithm (SPFA) in this paper. To address this issue, we explore a new model of mining general temporal association rules where the items are allowed to have different exhibition periods, and the determination of their supports is made in accordance with their exhibition periods. Explicitly, we introduce the notion of *maximal common exhibition period* (abbreviated as *MCP*) .The *MCP* of the itemset *X*, denoted by [*p*,*q*], is defined as the period between the *latest-exhibition-start time p* and the *earliest-exhibition-end time q* of all items belonging to X.

The items in a transaction database may have different exhibition periods. Without loss of generality, it is assumed that a certain time granularity is imposed by the application database. Let *n* be the number of partitions divided by the time granularity imposed. In the model considered,$db^{p,q}(1 \le p \le q \le n)$denotes the portion of the transaction database formed by a continuous region from the partition P_p to the partition P_q,and $X^{p,q}$ denotes the *temporal itemset* whose items are commonly exhibited from the partition Pp to the$\text{partition}P_q$.

As such, we can define the maximal temporal itemset $X^{p,q}$ and the corresponding temporal sub-itemsets as follows:

Definition 1: The temporal itemset X^{pq} is called a maximai temporal itemset (TI) if P_p is the latest starting partition and P_q is the earliest ending partition of all items belonging to *X*.[*p*,*q*] is referred to as the maximal common exhibition period(*MCP*) of the itemset *X*, denoted by *MCP*(*X*).

Definition 2: The maximal temporal itemset $X^{MCP(x)}$ is termed to be frequent if $\text{supp}(X^{MCP(x)}) \ge \text{min\_supp}$ where min-supp is the minimum support required.

Property 1: All temporal sub-itemsets of a frequent maximal temporal itemset are frequent.

Note that the calculation of the confidence of a general temporal association rule not only depends on the relative support of the corresponding maximal temporal itemset, but also relys on the relative supports of the corresponding temporal sub-itemsets. Property 1 ensures that relative supports of the corresponding temporal sub-itemsets can be obtained without extra database scans since all temporal sub-itemsets of the frequent maximal temporal itemset are also frequent.

3 SPFA for Mining General Temporal Association Rules

The problem of mining general temporal association rules is to discover all frequent general temporal association rules from the large database. Similarly, the problem of mining general temporal association can be decomposed into two steps: (1) Generate all frequent maximal temporal itemsets (TI) and the corresponding maximal temporal sub-itemsets (SI) with their relative supports; (2) Derive all frequent general temporal association rules that satisfy min-conf from these frequent TI.

Note that once that frequent TI and SI with their supports are obtained, deriving the frequent general temporal association rules is straight- forward. Therefore, in the rest of this paper we will concentrate our discussion on the algorithm for mining frequent TI and SI.

The major challenge of mining general temporal association rules is that the exhibition periods of the items in the transaction database are allowed to be different from one to another.

SPFA consists of two major procedures Segmentation (abbreviated as ProcSG) and Progressively Filtering (abbreviated as ProcPF). The basic idea is to first divide the database into partitions according to the time granularity imposed. Then, in light of the exhibition period of each item, SPFA employs ProcSG to segment the database into sub-databases in such a way that items in each sub-database will have either the common starting time or the common ending time. For each sub-database, SPFA utilizes ProcPF to progressively filter candidate 2-itemsets with cumulative filtering thresholds from one partition to another. After all sub-databases are process, SPFA unions all candidate 2-itemsets generated in each sub-database. As pointed out earlier, since infrequent 2-itemsets will be filtered out in the early processed partitions, the resulting candidate 2-itemsets will be very close to the frequent 2-itemsets. This feature allows us of adopting the scan reduction technique by generating all candidate k-itemsets (k>2) from candidate 2-itemsets directly. After all candidate itemsets are generated, they are transformed to TI, and the corresponding SI are generated based on these TI. Finally, the frequent TI and SI with their supports can be obtained by scanning the whole database once.

Finally, SPFA is completed by the integration of ProcPS and ProcPF. At first,SPFA segments the database into sub-databases by ProcSG. Then, for each sub-database, SPFA employs ProcPF to progressively filter out candidate 2-itemsets. Using the scan reduction technique[8], SPFA generates all candidate k-itemsets are transformed to TI, and the corresponding SI are generated. Finally, the database is scanned once to determine all frequent TI and SI.

3.1 Description of PROCSG for Segmentation

The motivation of ProcSG is to first reduce the problem of mining general temporal association rules to the one in which the exhibition periods of the items are only allowed to be either different in the starting time or different in the ending time. After such a reduction, we are able to employ ProcPF, in each sub-database, to progressively filter candidate 2-itemsets either forward or backward (in time) efficiently. However, as mentioned above, the advantageous feature of ProcPF is that it can progressively filter out infrequent 2-itemset in the early processed partitions. Thus， the more segments the whole database is divided into, the less significant the filtering effect will be. In view of this, ProcSG is devised to segment the whole database into the minimal number of sub-database as required for items in each sub-database to have either the common starting partition or the common ending parting. The function ProcSG(N) is as follow:

```
{S=ϕ;direction= -1; head=1;
for (i=0   to =n)
 {flag[i][L]=false; flag[i][R]=false;}
for (each item i∈I)
{   (p,q)=MCP(i);
   flag[p-1][L]=true;
   flag[q][R]=true;}
for(i=1 to n-1)
{ if direction==-1
   {if(flag[i][L]==true and flag[i][R]==true)
  {S=S∪(head,i,direction);
   head=i+1;}
elseif(flag[i][L]==true and flag[i][R]==false)
    direction=L;
elseif(flag[i][L]==false and flag[i][R]==true)
```

```
    direction=R;
    }else if direction==L;
{if(flag[i][R]==true)
  { S=S∪ (head,i,direction);
  head=i+1;
  direction=-1;}
}elseif (direction==R)
{if( flag[i][L]==true)
 {S=S∪ (head,i,direction);
  head=i+1;
  direction=-1;}
 }
}
 S=S∪ (head,n,direction);
return S;}
```

3.2 Description of PROCPF for progressively Filtering

After the entire database is segmented by ProcSG, ProcPF is designed to progressively filter candidates 2-itemsets from one partition to another in each sub-database. Specifically, PROCPF generates all 2-itemsets and counts their occurrences in the first partition. For those 2itesmsets whose numbers of occurrences are not smaller than the filtering threshold, they are viewed as candidate 2-itemsets and will be brought to the next partition for further processing. Then, ProcPF will generate new 2-itemsets in the second partition and count their occurrences as well. However, for those candidate 2-itemsets brought from the previous partition, their numbers of occurrences will be cumulated from the previous partition to the current one. Note that the filtering threshold for them will also be cumulated. Similarly, those 2-itemsets shoes numbers of occurrences are not smaller than their corresponding filtering thresholds will be brought to the next partition for further processing until there is no partition to be processed any more.

Let PS denote the cumulative set of candidate 2-itemsets. It is noted that PS is composed of the following two types of candidate 2-itemsets(1) the candidate 2-itemsets that were carried over from the previous partition and remain as candidate 2-itemsets after the current partition is included into consideration.(2) the candidate itemsets that were not in the cumulative candidate set in the previous partition but are newly selected after taking the current partition into account. Since a significant number of 2-itemsets will be filtered out in the early partitions, the resulting PS will be very close to the set of frequent 2-itemsets after processing all partitions. Taking advantage of this feature, we can employ the scan reduction technique to generate all candidate k-itemsets where k>2 from (k-1)itemsets at the same time[8].

The function to progressively filter out infrequent 2-itemsets is shown in ProcPF.ProcPF takes three arguments *p,q* and *direction* as the input, where *p* and *q* are the starting and ending partition to be processed (*p*<=*q*), and *direction* indicates the scanning direction. If the items in the sub-database have the same ending partition, the direction is forward. Otherwise ,the direction is backward. The function ProcPF is shown as follow:

Function ProcPF(*p,q,direction*)

{PS= ϕ *;*

if(direction==left)
head=p;tail=q;
else
head=p,tail=p;
for(h=head to tail)
for(each 2-itemset X_2 *in* P_h*)*
if ($X_2 \notin PS$)
{ X_2*.count=* $N_{p_h}(X_2)$*;*
X_2*.start=h;*
If (X_2*.count>=min_supp**|P_h|*)*
PS=PS∪ X_2*;}*
Else
{X_2*.coun=* X_2*.coun+* $N_{P_h}(X_2)$ *;*
$X_2.coun < \left| \min\_supp * \sum_{m=X_2.sart.h} P_m \right|$
PS=PS- X_2*;}*
Return PS;
}

Finally,SPFA is completed by the integration of ProcPS and ProcPF. At first,SPFA segments the database into sub-databases by ProcSG. Then, for each sub-database, SPFA employs ProcPF to progressively filter out candidate 2-itemsets. Using the scan reduction technique[8], SPFA generates all candidate k-itemsets are transformed to TI, and the corresponding SI are generated. Finally, the database is scanned once to determine all frequent TI and SI.

4 Conclusion

In this paper, we explored a new mode of mining general temporal association rules from large databases where the exhibition periods of the items are allowed to be different from one to another. We developed an efficient algorithm, referred to as SPFA in this paper to discover general temporal association rules effectively. The experimental results showed that SPFA significantly outperforms other schemes which are extended from prior methods in terms of the execution time and scalability. With the capability of mining

general temporal association rules for items with different exhibition periods,SPFA outperforms prior methods in its generality and superiority.

Acknowledgements

This paper is supported by Specialized scientific research Foundation.

Reference

[1] R. Agrawal, T.lmielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases.Proc.of ACM SIGMOD , pages207-216,May 1993.

[2] A.M.Ayad,N.M.E1-Makky, and Y.Taha. Incremental Mining of Constrained Association Rules,Proc.of the First SLAMConference on Data Mining.2001

[3] C.-H.Lee,C-R.Lin.Sliding-Window Flitering: An Efficient Algorithm for Incremental Mining.Proc.of the Tenth ACM International Conference on Information and Knowledge Management,November,2001.

[4]R.Srikant and R.Agrawal.Mining Generalized Association Rules.Proc. of the 21th International Conference on Very Large databases.Pages407-419.Sep.1995.

[5] E.C.et,al. Finding Interesting Associations without Support Pruning. IEEE Transactions on Knowledge and Data Engineering.Page64-78,Jan,2001.

[6]J.Ale and G..Rossi. An Approach to Discovering Temporal Association Rules.ACM Symposium on Applied Computing.2000.

[7]X.Chen and I.Petr. Discovering Temporal Association Rules:Algorithms,Language and system. Proc. Of 200 Int. Conf. on Data Engineering,2000.

[8] J.S.Park,M-S.Chen.Using a Hash-Based Method with Transaction Trimming for Mining Association Rules.IEEE Transaction on Knowledge and Data Engineering,((5):813-825. October 1997.