Mining Sequential Patterns from Super Market Datasets

Fokrul Alom Mazarbhuiya
College of Computer Science & IT
Albaha University, Albaha, KSA

Abstract: *Mining sequential patterns is an important data-mining problem and it has many application domains such as Supermarket Medical science, signal processing and speech analysis. The problem involves mining causal relationship between events. Mining sequence from supermarket is an interesting data mining problem. In this paper, we propose a method of mining such patterns. Our approach is completely different from others in the sense that we are interested to find inter-item sets patterns however in other cases patterns are inter-transactions. In our case we first find all frequent itemsets where each frequent itemsets is associated with the lists of time intervals in which it is frequent. Sequential patterns can be generated using the lists of time intervals associated with frequent itemsets. The efficacy of the method is established using experimental results*

Keywords: *Locally frequent itemsets, Temporal data mining, Frequent sequence, Maximal frequent sequence.*

I INTRODUCTION

Data mining, also is recognized as a important area for database researchers. It can be defined as efficiently finding interesting rules from large databases. Temporal Data Mining has recently been able to attract more people to work in this area. There are mainly two extensive directions of temporal data mining [1]. One involves the discovery of causal relationships among events which are temporally oriented .and others concerns the discovery of similar patterns within the same time sequence or among different time sequences. The underlying problem is to discover frequent sequential patterns in the temporal databases. which is termed as sequence mining.. In [2], the author has put forwarded the method discovering sequential pattern. In [3] the authors have discussed the problem of recognizing frequent episodes in an event sequence. In [4], authors present, an efficient algorithm for mining frequent sequences considering a variety of syntactic constraints in the form of length or width limitations on the sequences, minimum or maximum gap constraints, on the consecutive sequence elements. In [5], authors analyze the large sequence using Apriori All_Set algorithm.

In [6] *Agrawal et al* formulated the problem of mining frequent itemset from supermarket data..But usually supermarket data is temporal in the sense that each transaction in supermarket is associated with the time of transaction. In [7], author wrote a survey paper on weighted frequent mining. In [8], *Ale et al* proposed a method of extracting association rules from supermarket data which are termed as temporal association rules. These rules hold throughout the life-time of an item set where the life-time of an item set is defined as the time period between the first transaction and last transaction containing the item set (life-time of an item set may not be same as that of the dataset). Considering the time gap between two consecutive transactions containing items, a method is proposed [9, 10], which dynamically extracts all the frequent item sets along with the time period where the item sets are frequent. The algorithm described in [9, 10] extracts for each frequent item set a sequence of time intervals in which the item set is frequent and such frequent set are called as locally frequent sets. Problem of extracting patterns from lists of time intervals associated with different locally frequent item sets can be an interesting pattern-mining problem. For example the time intervals of two different item sets may be in some sequence i.e. the time period of one frequent item set starts just before that of another frequent item sets. Such type of patterns can provide a valuable information regarding inter-relation between different frequent item sets. In this paper we have devoted our studies made in this regard. In [11], authors devised a method of extracting such patters. However the work were mostly theoretical In this paper we use the algorithm [11] to extract patterns among different locally frequent item sets based on their interval lists. The proposed algorithm takes input as the output of algorithm [9, 10].

The paper is organized as follows. In section-II, we briefly discuss about related works. In section-III, we discuss about definitions and notations. In section-IV, we discuss about the proposed algorithm. In section-V, we have discussed about experiment conducted in this regard and in section-VI, conclusion is given.

II RELATED WORKS

The problem of mining sequences has been studied in [2, 4, 5, 12, 13]. The problem was put forwarded by *Agrawal* and *Srikant* [2] in 1995. In [13], an algorithm called GSP is proposed which considers minimum and maximum gaps, as well as sliding windows and it is shown empirically shown that GSP is 20 times faster than others. However in all the above cases, the patterns are inter-transactions.

Somewhat related to this work is the problem of mining association rules [6]. Since the problem is originated from supermarket to study the customer's behaviour with the help of transaction data, it is also *market basket problem*. Association rules are rules about what items are bought together within a transaction, and are thus intra-transaction patterns, unlike inter-transaction sequential patterns. The association rule mining includes frequent item set mining problem, which is a well-researched area of pattern mining. In [14], an algorithm is proposed to extract frequent item sets, which is called as *A-priori* algorithm.

Temporal association rule mining is an important extension of traditional association rule mining and is able to attract the attention of researchers. Considering the time aspect of datasets some more association rules can be extracted which otherwise cannot be extracted. In [8], *Ale et al* has proposed a method which considers the lifetime of an item sets. The lifetime of an item set is the time period between the first transaction containing the item set and last transaction containing the same and it is not necessarily equal to the lifetime of the whole dataset. The method proposed by *Ale et al* extracts much more rules than the traditional rule-mining algorithm. However the algorithm cannot extract various types of patterns that may exist in a temporal dataset viz. calendric, cyclic etc. In [3], [14], methods for extracting such patterns are discussed where the period is to be specified by user. In [9, 10], a method is proposed which can extracts various types of frequent item sets that may be present in the datasets. The algorithm [9, 10] dynamically extracts all the frequent item sets along with lists of time intervals and each frequent item set is associated with a list of time intervals where it is frequent. These item sets are called locally frequent item sets. In this paper, we propose an algorithm for extracting sequential patterns from the locally frequent item sets based on their associated lists of time intervals. So our study is basically inter-item sets study. The input to this algorithm is the output of the algorithm [9, 10]. Thus our method is a two-phase method. In the first phase the algorithm [9, 10] is used to extract all the locally frequent item sets. Next the time intervals associated with these locally frequent item sets are used to extract frequent sequences where the frequency of a sequence is defined in terms of corresponding interval lists associated with locally frequent item sets with which the sequence is formed. The algorithm discuss here is a level-wise algorithm. Since the algorithm [9, 10], extracts more frequent item sets than other known algorithms. Our method is likely to extract more sequential patterns than others. As we are extracting sequential patterns from locally frequent item sets, which are extracted from temporal dataset, the maximum gap, minimum gap and sliding windows can be incorporated here. Only we have to modify the definition of support of the sequences accordingly.

III DEFINITIONS AND NOTATIONS

(A) Old Definitions and Notations related to sequence mining

In below we discuss some of the definitions and notations related to sequence mining problem, which are available in the literature.

Let $\sum = \{i_1, i_2, \ldots i_m\}$ be a set of m distinct items comprising the alphabet.

An event is a non-empty, disordered collection of items. The items in an event are in some predefined order. An event is denoted as $(i_1, i_2, \ldots i_k)$, where i_j is an item in $\sum$.

Any event that is given as input is called a transaction. Thus, transactions and events have same structure, except that a transaction is known prior to the process and an event is generated during the algorithm. In below we review some definitions related sequence mining.

Definition1 Sequence

A *sequence* is an ordered list of events. A sequence s is denoted as $(\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_q)$, where α_i is an event. A sequence is called k-sequence, if the sum of the cardinalities α_i is k.

A *subsequence* is a sequence within a sequence, preserving the order. In the other words, its items need not be adjacent in time but their ordering in a sequence should not violate the time ordering of the supporting events. A subsequence can be obtained from a sequence by deleting some items and/or events.

A sequence s' is said to support another sequence s if s is a subsequence of s'.

Let D be the dataset of input sequences and a sequence is a set of temporally ordered transactions.

Definition2 Frequency of a sequence

The *frequency* of a sequence s with respect to this dataset D, is the total number of input sequences in D that support it.

Definition.3 Maximal Sequence

A *frequent sequence* is a sequence whose frequency exceeds some user-specified *threshold*. A *frequent sequence* is said to be maximal if it is not a subsequence of another *frequent sequence*.

The rationale behind *frequent sequences* lies in detecting precedence and causal relationships that make them statistically remarkable.

(A) New Definitions and Notations used in the Algorithm

Let $F_1 = \{A_1, A_2, \ldots, A_m\}$ be a set of locally frequent item sets obtained by using algorithm discussed in [1], where each A_i has an associated list of time-stamps.

Definition4 Sequence

We define a sequence of frequent sets as $s = (A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_p)$ where each A_i is locally frequent item set obtained by the method discussed in [1]. We call a sequence of size-p or p-sequence if the length of the sequence is p. So obviously the sequences of size-1 are nothing but the locally frequent sets themselves.

Definition5 Subsequence

We define a subsequence of a sequence having size less than the size of the sequence and having same ordering. So deleting some item sets from a sequence can form a subsequence of the sequence.

Definition6 Frequency of a Sequence

Let A_i and A_j be two locally frequent sets. From the sequence of time-stamps associated with the two item sets A_i and A_j, we extract two subsequences of time stamps LA_i and LA_j of same size satisfying the following conditions:

Let $st_k[LA_i]$ denote the time-stamp of the k-th interval in the subsequence LA_i, i, k being positive integers.

i) $st_k[LA_i] \leq st_k[LA_j]$ for k=1, 2,…l where l is the length of LA_i

ii) $st_{k+1}[LA_i] \geq st_k[LA_j]$ for k=1, 2,…l-1 and

iii) LA_i and LA_j are maximal sequences satisfying i) and ii)

Then frequency of $A_i \rightarrow A_j$ is defined as the ratio between the $Supp(A_i \rightarrow A_j)$ to the $Supp(A_i)$, where

$Supp(A_i \rightarrow A_j)$ = number of time-stamps $st_k[LA_i]$ satisfying the conditions i), ii) and iii)

$Supp(A_i)$ = number of time-stamps associated with A_i.

Thus the frequency of $(A_i \rightarrow A_j)$ is given by

$$fr(A_i \rightarrow A_j) = Supp(A_i \rightarrow A_j) \,/\, Supp(A_i)$$

In general, we define the frequency of a sequence $A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p$ as follows: Let $\{LA_i\}$;i=1,2,..,p be the subsequences of time-stamps of same size extracted from the sequence of time-stamps associated with the locally frequent sets $\{A_i\}$;i=1,2,…,p that satisfies the following conditions:

Let $st_k[LA_i]$ denote the time-stamp of the k-th interval LA_i k being a positive integer and i=1,2,….p.

1) $st_k[LA_1] \leq st_k[LA_2] \leq \ldots \leq st_k[LA_p]$ for k=1, 2,…l where l is the length of LA_1

2) $st_{k+1}[LA_1] \geq st_k[LA_p]$ for k=1, 2,…,l-1 and

3) $\{LA_i\}$; i=1,2,…,p are maximal sequences satisfying 1) and 2)

Then frequency of $A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p$ is defined as the ratio between the $Supp(A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p)$ to the $Supp(A_1)$, where

$Supp(A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_p)$ = length of LA_1 satisfying the conditions 1) and 2)

$Supp(A_1)$ = length of the sequence of time-stamps i.e. the number of time-stamps associated with A_1.

Thus $fr(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p) = Supp(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p) \,/\, Supp(A_1)$

Definition7 Frequent Sequence

A sequence of item sets is said to be frequent if its frequency exceeds some user-specified threshold σ.

Definition8 Maximal Frequent Sequence

A sequence $s = (A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p)$ of item sets is said to be maximal sequence if it cannot be enlarged or it is not a subsequence of any other frequent sequence.

Theorem 1

A sequence $s = (A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n)$ of item sets is frequent then any of its subsequence of size-(n-1) starting with A_1 is frequent.

Proof: Suppose that sequence s = $(A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_n)$ of item sets is frequent.

$fr(A_1 \rightarrow A_2 \ldots \rightarrow A_n) \geq \sigma$, for a given σ, $0 \leq \sigma \leq 1$

$$\Rightarrow \frac{Supp(A_1 \rightarrow A_2 .... \rightarrow A_n)}{Supp(A_1)} \geq \sigma \quad \ldots\ldots (1)$$

Now according to the definition of $Supp(A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_n)$, we can extract n subsequences of time-stamps say LA_1, LA_2, ... LA_n of equal length say l where LA_i denotes the list time-stamps associated with the item set A_i for i=1,2,…n where these are subsequences of the lists of starting time-stamps associated with the item sets A_1, A_2,…A_n respectively. The subsequences satisfy the following three conditions:

(i) $st_k[LA_i] \le st_k[LA_{i+1}]$ for i=1,2,…,n-1 and k=1, 2,…l.

(ii) $st_{k+1}[LA_i] \ge st_{k+1}[LA_l]$ for i=1,2,…,n-1 and k=1,2,….l-1

(iii) the sequences are maximal sequences satisfying (i) and (ii)

where $st_k[LA_i]$ denotes the k-th time stamp of the list LA_i.

Now let us consider any subsequences $B_1 \to B_2 \to \ldots \to B_m$ of $A_1 \to A_2 \to A_3 \to \ldots \to A_n$ starting with A_1 i.e. $B_1 = A_1$.

Now since each B_i for i=2,3,…m is some A_j for j=2,3,…n, we associate with each B_i a list of time-stamps say LB_i where LB_i is LA_j if B_i is A_j. Now it follows from above that the $B_1 \to B_2 \to \ldots \to B_m$ satisfy the following two conditions:

(i) $st_k[LB_i] \le st_k[LB_{i+1}]$ for i=1,2,…,m-1 and k=1,2,…l

(ii) $st_{k+1}[LB_i] \ge st_k[LB_m]$ for i=1,2,…m-1 and k=1,2,…l-1

This implies that $Supp(B_1 \to B_2 \to B_3 \ldots \to B_{n-1}) \ge Supp(A_1 \to A_2 \to A_3 \ldots \to A_n)$

$$\Rightarrow \frac{Supp(B_1 \to B_2 \to \ldots \to B_{n-1})}{Supp(B_1)} \ge \frac{Supp(A_1 \to A_2 \to \ldots \to A_n)}{Supp(A_1)}$$

$\ge \sigma$ [using (1)]

since $Supp(A_1) = Supp(B_1)$

$\Rightarrow fr(B_1 \to B_2 \to B_3 \ldots \to B_{n-1}) \ge \sigma$

$\Rightarrow s' = (B_1 \to B_2 \to B_3 \ldots \to B_{n-1})$ is a frequent sequence of item sets.

Hence every subsequence of a frequent sequence having same starting item set is frequent.

IV ALGORITHM PROPOSED

Here for the sake of convenience we discussed the algorithm [11]. The algorithm is a level-wise algorithm. In the first level, we will consider all the locally frequent item sets generated by the algorithm [9] and these are our frequent 1-sequences and we denote the set by F_1. Each frequent set in F_1 is associated with a list of time-stamps. Considering all possible pairs of disjoint frequent sets from the 1-sequences we generate a set of candidate 2-sequences. Then the frequency of every candidate is computed going though the list of time-stamps maintained for frequent 1-sequences and using the *Definition3.2.3*. Those candidates having frequency greater than the *min_threshold* σ are kept leaving others. These will be the set of frequent sequence in the second level we denote these as F_2. From F_2, we can generate candidates for third level as follows: If $A_1 \to A_2$ and $A_1 \to A_3$ are two members of F_2 then the corresponding candidates for third level will be $A_1 \to A_2 \to A_3$ and $A_1 \to A_3 \to A_2$. Then the frequency of each candidate is computed and compared with *min_threshold* σ to get F_3. From F_3 we can get candidates for fourth level as follows: If $A_1 \to A_2 \to A_3$ and $A_1 \to A_2 \to A_4$ are any two frequent 3-sequences, then the corresponding candidates for fourth level are $A_1 \to A_2 \to A_3 \to A_4$ and $A_1 \to A_2 \to A_4 \to A_3$. Before going to compute the frequency of each candidate we must ensure that all its subsequences of size one less than it having same starting item set are frequent. For example, for the sequence $A_1 \to A_2 \to A_3 \to A_4$ to be frequent, the 3-sequences $A_1 \to A_2 \to A_3$, $A_1 \to A_2 \to A_4$ and $A_1 \to A_3 \to A_4$ must belong to F_3. Thus pruning step actually begins at fourth level.

In general, from i-th level frequent sequences we can generate (i+1)-th level candidates in the following way: For any two frequent i-sequences $s_1 = (A_1 \to A_2 \to \ldots \to A_i)$ and $s_2 = (B_1 \to B_2 \to \ldots \to B_i)$ such that $A_1 = B_1$, $A_2 = B_2$, … $A_{i-1} = B_{i-1}$ and $A_i \ne B_i$, $(A_1 \to A_2 \to \ldots \to A_i \to B_i)$ and $(A_1 \to A_2 \to \ldots \to B_i \to A_i)$ will be two candidates for (i+1)-th level. In this way we generate all the candidates. Then the set of candidates is pruned. Then the frequency of each of the remaining candidates is computed and compared with *min_threshold* σ to get F_{i+1}. The process continues till no candidate is generated or a particular level becomes empty. The algorithm for sequence mining is given below.

Algorithm 1

Algorithm to find frequent 2-sequences of the type A_2 is followed by A_1 i.e. $A_1 \to A_2$

```
{lA1 = list of time-stamps associated with the item set A1.
 lA2 = list of time-stamps associated with the item set A2.
 llA1 = llA2 = null; count = 0;
 lpA1 = lA1.get(); // lpA1 will now point to the first node of lA1.
 lpA2= lA2.get(); // lpA2 will now point to the first node of lA2.
 while(lpA1 != null && lpA2 != null) do
  { t1= lpA1→ts();
    t2 = lpA2→ts();
      if(t1<t2)
          {if((temp = lpA1.get())!=null)
              if(temp→ts() >= t2)
                {count++; lpA1= temp;
                 lpA2 = lA2.get();
                 llA1.append(t1);
               llA2.append(t2);
                 continue;
               }
           else
              {lpA1 = temp; continue;}
       else
```

```
                {count++;   lpA1   =   temp;
        llA1.append(t1);
                  llA2.append(t2); continue;}
                }
            else
                lpA2 = lA2.get();
        }
    }
```

This algorithm will give the value of the $Supp(A_1 \rightarrow A_2)$ and it is divided by $Supp(A_1)$ to get $fr(A_1 \rightarrow A_2)$ which is required to find whether a sequence $(A_1 \rightarrow A_2)$ is frequent or not.

The algorithm also produces two lists of time-stamps *llA1* and *llA2* of length *count* from the lists *lA1* and *lA2* such that $(llA1)_i <= (llA2)_i$ for i=1,2,…*count*, where $(llA1)_i$ denotes the *i*-th time-stamp of *llA* and similarly $(llA2)_i$ has the same meaning and $(llA1)_{i+1} >= (llA2)_i$ for i=1,2,….*count*-1.

Algorithm to find Support of Sequences of any length

Associated with each frequent sequence of the form $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$ we have *k* number of lists of time-stamps satisfying 1), 2) and 3) of *Definition3.2.3*. While implementing the level-wise procedure, we maintain the two lists associated with the two extreme item sets A_1 and A_k. These two lists are needed for counting supports of candidate sequences in the next level, which are formed by joining this sequence with some other sequence of the same size.

Algorithm 2

Algorithm to compute support of (k+1)-sequences of the form $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{k+1}$ obtained by joining the two sequences $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{k-1} \rightarrow A_k$ and $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{k-1} \rightarrow A_{k'}$ with A_{k+1} being $A_{k'}$.

```
  {lA1←list of time-stamps associated with A1
  for the k-sequence A1→A2→....→Ak
    lAk←list of time-stamps associated with Ak
    for the k-sequence A1→A2→....→Ak
    lAkd←list of time-stamps associated with
    the 1-sequence Ak'
    count = 0
    t1=  lA1.get();  tk=  lAk.get();  tkd  =
    lAkd.get();
    lA1n = lAkp1= null;
    while((t1 != null)&&(tk != null)&&(tkd !=
    null)) do
            {if (tk->ts() <  tkd->ts())
              {if ((temp = lA1.get()) != null)
                {if (temp->ts() ≥ tkd->ts())
                    {count++;
                      lA1n.append(t1-
>ts());
                       lAkp1.append(tkd-
>ts());
                        tkd = lAkd.get();
                        t1 = temp;
                        tk = lAk.get();
                          continue;
                    }
                else
                { t1=temp;tk=  lAk.get();
continue;}
                }
              else
                {count++;  lA1n.append(t1-
>ts());
                  lAkp1.append(tkd->ts());
                  t1 = temp; continue;}
            }
          else
            tkd = lAkd.get();
      }
    }
```

The function *ts()* returns the time-stamp of the corresponding node in the list.

Algorithm 3

```
F2= the set of frequent 2-sequence obtained
using the Algorithm1.
k =3
do while Fk-1 ≠ ∅
    Fk:= ∅
    Ck:= gen_candidate_sequences(Fk-1)
    prune(Ck)
    for all candidates sk in Ck // sk is any
sequence of size-k
        if fr(sk) ≥σ  / /fr(sk) is computed
using the Algorithm2
            Fk:= Fk∪{ sk }
    k =k+1
end do
Answer:=∪ Fk
```

Candidate sequence generation with the given F_{k-1}

```
gen_candidate_sequences(Fk-1)
Ck:= ∅
  for all (k-1)-sequences sk-1∈ Fk-1
   for all (k-1)-sequences s'k-1∈ Fk-1
    ifsk-1[1]=s'k-1[1]∧sk-1[2]=s'k-1[2]∧...∧sk-
1[k-2]=s'k-1[k-2]∧sk-1[k-1]≠s'k-1[k-1]
      then sk =(sk-1[1] →sk-1[2] ... sk-1[k-2] →
sk-1[k-1]→ s'k-1[k-1])
        and s'k =(sk-1[1] →sk-1[2] ... sk-1[k-
2] → s'k-1[k-1]→ sk-1[k-1])
Ck:= Ck∪{ sk, s'k}
```

Pruning

```
prune(Ck)
   for all sk∈ Ck
```

for all (k-1)-subsequences s_{k-1} of s_k having same starting item set
do
if $s_{k-1} \notin F_{k-1}$
then $C_k := C_k \setminus \{s_k\}$

The algorithm gives all the frequent sequences along with the extracted lists of time-stamps.

5 EXPERIMENT CONDUCTED

(A) Dataset used

For experimental purpose, we have used a synthetic dataset T10I4D100K, available from FIMI[1] website. A summarized view of the dataset is presented in table 1 ..

TABLE 1. T10I4D100K DATASET CHARACTERISTICS

Dataset	#Items	#Transactions	Min|T|	max|T|	Avg|T|
T10I4D100K	**942**	**100 000**	**4**	**77**	**39**

Since the dataset is non-temporal we incorporate the temporal features on the dataset and execute the algorithm for the different sizes of transactions e. g. 10,000, 20,000, 30,000, 40,000, 50,000 and 100,000. The algorithm [9] is executed to extract all the locally frequent item sets along with their lists of time intervals. Then these locally frequent item sets are used as input for the algorithm discussed in this paper to extract sequential patterns.

(A) Patterns Obtained

Some of the sequential patterns obtained are given below for the different sizes of datasets:

TABLE 2: NUMBER OF FREQUENT SEQUENCES

10,000	2
20,000	2
30,000	4
40,000	5
50,000	5
100,000	8

VI CONCLUSION

In this paper, we mainly focused on extracting patterns from the locally frequent item sets based on the associated interval-lists obtained by the method discussed in [9]. Our method is two-phased method in the first phase we use the algorithm [9] to extract all the locally frequent item sets along with their intervals lists. In the second phase the item sets obtained in first phase, are fed to the algorithm discussed in this paper to extract sequential patterns among the item sets. The algorithm is a level-wise. In the first level we have all locally frequent item sets obtained by the algorithm [9], these are our sequences of size-1 or 1-sequnces. Considering all possible pairs of disjoint frequent item sets from the 1-sequences, candidate 2-sequnces are obtained. Then the frequency of every candidate is computed going through their lists of time intervals and computing supports (Definition of support of a sequence is given in section-3). Those candidates having supports greater than minimum threshold are frequent 2-sequnces. The process continues till no candidate is generated or a particular level is empty. It is to be mentioned here that the sequential patterns obtained here are not inter-transactions patterns rather they are inter-item sets patterns. We also report some of the experimental results in tabular form.

Lines for future works are as follows:

i) In future it can be looked for approaches other than level-wise approach.
ii) In future minimum gap, maximum gap and sliding windows can be incorporated.

REFERENCES

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

[2] R. Agrawal, and R. Srikant; Mining sequential patterns, *In Proc. of 11th Int'l Conf. on Data Engineering*, IEEE 1995, pp.3-14.

[3] H. Manilla, H. Toivonen and I. Verkamo: Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery: An International Journal* 1(3), (1997). pp. 259-289.

[4] M. J. Zaki; Efficient enumeration of frequent sequences, *In 7th Int'l Conf. on Information and Knowledge Management*, (Nov'1998).

[5] S. Mahajan, P. Pawar and S. Reshamwala; Analysis of Large Web Sequences using AprioriAll_Set algorithm, International Journal of Emerging Trends and Technology in Computer Science ISSN 2278-6856, Vol. 3(2), 2014, pp. 292-296.

[6] R. Agrawal, T. Imielinski and A. N. Swami, Mining association rules between sets of items in large databases, *In Proc. of 1993 ACM SIGMOD Int'l Conf on Management of Data*, Vol. 22(2) of SIGMOD Records, ACM Press, (1993), pp 207-216

[7] A Mohan and R. Visakh; Survey on Weighted Frequent Mining, International Journal of Computer Trends and Technology (IJCTT) – volume 9 number 3– Mar 2014.

[8] J. M. Ale and G. H. Rossi; An approach to discovering temporal association rules, *In Proc. of 2000 ACM symposium on Applied Computing* (2000).

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

[10] A. K. Mahanta, F. A. Mazarbhuiya, And H. K. Baruah (2008). Finding Calendar-based Periodic Patterns, *Pattern Recognition Letters, Vol.29(9), Elsevier publication, USA*, pp. 1274-1284.

[11] F. A. Mazarbhuiya, A. K. Mahanta; Finding Sequential Patterns from Temporal Datasets, Proceedings of The 2010 International Conference on Data Mining (DMIN'2010), pp. 386-391.
[12] M. J. Zaki; Efficient enumeration of frequent sequences, *In 7th Int'l Conf. on Information and Knowledge Management*, (Nov'1998).
[13] Srikant and R. Agrawal; Mining Sequential Patterns: Generalization and Performance Improvements, *In Proc. of 5th Int'l Conf. on Extending Database Technology*, (EDBT'96), (March'1996).
[14] R. Agrawal and R. Srikant; Fast Algorithms for Mining Association Rules, *In Proc. of the 20th VLDB Conf.*, Santiago, Chile, (1994).