SBLOCK – A Closed Sequential Pattern Mining Algorithm

Kailash C Kandpal[1], Rahul Agnihotri[2]

JJT University, Rajasthan

[1]kailashc.kandpal@gmail.com, [2]agnihotrirahul2006@gmail.com

***Abstract* - Previous studies said that the closed frequent patterns in comparison to frequent pattern mining leads to not only more compact yet complete result set but also better efficiency. Most of previous algorithm for closed sequential pattern mining is manly concentrated over candidate's generation and maintenance which takes much time and cost. SBLOCK is a an algorithm based on the concepts of memory blocks, This algorithm works on unique occurrences of items in a sequential database, with a notion that a bigger block can grasp the smaller of having same type.**

Keywords* - *Data Mining, Frequent Pattern Mining Closed Sequential Pattern Mining, Itemset Mining, Temporal Data Mining

I. INTRODUCTION

Many applications are involved in sequence data. Typical examples include customer shopping sequences, Web click streams, biological sequences, sequences of events in science and engineering and in nature and social development. Sequential pattern mining is the mining of frequently occurring ordered events or subsequences as patterns. In industry the sequential mining can be useful for target marketing, customer retention and many other tasks. Other areas where sequential analysis can be used are web access pattern analysis

Sequential pattern mining, since its introduction in [2], has become an essential data mining task, with broad applications, including market and customer analysis, web log analysis, pattern discovery in protein sequences, and mining XML query access patterns for caching. Many general sequential pattern mining [3, 4, 5], constraint-based sequential pattern mining [6], frequent episode mining [7], cyclic association rule mining [8], temporal relation mining [5], partial periodic pattern mining [7], and long sequential pattern mining in noisy environment [9] have been developed and studied.

Frequent patterns (both itemsets and sequences) are of hot research trends in past years but for complete results and better efficiency [10,11] one should not mine *all* frequent patterns as data redundancy is a major problem in such patterns, mining the *closed* one is a better alternate which reduces the redundant patterns but gives the same expressive power. However due to the complexity of the problem there are not so many methods proposed for mining closed sequential patterns like frequent pattern mining. To our best knowledge, CloSpan[12], BIDE[13], CHARM are currently the only such algorithms. Like most of the frequent closed item-set mining algorithms, it follows a *candidate maintenance-and-test* paradigm, i.e., it needs to maintain the set of already mined closed sequence candidates which can be used to prune search space and check if a newly found frequent sequence is promising to be closed. Unfortunately, a closed pattern mining algorithm under such a paradigm has rather poor scalability in the number of frequent closed patterns because a large number of frequent closed patterns (or just candidates) will occupy much memory and lead to large search space for the closure checking of new patterns, which is usually the case when the support threshold is low or the patterns become long.

In this paper, we present a nice solution which leads to an algorithm, SBLOCK that mines efficiently the complete set of frequent closed sequences.

The rest of this paper is organized as follows: In section 2 we present the problem definition of frequent closed sequence mining and discuss the related work and our contributions to this problem. Section 3 is focused on the SBLOCK algorithm. Some possible extensions are also discussed in this section. In section 4 we present an extensive experimental study. Finally, we conclude the study in section 5.

II. PROBLEM DEFINITION AND RELATED WORK

Let I={i1,i2,i3,i4…..in} be a set of distinct items. A sequence S is an ordered list of events , denoted as {e1,e2,e3,….,em} where ei is an item, i.e. ei € I for 1 ≤ i ≤ m From the definition we know that an item can occur multiple times in different events of a sequence (Fig 1.1). The number of events (i.e., instances of items) in a sequence is called the length of the sequence and a sequence with a length l is also called an l-sequence, for example AABBCA is a 6 sequence. A sequence sa = a1a2a3….an is contained in another sequence Sb=b1,b2….bm if there exist integers 1 ≤ i1 < i2 ≤ …. ≤ in ≤ m such that a1=bi1, a2= bi2 and so on .

If sequence S is contained in S', S is called subsequence of S' and S' is super-sequence of S, denoted as S [S

An input sequence database SDB (Fig 1.1) is a set of tuples with a sequence identifier and input sequences. The number of tuples in SDB is called base size of SDB, denoted as |SDB|. A tuple(Sid and S) is said to contain a sequence S' if S is a super-sequence of S', for example in fig 1.1 tuple (a, CAABC) contain a sequence CAA because CAABC is a super sequence of CAA. The number of tuples in SDB that contain S' is called an absolute support of sequence S in SDB.

Min_sup is a minimum support threshold already provided or assumed. A sequence S is a frequent sequence on SDB if sup(S) ≥ min_sup. If sequence S is frequent and there exist no proper super-sequence of S with same support i.e. there exists no super-pattern s' such that s' ɔ s, and s' and s have the same support it is called a closed pattern.

TABLE I
A SAMPLE DATABASE

Sequence identifier	Sequence
1	CAABC
2	ABAC
3	CABC
4	ABBCA

In this paper we are trying to implement the sequential closed pattern mining by using separate memory blocks for separate patterns like for above fig 1.1 we will have 4 memory blocks for all patterns starts with A and another block for all patterns starts with B and so on . On placing these sequential patterns in memory blocks it will now easier to find closed patterns while placing them to these blocks.

III. RELATED WORK

The sequential pattern mining problem was first proposed by Agrawal and Srikant in [2] they provided the concept introduction and an initial Apriori-like algorithm for mining frequent itemsets. Aftre that A generalized and refined algorithm based on Apriori method was suggested by Agrawal and Srikant in GSP (Generalized Sequential Patterns) [20]. Meanwhile sequential pattern mining algorithm FreeSpan & PrefixSpan[17] were developed by Han et al.KDD'00 which were based on pattern growth methods . A vertical format based mining algorithm SPADE[24] was then proposed by Zaki which uses a vertical id-list format and uses a lattice-theoretic approach to decompose the original search space into smaller spaces. Some constraint based sequential mining algorithms were also developed like SPIRIT by Garofalakis, Rastogi, Shim.

Since the introduction of frequent closed itemset mining [15], several efficient frequent closed itemset mining algorithms have been proposed, such as A-Close [15], CLOSET [16], CHARM [25], and CLOSET+ [21]. Most of these algorithms need to maintain the already mined frequent closed patterns in order to do pattern closure checking. Clospan [22] is a latest proposed algorithm for mining closed sequential patterns developed by Yan, Han & Afshar. It follows the *candidate maintenance-and-test* approach: First generate a set of closed sequence candidates which is stored in a hash indexed result-tree structure and then do post-pruning on it. It uses some pruning methods like *Common Prefix* and *Backward Sub-Pattern pruning* to prune the search space. Because CloSpan needs to maintain the set of historical closed sequence candidates, it will consume much memory and lead to huge search space for pattern closure checking when there are many frequent closed sequences. As a result, it does not scale very well with respect to the number of frequent closed sequences.

IV. ALGORITHM

There are mainly three steps which SBLOCK follows to generate closed sequential pattern they are:

- Identify the frequent unique items.
- Generate frequent patterns on projected database for each item.
- Determine closed patterns and remove them.

In this section we introduce the algorithm step by step by providing the answers of some questions related to sequential pattern mining. They are How to enumerate the complete set of frequent sequences? How to get a closed sequence from that frequent sequence? And how to optimize the process for accelerate the mining process?

A. Finding Frequent Patterns

Initially we are getting all unique occurrences of items with their support that we are placing at top as root node or first level, recursively we can extend the memory boxes at level L in the flow by adding an item I at top level.

- A subroutine Findunique(I) will be called to find all unique items from database SD from fig1.1 all unique occurrences are A B and C.After finding all unique occurrences of sequential database the next step will be to find all frequent patterns related to each unique occurrence
- Subroutine FindPattern(UI, I) will be called to find all frequent patterns where UI is the unique item generated in first step ie A, B and C in our example table. This will return all possible frequent patterns for one block; this process can be nested to find other possible frequent patterns of different unique items generated earlier. After

completion of this step we will get the output as in level 2 of our sample diagram, where all unique occurrences will have their own sequential frequent patterns. To improve the performance we implements a checking for closed sequential patterns in same level , Items with underline are found closed so will be removed from the block and in step 2 we will get all sequential closed patterns of all blocks.

Further steps are taken as sub process of step 2 in which checking is performed for finding the closed frequent item sets which may occur in other blocks like BB:2 and BC:4 from block B are repeated in Block A as ABB:2 and ABC:4 so are closed and removed. Finally after reaching to the last we will get all frequent closed patterns. From Fig1we can found a final closed sequence for our first extracted item i.e. 'A' here. We can further repeat the same process for other left items B and C.

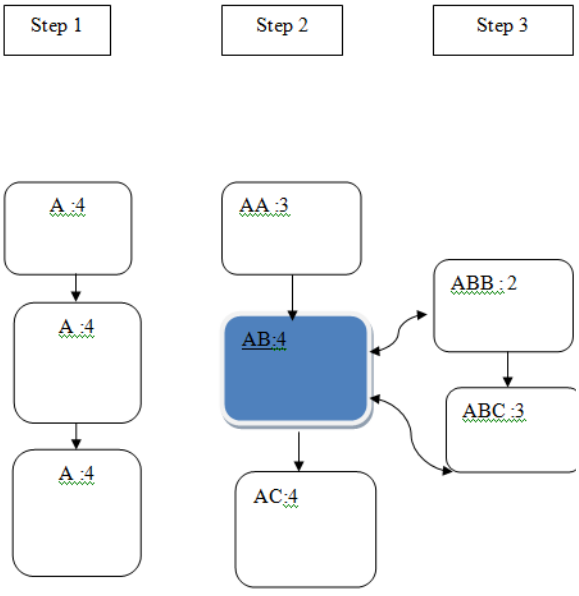

Fig. 1 Output- final closed sequence

B. SBLOCK Algorithm

As explained above now we are presenting the original algorithm which can further implemented in any programming language over any database.

1) Bunique(SDB,USQ)

Input : an input sequence db SDB
Output : The complete unique set of items I in SDB

- Usq=θ;
- Select distinct item I in SDB;
- Usq=I;
- Return usq;

A B and C will be returned so 3 blocks

2) Frequent-Block(SDB,min_sup,Usq,FCS)

Input: SDB, minimum support min_sup threshold, Unique sequence Usq
Output: Frequent Closed Sequence

- For each unique item I in Usq
- Call Callstack(I,st1,st2,SDB,USQ,Sto)
- Store sto elements in FCS
- return FCS

3) Callstack(I, st1,st2,SDB,Usq,sto)

Input: Input sequence I, Stack for Frequent Items st1, stack for support st2
Output : stack of closed frequent sequence of one block sto

- Create length(Usq)node for st1 and st2 and store I on all of them
- For (no p from 1 to no of length (Usq)) do
- Find all combination upto p+1 for all Item starts with I
- Check if min_sup is same as previous stack
- If(true) replace previous with new one
- Else create a new node and store item
- Return st1

V. Experimental Results

Our performance study includes both synthetic and real datasets. We used three synthetic datasets generated an online available tool [14]. The characteristics of these datasets are shown in table II. All experiment was based on windows7 OS with core 2 duo processor and 1 GB main memory. We used java compiler to perform this experiment.

TABLE II
DATA-SET FOR EXPERIMENT

A	B	C	Class
a	i	b	c1
f	e	s	c3
f	e	b	c3
f	e	g	c3

We then applied above dataset in our target language and measured result with continuously increasing data count and minimum support, for every condition SBLOCK is working fine In particular we compare the performance of SBLOCK with CloSpan. The comparison is based on assigning optimal minimum support to CloSpan to get similar result as SBLOCK

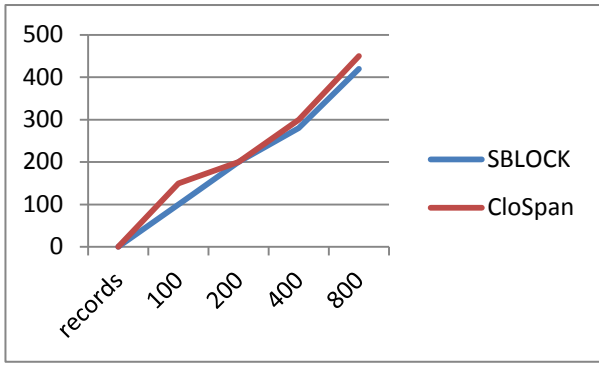

Fig. 2 Comparison of execution time

As we can see in above diagram SBLOCK is taking les time to execute same records

VI. CONCLUSIONS

For huge frequent patterns where the length of frequent closed patterns is also large , our study shows it is true that closed pattern mining has better expressive power as that of frequent pattern mining, as it do have a better efficiency and same analytical result. In this paper we introduce an algorithm for mining frequent closed sequences with the help of stack as memory blocks and then sequential patterns were generated to found if any of them is closed as like forward attribute selection.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In VLDB'94, Santiago, Chile, Sept. 1994

[2] R. Agrawal, and R. Srikant, Mining sequential patterns. In ICDE'95, Taipei, Taiwan, Mar. 1995.

[3] F. Masseglia, F. Cathala, and P. Poncelet, The psp approach for mining sequential patterns. In PKDD'98, Nantes, France, Sept. 1995

[4] R. Srikant, and R. Agrawal, Mining sequential patterns:Generalizations and performance improvements. In EDBT'96, Avignon, France, Mar. 1996.

[5] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, FreeSpan: Frequent pattern-projected sequential pattern mining . In SIGKDD'00, Boston, MA, Aug. 2000.

[6] M. Garofalakis, R. Rastogi, and K. Shim, SPIRIT: Sequential PAttern Mining with regular expression constraints. In VLDB'99, San Francisco, CA, Sept. 1999.

[7] H. Mannila, H. Toivonen, and A.I. Verkamo, Discovering frequent episodes in sequences . In SIGKDD'95, Montreal, Canada, Aug. 1995.

[8] B. Ozden, S. Ramaswamy, and A. Silberschatz, Cyclic association rules. In ICDE'98, Olando, FL, Feb. 1998.

[9] J. Yang, P.S. Yu, W. Wang and J. Han, Mining long sequential patterns in a noisy environment. In SIGMOD' 02, Madison, WI, June 2002.

[10] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discoving frequent closed itemsets for association rules. In ICDT'99, Jerusalem, Israel, Jan. 1999.

[11] M. Zaki, and C. Hsiao, CHARM: An efficient algorithm for closed itemset mining. In SDM'02, Arlington, VA, April 2002.

[12] H. Mannila, H. Toivonen, and A.I. Verkamo, Discovering frequent episodes in sequences. In SIGKDD'95, Montreal, Canada, Aug. 1995.

[13] BIDE: Jianyong Wang_and Jiawei Han Efficient Mining of Frequent Closed Sequences

[14] http://www.datasetgenerator.com/