International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

A Survey on Mining User-Aware Rare Sequential Pattern

Salmath Amina KP[1], Farzin Ahammed T[2]

[1, 2]AWH Engineering College, KTU University, Department of Computer science & Engineering, Kuttikkatoor, Kozhikode, India

Abstract: ***Sequential pattern mining is exploring enthusiasm for finding the most frequently using patterns from the data sets. We explore the sequential pattern mining using a bitmap representation. Our algorithm incrementally output new frequent itemset in an online fashion using depth-first search strategy. To find out the sequential pattern in data streams, we propose a sequential data mining algorithm using weighted sliding window. SWSS algorithm provides more space for users to specify which sequences they are more interested in. To address the problem of finding frequent and rare itemsets from the data streams, we propose another algorithm called New_SPAM. New_SPAM Algorithm is the combination of both Per_SPAM and Min_SPAM algorithm.***

Keywords: Frequent Itemset Mining, SWSS, Rare itemset, Rare Sequential pattern mining, Weighted Sliding Window

1. Introduction

Data mining [1] is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

The problem of mining sequential patterns over data is solved by this paper. A new algorithm for mining sequential patterns algorithm is especially efficient when the sequential patterns in the database are very long. Here we introducing a novel depth-first search strategy that integrates a depth-first traversal of the search space with effective pruning mechanisms.

Mining data streams for knowledge discovery is important to many applications, including Web click stream mining. The algorithm SWSS (Sequential pattern mining[2] with the weighted sliding window model in SPAM) to mine frequent sequential patterns based on the weighted sliding windows model. This algorithm provides more space for users to specify which sequences they are more interested in.

Recently, data mining communities have focused on a new data model, where data arrives in the form of continuous streams. Real life datasets contain both frequent and infrequent items. The infrequent items can be called as rare items and these are of importance. Rare itemset problem occurs when the frequency of an item varies largely. For a very high minimum support, frequent patterns involving rare items are missed. So, a lower minimum support is set to find frequent patterns involving both rare and frequent patterns. But this may cause item explosions. So, we have proposed approaches to efficiently extract items.

Document streams are generated in various forms on the Internet, such as news streams, micro blog articles, instant messages, research paper archives, web forum discussion threads, and so forth. These document streams generally concentrate on specific topics.

One of the reasons behind maintaining any database is to enable the user to find interesting patterns and trends in the data. For example, in a supermarket, the user can figure out which items are being sold most frequently. But this is not the only type of `trend' which one can possibly think of. The goal of database mining is to automate this process of finding interesting patterns and trends. Once this information is available, we can perhaps get rid of the original database. The output of the data-mining process should be a "summary" of the database. This goal is difficult to achieve due to the vagueness associated with the term `interesting'. The solution is to define various types of trends and to look for only those trends in the database. One such type constitutes the association rule.

Real life datasets contain both frequent and infrequent items. The infrequent items can be called as rare items and these are of importance. Rare itemset problem occurs when the frequency of an item varies largely. For a very high minimum support, frequent patterns involving rare items are missed. So, a lower minimum support is set to find frequent patterns involving both rare and frequent patterns. But this may cause item explosions. So, we have proposed approaches to efficiently extract items.

Some STPs occur frequently in a document stream and thus reflect common behaviors of users. Besides, there are still some others which are rare for the general population, but occur relatively often for some specific user or some specific group of users. Compared to frequent ones, mining these user-related rare STPs are more interesting. Theoretically, it defines a new kind of patterns for event mining, which can characterize those individual and personalized behaviors in a certain context.

2. Bitmap Representation

A database D is a set of tuples (c_{id}, t_{id}, X), where c_{id} is a customer-id, t_{id} is a transaction-id based on the transaction time, and X is an itemset . Each tuple in D is referred to as a transaction. For a given customer id, there are no transactions with the same transaction-id. All the transactions with the same c_{id} can be viewed as a sequence of itemsets ordered by increasing t_{id}. An analogous representation for the database is thus a set of sequences of transactions, one sequence per customer, and users refer to this dual representation of D as its sequence representation.

Given a support threshold minSup, a sequence s_a is called a frequent sequential pattern on D if supD(sa) minSup. The problem of mining sequential patterns is to find all frequent sequential patterns for a database D, given a support threshold sup.

Suppose we want to find the support of the sequence s_a = (a, b, c). From Table below,

Table 1: *Sequence for each customer*

CID	Sequence
1	$(\{a,b,d\},\{b,c,d\},\{b,c,d\})$
2	$(\{b\},\{a,b,c\})$
3	$(\{a,b\},\{b,c,d\})$

The support of *sa* in *D* is denoted by *supD*(*sa*). Given a support threshold *minSup*, a sequence *sa* is called a *frequent sequential pattern on D* if *supD*(*sa*) ≥ *minSup*. The problem of mining sequential patterns is to find all frequent sequential patterns for a database *D*, given a support threshold *sup*.

2.1 The SPAM Algorithm

The Algorithm describes the lexicographic tree of sequences upon which our algorithm is based. The algorithm based on lexicographic tree representation, depth first traversal and pruning technique.

The part Lexicographic Tree for Sequences This part describes the conceptual framework of the sequence lattice upon which our approach is based. A similar approach has been used for the problem of mining frequent itemsets in MaxMiner[3] and MAFIA[5]. One can use this framework to describe our algorithm and some pertinent related works. Assume that there is a lexicographical ordering ≤ of the items I in the database. If item i occurs before item j in the ordering, then denote this by i ≤ Ij. This ordering can be extended to sequences by defining sa ≤ sb if sa is a subsequence of sb. If sa is not a subsequence of sb, then there is no relationship in this ordering. Example for a lexical tree is given below

In the part depth first traversal SPAM traverses the sequence tree described above in a standard depth-first manner. At each node n, the support of each sequence-extended child and each itemset-extended child is tested. If the support of a generated sequence s is greater than or equal to minSup, we store that sequence and repeat DFS recursively on s. If the support of s is less than minSup, then do not need to repeat DFS on s by the Apriori principle, since any child sequence generated from s will not be frequent. If none of the generated children are frequent, then the node is a leaf and we can backtrack up the tree.

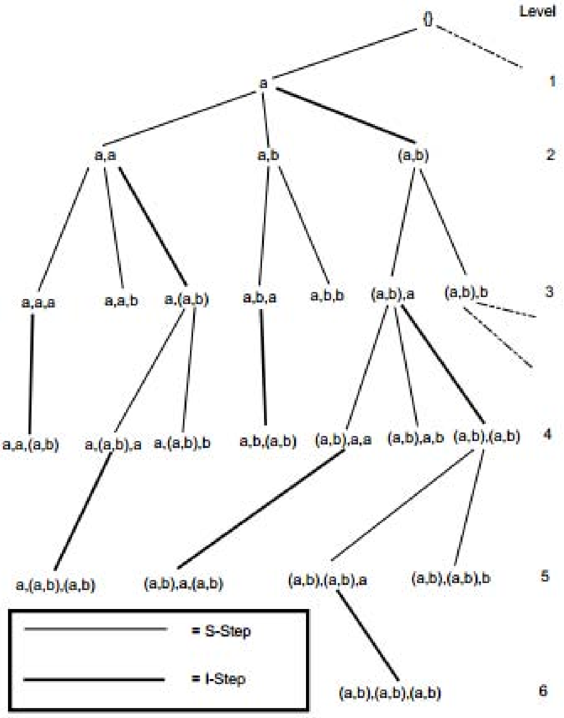


Figure 1: Example for lexicographic tree

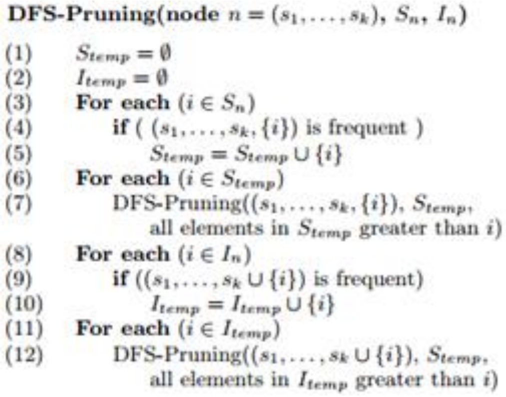
If we generate sequences by traversing the tree, then each node in the tree can generate sequence-extended children sequences and itemset-extended children sequences. User refer to the process of generating sequence-extended sequences as the sequence-extension step, and refer to the process of generating itemset-extended sequences as the itemset-extension step. Thus one can associate with each node n in the tree two sets: Sn, the set of candidate items that are considered for a possible S-step extension of node n, and I_n, which identifies the set of candidate items that are considered for a possible I-step extensions.

The pruning technique consists of S-step pruning and I-step pruning. Figure 1 shows below the pseudo code for our algorithm after both I-step and l-step pruning are added.

The S-step pruning is one technique used prunes S-step children. Consider a sequence s at node n and suppose its sequence-extended sequences are sa = (s, (i) g) and sb = (s (i)). Suppose sa is frequent but sb is not frequent. By the Apriori principle, (s, (i) (i)) and (s(i) (k)) cannot be frequent, since both contain the subsequence sb.

In I-step pruning The second technique prunes I-step children. Figure below shows the pseudocode of our algorithm after both I-step pruning and S-step pruning are added. After pruning the correct items from Sm and Im, we pass the new lists to the next recursive call. The lists are pruned exactly as described in the previous sections.

Table 2: Pseudo code for DFS with pruning



```
DFS-Pruning(node $n = (s_1, \ldots, s_k)$, $S_n$, $I_n$)

    $S_{temp} = \emptyset$
    $I_{temp} = \emptyset$
    For each $(i \in S_n)$
        if ( $(s_1, \ldots, s_k, \{i\})$ is frequent )
            $S_{temp} = S_{temp} \cup \{i\}$
    For each $(i \in S_{temp})$
        DFS-Pruning($(s_1, \ldots, s_k, \{i\})$, $S_{temp}$,
            all elements in $S_{temp}$ greater than $i$)
    For each $(i \in I_n)$
        if ($(s_1, \ldots, s_k \cup \{i\})$ is frequent)
            $I_{temp} = I_{temp} \cup \{i\}$
    For each $(i \in I_{temp})$
        DFS-Pruning($(s_1, \ldots, s_k \cup \{i\})$, $S_{temp}$,
            all elements in $I_{temp}$ greater than $i$)
```

After pruning the correct items from *Sm* and *Im*, we pass the new lists to the next recursive call. The lists are pruned exactly as described in the previous sections.

3. Weighted Sliding Window Model

The sliding window model mainly deals with the data generated from the current moment to a specified time point and the data continuously changes when new elements come and old ones are deleted. The size of the window could be specified by users to be a given number of transactions or a fixed time period.

Since the weight of the sliding windows can be specified by users according to their interests, one frequent sequence in one window is not really frequent in other windows. Similarly, one infrequent sequence in one window is probably frequent in other windows. Thus user have to consider the weights of the windows and get the final results by calculating the support of one sequence based on the weights of different windows.

3.1 SWSS Algorithm

If item *i* appears in a transaction *j*, then, the bit corresponding to transaction *j* of the bitmap for item *i* is set to one, otherwise the bit is set to zero. And the above kind of bitmap can naturally be extended to itemsets and sequence. A sequence support counting is a calculating process that is the bitwise AND of these two bitmaps.

From Apriori algorithm property we can easily get the two extension conclusions above which are also applicable in the frequent sequential patterns mining over the weighted sliding windows. Both lemmas are applicable in every window for the same reason. Besides, the candidate sequences are generated during the mining process in the manner of the depth first traversal through a lexicographic tree.
Each sequence can be extended through S-step or I-step. If the extended sequence is infrequent, the item a will be deleted during the process. Obviously each node of lexicographic tree represents one frequent sequence pattern.

In MaxMiner and MAFIA two similar algorithms have been used for mining frequent itemsets problem. In the lexicographic tree, SPAM generates sequences by traversing the tree, each node can be generated by two categories of extending methods: the sequence extension step (abbreviated, S-step) and the itemset–extension step (abbreviated, I-step). Suppose bitmaps B(s) and B(i) is respectively a sequence and a item i. The S-step appends the itemset {i} to the sequence s. And the I-step appends item I to the last itemset of s to generate a new sequence sn.

In our work, we propose SWSS algorithm to mine frequent sequential patterns in the weighted sliding window model. As far as our knowledge, this is the first piece of work for mining frequent sequential patterns over weighted sliding windows. In this paper, we make full use of the fast counting of SPAM[9] and its lexicographical tree for storing the frequent sequential patterns. We use SWSS to mine frequent sequential patterns over the weighted sliding window model.

Table 3: SWSS Algorithm

```
(1) Initialization:
    a: for each window from the data stream do
    b: get *B-List* (initialize the bitmaps of all items of each
    window respectively.)
    c: get *minSup_weight*(get *minSup_weight* of the sliding
    windows)

(2) Mining:
    a: For each window from the data stream do
    b: Get *All-List* (Find out all items which are not different
    from each other in every window and store them in *All
    List*. The items are used as the initial values for *S-step* and
    *I-step*)
    c: Initialize *W-Tree* (the root of the tree is NULL)
    d: Build *W-Tree* by depth first traversal (*S-step* then *I-
    step*)
       If a new sequence support over *minSup_weight*
          Add the sequence into *W-Tree* as a node
          Add the sequence into *All-List*
          Transform the sequence a bitmap into *B-List*
     e: If a new frequent sequence exists
           Go to step d
           Else
            Build *W–Tree* process is completed.
```

A frequent sequential pattern in the weighted sliding windows includes a sequences whose weighted support count (abbreviated, sup_weight(s)) is not smaller than the minimum weighted support count of the windows (abbreviated, minSup_weight).

Among the transactions every item is represented by the vertical bitmap. Every bitmap is partitioned into n sections and n means the number of the customers. A sequence support counting is a calculating process that is the bitwise AND of these two bitmaps.

Lemma 1: If sequence s is an infrequent sequence, then any super sequence of it is an infrequent sequence [10].
Lemma 2: If sequence s is a frequent sequence, then any subsequence of it is a frequent sequence [10].

Both lemmas are applicable in every window for the same reason. Besides, the candidate sequences are generated during the mining process in the manner of the depth first traversal through a lexicographic tree.

4. New_SPAM

SPAM [9] algorithm proposed by Ayers et al. in 2002 uses a binary vertical database format representation for every item to store each sequence. It uses a depth first search technique to traverse the lexicographic sequence tree. The supports of candidate patterns are obtained rapidly by executing bitwise operations on the bitmaps. SPAM reduces the number of candidates generated by using an efficient pruning mechanism. This algorithm requires large amount of memory and causes a bottleneck issue of creating a huge amount of infrequent candidates as it uses a generate candidate and test method. To extract frequent itemsets including both frequent and rare items, a new approach has been proposed in this paper.

4.1 Periodic Approach

When a high minimum support, we fail to extract frequent patterns containing rare items. So, we have proposed a new approach called *Per_SPAM* (periodic) that allows the rare itemsets to fulfill the minimum support by decreasing the count number of support value. This approach leads to efficient extraction of rare itemsets as it decreases the value of support count by satisfying the lower minimum support criteria. Hence, this decreases the distance between the minimum and maximum support value.

4.2 Minimum Item Support (Mis) Approach

A new approach is proposed called Mis_SPAM that allows extraction of frequent patterns involving both frequent and rare items [21]. In this approach, there is unchanging difference between the items MIS value and its support value. The difference remains constant for varying frequencies of items. Each item is assigned a minimum support (MIS) value.

Here we use the term Weight (W) which can be defined as the measure of importance upon which the itemset can be considered as frequent. W can be calculated as shown in equation 1.

$$W=\beta(1-\alpha)$$

Where β is the running average of supports of items and α is the user specified value ranging from (0,1) that is suitable to user according to dataset.

4.3 New Algorithm

A new algorithm called New_SPAM which is an improved algorithm. We have integrated the two proposed approaches Per_SPAM and Mis_SPAM into one algorithm and introduced a new algorithm called New_SPAM which is used to extract frequent and rare items by satisfying periodic properties of an item sequence.

The pseudo code of New_SPAM is described below where *minsup* is the relative minimum support, SDB is the sequence database to be loaded, I is the item, F1 is the list of frequent items, e is the frequent item, s is the sequence, lex means lexicographical, pat is the sequential pattern, Sn represents items to be appended to pat by s-extension, In represents items to be appended to pat by i-extension.

Table 4: New_SPAM Algorithm

```
New_SPAM (fileHandle, minsup)
count = 0
SDB = new SDB ( )
FOR i FROM 0 TO CALC_MAX_ITEMS (fileHandle)
    ADD_ENTRY_IN_SDB (READ_ITEM (fileHandle)),
    INCREMENT count.
    IF count = (N +1),
        i = i – (N-1).
Scan SDB to create Vertical(SDB) and identify F1.
FOR each s in F1,
    SEARCH ({s}, F1, {e in F1 | e >lex s}, minsup, MIS).
SEARCH (pat, Sn, In, minsup, MIS)
    Output pattern pat.
    Stemp := Itemp = {} // Null Set
    FOR each item j in Sn,
        IF the s-extension of pat is frequent THEN Stemp := Stemp UNION {i}.
    FOR each item j in Stemp,
        SEARCH (the s-extension of pat with j, Stemp, {e in Stemp | e >lex j}, minsup, MIS).
    FOR each item j in In,
        IF the i-extension of pat is frequent THEN Itemp := Itemp UNION {i}.
    FOR each item j in Itemp,
        SEARCH (i-extension of pat with j, Stemp, {e in Itemp | e >lex j}, minsup, MIS).
```

5. Conclusion

In the bitmap representation approach based paper we presented an algorithm to quickly find all frequent sequences in a list of transactions. The algorithm utilizes a depth-first traversal of the search space combined with a vertical bitmap representation to store each sequence. In this paper SWSS, we propose an efficient single pass algorithm SWSS to mine all the frequent sequential patterns over the weighted sliding windows. SWSS allows users to specify what they are more interested in. In New_SPAM algorithm, we have presented two approaches. Periodic approach is used to discover frequent patterns in item and Minimum item support approach is used to extract frequent patterns. We have proposed a new algorithm named New_SPAM which includes both of the periodic and MIS approaches to find both frequent as well as rare items.

References

[1] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2006).

[2] R. Agrawal and R. Srikant. Mining Sequential Patterns. In ICDE 1995, Taipei, Taiwan, March 1995.

[3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the Twentieth International Conference on Very Large Databases, pages 487–499, Santiago, Chile, 1994.

[4] R. Agrawal and R. Srikant. Mining Sequential Patterns. In ICDE 1995, Taipei, Taiwan, March 1995.

[5] R. J. Bayardo. Efficiently mining long patterns from databases. In SIGMOD 1998, pages 85–93, 1998.

[6] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” in Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, 2006, pp. 79–88.

[7] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In ICDE 2001, Heidelberg, Germany, 2001.

[8] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In ICDE 1999, pages 106–115, Sydney, Australia, Mar.

[9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan mining sequential patterns efficiently by prefix projected pattern growth. In ICDE 2001, pages 215–226, Heidelberg, Germany, Apr. 2001.

[10] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. Machine Learning, 42(1/2):31–60, 2001

[11] J. Ayres, J. Flannick, J. Gehrke, & T. Yiu, Sequential PAttern Mining Using A Bitmap Representation, In Proc. Int. Conf. Knowledge Discovery and Data Mining, 2002, pp.429-435D.

[12] Dong. Guozhu, Pei. Jian, Sequence Data Mining, Series: Advances in Database Systems, Vol. 33, 2007, p.119

[13] Y. Zhu, & D. Shasha, StartStream: Statistical monitoring of thousands of data streams in real time, In Proceedings of the VLDB conference, 2002, pp.358-369.

[14] P. Domingos, & G. Hulten, Mining high-speed data streams, In Proceedings of ACM SIGKDD, 2000, pp.71-80.

[15] Y.Chi, H.Wang, P. S. Yu, & R. R. Muntz, Catch the moment: Maintaining closed frequent itemsets over a data stream sliding window, Knowledge and Information Systems10(3), 2006, pp.265-294.

[16] A. Marascu & F. Masseglia, Mining Sequential Patterns from Temporal Streaming Data, In Proceedings of the 1st ECML/PKDD Workshop on Mining Complex Data (IEEE MCD), 2005

[17] C. Ezeife & M. Monwar, SSM: a frequent sequential data stream patterns miner, CIDM, 2007, pp.120-126.

[18] C. C. Ho, H. F. Li, F. F. Kuo, & S. Y. Lee, Incremental mining of sequential patterns over a stream sliding window, In Proceedings of IEEE international workshop on mining evolving and streaming data, 2006.

[19] G. Chen, X. Wu, & X. Zhu, Mining Sequential Patterns across Data Streams, Technical Report, 2004.

[20] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2006).

[21] Yun, H., Ha, D., Hwang, B., and Ryu K. H. “Mining association rules on significant rare data using relative support.”, The Journal of Systems and Software 67, 2003, pp. 181-191.

Author Profile

Salmath Amina KP received the B Tech degree in Computer Science and Engineering from University of Calicut in 2014. Currently pursuing M Tech in Computer Science and Engineering from Kerala Technical University.

Farzin Ahammed T received B.Tech Degree in Computer Science and Engineering in 2012 from Govt. Engineering College Thrissur and M.Tech in Computer Science and Engineering from TKM College of Engineering Kollam in 2015. Now he is currently working as Assistant Professor in AWH Engineering College.