Sequence Pattern Mining: An Incremental Approach

Anjuman Ranavadiya¹ Prof. J. S. Dhobi²

¹,² Computer Science & Engineering Department
¹,² G. E. C., Modasa

Abstract—The basic idea of sequential pattern mining was first introduced by Agrawal and Srikant [1]. The sequence mining task is to discover a set of attributes, shared across time among a large number of objects in a given database. For example, consider the sales database of a bookstore, where the objects represent customers and the attributes represent authors or books. Let’s say that the database records the books bought by each customer over a period of time. The discovered patterns are the sequences of books most frequently bought by the customers. An example could be that, “70% of the people who buy Jane Austen’s Pride and Prejudice also buy Emma within a month.” Stores can use these patterns for promotions, shelf placement, etc. Sequential mining algorithms can mine a static database. But, nowadays, almost all databases are dynamic in nature and they grow incrementally. One way to handle this is to mine the whole database every time an update occurs. But it is highly inefficient and also undesirable. We must find a way to use the already mined information. An incremental mining algorithm does the same. It utilizes the mined information to get new set of frequent sequential patterns instead of mining the whole database from scratch. Note that the ultimate aim of using an incremental mining algorithm instead of non-incremental one is to gain efficiency with respect to time. Otherwise a non-incremental mining algorithm can also serve the purpose of mining very easily. So for incremental mining algorithm the time taken by the algorithm to mine complete set of frequent patterns must be considered[5] and there are various algorithm for sequence pattern non incremental and as well incremental Mining

A. Notation : The notation used in this approach is defined below.
- D: the original customer sequences.
- T: the set of newly merged customer sequences from the newly inserted customer sequences.
- U: the entire updated customer sequences.
- q: the number of newly added customer sequences belonging to old customers in the original database.
- S_u: the upper support threshold for large sequences.
- S_l: the lower support threshold for pre-large sequences, S_l < S_u.
- L_k^D: the set of large k-sequences from D.
- L_k^T: the set of large k-sequences from T.
- L_k^U: the set of large k-sequences from U.
- P_k^D: the set of pre-large k-sequences from D.
- P_k^T: the set of pre-large k-sequences from T.
- P_k^U: the set of pre-large k-sequences from U.
- C_k: the set of all candidate k-sequences from T.
- I: a sequence.
- S^l(I): the number of occurrences of I in D.
- S^T(I): the number of occurrence increments of I in T.
- S^U(I): the number of occurrences of I in U.

I. INTRODUCTION

An itemset is a non-empty set of items. A sequence is an ordered list of itemsets. Without loss of generality, we assume that the set of items is mapped to a set of contiguous integers. We denote an itemset i by (i₁, i₂, i₃, ..., iₙ) where i_j is an item. We denote a sequence s by <s₁, s₂, s₃, ..., s_m> where s_j is an itemset.

For the sequence pattern mining sequence database is used. A sequence database is used. A sequence Database S is set of tuple <SID,s> where SID is sequence id and s is a sequence any item is frequent in sequence database S if support>=min_sup. A frequent Sequence is called a Sequential Pattern[1]

II. SOME SEQUENCE PATTERN MINING ALGORITHMS

Sequential pattern mining is an important data mining problem, which detects frequent sub sequences in a sequence database. The major technique for sequential pattern mining are
- AprioriALL
- GSP
- FreeSpan
- PrefixSpan
- SPADE
- SPAM

A. Aprioriall: Sequential pattern mining was first introduced by Agrawal and Srikant[1]. The authors proposed three Apriori-based algorithms. There are five phases in the whole work flow of the algorithms[1]. Mainly sort phase, L-itemset Phase, Transformation Phase, Sequence Phase, Maximal Phase. The kind of disadvantage of AprioriAll is that there are many passes over the database and many candidates generated, which are time consuming database [1]

B. Gasp: It is an Apriori based algorithm for sequential pattern mining [2]. The difference is that GSP inserts some constraints into the mining process, i.e., time constraints, and relaxes the definition of transaction. Moreover, it takes the taxonomy into account. For time constraints, maximum gap and minimal gap are defined to specified the gap between any two adjacent transactions in the sequence. If the distance between two transactions is not in the range between the maximum gap and the minimal gap, then the
two transactions can not be taken as two consecutive transactions in a sequence. It scan Database multiple times, so there is need of more efficient Algorithm.

C. Freespan: FreeSpan [9] was developed to substantially reduce the expensive candidate generation and testing of Apriori, while maintaining its basic heuristic. In general, FreeSpan uses frequent items to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database. Each projection partitions the database and confines further testing to progressively smaller and more manageable units. General idea is to use frequent items to recursively project sequence databases into smaller projected databases, and grow subsequence fragments in each projected database.

D. Prefixspan: PrefixSpan [3] utilizes the method of database projection to make the database for next pass much smaller and consequently make the algorithm more speedy. The au-thors claimed that in PrefixSpan there is no need for candidates generation. It recursively projects the database by already found short length patterns. This pattern growth idea is similar to that in Apriori heuristic. There is no candidate Generation.


F. Span: SPAM [6] algorithm based on the key idea of SPADE. The difference is that SPAM utilizes a bitmap representation of the database instead of (SI D, T I D) pairs used in the SPADE algorithm. Hence, SPAM can perform much better than SPADE and others by employing bitwise operations[5]. When the sequential patterns in the database are very long than BitMap Represent Used. Bitmap representation, Depth First Traversal, outperforms than SPADE on large Datasets, Memory Utilization High. This all algorithm are based on sequence pattern mining with s incremental tactic Database than after when the concept of dynamic database are arrives than Incremental Approach is introduced.

III. INCREMENTAL MINING ALGORITHMS

Note that all the above Discuss Algorithm can mine set of Frequent Sequential pattern from a static database, but these days all the real world database are dynamic in nature, so Incremental Mining concept is introduced. Incremental mining algorithm utilizes the already mined set of frequent sequential patterns for the original database to mine the complete set of frequent sequential patterns for the updated database.

Some Incremental Mining Algorithm are ISM, ISE and INCSPAN etc.

A. Ism: Incremental Sequence Mining [6] mining algorithm is based on the Negative Border Concept. Negative border is defined as the set of sequences which are not frequent but both of whose generating subsequences are frequent. The generating subsequences of a sequence of length k are two subsequences of length (k-1) obtained by dropping exactly one of its first or second item. In this

As new transactions are added, the proposed approach first transforms them into new customer sequences and merges them with the corresponding old sequences algorithm they have maintained a lattice of sequences to keep the set of frequent sequences and the negative border. This concept of negative border turns out to be helpful but it has its own drawbacks also. The main drawback of this approach is that the total number of sequences in the negative border is very large. So we have to store these infrequent sequences along with the set of frequent sequential patterns. Moreover the sequences in negative border may have very low support and they may not become frequent after many subsequent updates to the database. So there is no point in keeping those highly infrequent sequences.

B. Ise: Incremental Sequence Extraction[6] is another Incremental Mining Algorithm. In this algorithm the candidate-generate and test approach is used. The disadvantages of this algorithm include the huge number of candidate sequences to be tested and need of multiple scan of the whole database. So this algorithm turns out to be very costly with respect to time and space requirement.

C. Incspan: Incremental mining algorithm called IncSpan based on an existing algorithm called PrefixSpan [4]. To gain efficiency the concept of semi-frequent patterns is introduced. Semi-frequent patterns are the patterns which are not frequent but whose support is greater than the product of minimum support and a user specified factor μ (0 < μ < 1) i.e. patterns that are not frequent but are almost frequent. Note that these almost frequent patterns are most likely to be frequent in the updated database. The experimental results in [2] show that IncSpan outperforms the non-incremental algorithm PrefixSpan and an incremental mining algorithm ISM.

Based on Non Incremental PrefixSpan and Incremental ISM, IncSpan has a major drawback of not able to mine the complete set of frequent sequential patterns. For that proposed approach is introduced.

IV. PROPOSED APPROACH

Proposed Approach focuses on newly added customer sequences, which are transformed from newly added transactions sequences, that is used purely incremental approach and that generate complete set of frequent pattern. The Objective of Proposed Approach is To observe the effect of various existing algorithms for mining frequent sequences on various datasets. To propose an incremental approach for mining the frequent sequences for sequence database i.e. for the above problem and to validate the incremental approach on different datasets used in various ways. existing in the original database. The newly merged customer sequences are then scanned to generate candidate 1-sequences with occurrence increments. These candidate sequences are compared to the large and pre-large 1-sequences which were previously retained.

These candidate sequences are divided into three parts according to whether they are large, pre-large or small in the original database. If a candidate 1-sequence is also among the previously retained large or prelarge 1-sequences, its new total count for the entire updated database can easily be calculated from its current count increment and previous count, since all previous large and pre-large sequences with their counts have been retained. Whether an original large or pre-large sequence is still large or pre-large after new transactions are added is then
determined from its new support ratio, which is derived from its total count over the total number of customer sequences.

If a candidate 1-sequence does not exist among the previously retained large or pre-large 1-sequences, then the sequence is absolutely not large for the entire updated database when the number of newly merged customer sequences is within the safety bound. In this situation, no action is needed. When new transaction data are incrementally added and the total number of newly added customer sequences exceeds the safety bound, the original database must be re-scanned to find new large and pre-large sequences.

The proposed approach can thus find all large 1-sequences for the entire updated database. After that, candidate 2-sequences from the newly merged customer sequences are formed, and the same procedure is used to find all large 2-sequences. This procedure is repeated until all large sequences have been found.

Two global variables, c and b, are used to accumulate, respectively, the number of newly added customer sequences and the number of newly added customer sequences belonging to old customers since the last re-scan of the original database.

A. Steps Of Algorithm:

1) Input: A lower support threshold $S_l$, an upper support threshold $S_u$, a set of large and pre-large sequences in the original database $D$ consisting of $(d + c)$ customer sequences, the accumulative amount $b$ of new customer sequences belonging to old customers, and a set of $t$ newly added customer sequences transformed from new transactions.

2) Output: A set of final large sequential patterns for the updated database.

- Step 1: Calculate the value of the term as: $f = [(S_u - S_l) / d] / (1 - S_u)$
- Step 2: Merge the newly added customer sequences with the old sequences in the original database and count the value $q$, which is the number of the newly added customer sequences belonging to old customers.
- Step 3: Set $b = b + q$ and calculate the value of the term as: $h = bS_u / (1 - S_u)$
- Step 4: Set $k = 1$, where $k$ is used to record the number of itemsets in the sequences currently being processed.
- Step 5: Find all candidate $k$-sequences $C_k$ and their count increments from the newly merged customer sequences $T$.
- Step 6: Divide the candidate $k$-sequences into three parts according to whether they are large, pre-large or small in the original database.
- Step 7: Do the following substeps for each $k$-sequence $I$ in the original large $k$-sequences $L_k^D$:
  - Substep 7-1: Set the new count $S^I(I) = S^T(I) + S^D(I)$.
  - Substep 7-2: If $S^I(I)/(d + c + t - b) \geq S_u$, then assign $I$ as a large sequence, set $S^D(I) = S^I(I)$ and keep $I$ with $S^D(I)$; otherwise, if $S^I(I)/(d + c + t - b) \geq S_l$, then assign $I$ as a pre-large sequence, set $S^D(I) = S^I(I)$ and keep $I$ with $S^D(I)$; otherwise, ignore $I$.
- Step 8: Do the following substeps for each $k$-sequence $I$ in the original pre-large sequences $P_k^D$:
  - Substep 8-1: Set the new count $S^U(I) = S^T(I) + S^D(I)$.
  - Substep 8-2: If $S^U(I)/(d + c + t - b) \geq S_u$, then assign $I$ as a large sequence, set $S^U(I) = S^U(I)$ and keep $I$ with $S^U(I)$; otherwise, if $S^U(I)/(d + c + t - b) \geq S_u$, then assign $I$ as a pre-large sequence, set $S^U(I) = S^U(I)$ and keep $I$ with $S^U(I)$; otherwise, ignore $I$.
- Step 9: Put $I$ in the rescan-set $R$ for each $k$-sequence $I$ in the candidate $k$-sequences $C_k$ that is neither in the original large sequences $L_k^D$ nor in the pre-large sequences $P_k^D$, for use when rescanning in Step 10 is necessary.
- Step 10: If $c + t \leq f - h$ or $R$ is null, then do nothing; otherwise, rescan the original database to determine whether the sequences in the rescan-set $R$ are large or pre-large.
- Step 11: Form candidate $(k + 1)$-sequences $C_{k+1}$ from finally large and pre-large $k$-sequences $(L_k^D \cup P_k^D)$ that appear in the newly merged transactions.
- Step 12: Set $k = k + 1$.
- Step 13: Repeat STEPS 5 to 12 until no new large or pre-large sequences are found.
- Step 14: Modify the maximal large sequence patterns according to the modified large sequences.
- Step 15: If $c + t > f - h$, then set $d = d + c + t$, $c = 0$ and $b = 0$; otherwise, set $c = c + t$.

After Step 15, the finally maximal large sequences for the updated database can be determined.

V. EXPERIMENTS RESULT

INCSPAN and Proposed Approach are compared for various attributes on a set of customer data and the results found are as follows:

The chart below shows the time required by the algorithms on varying minimum support counts.

VI. CONCLUSION

The Proposed Method is based on the Incremental approach that is improvement over IncSpan Algorithm. The IncSpan Algorithm is used Projected Database that’s the running time of algorithm is very high and According to our observations, the performances of the algorithms are strongly depends on the support levels and the features of the data sets (the nature and the size of the data sets). Therefore we employed it in incremental approach to guarantee the time saving in the case of sparse and dense...
data sets. Thus it saves much time and considered as an efficient method as proved from the results.

REFERENCES


[7] Hong Cheng, Xifeng Yan, Jiawei Han:” IncSpan: Incremental Mining of Sequential Patterns in Large Database “by U.S. National Science Foundation NSF


[15] Jiawei Han and Micheline Kamber, Data Mining Concepts and Techniques - Third Edition, ELSEVIER Morgan Kaufman Publisher, July 6, 2011