Recursive Prefix Suffix Pattern Detection Approach for Mining Sequential Patterns

Dr. P.Padmaja
Associate professor
Department of IT
GITAM University
Visakhapatnam, INDIA

P.Naga Jyothi
Assistant professor
Department of IT
GITAM University
Visakhapatnam, INDIA

M.Bhargava
Department of IT
GITAM University
Visakhapatnam, INDIA

ABSTRACT
The need for real-time data mining has long been recognized in various application domains. Some of the applications like customer shopping sequences, medical treatments, natural disasters, telephone calling patterns, Weblog click streams, DNA sequences and gene structures require sequential pattern mining techniques. These techniques find the complete set of frequent subsequences for the given set of sequences and support threshold. Traditional pattern growth based approaches for sequential pattern mining derive length (n+1) pattern based on projected databases of length n-patterns recursively. As result lot of recursions occur which may lead to certain complexities. Thus, in order to reduce the number of iterations, an efficient bidirectional sequential pattern mining approach namely Recursive Prefix Suffix Pattern detection, RPSP algorithm is proposed. The RPSP algorithm first finds all Frequent Itemsets (FI's) according to the given minimum support and transforms the database such that each transaction is replaced by all the FI's it contains and then finds the patterns. The pattern further is detected based on i^{th} projected databases, and constructs suffix and prefix databases based on the apriori property. RPSP will increase the number of frequent patterns by reducing the minimum support and vice versa. Recursion is terminated when the detected FI set of prefix or suffix projected database of parent database is null. All the patterns that correspond to a particular i^{th} projected database of transformed database are formed into a set, which is disjoint from all other sets. The union of all the disjoint subsets is the resultant set of frequent patterns. The proposed algorithm was tested on the hypothetical data and results obtained were found satisfactory. Thus, RPSP algorithm can be applicable to many real world sequential data sets.

Keywords - Sequential patterns, Frequent itemsets, Prefix and Suffix databases, Projected database, Transformed database.

1. INTRODUCTION
Data mining , is a powerful tool for extraction of hidden predictive information from large databases. Data mining tools predict future trends and behaviours, allowing businesses to make proactive, knowledge-driven decisions. It is the collection of concepts and techniques for uncovering the interesting data patterns hidden in large data sets. One of the popular data mining techniques is Association rule mining, is a process of identifying the dependency of one item with respect to the occurrence of the other item. Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.Some of the applications like, Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering and classification etc.

Sequential pattern mining is more difficult than association rule mining because the patterns are formed not only by combining items but also by permuting item-sets. Enormous patterns can be formed as the length of a sequence is not limited and the items in a sequence are not necessarily distinct. Owing to the challenge posed by exponential possible combinations, improving the efficiency of sequential pattern mining has been the focus of recent research in data mining. Sequential pattern mining is an important Data Mining problem which detects frequent sub- sequences in a sequence database [1][2]. In this proposed approach certain frequent Sequential Patterns at faster pattern growth by recursively determining the prefix-projected database and suffix-projected database for every ith database and determine the frequent patterns containing i. The final set of frequent sequential patterns is the union of the disjoint subsets[8]. Sequential pattern mining problem can be widely used in different areas, such as mining user access patterns for the web sites, using the history of symptoms to predict certain kind of disease, also by using sequential pattern mining, the retailers can make their inventory control more efficient[5].Which results efficient mining and better scale up property than conventional algorithms.

The rest of the paper is organised as follows section II presents proposed approach and its algorithm, experimental study, performance evaluation and time complexity are presented in Section III. Finally, conclusion and future work in section IV.

2. PROPOSED APPROACH
In this section the proposed Recursive Prefix Suffix Pattern based detection algorithm (RPSP) is discussed. The RPSP algorithm has two major steps. In the first step the database is transformed by finding the frequent items with the support larger than the minimum support as per the apriori principle [3][4]. In the second step the algorithm partitions the patterns into prefix and suffixes of the each frequent item. The algorithm proceeds by constructing the prefix and suffixes of the projected databases, without the construction of DAG(Directed Acyclic Graph) [8]. The proposed approach finally detects patterns of each subset by recursively calling prefix and suffix subroutines.

2.1 RPSP Algorithm
1. Start
2. Read the Sequence Database (DB) and Minimum Support(MS)
3. FI= Find_FI (DB, MS, true)
4. Assign indexes to frequent itemsets after they are being arranged in lexicographic order.

5. Transform the DB into Transform DataBase(TDB)
6. P= Ø
7. For each 'x' in FI do
Px= Ø
xDB = call get_iDB(x,TDB,true)
PPx = call Prefix (xDB, MS, x)
SPx = call Suffix(xDB, MS, x)
PSPx= call Combination_of (x,PPx,SPx,TDB,MS)
Px= union of (PPx, SPx, PSPx, x)
End of for
8. P=Union of (P, Px)
9. Stop.

The RPSP algorithm finds Frequent Items (FI's) in the transaction database with given minimum support(MS) and transforms the DataBase(DB) by assigning unique Identification numbers (Id's) to the corresponding FI's .The algorithm further proceeds by finding the projected databases for each FI.

2.1.1 Find_FI(String tempDB[][], int ms, Boolean ismultiple)

1. tempFI[] = Get all the single items that have count greater than the ms .
2. if (ismultiple)
3. tempFI2[]=Get all the multiple item itemsets whose count is greater than the ms support value4. tempFI[] = tempFI[] + tempFI2[] (concatenation)
5.Perform String Sort on the tempFI[]
6. Return tempFI.

The subroutine Find_FI() finds the frequent itemsets in the input database. It takes the database from which FI's are to be found, minimum support and a Boolean Ismultiple as parameters and returns the set of frequent itemsets. If the ismultiple value is true then even number of itemsets are considered as candidates for frequent itemsets. If the value is false then they are not considered as candidates for frequent itemsets[11]. Therefore, with the candidates obtained for each corresponding FI's, the subroutine proceeds with detection of prefix and suffix projected databases[12].

2.1.2 Prefix(tempDB,MS,temp)

1.TDB = call get_iDB(tempDB,temp,false)
2.PDB = get all the itemsets before to last occurance of temp, and form them into a database
3.FI= call Find_FI(PDB,MS,false)
4.Repeat until FI=Ø
Py=Ø
For each 'y' in FI
yBD= call get_iDB(y,PDB,false)
PPy= call Prefix(yDB,MS,y)
SPy= call Suffix(yDB,MS,y)
PSPy= call Combination_Of(x,PPy,SPy,PDB,MS)
Py= union of(PPy, SPy, PSPy, y)
End of for
5. Return Py

2.1.3 Suffix(tempdDB, MS ,temp)

1.call Suffix(tempdDB, MS ,temp)
2TDB = call get_iDB(tempDB,temp,false)
3.SDB = {form a Database where all the tuples correspond the respective tuples in Step7.4.3: TDB such that each tuple contains the itemset that occur after 'temp' in TDB tuple
4.FI = call Find_FI(SDB,Ms,false)
5.Repeat till FI=Ø
Py=Ø
For each 'y' in FI do
yDB= call get_iDB(y,SDB,false)
PPy= call Prefix(yDB, Ms,y)
SPy= call Suffix(yDB,MS,y)
PSPy=call Combination_Of(y,PPy,SPy,PDB,MS)
Py= union of(PPy, SPy, PSPy, y)
End of for
6. Return Py.
The subroutine Prefix(tempDB,MS,temp) detects all FI's which are prefixes to the ith projected databases .Similarly Suffix(tempdDB, MS ,temp) [10]detects all patterns which are suffixes to the ith projected databases. The prefix and suffix pattern construction process continues until database becomes empty and FI reaches to null set of all itemsets.

2.1.4 get_idB(String temp,String[][] tempDB, Boolean isGre)

1.Temp1dB = {get all the tuples that contain atleast one item 'temp'From tempDB}
2.If (isGre)
Temp1dB = { set of tuples which is resultant after removing all the itemsets that are numerically greater than 'temp' in Temp1dB}
Return Temp1dB.

The subroutine get_iDB() takes the database, the root value for which the ith Database is to be constructed and a Boolean variable isGre as Parameters and returns the corresponding ith database. If isGre value is true then all the items that are greater than the root value are ignored in the resultant ith database otherwise, the items are included.

2.1.5 Combination_Of(String root, String a[], String b[], String tempDB[][], int MS)

1. Let x, y represent an item in a[], b[] respectively.
2. String abc[]<-- get all possible combinations of x and y for all possible values of x and y. The retrieved combination should be like x + root + y
3. abc[]<--get all the patterns(entries) from abc[] whose count in tempDB[] is greater than the given MS value
Return abc.

The subroutine Combination_Of () takes the root element, prefixes patterns, suffix patterns, parent database and the minimum support value as parameters and returns the bidirectional patterns. It forms all the possible combinations for every possible value in prefix patterns and the suffix

patterns along with the root element. The concatenation of Prefix Pattern root and Suffix Pattern root along with the root (PProot + root + PSroot) is checked with the count and minimum support in the parent database. If the count is greater than minimum support the final pattern set is returned as a bidirectional frequent pattern otherwise rejected.

3. EXPERIMENTAL STUDY AND ANALYSIS

To evaluate the effectiveness of RPSP algorithm, it is tested on hypothetical and real world data sets. The results were found satisfactory.

3.1 Experimental Study on hypothetical data:

Input hypothetical Data Base

Seq Id	Sequence
1	<(1) (1,2,3) (4) (7,8) (3)>
2	<(3) (5,8,9) (1,2) (2,3)>
3	<(5) (1,2) (3,5,6) (1,2) (6)>

For the above given sequence database the detected Frequent Itemsets (FIs) are (1), (1,2), (2), (3). Unique Id is assigned to each FI as follows :- (1)-1, (1,2)-2, (2)-3, (3)-4[7]. The algorithm consists of two major steps.

3.1.1 Transformation of Database: In this step the detected FI's with their assigned id's are transformed and infrequent items are eliminated. Thus, the transformed database is 1: < (1) (1,2,3,4) (4) > ; 2: < (4) (1,2,3) (3,4) > ; 3: < (1,2,3) (4) (1,2,3) >.

3.1.2 Pattern Partitioning: With the detected FI's the number patterns are chosen, and then algorithm proceeds in finding their prefix and suffixes of each projected database. In the mentioned example the number of detected FIs are 4, and thus the required final Patternset (P) consists of 4 disjoint subsets i.e., from P_1, P_2,P_3 and P_4, where Pi includes 'i^{th}' element and element less than 'i^{th}' element .[6]

1) Finding subsets of patterns

a) The construction of pattern set P starts with subset P_4 and ends up with the subset P_1. In the process of finding the 4^{th} projected database(P_4) the transformed 4^{th} DB is {1 : <(1)(1,2,3,4)(4)> ;2 :<(4),(1,2,3)(3,4)>;3 <(1,2,3)(4)(1,2,3)> }. Prefix of 4^{th} DB is {1: < (1) (1,2,3,4) >; 2: < (4) (1,2,3) > ; 3: < (1,2,3) >} and Suffix of 4^{th} DB is {1: <(4)> ;2: <(1,2,3) (3,4)> ;3: <(1,2,3)>} . Thus, FI set contains {3} as there are no frequent patterns for suffix of the 4th projected data base. But, the prefix of the 4th projected database is having frequent items in 3db,2db and 1db. Therefore, the FI's for 3db {i.e 1: <(1)(1,2,3)>; 2: <(1,2,3)(3)>; 3 :<(1,2,3)(1,2,3)>;} has prefixes and suffixes as follows : pre(3pre (4D)) is 1: <(1)> ;2: <4>; suf (3pre (4D)).Frequent patterns are not found further as there are no databases to process 3db [9][11] .

For pre(2pre (4D)) as 1:<(1)> ;2:<(4)> : suf (2pre (4D)) no suffix items for 1db with (1pre(4D) are pre(1pre (4D)) 1: <(1)> ;2- <4> ;suf (1pre (4D)) 1:<(1,2,3,4).Thus, P_4:<1 4> <2 4> <3 4> <4>;

b) Similarly patterns for P_3, P_2, P_1 are found as P_3 : <1 3> <3> ; P_2:<2> ;P_1 <1> .

Thus, the final pattern set P is the union of all disjoint subsets (P_4, P_3, P_2, P_1)of FI set.

P: {<1 3> ,<1 4> ,<1> ,<2 4> ,<2> ,<3 4>, <3> ,<4>} .

3.2 Performance Evaluation and Time complexity:

This subsection deals with impact of different parameters in the dataset on the performance of the algorithm .The actual data is from MSNBC (an anonymous web data) dataset for RPSP algorithm, with many experimental proofs the parameters are evaluated. The experimental parameters are minimum support, number of frequent items, time, memory and number of transaction in a sequence. The experiments shows that RPSP algorithm can find all frequent item sets with given minimum support value. If the value of minimum support increases, the average number frequent items sets tends to decrease as shown in the figure 3.2.1. The time for finding the number frequent itemsets increases with the number of sequences which is shown in figure 3.2.2. It is observed that, more memory is required as the number transactions in sequence on the dataset increases as shown in figure 3.2.3.

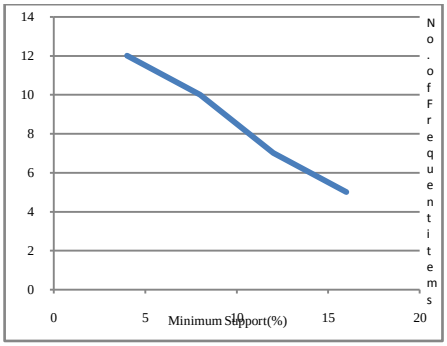

Figure: 3.2.1 Average number of frequent items for the different minimum support values

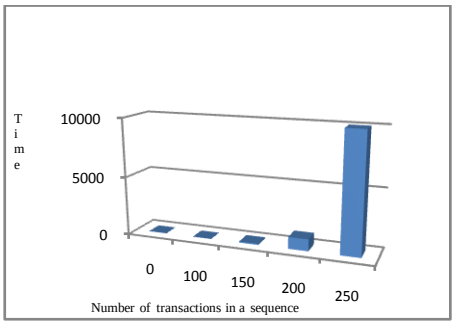

Figure: 3.2.2 Performance with time varying for the number of transactions in a sequence

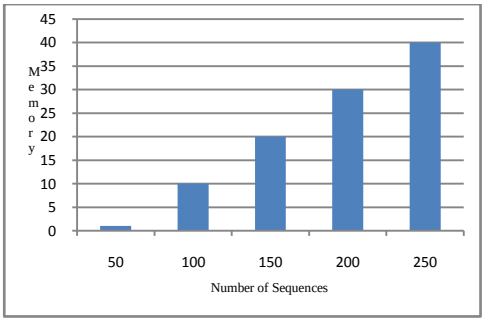

Figure: 3.2.3 Memory usage for the MSNBSC data set

The time complexity of the RPSP algorithm is proportional to the total time of checking items in projected databases. For the given a sequence, if the longest pattern in the sequence is M, then the maximal level of projections that may occur on the sequence will be M. This means each item in the sequence is checked at most M times. Given a database, the total number of items is T. Let L be the average length of detected patterns, then on average an item is checked at most L times, and the total instances of items checked is at most LT times. Practically it is close to O((logL)T)) because the minimal levels of projections to detect a pattern with length M is about $\log_2 M+1$.

4. CONCLUSION

With the increase of large data, it is difficult to maintain and access the information in real life situations. Due to that, there is need of various mining techniques especially designed for different types of data. Moreover, frequent itemset mining plays an essential role in the mining of various patterns and useful in many applications. In this paper a new approach is proposed for sequential pattern mining namely Recursive Prefix Suffix Patterns pattern based detection (RPSP) algorithm. RPSP algorithm requires simpler steps for finding the FI's with the given minimum support. The algorithm constructs Prefix and suffix projected databases and then detects frequent patterns hierarchically by pattern partitioning, thus it is an improvement in terms of levels of recursion. The time of execution decreases for large datasets and by increasing the minimum support level. Our approach can be extended to other type of sequential pattern mining problems, like mining with constraints, maximal pattern mining and domain specific pattern mining etc. The implementation of RPSP approach is done with java swings environment.

5. REFERENCES

[1] R.Agarwal and R.Srikanth, "Mining Sequential Patterns" ICDE'95, Pg 3-14,1995.

[2] R. Agrawal, and R. Srikant, Fast algorithms for mining association rules, Proc. of 20th Intl. Conf. on VLDB, pp. 487-499, 1994.

[3] G. Grahne and J. Zhu., "Efficiently Using Prefix-trees in Mining Frequent Itemsets," Prof.FIMI'03 Workshop on Frequent Itemset Mining Implementations, 2003.

[4] J. Chen, T. Cook. Mining Contiguous Sequential Patterns from Web Logs. In Proc. Of WWW2007 Poster session, May 8-12, 2007, Banff, Alberta, Canada.

[5] C. Antunes and A. L. Oliveira, "Sequential Pattern Mining Algorithms: Trade-offs between Speed and Memory," Proc. 2nd Intl. Workshop on Mining Graphs, Trees and Sequences, 2004.

[6] M.Y. Lin and S.Y. Lee, "Fast Discovery of Sequential Patterns through Memory Indexing and Database Partitioning," J. of Information Science and Engineering, vol. 21,2005.

[7] M. Zaki, "Spade: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, vol. 40, pp.31-60, 2001.

[8] Jinlin Chen, "An UpDown Directed Acyclic Graph Approach for Sequential Pattern Mining", IEEE 2009, vol-ll, Pg 1-16.

[9] D.Y. Chiu, Y.H. Wu and A.L.P. Chen, "An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting," Proc. ICDE 2004, pp. 375, 2004.

[10] 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," Proc. 2001 Int'l Conf. Data Eng. (ICDE '01), pp. 215-224, 2001.

[11] Z. Zhang, Y. Wang, and M. Kitsuregawa, "EffectiveSequential Pattern Mining Algorithms for Dense Database," Proc. Japanese National Data Engineering WorkShop (DEWS'06). 2006.

[12] Chen, T. Cook. Mining Contiguous Sequential Patterns from Web Logs. In Proc. of WWW2007 Poster session, May 8-12,2007, Banff, Alberta, Canada.