International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 6 394 - 397

Efficient Mining of Sequential Patterns in a Sequence Database with Weight Constraint

A. Sirisha
Assistant Professor, Department of IT
Chaitanya Bharti Institute of Technology
Hyderabad, India
e-mail: asirishanaidu@gmail.com

Suresh Pabboju
Professor, Department of IT
Chaitanya Bharti Institute of Technology
Hyderabad, India
e-mail:plpsureshrs@gmail.com

G. Narsimha
Associate Professor, Department of CSE
JNTUH, College of Engineering
Hyderabad, India
e-mail: narsimha06@gmail.com

***Abstract*—** Sequence pattern mining is one of the essential data mining tasks with broad applications. Many sequence mining algorithms have been developed to find a set of frequent sub-sequences satisfying the support threshold in a sequence database. The main problem in most of these algorithms is they generate huge number of sequential patterns when the support threshold is low and all the sequence patterns are treated uniformly while real sequential patterns have different importance. In this paper, we propose an algorithm which aims to find more interesting sequential patterns, considering the different significance of each data element in a sequence database. Unlike the conventional weighted sequential pattern mining, where the weights of items are preassigned according to the priority or importance, in our approach the weights are set according to the real data and during the mining process not only the supports but also weights of patterns are considered. The experimental results show that the algorithm is efficient and effective in generating more interesting patterns.

Keywords- *Data mining;Apriori property; Sequential Pattern Mining;Weighted sequential pattern;weight constraint*

*****

I. INTRODUCTION

Sequential pattern mining is one of the most important data mining techniques, which has drawn a great number of researches to pursue it, because of its numerous applications including usage in customer purchase behavior analysis, webpage access pattern detection, disease treatment pattern analysis, DNA sequence analysis and stock sequence analysis. In general, sequential pattern mining finds all frequent sequential patterns whose occurrence frequency is no less than the given threshold frequency from a sequence database. The sequential pattern mining problem was first introduced by Srikanth and Agrawal [1]. Later many Sequential pattern mining algorithms have been extensively developed due to its huge applications. The previous sequential pattern mining approaches do not reflect the characteristic of real datasets. i.e they consider the sequential patterns and items in a sequential pattern uniformly However, they have different importance in real world applications. For example, when finding the traversal patterns in the World Wide Web, different pages can have different importance. In biomedical and DNA data analysis, some genes are more important than others in causing a particular disease and some genes are more effective than others in fighting diseases. With these observations a new approach is proposed in this paper. In this approach weights are assigned to the items according to the real data to reflect their importance, weight range is used to restrict the weight values of items and the values are normalized. During mining the mean weight is used to prune the weighted infrequent patterns. The effectiveness of this approach is analyzed in terms of number of resulting sequential pattern and processing time. And the experiment results have showed that that the number of weighted sequential patterns can be easily adjusted by setting a weight range and the runtime is efficient.

The rest of this paper is organized as follows: Section 2 gives a brief summary of related work, and a problem definition is given in Section 3. In Section 4, a new approach for finding weighted sequential patterns is described. Section 5 summarizes a series of experimental results, and finally conclusions are described in Section 6.

II. RELATED WORK

Many algorithms have been proposed for mining sequential patterns. And most of the basic and earlier algorithms are based on the Apriori property that any super-pattern of a nonfrequent pattern cannot be frequent. The apriori-based methods like GSP[4] have some drawbacks of generating large sets of candidate sequences and performing multiple scans of a data set to get a resulting set of frequent sequences. To overcome these drawbacks a series of data projection based algorithms FreeSpan[2] and PrefixSpan[3] were proposed. PrefixSpan is one of the pattern-growth approaches that recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only the locally frequent fragments. However, the PrefixSpan method also partially requires multiple scans of data sets, i.e., projected data sets. Among the algorithms for sequential pattern mining, SPADE [5] and PrefixSpan are more efficient than others in terms of processing time. SPADE is one of the vertical-format based algorithms and uses equivalence classes in the mining process. All the sequential pattern mining algorithms [6, 7, 8, 10] suggested so far have given same importance to the sequences and the elements in a sequence. However, it is important to distinguish important sequences from a large number of sequence patterns. To solve this problem Wspan[9] algorithm is proposed but in this algorithm the weights of the items are set manually based on the domain knowledge.

III. PROBLEM DEFINITION

The problem of sequential pattern mining is to find the complete set of sequential patterns in the sequence database with a support constraint. The most general form of the sequence mining problem can be stated as follows:

Let I = {i_1, i_2,..., i_m} be a set of m-distinct items. An itemset s_j is a set of items, $s_j \subseteq I$.

A sequence S is an ordered list of itemsets denoted by S=<s_1, s_2, …,s_n> where s_j is an itemset, also called an element of the sequence. Each itemset in a sequence represents a set of events occurring at the same timestamp, while different itemsets occur at different times. We assume that the items in each itemset are sorted in certain order. An item can occur at most one time in an element of a sequence but it can occur multiple times in different elements of a sequence. The size of a sequence is the number of elements in the sequence. The sequence with a total of k items is referred to as a k-length sequence or k-sequence, and each occurrence of the items in the sequence is counted.

A sequence $S = <s_1, s_2,\ldots, s_m>$ is a *subsequence* of another sequence $T = <t_1, t_2,\ldots, t_n>$, denoted by $S \subseteq T$ if and only if there exists set of integers $i_1, i_2,\ldots, i_m$, such that $1 \le i_1 < i_2 < \ldots < i_m \le n$ and $s_1 \subseteq t_{i1}, s_2 \subseteq t_{i2},\ldots, s_m \subseteq t_{im}$. A sequence database is a set of tuples <sid, S> where sid is a sequence identifier and S is a sequence. Table 1 shows an example sequence database. It consists of four sequences, and their sids are 10, 20, 30, and 40, respectively.

For a sequence database, a tuple <sid,S> is said to contain a sequence T if T is a subsequence of S. The support of a sequence T in a sequence database is the number of tuples in the database containing T. If the support of a sequence *S* satisfies a pre-specified *min_sup* threshold, *S* is a *frequent* sequential pattern.

TABLE 1
A SEQUENCE DATABASE AS A RUNNING EXAMPLE

Sid	Sequence
10	<(*A B C D E F*) (*A E G*)>
20	<(A B C D F) (B C E F)>
30	<(*A B C E F H*) (*A D*)>
40	<A (ABD) B C>

Example 1. Table 1 shows the input sequence database *SDB* in our running example. Assume that a *min_sup* is 2. The *SDB* has 8 unique items, and 4 input sequences. The *size* of < (*A B C D E F*) (*A E G*)> is 2 and the *length* of this sequence is 9. Sequence < (*B C E F*) (A)> is a sub sequence of <(*A B C D E F*) (*A E G*)>since(*B C E F*) $\subseteq$(*A B C D E F*), (A) $\subseteq$ (*A E G*). And the sequence <(*B C E F*) (A)> is a frequent sequential pattern because sequences 10 and 30 contain it as a subsequence and the support (2) of the sequence is no less than the *min_sup* 2.

IV. MINING WEIGHTED SEQUENTIAL PATTERNS

In this section, we propose a new approach to find weighted sequential pattern in which the weight constraint is pushed deeply into sequential pattern mining. In our approach, we first assign weight values to the items as per the real data , and then we define weighted sequential patterns and pruning conditions. Based on the approach, we present an algorithm for detecting weighted sequential patterns.

A. *Weighted Sequential Pattern*

Weight of an item is a non-negative real number that reflects it's importance. For example, in market basket data to set up weights of retail items, an attribute value prices of items can be used .However, the real values of items are not suitable for weight values because of the variations among the data from various sources. So we use normalized weights in our approach.

Let *i* be a single item, S_1, S_2, …, S_n are *n* sequences in sequence database *SDB*, the weight of *i* is defined as: $W(i) = T(i) / \Sigma_{1 \le j \le n} L(S_j)$, where $T(i)$ is the total number of occurrences of *i* in *SDB*, $L(S_j)$ is the length of sequence S_j. For example, the occurrence of single item *A* in the SDB in Table 1 are 2, 1, 2,2 respectively, and the lengths of the four sequences are 9, 9, 8,6 respectively, thus, the weight of *A* is $W(A) = (2+1+2+2)/(9+9+8+6)=0.21875$.

Definition 1: Weight of a Sequence
Let $S = <I_1, I_2,\ldots, I_m>$ be a sequence, I_k $(1 \le k \le m)$ is an element of *S*, composed of *n* single items $i_1, i_2, \ldots, i_n$, the weight of I_k is defined as $W(I_k) = \Sigma_{1 \le j \le n} W(i_j)/n$. While the weight of *S* is defined as $W(S) = \Sigma_{1 \le k \le m} W(I_k)/m$.
For example, the weight of an element <(*A B C D F*) of the sequence with sid 10 in Table 1 is $(W(A)+W(B)+W(C)+W(D)+W(F))/5$ and weight of the sequence <(*A B C D F*) (*B C E F*)> in Table 1 is $(W(A\ B\ C\ D\ F) + W(B\ C\ E\ F)) / 2 = 0.426$(approx.)

Let ***SDB*** be a sequence database composed of *n* single items i_k $(1 \le k \le n)$, we define maximal weight as $Max\_W = max_{1 \le k \le n} (W(i_k))$, and minimum weight as $Min\_W = min_{1 \le k \le n} (W(i_k))$ respectively. Then the mean weight is defined as $Mean\_W = (Max\_W + Min\_W)/2$.

Definition 2: Frequent weighted sequential pattern
A sequence *S* is called a frequent weighted sequential pattern, if and only if $sup(S)*Mean\_W \ge min\_sup$, *otherwise S* is called an infrequent weighted.

B. *Mining Sequential Pattern with Weight Constraint*

In our approach, we pushed the weight constraint into the prefix projected sequential pattern growth approach. A sequence database is recursively projected into a set of smaller weighted projected databases and weighted sequential patterns are grown in each weighted projected database. Given a sequential pattern α in a sequence database, α-projected database (S|α) is the collection of suffixes of sequences in S with the prefix α. The support ($support_{s|\alpha}(\beta)$) of a sequential pattern β in the α-projected database (S|α) is the number of weighted sequences γ in S|a.

SPMW Algorithm: Sequential patterns mining with weight constraint from large sequence databases.

Input: A sequence database: SDB, The minimum support threshold: min_sup.
Output: The complete set of weighted sequential patterns.
Begin

1. Let WSP be the set of Weighted Sequential Patterns. Initialize WSP is {};

2. Scan SDB once, count the support of each item, check the weight of each item and find each weighted frequent item β, in sequences satisfying the following condition:

support * Mean_W ≥ min_sup

3. For each weighted frequent item β, in SDB
Call SPMW (WSP, <β>, 1, SDB)
End for
End

Procedure SPMW (WSP, α, L, S|α)
Parameters:

(1) α is a weighted sequential pattern,
(2) L is the length of α
(3) S|α is the sequence database SDB if α is null, otherwise, it is the α-projected database.

1. Scan S|α, once, count the support of each item, and find each weighted frequent item, β in sequences.
β is a weighted sequential item if the following pruning condition not satisfied.
Pruning condition: (support * Mean_W < min_sup)
(a) β can be assembled to the last element of α to form a sequential pattern or
(b) <β> can be appended to α to form a sequential pattern.

2. For each weighted frequent item β, Add it to α to form a sequential pattern α ', and output α '.
End for

3. For each α ', Construct α '-projected database S|α`;
Call SPMW(α`, L+1, S|α`)
End for.

V. PERFORMANCE EVALUATION

After the text edit has been completed, the paper is ready In this section we present our performance study based on the experiment conducted on a randomly generated data set of 50,000 transactions generated by dataset generator class using an applet shown in Fig 1. This applet takes the total number of different items that can be present in the database, the maximum number of items that can be present in an itemset and the maximum number of itemsets in a sequence. The total numbers of transactions are also entered. The main purpose of this experiment is to demonstrate how effectively the weighted sequential patterns can be generated by incorporating a weight measure along with a support measure. The experiment was performed on a machine operating at 2GHZ with 4GB of memory and the algorithm was implemented in java. We compared our approach with PrefixSpan for performance evaluation.

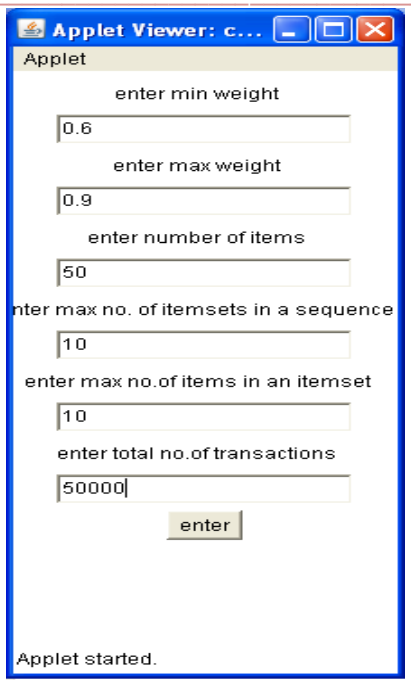

Fig. 1. Applet for generating Data set

TABLE 2
TEST RESULTS SHOWING THE FREQUENT PATTERNS GENERATED FOR DIFFERENT SUPPORTS

Support threshold	0.07	0.08	0.1
Prefix Span	2550	1430	50
Spmw 0.6-0.9	580	50	50
Spmw 0.8-1.2	2550	1229	50

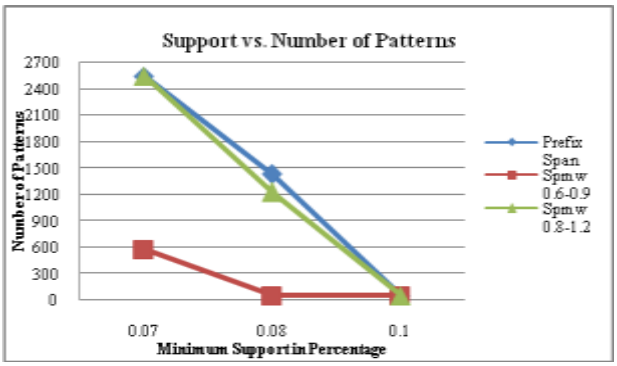

Fig. 2. Support v/s Number of Frequent Patterns

Figs. 2 and 3 show that our algorithm SPMW generates fewer sequential patterns and runs faster than Prefixspan. Specifically, from the Table 2 it is observed that much fewer sequential patterns are generated as the weight range is decreased. SPMW generates fewer patterns than Prefixspan by adjusting the weight range. Table 3 shows that Prefixspan is slower as the minimum support is decreased and SPMW is faster than Prefixspan.

TABLE 3

TEST RESULTS SHOWING THE EXECUTION TIME FOR DIFFERENT SUPPORTS

Support threshold	0.06	0.07	0.08
Prefix Span	303629	296224	190733
Spmw 0.6-0.9	115813	24235	24297
Spmw 0.8-1.2	211391	212422	114594

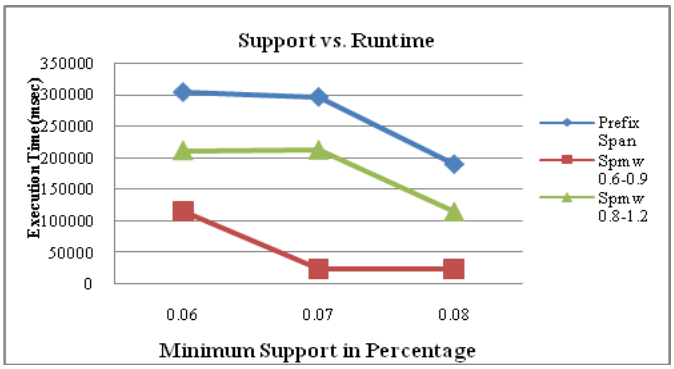

Fig. 3. Support v/s Execution Time

VI. CONCLUSION

The main limitations of the traditional approach for mining sequential patterns is that all items are treated uniformly, while real items have different characteristics and they generate a very large number of patterns as the minimum support becomes lower. Although frequent closed sequential pattern mining algorithms have been suggested, in large databases, they still generate too many patterns when support is low or the sequential pattern becomes long. In this paper, we described an approach to mine more interesting sequential patterns. Our algorithm SPMW focuses on mining the weighted frequent patterns based on the prefix projected sequential pattern growth approach. This algorithm mixes the Prefixspan algorithm and the weights to produce the projected database which has fewer itemsets. In our algorithm, the weights are calculated according to real data, rather than set by experts and a weight range is used to adjust the number of sequential patterns. The experiment results have showed that it is more efficient and scalable.

REFERENCES

[1] R. Agrawal, R. Srikant, *"Mining sequential patterns",* in Proceedings of the 1995, International Conference on Data Engineering (ICDE '95), 1995, pp. 3–14.

[2] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, *"FreeSpan: frequent pattern-projected sequential pattern mining"* in Proceedings of the 2000 ACM, SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00), 2000, pp. 355–359.

[3] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, *"Mining sequential patterns by pattern-growth: the PrefixSpan approach"*, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) pp.1424–1440.

[4] R. Srikant, and R. Agrawal, *"Mining Sequential Patterns: Generalizations and Performance Improvements"*, EDBT, 1996.

[5] M.J. Zaki, *"SPADE: an efficient algorithm for mining frequent sequences",* Machine Learning 42 (1/2) (2001) pp. 31–60.

[6] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, *Sequential Pattern Mining using A Bitmap Representation"*, IGKDD'02, 2002.

[7] H. Cheng, X. Yan, and J. Han, *"IncSpan: Incremental Mining of Sequential Patterns in Large Databases",* SIGKDD'04, 2004.

[8] M. Ester, *"A Top-Down Method for Mining Most Specific Frequent Patterns in Biological Sequence Data"*, SDM'04, 2004.

[9] U. Yun and John J. Leggett ,*"WSpan: Weighted Sequential pattern mining in large sequence databases"*,3rd International IEEE Conference on Intelligent Systems,September 2006, pp. 512–517.

[10] X. Yan, J. Han, and R. Afshar, *"CloSpan: mining closed sequential patterns in large databases",* Proc. 3rd SIAM Intl. Conf. Data Mining (SDM 03), SIAM, May 2003, pp. 166-177.