An Efficient Algorithm for Mining Fuzzy Temporal Data

Fokrul Alom Mazarbhuiya
Department of Information Technology, College of Computer Science and IT, Albaha University, Albaha, KSA,

Abstract- *Mining patterns from fuzzy temporal data is an important data mining problem. One of these mining task is to find locally frequent sets, In most of the earlier works fuzziness was considered in the time attribute of the datasets .Although a couple of works have been done in dealing with such data, little has been done on the implementation side. In this article, we propose an efficient implementation of an algorithm for extracting locally frequent item sets from fuzzy temporal datasets. Our implementation is a Trie-based (Prefix-tree) implementation. The efficacy of the method is established with an experiment conducted on a synthetic dataset.*

Keywords- *Temporal dataset, Fuzzy temporal dataset. Frequent item set, Locally frequent item set, Core of a fuzzy number, Data mining, Fuzzy membership function, α-cut of a fuzzy number.*

I. INTRODUCTION

Association rules mining problem from datasets has been studied initially [1] by R. Agarwal *et al* for application in large datasets of super markets. Such datasets are temporal in the sense that each transaction in dataset is associated with the time of transaction. Frequent itemsets mining from such dataset is an important data mining problem.

In this paper, we consider datasets, having fuzzy time of transaction. That is the dataset is fuzzy temporal. In [2], authors proposed a technique of finding locally frequent itemsets from such dataset. In [3], authors proposed a hash tree-based implementation of the algorithm [2]. In this article, we propose an another type of implementation. The implementation discussed here is a trie-based implementation. We use here a synthetic dataset to show the efficacy of our implementation.

The paper is structured as follows: In section-II we give a brief idea on the recent works in Mining Temporal Data and Mining fuzzy temporal data. In section-III we explain the terms and notations used in this paper. In section-IV, we give the proposed algorithm [2]. In section-V, we discuss the implementation detail along with experimental results. We conclude with conclusion and lines for future work in section-VI.

II. RECENT WORKS

In [1], authors formulated the problem of association rules discovery by proposing .an algorithm of discovering association rules known as the A priori algorithm. Temporal Data Mining is an important extension of conventional data mining and has recently been able to attract more people to work in this area. Considering the time attributes, more appealing time dependent patterns can be extracted. Primarily there are two broad directions of temporal data mining [4]. One concerns the extraction of causal relationships among temporally oriented events. The other concerns the extraction of similar patterns within the same time sequence or among different time sequences. The later problem is known as sequence mining problem. In [5] the authors devised a method of recognizing frequent episodes in an event sequence. In temporal association rules each rule has associated with it a time interval in which the rule is valid. The problem is to find frequent item sets which are frequent certain valid time periods then extracting rules from such frequent item sets. In [6 7, 8, 9], the problem of temporal data mining is addressed in detail and techniques have been proposed for this. In [10] an algorithm for the discovery of temporal association rules is discussed.. In [11, 12], an efficient method for finding locally and periodically frequent sets and periodic association rules are discussed. In [2], an algorithm for mining locally frequent itemsets from fuzzy temporal data is discussed. In [3]. an implementation of [2] is given.

III. TERMS,NOTATION AND SYMBOL USED

A. Some Definitions related to Fuzziness

A fuzzy interval is in fact a fuzzy number with a flat area. We denote a fuzzy interval A is by $A = [a, b, c, d]$ where $a < b < c < d$ and $A(a) = A(d) = 0$, $A(x) = 1$ for all $x \in [b, c]$. $A(x)$ for all $x \in [a, b]$ is known as left reference function and $A(x)$ for $x \in [c, d]$ is known as the right reference function. [13].

An α-cut of the fuzzy interval $[t_1\text{-}a, t_1, t_2, t_2+a]$ is a actually a closed interval $[t_1+(\alpha\text{-}1).a, t_2+(1\text{-}\alpha).a]$.

The core of a fuzzy number A is defined as the set of elements of A having membership value one i.e.

$$\text{Core}(A) = \{(x, A(x); A(x) = 1\}$$

Any fuzzy set A, $A = \bigcup_{\alpha\in[0,1]} {}_{\alpha}A$ where ${}_{\alpha}A(x) = \alpha.\ {}^{\alpha}A(x)$ and ${}_{\alpha}A$ is a special fuzzy set. It is to be mentioned

here that a fuzzy number or fuzzy interval is actually convex and normalized fuzzy sets

For any pair of fuzzy sets A and B and for all $\alpha \in [0, 1]$, ${}^{\alpha}(A \cup B) = {}^{\alpha}A \cup {}^{\alpha}B$ and ${}^{\alpha}(A \cap B) = {}^{\alpha}A \cap {}^{\alpha}B$

The membership functions $A(x)$ of A and $B(x)$ of B are said to be similar if they satisfies the following two conditions

i) the slope of the left reference function of $A(x)$ is equal to the that of $B(x)$ and

ii) the slope of right reference of $A(x)$ is equal that of $B(x)$.

Thus for any two fuzzy numbers A and B having similar membership functions $|{}^{\alpha}A| = |{}^{\alpha}B|$, $\forall \alpha \in [0, 1]$.

B. Some Definitions related to Association Rule Mining over Fuzzy time period

Suppose that $T = <T_o, T_1, \ldots\ldots\ldots\ldots>$ is a sequence of imprecise or fuzzy time stamps over which a linear ordering $<$ is defined. We also assume that all the fuzzy time stamps have similar membership functions. Let I be a finite set of items and D, the transaction dataset is the set of transactions with the property that each transaction has two parts, one subset of the itemset I and the other fuzzy time-stamp indicating the approximate time in which the transaction had taken place. We also suppose that D is ordered in the ascending order of the core of fuzzy time stamps. A transaction is said to be in the fuzzy time interval $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$ if the α-cut of the fuzzy time stamp of the transaction is within the α-cut of $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$ for some user's specified value of α.

The local support of an itemset in a fuzzy time interval $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$ is defined as the ratio of the number of transactions in the time interval $[T_1\text{+}(\alpha\text{-}1).a, T_2\text{+}(1\text{-}\alpha).a]$ containing the itemset to the total number of transactions in $[T_1\text{+}(\alpha\text{-}1).a, T_2\text{+}(1\text{-}\alpha).a]$ for the whole dataset D for a given value of α. The notation $Sup_{[T_1-a,T_1,T_2,T_2+a]}(X)$ is used to denote the support of the itemset X in the fuzzy time interval $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$. Given a threshold σ we say that an itemset X is frequent in the fuzzy time interval $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$ if $Sup_{[T_1-a,T_1,T_2,T_2+a]}(X) \geq (\sigma/100)$* tc where tc denotes the total number of transactions in D that are in the fuzzy time interval $[T_1\text{-}a, T_1, T_2, T_2\text{+}a]$.

IV. ALGORITHM PROPOSED

A Generating Locally Frequent Sets

Before proceeding further, for the sake of convenience, we describe the algorithm proposed in [2].

While constructing locally frequent sets over fuzzy temporal data, we maintain a list of fuzzy time-intervals for each locally frequent itemset in which the itemset is frequent. For this purpose, two user's specified thresholds α and *minthd* are used. During the algorithm execution, while making a pass through the dataset, if for a particular itemset the α-cut of fuzzy time-stamp of current transaction, $[{}^{\alpha}Lcurrent, {}^{\alpha}Rcurrent]$ and the α-cut, $[{}^{\alpha}Llastappear, {}^{\alpha}Rlastappear]$ of its fuzzy time, when it was last appeared in the transaction, overlap then the current transaction is included in the current time-interval under consideration which is extended with replacement of ${}^{\alpha}Rlastappear$ by ${}^{\alpha}Rcurrent$; otherwise a new time-interval is started with ${}^{\alpha}Lcurrent$ as the start point. The support count of the itemset in the previous time interval will be checked to see whether it is frequent or not in that interval. If it is frequent then it will be fuzzified and added to the list maintained for that itemset. Also for each locally frequent itemsets over fuzzy time intervals, an user-specified minimum core length of the fuzzy time interval given as *minthd* and fuzzy time intervals having core length greater than or equal to *minthd* are only kept. If we don't use *minthd* then an item appearing once in the whole dataset will also become locally frequent over fuzzy point of time.

Method to compute L_1, the set of all locally frequent item sets of size 1.

For each item while going through the dataset, we always keep an α-cut ${}^{\alpha}lastappear$ which is $[{}^{\alpha}Llastappear, {}^{\alpha}Rlastappear]$ that corresponds to the fuzzy time stamp when the item was last appeared. When an item is found in a transaction with the fuzzy time-stamp Tm and if its α-cut ${}^{\alpha}Tm=[{}^{\alpha}LTm, {}^{\alpha}RTm]$ has empty intersection with $[{}^{\alpha}Llastappear, {}^{\alpha}Rlastappear]$, then a new time interval is started by setting *start* of this time interval as ${}^{\alpha}LTm$. The *end* of the previous time interval will be ${}^{\alpha}Rlastappear$. Then the previous time interval is fuzzified if the support of the item is greater than or equal to the user-specified minimum support. The fuzzified interval is then added to the list maintained for that item provided that the length of its core is greater than or equal to *minthd*. Otherwise ${}^{\alpha}Rlastappear$ is set to ${}^{\alpha}RTm$, the counters for counting transactions are increased suitably and the process will be continued.

In below, we give the pseudo code for the algorithm to compute L_1, the list of locally frequent sets of size 1. We assume that the number of items in the dataset under consideration is n and there is an ordering among the items.

Algorithm 1

$C_1 = \{(i[k], tp[k]) : k = 1,2,\ldots,n\}$

where i[k]= the k-th item and tp[k] points to a list of fuzzy time intervals initially empty.}

for(k = 1; k≤n;k++) do

set ${}^{\alpha}lastappear[k]=\phi$;

for each transactions t in the dataset with fuzzy time stamp Tm

initially icount[k]==0; tcount[k]==0;

do

{for(k = 1; k≤n;k++) do

{ if {i[k]} ⊆ t then

```
  { if(^α lastappear[k] == ϕ)
   {^α lastappear[k] = ^α firstappear[k] = ^α Tm;
   icount[k] = tcount[k] = 1;
    }
 else
 if([^α Llastappear[k],^α Rlastappear[k]]∩ [^α LTm[k], ^α RTm[k]]!=ϕ)
    {^α Rlastappear[k]=^α RTm[k];  icount[k]++;
tcount[k]++;
   }
else
{ if (icount[k]/tcount[k]*100 ≥ σ)
 fuzzify([^α Llastappear[k],^α Rlastappear[k]],∀α∈[0, 1])
  if(|core(fuzzified interval)|≥ minthd)
   add(fuzzified interval) to tp[k];
   icount[k] = tcount[k] = 1;
   lastappear[k] = firstappear[k] = Tm;
  }
 }
 else tcount[k]++;   }
} // end of do loop //
for( k = 1; k≤ n; k++)
{ if (icount[k]/tcount[k]*100≥σ)
  fuzzify([^α Llastappear[k], ^α Rlastappear[k]], ∀α∈[0, 1])
   if(|core(fuzzified interval)|≥ minthd)
  add(fuzzified interval) to tp[k];
  if(tp[k] != 0) add {i[k], tp[k]} to L1
   }
 fuzzify([^α a1,^α a2], α)
   { fuzzified interval= Y_{α∈[0,1]} α[ a1, a2 ];
     where α[a1,a2](x) = α. ^α[a1, a2](x)
     return(fuzzified interval)
   }
```

Two support counts are kept, *icount* and *tcount*. If the count percentage of an item in an α-cut of a fuzzy time interval is greater than or equal to the user-specified minimum support only then the set is considered as a locally frequent set over the fuzzy time interval.

L_1 as computed above will have all 1-sized locally frequent sets over fuzzy time intervals and with each itemset there will be an ordered list of fuzzy time intervals in which the itemset is frequent. Then we apply the A priori candidate generation algorithm to find candidate frequent itemset of size-2. With each candidate frequent itemset of size-2, we associate a list of fuzzy time intervals obtained by pruning phase. In the generation phase this list is empty. If all subsets of a candidate item set are found to be frequent in the previous level then this set will be constructed. The process is as follows: when the first subset appearing in the previous level is found then the list of this subset is taken as the list of fuzzy time intervals associated with the set. When subsequent subsets are found then the list of fuzzy time intervals is reconstructed by taking all possible pair wise intersection of the fuzzy time intervals one from each list. Itemsets for which this list is found to be empty are further pruned.

Using this concept we express below the modified A-priori algorithm for the problem under consideration.

Algorithm 2
Modified A priori
Initialize
 k = 1;
C_1 = { itemsets of size- 1}
L_1 = {frequent itemsets of size 1 where
with every itemset {i[k]} a list tp[k] is maintained which gives all time fuzzy
intervals in which i[k] is frequent}
/ L_1 is computed using algorithm 1 */*
for(k = 2; $L_{k-1} \neq \phi$; k++) do
{ C_k = apriorigen(L_{k-1})
/ same as A priori candidate generation algorithm setting tp[i] to zero ∀ i*/*
prune(C_k);
drop all lists of fuzzy time intervals maintained with the itemsets in C_k
Compute L_k from C_k.
/L_k can be computed from C_k using the similar method used for L_1* /*
k = k + 1
}

$$Answer = \bigcup_k L_k$$

Prune(C_k)
{Let m be the number of itemsets in C_k and let the sets be s[1], s[2],…, s[m]. Initialize the pointers tp[i] for each s[i] to null
for(i = 1; i ≤ m; i++) do
{for each (k-1) subset a of s[i] do
{if a ∉ L_{k-1} then
{C_k = C_k - {s[i],tp[i]}; break;}
else
{ if (tp[i] == null) then set tp[i] to point to the list of fuzzy time intervals maintained for a
else
{ take all possible pair-wise intersection of fuzzy time intervals one from each list, one list maintained with tp[i] and the other maintained with a and take this as the list for tp[i]
delete all fuzzy time intervals whose core length is less than the value of minthd
if tp[i] is empty then {C_k = C_k - {s[i],tp[i]};
break;
}
}
}
}
}
}

V. IMPLEMENTATION

A. *Data structure used*

Candidate generation, pruning and support count requires an competent data structures in which all candidates are stored. In general, two nice data structures have been used for this purpose namely hash-tree and trie data structure. In this implementation, we have used prefix-tree (Trie) data structure.

1. Trie data structures

All the items are marked as a1, a2, ….a*n* where *n* = total number of items in the dataset. We also assumed that the items in the transactions are ordered in ascending. In a *trie* data structure, every *k*-itemset has a node linked with it, as does its (*k*-1)-prefix. The empty itemset is the *root node*. All the itemsets of size-1, are attached to the *root node* as its children. Every other itemset of size-k, is attached to its (*k*-1)-prefix. Each node represents an itemset. Each node is storing the last item in the itemset it represent, a pointer to its *childlist*, a pointer to its parent, a pointer time intervals list where the itemset is frequent and a pointer to its *right sibling*. The *siblings* of every node are

implemented as *linked list*. So, each level consists of a set of lists. In our implementation, we maintain all the list in a level as a list of lists. The level-1 is a single list consisting of all the itemsets of size-1. At *k*th iteration, all the candidate itemsets of size-k are stored at depth *k* in the *trie*. To count the supports of itemsets in a particular level, the itemsets represented by the nodes are found by moving upwards towards the *root* using parent pointers. Also in candidate generation procedure the join step becomes very simple. As all itemsets of size-k with the same (*k*-1)-prefix are represented as a *linked list*, to create all candidate k-size itemsets with (*k*-1)-prefix *X* we merely copy all *right siblings* of the node representing *X* and add them as child of *X*. Candidate generation method also computes pair-wise intersection of the fuzzy time intervals lists associated with the two itemsets that are merged to get the candidate. If intersection of the time intervals lists is found to be empty or the core length of fuzzified intervals in the list is found to be less than or equal to *minthd* then the newly added node is deleted. The method also includes *A-priori* pruning, which checks whether all the subsets of the candidates are present in the previous level or not.

B Analysis of Results

For experimented conducted in the paper, we used a synthetic dataset which is available at http://fimi.cs.helsinki.fi/testdata.html. As the dataset does not have fuzzy time contents, we incorporate the fuzzy time on it. We consider the different transactions sizes like 20,000, 40,000,60,000, 80,000, 100,000 and execute the algorithm. We keep the life span of dataset as one year (2012). The results obtained by the method are given in table1 and figue1.

TABLE I
FREQUENT ITEMSETS EXTRACTED BY THE METHOD [2]

Transaction sizes	Number frequent itemsets
20,000	2
40,000	2
60,000	4
80,000	5
100,000	9

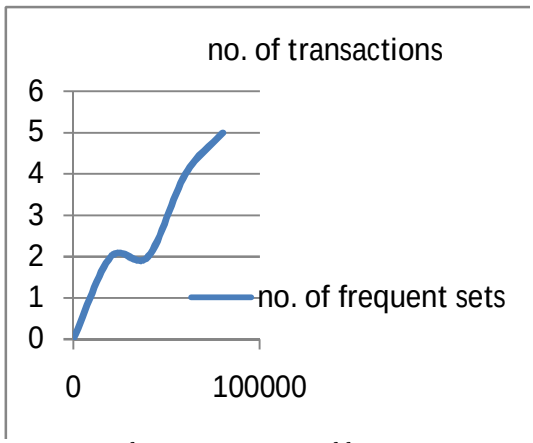

Fig.1: no. of transactions vs. no. of frequent itemsets

VI CONCLUSION

An implementation of the algorithm [2] is given in this paper, The algorithm is used for finding frequent itemsets which are frequent in certain fuzzy time periods. The implementation discussed here is Trie-based implementation. The effectiveness of the implementation is given with the experiment conducted on a synthetic data available in internet. As the dataset does not have fuzzy temporal features, we incorporate the fuzzy time stamp on it to make it appropriate for our experiment. In future, we will try to implement the same algorithm using other type of data structures PF-tree based implementation and compare with existing implementations.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. Swami; Mining association rules between sets of items in large databases; Proceedings of the ACM SIGMOD '93, Washington, USA (May 1993).

[2] F. A Mazharbhuiya, M. Shenify and Mohammed Husamuddin, Finding Local and Periodic Association Rules from Fuzzy Temporal Data, *The 2014 International Conference on Advances in Big Data Analytics* July 21-24, 2014, Las Vegas, Nevada, USA.

[3] F. A. Mazarbhuiya (2016); Mining local patterns from fuzzy temporal data, International Journal of Engineering and Applied Sciences (IJEAS), ISSN: 2394-3661, Volume-1, Issue-1, January 2016, INDIA, pp. 70-73.

[4] J. F. Roddick, M. Spillopoulou; A Biblography of Temporal, Spatial and Spatio-Temporal Data Mining Research; ACM SIGKDD (June 1999).

[5] H. Manilla, H. Toivonen and I. Verkamo; Discovering frequent episodes in sequences; KDD'95; AAAI, 210-215 (August 1995).

[6] J. M. Ale and G.H. Rossi; An approach to discovering temporal association rules; Proceedings of the 2000 ACM symposium on Applied Computing (March 2000).

[7] X. Chen and I. Petrounias; A framework for Temporal Data Mining; Proceedings of the 9th International Conference on Databases and Expert Systems Applications, DEXA '98, Vienna, Austria. Springer-Verlag, Berlin; Lecture Notes in Computer Science 1460 (1998), 796-805.

[8] X. Chen and I. Petrounias; Language support for Temporal Data Mining; Proceedings of 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, PKDD '98, Springer Verlag, Berlin (1998), 282-290.

[9] X. Chen, I. Petrounias and H. Healthfield; Discovering temporal Association rules in temporal databases; Proceedings of IADT'98 (International Workshop on Issues and Applications of Database Technology (1998), 312-319.

[10] J. M. Ale, and G. H. Rossi; An Approach to Discovering Temporal Association Rules, *In Proc. of 2000 ACM symposium on Applied Computing* (2000).

[11] A. K. Mahanta, F. A. Mazarbhuiya and H. K. Baruah; Finding Locally and Periodically Frequent Sets and Periodic Association Rules, Proceeding of 1st Int'l Conf on Pattern Recognition and Machine Intelligence (PreMI'05),LNCS 3776 (2005), 576-582.

[12] F. A. Mazarbhuiya Yusuf Pervaiz (2015); An Efficient Method for Generating Local Association Rules, International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA Volume 9 – No.2, June 2015.

[13] D. Dubois and H. Prade; Ranking fuzzy numbers in the setting of possibility theory, *Information Science* 30(1983), 183-224.

Author's Profile

Fokrul Alom Mazarbhuiya received B.Sc. degree in Mathematics from Assam University, India and M.Sc. degree in Mathematics from Aligarh Muslim University, India. After this he obtained his Ph.D. degree in Computer Science from Gauhati University, India. He had been serving as an Assistant Professor in College of Computer Science and Information Systems, King Khalid University, Abha, kingdom of Saudi Arabia from 2008 to 2011. Currently, he is working as an Assistant Professor, Information Technology, College of Computer Science aind Information Technology, Al Baha University, Al Baha, Kingdom of Saudi Arabia. His research interest includes Data Mining, Information security, Fuzzy Mathematics and Fuzzy logic.