A FAST & EFFICIENT METHODOLOGY FOR GENERATING ALL SEQUENTIAL PATTERNS FROM A SEQUENTIAL DATA SET

[1]Deepti chandnani, [2]Prof. Ravi Gedam

[1]M.Tech Scholar, [2]Assistant Professor
SBITM, Betul (M.P)

***Abstract*: The discovery of sequential patterns is one of the most vibrant and useful segment of modern data mining. It has a vast array of real world applications. It is worthy of study on extending the memory indexing approach for efficient mining of generalized sequential patterns. This paper proposes a a new method for sequential pattern mining. In this paper the original data set is converted in to the transformed data set, by pruning the useless data in the early stage of data mining. The results have shown that this approach is resulting in saving a lot of computational time.**

1. Introduction:

Nowadays, computers are used widely in different areas. Fast, reliable and unlimited secondary storage provides a perfect environment for the users to collect and store large amounts of the data. Computers are also used to extract the useful information from the mass of data. This is the knowledge discovery in database (KDD) or data mining.

The knowledge discovery in database (KDD) [1,2,3] is defined as the process of nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. Fayyad et. al suggested that the KDD process can be divided into several steps, as shown in Figure 1. The whole KDD steps include selection, preprocessing, transformation, data mining and the interpretation or evaluation. The researchers are focused on the data mining process, as it is the application of algorithms for extracting patterns from the data, which is not a trivial task.

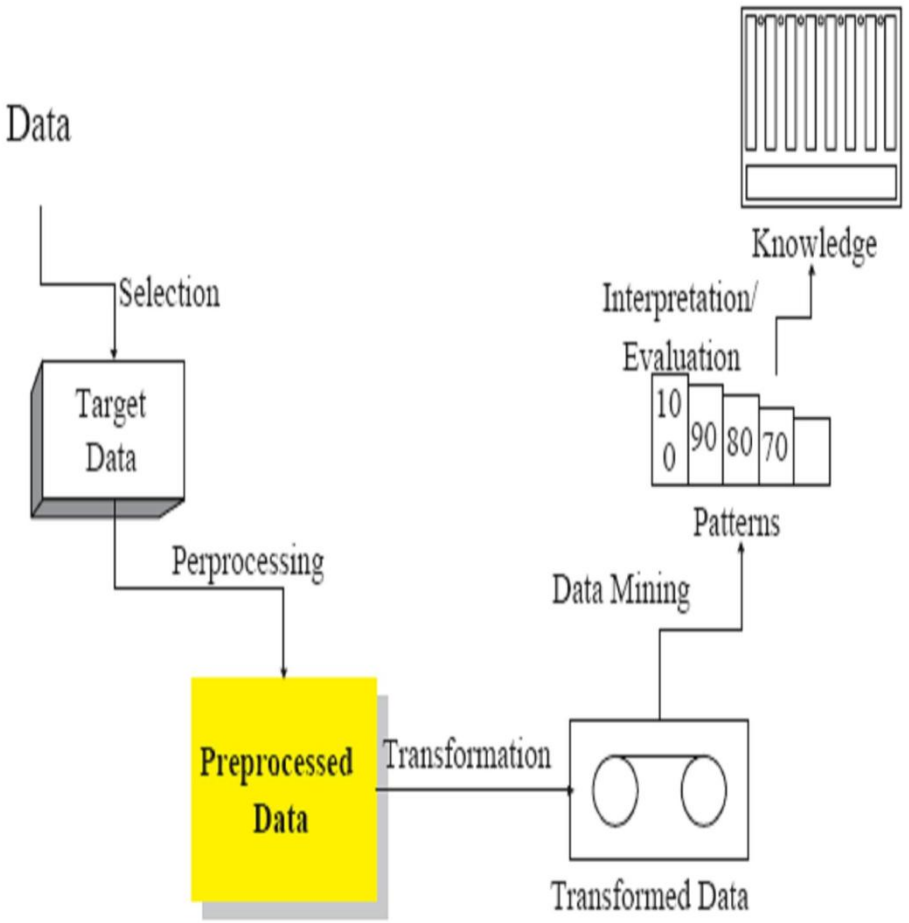

Figure 1: KDD Process

There are two important principles in frequent itemset mining:

- Monotonicity Principle: Let J, I be two itemsets where J is subset of I , the support of I will be at most as high as the support of J.
- Apriori Property: All nonempty subsets of a frequent itemset must also be frequent. These properties have been widely used in improving the efficiency of algorithms by pruning those "infrequent" branches in time so to narrow the search space.

Sequence Database Each sequence is a time-ordered list of item sets. An item set is an unordered set of items (symbols), considered to occur simultaneously.

S.No	ID	Sequences
01	Seq1	{a,b},{c},{f},{g},{e}
02	Seq2	{a,d},{c},{b},{a,b,e,f}
03	Seq3	{a},{b},{f},{e}
04	Seq4	{b},{f,g}

Table 1: Data Mining Sequences

Sequential Pattern Mining (SPM) [2,3,7] is perhaps the foremost standard set of techniques for locating temporal patterns in sequence databases. SPM finds sub- sequences that are common to over minsup sequences. SPM is restricted for creating predictions. for instance, take into account the pattern. It"s attainable that y seems often when an x but that there are also several cases wherever x isn't followed by y. For prediction, we'd like a mensuration of the confidence that if x happens, y can occur afterward.

A sequential rule usually has the shape X->Y . A sequential rule X⇒Y has 2 properties:

1. Support: the number of sequences where X happens before Y, divided by the number of sequences.
2. Confidence the number of sequences where X happens before Y, divided by the number of sequences where X occurs.

Background [4,5,6] & Related Work

Frequent itemsets[4,5,6] and association rules focus on transactions and the items that appear there.Databases of transactions usually have a temporal information. Sequential pattern or sequential rules exploit these temporal information.

Example data:

- Market basket transactions
- Web server logs
- Tweets
- Workflow production logs

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7

Table 2: A Sequence Data Base

Formal Definition of a Sequence

A sequence is an ordered [8,9] list of elements (transactions). Each element contains a collection of events (items). Each element is attributed to a specific time or location. Length of a sequence, |s|, is given by the number of elements of the sequence

A **sequential rule is an implication of the form** 1=>2 .It has following two associated terms:

- **Support (1=>2) = F(1 => 2)/(N)**

Where F(1=>2) is the number of transactions in which 2 comes after 1. N is the total number of transactions.

- **Confidence (1=> 2) = F(1 => 2)/ F(1)**

Where F(1=>2) is the number of transactions in which 2 comes after 1. F(1) is the number of transactions containing 1.

Sequential Rule Mining finds all rules whose support and confidence is greater than the minimum support threshold and minimum confidence threshold respectively

A **sequential rule** typically has the form X->Y .A sequential rule X⇒Y has **two properties**:

- **Support:** the number of sequences where X occurs before Y, divided by the number of sequences.

- **Confidence** the number of sequences where X occurs before Y, divided by the number of sequences where X occurs.

Sequential Rule Mining finds all **valid rules**, rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf* .

Association rule mining (Agrawal et al., 1993) is a popular knowledge discovery technique for discovering associations between items from a transaction database. Formally, a transaction database D is defined as a set of transactions T={t1,t2…tn} and a set of items I={i1, i2,…in}, where t1,t2,…,tn ⊆ I. The support of an itemset X ⊆ I for a database is denoted as sup(X) and is calculated as the number of transactions that contains X. The problem of mining association rules from a transaction database is to find all association rules X→Y, such that X,Y ⊆ I, X∩Y=∅, and that the rules respect some minimal interestingness criteria. The two interestingness criteria initially proposed (Agrawal et al. 1993) are that mined rules have a support greater or equal to a user-defined threshold *minsup* and a confidence greater or equal to a user-defined threshold *minconf*. The support of a rule X→Y is defined as sup(X ∪ Y) / |T|. The confidence of a rule is defined as conf(X→Y) = sup(X ∪ Y) / sup(X). Since |T| ≥ sup(X) for any X ⊆ I, the relation conf(r) ≥ sup(r) hold for any association rule r

Association rules are mined from transaction databases. A generalization of a transaction database that contains time information about the occurrence of items is a sequence database (Agrawal & Srikant, 1995). A sequence database SD is defined as a set of sequences S={s1, s2…sn} and a set of items I={i1, i2,…in}, where each sequence sx is an ordered list of transactions sx={X1, X2, … Xn} such that X1, X2, …Xn ⊆ I .

We propose the following definition of a sequential rule [10,11] to be discovered in a sequence database. A sequential rule X⇒Y is a relationship between two itemsets X,Y such that X,Y ⊆ I and X∩Y = ∅. The interpretation of a rule X⇒Y is that if the items of X occur in some transactions of a sequence, the items in Y will occur in some transactions afterward from the same sequence. Note that there is no ordering restriction between items in X and between items in Y. We define two interestingness measures for such a rule, which are an adaptation for multiple sequences of the measures used for other sequential rule mining algorithms (Mannila et al., 1997; Das et al., 1998; Harms et al., 2002). The first measure is the rule's sequential support and is defined as: seqSup(X ⇒ Y) = sup(X■Y) / |S|. The second measure is the rule's sequential confidence and is defined as: seqConf(X ⇒ Y) = sup(X■Y) / sup(X). Here, the notation sup(X■Y) denotes the number of sequences from a sequence database where all the items of X appear before all the items of Y (note that items from X or from Y do not need to be in the same transaction). The notation sup(X) represents the number of sequences that contains X. Since |S| ≥ sup(X) for any X ⊆ I , the relation seqConf(r) ≥ seqSup(r) holds for any sequential rule r.

3. **Proposed Algorithm:**

The steps are as follows

STEP 1: START

STEP 2: INPUTS ARE:

· **SEQUENTIAL DATA SET D&**

· **MINIMUM SUPPORT THRESHOLD.**

STEP 3: FIRST THE PROPOSED ALGORITHM SCANS THE SEQUENTIAL DATA BASE D AND CALULATES THE SUPPORT OF EACH SINGLE SIZE ITEM FROM D.

STEP 4: NOW ELIMINATE ALL THE INFREQUNT ITEMS FOUND IN STEP 3 FROM D SO THAT D WILL BE CONVERTED IN TO A COMPRESSED SEQUENTIAL DATA BASE.

STEP 5: ALGORITHM IS CALLED RECURSIVELY TO GENERATE BIGGER SEQUENTIAL PATTERNS BY USING THE UNION OR EXPANSION OF LOWER SIZE ITEMS.

Example:
Consider the following sequential data set with the minimum support 3
Table : Sequential Data Base

Sequence ID	**Sequences**
S1	<(1) (1,2,3) (4) (7,8) (3)>
S2	<(3) (5,8,9) (1,2) (2,3)>
S3	<(5) (1,2) (3,5,6) (1,2) (6)>

The data set is scanned to find the sequential frequent patterns of the size 1:

1- Suppot 3
2- Support 3
3- Support3

The items 4,5,6,7,8 have support less than the 3.

In pruning step, all the infrequent items are eliminated from the original data set. Because it is clear that they will not appear in any frequent sequential pattern.

Bu doing this, the original data set is converted in to the transformed & much compressed data set. It is as follows:

Table 2

Sequence ID	**Sequences**
S1	<(1) (1,2,3) (3)>
S2	<(3) (1,2) (2,3)>
S3	<(1,2) (3) (1,2)>

Now expand the size 1 items to get the patterns of larger size:

For example 1 is expanded in to (1,2) & (1,3). Then these two are checked for the minimum support in table2. We see that the support is 3. So this two are also frequent sequential patterns.

The algorithm is continued until there are items to be expanded. At the end, we get following sequential patterns

(1), (2), (3), (1,2), (1,3), (2,3), (1,2,3)

Conclusion:

In this paper, we presented a data elimination based technique for mining sequential patterns from a data set. The data set is transformed into a more compact data set by eliminating all the infrequent patterns at the early stage of the sequential pattern mining. We have evaluated the performance of our proposed algorithm. It is fast. Also it is taking less main memory for computation in comparison to previous algorithm.

References:

1. Srikant R, Agrawal R., 1995. "Mining generalized association rules". In: Dayal U, Gray P M D, Nishio Seds. Proceedings of the International Conference on Very Large Databases. San Francisco, CA: Morgan Kanfman Press, pp. 406-419.
2. M. Houtsma, and Arun Swami, 1995. "Set-Oriented Mining for Association Rules in

Relational Databases". IEEE International Conference on Data Engineering, pp. 25–33.
3. S. Brin, R. Motwani, J.D. Ullman, and S. Tsur, 1997. "Dynamic itemset counting and implication rules for market basket data". In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, volume 26(2) of SIGMOD Record, pp. 255–264. ACM Press.
4. Chen, E., Cao, H., Li, Q., & Qian, T. (2008). Efficient strategies for tough aggregate constraint-based sequential pattern mining. Inf. Sci., 178(6), 1498-1518.

5. Masseglia, F., Poncelet, P., & Teisseire, M. (2003). Incremental mining of sequential patterns in large databases. Data Knowl. Eng., 46(1), 97–121.

6. Wang, J. L., Chirn, G., Marr, T., Shapiro, B., Shasha, D., & Zhang, K. (1994). Combinatorial pattern discovery for scientific data: Some preliminary results. Proc. ACM SIGMOD Int'l Conf. Management of Data, (pp. 115-125).
7. Yang, J., Wang, W., & Yu, P. S. (2001). Infominer: mining surprising periodic patterns. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
8. Srikant, R., & Agrawal, R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology.
9. Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences.
Journal of Machine Learning, 42(1-2), 31-60.

10. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). FreeSpan: frequent pattern-projected sequential pattern mining. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
11. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., et al. (2001). PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. Proceedings of the 17th International Conference on Data Engineering.