EFFICIENTLY MINING CLOSED SEQUENTIAL PATTERNS USING PREFIX TREE

PHAM THI THIET, VAN VO
Industrial University of Ho Chi Minh City;
phamthithiet@iuh.edu.vn, vttvan@iuh.edu.vn

Abstract. Mining closed sequential patterns is one of important tasks in data mining. It is proposed to resolve difficult problems in mining sequential pattern such as mining long frequent sequences that contain a combinatorial number of frequent subsequences or using very low support thresholds to mine sequential patterns is usually both time- and memory-consuming. So, using the parent–child relationship on prefix tree structure to improve the performance of the mining sequential patterns process from the sequence database is also one of important methods in data mining, specially in the mining closed sequential patterns. This paper produces an effective algorithm and experimental results for mining closed sequential patterns from the sequence database using the parent–child relationship on the prefix tree structure. Experimental results show that the performance runtime of the proposed algorithm is much faster than that of other algorithms by more than one order of magnitude and the number of sequential patterns is reduced significantly.
Keywords. sequential pattern, closed sequential pattern, prefix tree, sequence database.

1. INTRODUCTION

Sequential pattern mining, since it was first introduced by Agrawal et al. [1], has played an important role in data mining tasks with broad applications including market and customer analysis, web log analysis, pattern discovery in protein sequences, and mining XML query access patterns for caching and so on. The sequential pattern mining algorithms proposed so far have a good performance in databases with short frequent sequences [2,3,9-10,16]. However, when mining long frequent sequences that contain a combinatorial number of frequent subsequences, such a mining will generate an explosive number of frequent subsequences for long patterns, or when using very low support thresholds to mine sequential patterns, which is prohibitively expensive in both time and space. So, the performance of such algorithms often degrades dramatically. To overcome this difficultly, the problem for mining closed sequential patterns have also been proposed. A sequence S is called closed if there exists no supersequence of S with the same support in the database.

Several studies have been recently proposed to mine closed sequential patterns [4,11-12,14-15]. But these algorithms used the corresponding projected databases of frequent subsequences to find closed sequences. It consumes much time to construct projected databases of frequent subsequences for a set of sequence. In this paper, we introduce an efficient algorithm and experimental results for the mining closed sequential patterns problem. Based on the combination of the parent–child relationship and its propertied on prefix tree structure, the closed sequential patterns could have been found directly at the generating sequential patterns process. On the prefix tree in this approach, each node stores a sequential pattern and its corresponding support value. Besides, it will be added one field to consider whether this node is a closed sequential pattern (*IsCSP*). Based on the *IsCSP* field added to each node, the algorithm easily determines if a node is a closed sequential pattern so the mining time is reduced significantly. This algorithm also uses join operations over the prime block encoding approach of the prime factorization theory to represent candidate sequences and determine the frequency for each candidate. The experimental results showed that the performance for mining closed sequential patterns in this algorithm is much better.

The rest of this paper is organized as follows. Section 2 reviews some works related to mine closed sequential patterns. Section 3 presents some problem definitions related to sequential patterns/closed sequential pattern and prefix tree. The algorithm for mining closed sequential patterns is discussed in

Section 4. The experimental results for mining closed sequential patterns using the prefix tree structure are presented in Section 5, and conclusion and future work are presented in Section 6.

2. RELATED WORK

Mining sequential patterns with closed patterns may significantly reduce the number of patterns generated in the mining process without losing any information because it can be used to derive the complete set of sequential patterns; so, the number of closed sequential patterns is usually fewer than the number of sequential patterns. Several studies have been recently proposed to mine closed sequential patterns [4,11-12,14-15]. The *CloSpan* algorithm [15] has been proposed. Like most of the frequent closed itemset mining algorithms *CLOSET* [6] and *CHARM* [17], *CloSpan* algorithm used the candidate maintenance and test approach. It needs to maintain the set of already mined closed sequence candidates for doing the backward subpattern and backward superpattern check to verify if a newly found frequent sequence is promising to be closed or not. Because CloSpan needs to maintain the set of historical closed sequence candidates, when there are many frequent closed sequences, it will consume much memory and lead to huge search space for pattern closure checking. *BIDE* [14] is another faster closed sequence mining algorithm. Different from *CloSpan*, it used a novel sequence closure checking scheme called BI-Directional Extension, and pruned the search space more by using the BackScan pruning method and the ScanSkip optimization technique to directly get the complete set of the frequent closed sequence patterns without candidate maintenance. Thus, in most cases, *BIDE* is more efficient than *CloSpan*, especially when a database is dense or the minimum support value is low. But to implement the closure check, the BIDE algorithm spends a lot of time on scanning the pseudo-projected database repeatedly to verify the existence of extension of position with a prefix sequence, which costs much time in the mining process. To reduce the time consumed on scanning the pseudo-projected database for verifying in the *BIDE* algorithm, the *FCSM-PD* algorithm was proposed by Huang et al. [4]; the positional data was used to reserve the position information of items in the data sequences. In the pattern growth process, the extension of position with a prefix sequence is checked directly and all the position information of the new prefix sequences will be recorded. The *FCSM-PD* algorithm must store all the position information of a prefix sequence in the process of pattern growth in advance; so it consumes more memory in this algorithm.

3. PROBLEM DEFINITIONS

A sequence database *SD* is a set of sequences $S=\{s_1, s_2, ..., s_n\}$ and a set of items $I=\{i_1, i_2, ..., i_n\}$, where each sequence s_x is an ordered list of itemsets $s_x=\{x_1, x_2, ..., x_n\}$, and s_1 occurs before s_2, which occurs before s_3, and so on, such that $x_1, x_2, ..., x_n \subseteq I$. The size of a sequence is the number of itemsets in the sequence. The length of a sequence is the number of instances of items in the sequence. A sequence with length l is called an l-pattern sequence. A sequence with size *k* is called a *k*-sequence.

Given two sequences $\alpha = \langle a_1\ a_2\ ...\ a_n\rangle$ and $\beta = \langle b_1\ b_2\ ...\ b_m\rangle$ *(where a_i, b_i are itemsets)*, sequence α is called a subsequence of β and β is a supersequence of α, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \le j_1 < j_2 < ... < j_n \le m$ $(n \le m)$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, . . . , $a_n \subseteq b_{jn}$. For example, if $\alpha = \langle (ab), d\rangle$ and $\beta = \langle (abc), (de)\rangle$, where *a, b, c, d*, and *e* are items, then α is a subsequence of β and β is a supersequence of α. The support of a sequence α (denoted by *Sup(α))* in a sequence database is the number of sequences in the database containing α. Sequence α is a frequent sequence in sequence database *SD*, if $Sup(\alpha) \ge minSup$ where *minSup* is the minimum support threshold defined by user. A frequent sequence is called a sequential pattern. A sequential pattern α is called a closed sequential pattern if and only if $\neg\exists\, \beta$ such that $\alpha \subseteq \beta$ (i.e., β contains α) and $Sup(\alpha) = Sup(\beta)$.

Sequence α is a prefix of β if and only if $a_i = b_i$ for all $1 \le j \le n$. After eliminating the prefix part α of sequence β, the remainder of β is a postfix of β. From the above definition, we know that a sequence of size *k* has *(k-1)* prefixes. For example, a sequence $\langle (A)(BC)(D)\rangle$ has 2 prefixes: $\langle (A)\rangle$ and $\langle (A)(BC)\rangle$. Therefore, $\langle (BC)(D)\rangle$ is the postfix for prefix $\langle (A)\rangle$, and $\langle (D)\rangle$ is the postfix for prefix $\langle (A)(BC)\rangle$.

A prefix tree is similar to a lexicographic tree [7-8], which starts from the tree root at level 0. In this paper, the prefix tree is started at the root with a null sequence ∅. Each child node stores a sequential pattern, its support value, one field to consider whether this node is a closed sequential pattern (*IsCSP*).

At level 1, each node is set with a single frequent item; at level k, each node is set with a k-pattern sequence. Recursively, there are nodes at the next level $(k+1)$ after a k-pattern sequence is extended with a single frequent item. There are two ways to extend a k-pattern sequence, namely sequence extension and itemset extension [3]. In sequence extension, a single frequent item from I is added to the k-pattern sequence as a new itemset, increasing the size of the sequence. A sequence α is a prefix of all sequence-extended sequences of α, and α is the prefix of all subnodes of the nodes that are sequence-extended in α. In itemset extension, single frequent item from I that is greater than all items in the last itemset is added to the last itemset in the k-pattern sequence. The size of itemset-extended sequences does not change and α is an incomplete prefix of all subnodes of itemset-extended nodes in α.

4. MINING CLOSED SEQUENTIAL PATTERNS BASED ON THE PREFIX TREE STRUCTURE

In this section, A briefly description of the idea of algorithm to mine closed sequential patterns using the prefix tree is done. This algorithm uses the extension of a sequence on the prefix tree by performing a depth-first search and the attribute of closed sequential patterns to generate all closed sequential patterns. Using the prefix tree, new sequences, which are children nodes *Cnode*, can be easily created by appending an item to the last position of a parent node *Pnode* as an itemset extension or a sequence extension. When a new node *Pnode* is created, if *Sup(Pnode) = Sup(Cnode),* then we set the *IsCSP* of *Cnode* to false and the *IsCSP* of *Pnode* to true.

The details of the algorithm for mining closed sequential patterns are introduced in Figure 1. First, the algorithm initializes the prefix tree *pretree* with the root node being null and children nodes being sequential 1-patterns with its *IsCSP* field as *true.* Each child node *cn* on *pretree* is considered as a root node for *EXTENDTREE(cn, pretree)* function to create its children nodes and extend the *pretree* tree. In this function, each child node of root node *P* is created by itemset extension or sequence extension. To represent candidate sequences as well as determine the frequency for each candidate, it uses the prime block encoding approach and the join operations over the prime blocks in [3]. With each new child node is created *Pnew* from *P,* if *Sup(Pnew)=Sup(P),* then the value of *Pnew.IsCSP* is set to *true* and the value of *P.IsCSP* is set to *false*. The *ISCLOSED(Pnew, pretree)* function is called to update closed sequential patterns on the *pretree* tree. Finally, the algorithm returns the corresponding *pretree* tree with the sequential patters which have the corresponding IsCSP values.

Input: *SD*, *minSup*.

Output: Set of the c*losed sequential patterns*.

Method:

MCSP_PreTree(SD, minSup)

```
    pretree ← ∅;
    SPs ← all frequent 1-pattern sequences;
    for each pattern P in SPs
            Add P into pretree as a child node;
    for each child node r in pretree
            EXTENDTREE(r, pretree);
    return pretree;
//.....
EXTENDTREE(Root, pretree)
  EXTENDITEMSET(Root, pretree);
    EXTENDSEQUENCE(Root, pretree );
    For each node Pi that is an itemset extension of Root
            EXTENDTREE(Pi, pretree);
```

```
    For each node Ps that is a sequence extension of Root
        EXTENDTREE(Ps, pretree);
// ..........
EXTENDITEMSET(P, pretree)
    For each pattern Pi in SPs with Pi>last item in last itemset of P
        Pnew is a new node created by adding Pi into last position in last itemset of P and using
        block encoding based on prism factorization in [3] to count the support;
      If (sup(Pnew) ≥ minSup)
            Set Pnew as a closed pattern;
          If (sup(P) = sup(Pnew))
            Set P as a non-closed pattern;
          Else
            ISCLOSED(Pnew, pretree);
          Add Pnew into pretree as a itemset-extended child node of P;
//..........
EXTENDSEQUENCE(P, pretree)
  For each 1-pattern Pi in SPs
        Pnew is a new node created by adding Pi as last itemset into P and using block encoding
        based on prism factorization in [3] to count the support;
        If (sup(Pnew) ≥ minSup)
            Set Pnew as a closed pattern;
        If (sup(P) = sup(Pnew))
            Set P as non-closed pattern;
        Else
            ISCLOSED(Pnew, pretree);
            Add Pnew into pretree as a sequence-extended child node of P;
//..........
ISCLOSED (Pnew, pretree)
  For each node P in pretree
    If Pnew is a supersequence of P
      If P is a closed pattern and sup(P) = sup(Pnew)
          Set P as a non-closed pattern;
    subtree = the tree which rooted at P;
    ISCLOSED(Pnew, subtree);
```

Figure 1. The pseudo code for generating set of closed sequential patterns.

5. EXPERIMENTAL RESULTS

All experiments were performed on PC with a core i5 2.6 GHz CPU and 8 GB of RAM running Windows 10 and implemented using C# (2012). The experiments were performed on synthetic and real databases, namely *C6T5S4I4N1kD1k, Chess*, and *Mushroom*. *C6T5S4I4N1kD1k* was generated using the synthetic data generator developed by IBM to mimic transactions in a retail environment with the following parameters: *C*, the average number of itemsets per sequence, was set to 6 (denoted as *C6*); *T*,

the average number of items per itemset, was set to 5 (denoted as *T5*); *S*, the average number of itemsets in maximal sequences, was set to 4 (denoted as *S4*); *I*, the average number of items in maximal sequences, was set to 4 (denoted as *I4*); *N*, the number of distinct items, was set to 1,000 (denoted as *N1k*); and *D*, the number of sequences, was set to 1,000 (denoted as *D1k*). *Chess* and *Mushroom* databases were downloaded from http://fimi.ua.ac.be/data/, where each itemset in a sequence of these databases is a single item. The *Chess* database includes 3196 sequences with 76 distinct items and the *Mushroom* database includes 8124 sequences and 120 distinct items.

Table 1 shows the number of sequential patterns, closed sequential patterns, and the mining time of the *CloSpan* [15] and the proposed (*MCSP_PreTree*) algorithms with corresponding minimum support thresholds in sequence databases *C6T5S4I4N1kD1k, Chess,* and *Mushroom*. From the achieved results in Table 1, we see that the number of closed sequential patterns in three databases is fewer than the number of sequential patterns a lot especially when mining with low support thresholds. For example, when mining with *minSup* is 10% for the sequence database *Mushroom*, the number of closed sequential patterns is 29756 while the number of sequential patterns is 113203.

Table 1. Number of sequential patterns, closed sequential patterns and mining time.

Database	minsup	Sequential patterns	Closed Sequential patterns	Mining time	
				CloSpan	MCSP_PreTree
Chess	80	8227	5113	278.6	139.0
	75	21000	11598	948.2	482.6
	70	49020	24763	2754.4	1449.2
	65	112103	53309	7422.0	4359.7
Mushroom	40	499	221	43.9	21.7
	30	1777	736	174.5	86.4
	20	9003	3273	1035.7	509.1
	10	113203	29756	12101.6	6625.3
C6T5S4I4N1kD1K	0.6	20644	20347	1685.9	1463.6
	0.5	31311	30599	2632.1	2103.7
	0.4	54566	51639	4866.1	3460.9
	0.3	124537	105300	13159.8	9988.5

On the basis of the experimental results in Figure Table 1, we can see that our *MCSP_PreTree* algorithm outperforms the *CloSpan* algorithm by more than an order of magnitude, especially when mining with low support, the number of sequential patterns and closed sequential patterns generated from sequence databases is large. Because the *CloSpan* algorithm must spend more time for doing the backward subpattern and backward superpattern check on the set of historical closed sequence candidates to verify if a newly found frequent sequence is promising to be closed or not while the *MCSP_PreTree* algorithm applies the parent–child relationship between nodes on prefix tree to determine whether a sequential pattern to be a closed sequential pattern by adding *IsCSP* field into each sequential pattern node on the prefix-tree. For example, considering that the sequence database *Chess* with *minSup* is 65%, the number of sequential is 112103, the number of closed sequential patterns is 53309, the time to mine closed sequential patterns of the *MCSP_PreTree* algorithm is 4359.7 seconds while that of the *CloSpan* algorithm is 7422.0 seconds. The details of the experimental results for this database are shown in table 1.

6. CONCLUSIONS AND FUTURE WORK

This paper introduced an efficiently algorithm called *MCSP_PreTree* and experimental results for mining closed sequential patterns algorithm. This algorithm combined the parent–child relationship between nodes on prefix tree and the definition of closed sequential pattern to determine whether a sequential pattern is a closed sequential pattern by adding *IsCSP* field into each sequential pattern node on the prefix tree. The algorithm also applied the prime block encoding approach and the join operations over the prime blocks in [3] for generating candidate sequences and determining the frequency for each candidate. Based on this algorithm, we have performed more experimental results with the various lsequence databases such as synthetic and real databases, namely C6T5S4I4N1kD1k, Chess, and Mushroom and so on. C6T5S4I4N1kD1k was generated using the synthetic data generator developed by IBM to mimic transactions in a retail environment Chess and Mushroom databases were downloaded from http://fimi.ua.ac.be/data/. Experimental results are examined on the sequence database also showed that the number of closed sequential patterns is much smaller than that of the sequential patterns, and the mining time of the *MSCP_PreTree* algorithm is also better that of *CloSpan* for mining the closed sequential patterns.

In future, by using these experimental results, we will generate sequential rules. Besides, the lattice-based approach has been proposed for mining association rules and classification association rules in recent years [5,13]. We will study how to apply this approach for mining sequential patterns, closed sequential patterns and sequential rules in the future.

ACKNOWLEDGEMENTS

This research is funded by Industrial University of Ho Chi Minh City under grant number 182.CNTT03.

REFERENCES

[1] Agrawal R. and Srikant R. - Mining Sequential Patterns. In: Proc. of 11th Int'l Conf. Data Engineering, DC, USA, pp. 3 –14, 1995.

[2] Ayres J., Gehrke J.E., Yiu T. and Flannick J. - Sequential Pattern Mining using a Bitmap Representaion. In: SIGKDD Conf., NY, USA, pp. 1–7, 2002.

[3] Gouda K., Hassaan M. and Zaki M.J. - PRISM: A Primal-Encoding Approach for Frequent Sequence Mining. In: Journal of Computer and System Sciences, 76(1), pp. 88-102, 2010.

[4] Huang G.-Y., Yang F., Hu C.-Z. and Ren J.-D. - Fast Discovery of Frequent Closed Sequential Patterns based on Positional. Proc. of the International Conference on Machine Learning and Cybernetics, ICMLC 2010, Qingdao, China, pp. 444 – 449, 2010.

[5] Nguyen L.T.T, Vo B., Hong T.P. and Thanh H.C. - Classification based on association rules: A lattice-based. Expert Systems with Applications, 39(13), pp. 11357 –1136, 2012.

[6] Pei J., Han J. and Mao R. - CLOSET: An efficient algorithm for mining frequent closed itemsets. In DMKD'01 workshop, Dallas, TX, 2001.

[7] Pham T.-T., Luo J. and Vo B. - An effective algorithm for mining closed sequential patterns and their minimal generators based on prefix trees. International Journal of Intelligent Information and Database Systems, 7(4), 324-339, 2013.

[8] Pham T.T., Luo J., Hong T.-P. and Vo B. - An Efficient Method for Mining Non-Redundant Sequential Rules Using Prefix-Trees, 32, 88 - 99, 2014.

[9] Pei J., et al - Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. Knowledge and Data Engineering, 16(10), pp. 1424 –1440, 2004.

[10] Srikant R. and Agrawal R. - Mining Sequential Patterns: Generalizations and Performance Improvements. In: Proc. of 5th Int'l Conf. Extending Database Technology, London, UK, pp.3–17, 1996.

[11] Thilagu M., Nadarajan R., Ahmed M.S.I. and Bama S.S. - PBFMCSP: Prefix Based Fast Mining of Closed Sequential Patterns. In The International Conference on Advances in Computing, Control, and Telecommunication Technologies ATC'09, Trivandrum, Kerala, India, pp. 484 – 488, 2009.

[12] Tzvetkov P., Yan X. and Han J. - TSP: Mining Top -K Closed Sequential Patterns. Knowledge and Information Systems, 7(4), pp. 438-457, 2005.

[13] Vo B. and Le B. - Interestingness measures for association rules: Combination between lattice and hash tables. Expert Systems with Applications, 38(9), pp. 1630 - 11 640, 2011.

[14] Wang J. and Han J. - BIDE: Efficient mining of frequent closed sequences. In proc of the 20th Int' Conf on Data Engineering (ICDE95): IEEE Computer Society Press, DC, USA, pp. 79-91, 2004.

[15] Yan X., Han J. and Afshar R. - CloSpan: Mining closed sequential patterns in large datasets. In Proc of the 3th SIAM International Conference on Data Mining, San Francisco, CA, USA: SIAM Press, pp. 166 -177, 2003.

[16] Zaki M.J. - SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning Journal, 42(1/2), pp. 31- 60, 2000.

[17] Zaki M.J. and Hsiao C. - CHARM: An efficient algorithm for closed itemset mining, In SDM '02, Arlington, VA, pp. 457 - 473, 2002.

KHAI THÁC CÁC MẪU TUẦN TỰ ĐÓNG HIỆU QUẢ SỬ DỤNG CÂY TIỀN TỐ

Tóm tắt. Khai thác mẫu tuần tự đóng là một trong những công việc quan trọng trong lãnh vực khai thác dữ liệu. Khai thác mẫu tuần tự đóng được đề xuất để giải quyết các vấn đề tiêu hao bộ nhớ và thời gian khai thác trong khai thác mẫu tuần tự từ cơ sở dữ liệu chuỗi cụ thể như khi khai thác với chuỗi tuần tự phổ biến dài sẽ chứa một tổ hợp lớn các chuỗi con phổ biến hoặc khi sử dụng các ngưỡng hỗ trợ rất thấp để khai thác các mẫu tuần tự thì số lượng mẫu tuần tự rất lớn và tốn nhiều thời gian để khai phá hơn. Vì vậy, việc sử dụng mối quan hệ cha con trên cấu trúc cây tiền tố để cải tiến hiệu suất của quá trình khai thác mẫu tuần tự từ cơ sở dữ liệu chuỗi cũng là một phương pháp quan trọng trong khái thác dữ liệu. Bằng cách sử dụng mối quan hệ cha con trên cấu trúc cây tiền tổ, bài viết này đưa ra một thuật toán hiệu quả cho việc khai phá các mẫu tuần tự đóng từ cơ sở dữ liệu chuỗi. Các kết quả trong phần thực nghiệm cho thấy hiệu suất thời gian chạy của thuật toán đề xuất rất lớn và số lượng các mẫu tuần tự cũng giảm đáng kể.

Từ khóa. Sequential pattern, closed sequential pattern, prefix tree, sequence database.

Ngày nhận bài: 17/06/2017

Ngày chấp nhận đăng: 11/08/2017