Mining Sequential Pattern Algorithms Comparison

Haytham Qalalwi

Abstract—Sequential pattern mining is a very important mining technique with broad applications. It found very useful in various domains like natural disaster, sales record analysis, marketing strategy, shopping sequences, medical treatment and DNA sequences etc. It discovers the sequence's and frequent relevant pattern from the given sequences. That we have provided the sequence database having sequences, in which each sequence is a list of the transactions ordered by the transaction time. Each transaction consists of the number of the items. The problem is to discover the all sequential pattern which satisfy the user specified constraint, from the given sequence database. There are various sequential pattern mining algorithms proposed earlier, some of them are GSP, SPADE and PrefixSpan. They are proposed to find the relevant frequent pattern from the sequences. In these algorithm the timestamp is an important attribute of each dataset, and also it is important in the process of data mining for giving the more accurate and useful information. The detail survey of these entire algorithms is presented in this paper. First we categorized these algorithms by their used approaches to solve the mining problem and then we have compared each one with another by their various provided features and performance point of view.

Keywords— Sequential Pattern Mining; Sequence Database; Apriori; GSP; SPADE; PrefixSpan

I. INTRODUCTION

Data mining is the process of running powerful algorithms on data to extract useful knowledge. It is useful in various domains such as market analysis, decision support, fraud detection, business management and so on [1]. Many approaches have been proposed to extract information from input sequences and sequential pattern mining is one of the most important methods. This mining algorithm solves the problem of discovering the presence of frequent sequences in the given database [2]. The database given to this algorithm is a set of sequences time-ordered, called data-sequences, as the input data. Each data-sequence is a list of transactions, where each transaction contains a set of literals, called items. Note that the order of items in a transaction (item set) does not matter. In another word A sequence α is a frequent sequence if a sequence α is an ordered list of events <a_1,a_2,...,a_m>. An event is a non-empty unordered set of items a_i⊆{i_1,i_2,...,i_k}. A sequence α = <a_1,a_2,...,a_m> is a subsequence of β = <b_1, b_2,...,b_n> if and only if exists i_1,i_2,...,i_m such that 1 ≤ i_1 < i_2 < ... < i_m ≤ n and a_1 ⊆ b_1, a_2 ⊆ b_2 and a_m ⊆ b_m. Given a sequence database D = s_1,s_2,...,s_n, the support of a sequence α is the number of sequences of D which contains α as a subsequence. If the support of α is bigger than a threshold MaxSup. A sequence database consists of tuples, where each tuple consists of a sequence id, transaction id, and an item set. An example of a sequence is { (ABD), (CDA)}, where items ABD indicate the items that were purchased in the first transaction of a customer and CDA indicate the items that were purchased in a later transaction of the same customer.

II. MINING APPROACHES

Sequential Pattern Mining is defined as discovering the whole set of frequent subsequence in the set of sequential transactional database. The sequential pattern mining discovers the correlation between the different transactions. It is discovers which items a single customer, having those items come from various transactions, brings in a particular order. The resulting pattern found after mining is the sequence of item sets that normally found frequent in specific order. Sequential pattern mining is used in various areas for different purposes. It can be used for identifying Customer Shopping Sequence to determine which item particular customer brings one after another in sequence [3][4]. Various algorithms have been implemented for identifying frequent sequences from sequence database.

The problem of mining sequential patterns is stated in [2] and improved on through various other algorithms. Many studies and improvement methods have contributed to sequential pattern mining to make it more efficient. Sequential pattern mining algorithm can be broadly divided into three approaches: Apriori based (GSP, SPADE, SPAM), pattern growth (FreeSpan, PrefixSpan), early pruning (LAPIN-SPAM). There are lots of algorithms for sequential pattern mining but here I show some of good algorithms that I studied and advantages and disadvantages of those algorithms.

III. APRIORI-BASED APPROACHES

The Apriori algorithm was first proposed by Agrawal in [2], for the discovery of frequent itemsets. It is the most widely used algorithm for the discovery of frequent itemsets and association rules. The main concepts of the Apriori algorithm is: Any subset of a frequent itemset is a frequent itemset.
It advances by identifying all frequent individual items in the database and extending them to larger item sets as long as these item sets appear amply often in the database of sequences.

Figure (2): The Apriori algorithm step

A. GSP (GENERALISED SEQUENTIAL PATTERN) ALGORITHM

GSP discovers sequential pattern. It has a very good scale up properties with respect to the number of transactions per database and number of items per transaction (the size of the data increases). The GSP algorithm is not a main-memory algorithm, it makes multiple passes over the data, if the candidates do not fit in memory, the algorithm generates only as many candidates as will fit in memory and the data is scanned to count the support of these candidates. Frequent sequences resulting from these candidates are written to disk, while those candidates without minimum support are deleted. This procedure is repeated until all the candidates are been counted.

A.1 Algorithm Procedure:

Step 1: The algorithm at the first pass to the sequence database determines the support of each item (the number of data-sequences that include the item). At the end of the first pass, the algorithm knows which items are frequent (have minimum support). We call the data result from this step frequent sequence (F) or seed set for next iteration, which it used, in the next step.

Step 2: generate the candidate sequence (C) in length (k), by joining the seed set from the length (k-1) with itself.

Step 3: generate the frequent sequence (Fk) by deleting the sequence that do not have minimum support from the candidate sequence in length (Ck) we name it pruning. This becomes the next seed set for the next iteration.

At each next iteration, each candidate sequence has one more item than the seed sequence. Hence, at end of every step, each transaction consists of only the frequent elements that were originally contained. The step 2, 3 is repeated until there are no frequent sequences at the end of a pass, or when there are no candidate sequences generated.

Three key innovations in the GSP algorithm are how candidates are generated, how pruning method done and how candidates are counted.

1) Candidate Generation: Given the set of frequent (k−1)-sequences Fk−1, the candidates for the next pass are generated by joining Fk−1 with itself. For fast counting, the candidate sequences are stored in a hash-tree. We can see it in Figure (3).

2) A pruning phase eliminates any sequence at least one of whose subsequences is not frequent.

3) Support Counting: To find all candidates contained in a input-sequence E, conceptually all k-subsequences of E are generated. For each such subsequence, a search is made in the hash-tree. If a candidate in the hash-tree matches the subsequence, its count is incremented, a hash tree [2] is created whose leaves index the candidates. This speeds up the process of counting the min-support of the sequences, and to reduce the number of sequences to be checked when processing a transaction.

Hence, the GSP algorithm solves not only the basic frequent mining problems but can also handle stringent time constraints, and item classifications. But this Apriori-like sequential pattern mining method bares three non-trivial inherent costs, which are independent of detailed implementation techniques:

1) Potential large indexing storage of the sets of candidate sequences.
2) Multiple scans of databases.
3) Difficulties at mining long sequential patterns (as it needs to generate a large number of small candidates).

These problems of repeated database scans and complex internal data structures were overcome by proposing a new algorithm based on the same concepts of Apriori called SPADE (Sequential Pattern Discovery using Equivalence classes) presented below.

B. SPADE (Sequential Pattern Discovery using Equivalence classes)

While the GSP algorithm uses Apriori horizontal data format for sequential pattern mining, the SPADE [3] algorithm
uses a sequential database mapped as vertical data format, thereby considering each item as the center of observation and taking its associated sequence and event identifiers as datasets.

This algorithm was proposed by Parthasarathy, Zaki, & Li in 1998 and overcame all the limitations of GSP along with solving the goal of sequential pattern mining. SPADE not only minimizes I/O costs by reducing database scans, but also minimizes computational costs by using efficient search patterns.

The vertical database format used consists of an id-list for each item is maintained. Each sequence is a list of objects with their time stamps <SID (Sequence ID), EID (Event ID)> we can see it in Figure (4).

Figure (4): SPADE vertical database format.

B.1 Algorithm Procedure:

Step 1: compute the frequencies of 1-sequences, which are sequences with only one item. This is done in a single database scan.

Step 2: consists of counting 2-sequences. This is done by transforming the vertical representation into an horizontal representation in memory, and counting the number of sequences for each pair of items using a 2D matrix. Therefore, this step can also be executed in only one scan.

Step 3: Subsequent n-sequences can then be formed by joining (n-1)-sequences using their id-lists. The size of the id-lists is the number of sequences in which an item appears. If this number is greater than minsup, the sequence is a frequent one. The algorithm stops when no frequent sequences can be found anymore. The algorithm can use a breadth-first or a depth-first search method for finding new sequences [5].

The improvement characteristics are as follows:

1) It reduces the I/O costs by reducing the number of database scans, where all the sequences are discovered in only three databases scans.

2) The second characteristic is a modular approach to the algorithm data structure construction allowing easier problem decomposition, where each part of the algorithm can be independently developed and tuned.

3) And as a last point, algorithm doesn’t make use of complex hash and/or data structures, allowing not only a simpler comprehension but also a simpler implementation.

Both the GSP algorithm and SPADE algorithm share strengths and weaknesses as they are based on the Apriori method of pattern extraction.

IV. PATTERN GROWTH BASED APPROACH

The general idea of pattern growth based approach is that instead of repeatedly scanning the entire database and generating and testing large sets of candidate sequences, one can recursively project a sequence in a database into a set of smaller databases associated with the set of patterns mined so far [7]. The underlying algorithm can then mine locally frequent patterns in each of these projected databases.

Pattern growth approaches are a recent addition to the literature of sequential pattern mining problems. The candidate generation step is generally omitted and the focus is more on a restricted portion of the initial database.

A. PREFIXSPAN (Prefix Projected Sequential pattern Mining)

The PrefixSpan algorithms presented by Jian Pei, Jiavei Han and Helen Pinto [6]. The idea of the algorithm is instead of projecting sequence databases by considering all possible occurrences of frequent subsequences, the projection be based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix, they propose a method that has the following two key features.

1) projection-based: the sequence database is iteratively projected into smaller projected databases by examines only the prefix subsequences and projects only their corresponding postfix subsequences into projected databases.

2) grows patterns sequentially in the projected databases, by investigating only locally frequent segments.

The PrefixSpan algorithm outperformed the other methods mainly in three ways:

1) It grows patterns without candidate generation.

2) The data reduction can be performed effectively by the projected databases that leads to efficient processing.

3) The memory space utilization is approximately steady.

Figure (5): PREFIXSPAN.
A.1 Cost:

No candidate sequence needs to be generated by PrefixSpan. Projected databases keep shrinking. The major cost of PrefixSpan is the construction of projected databases. Mining efficiency is improved by two kinds of database projections: level-by-level projection and bi-level projection. Moreover, a main-memory-based pseudo-projection (using pointers rather than physically copying postfix sequences) technique is developed for reducing the number and size of the projected databases to save the cost of projection and speeding up processing when the projected (sub)-database and its associated pseudo-projection processing structure can fit in main memory.

A.2 PREFIX CONCEPT OVERVIEW:

A \(\alpha\)-projected database is the set of subsequences in the database, which are suffixes of the sequences that have prefix \(\alpha\). Each projected database has the same prefix subsequence. At each step, the algorithm looks for the frequent sequences with prefix \(\alpha\), in the corresponding projected database. In this way, the search space is reduced at each step, allowing for better performances in the presence of small support thresholds. In general, pattern-growth methods can be seen as depth-first traversal algorithms, since they construct each pattern separately, in a recursive way [6].

A.3 Algorithm Procedure:

Step 1: Scan the sequential database \(S\) to get the length-1 sequences, considering the minimum support that has been specified.

Step 2: Sequential database is divided into different partitions (prefixes) according to the number of length-1 sequence to get projected databases.

Step 3: Generate projected database by:
   a. Examine only the frequent prefix subsequences.
   b. Project only their corresponding postfix subsequences.

Step 4: The subsets of sequential patterns can be searched by constructing projected databases of the prefixes supported. Searching is either can be depth first or breadth first search.

Step 5: Mine projected database repeatedly for sequential patterns until frequent sequences of the database are not found.

V. COMPARATIVE ANALYSIS OF SEQUENTIAL PATTERN MINING ALGORITHMS

For the comparison sequential pattern mining algorithm is categorized into two broad categories, as Apriori based and pattern growth based algorithm. The all features used to classify these algorithms are discussed first and then comparison is done for the following algorithms.

G.S.P.: Generalized sequential pattern.

SPADE: Use of the equivalence classes for the discover of the sequential pattern.

PrefixSpan: By prefix-projected sequential pattern mining.

A. Features of Sequential Pattern Mining Algorithms:

1) Breath-First Search Based Approach vs Depth First Search Based Approach: In the breath-first search traversal technique level-by-level search is conducted to find the complete set of pattern i.e. All the neighboring nodes are processed before moving to the next level. Instead, in the depth-first search traversal technique, all the inner-node (branch node) must be explored before in the path moving to the next one. The depth first search is that it can reach very quickly to large frequent fragments and therefore some expansion in the other path in the tree can be avoided.

2) Apriori-Based vs Pattern-growth Based: In the category of Apriori based type algorithm the main theme is to candidate-generate and test which uses the downward closer property. If an item set \(\alpha\) is frequent, then and then only the superset of \(\alpha\) is frequent, otherwise if not be frequent either. Pattern-growth strategy takes better approach in creating possible frequent sequences, and uses the divide-and-conquer approach(recursively breaking down a problem into two or more sub-problems of the same (or related) type, until these become simple enough to be solved directly). For the reduce of search space this pattern-growth algorithm do the projection on the database.

3) Top-Down Search vs Bottom-up search: The Apriori based algorithms uses a bottom-up search by ensuring each single frequent sequence. It means that for the produce a frequent sequence of length 1, all 21 subsequence’s have to be generated. From that it can be stated that this exponential complexity is limiting at the Apriori based algorithms to find out only short pattern, since they just find the subset infrequent pruning by deleting any candidate sequence for which there exist a subsequence that does not belongs to the set of frequent sequences. In case of the top-down approach the subset of sequential pattern can be mined by generating the relative set of projected databases and mining each recursively for top to bottom.

4) Anti-Monotone vs. Prefix-Monotone Property: According to the property of anti-monotone, it states that the each non-empty subsequence of the sequential pattern is a sequential pattern. And in the prefix-monotone states that every sequence which is having \(\alpha\) as a prefix satisfies the constraints if \(\alpha\) sequence satisfy the constraint.

5) Regular Expression Constraints: The number of state changes in the relative deterministic finite automata help to calculate the complexity of regular expression constraints. It has the nice property known as growth-based anti-monotonic if it satisfy the following property. The sequence must be reachable by growing from any component, which matches the part of the regular expression when it satisfies the constraints first.
From our comparative study, we found that PrefixSpan algorithm uses depth-first search based approach. Top-down technique is efficient technique to find frequent subsequence’s as sequential pattern from the large database. Also the regular expression constraint and prefix monotone property is use by PrefixSpan algorithm, which makes this algorithm best choice for applying user defined constraint for mining only concerned sequential pattern.

B. EXPERIMENTAL STUDY DONE BY RESEARCHERS:

To analyses the correctness and performance of various sequential pattern mining algorithm, many studies of performance is done on the three algorithms, on GSP, PrefixSpan and SPADE. By using real and synthetic datasets. We illustrates the conclusions for this studies

1) Analysis on the memory uses by algorithms: From the experimental result, it is found that the PrefixSpan is better stable in memory usage than the other algorithms SPADE and GSP. While the PrefixSpan only uses less than the one third of the memory used by the GSP algorithm. In the analysis done by the researchers it found that, the PrefixSpan algorithm needs memory space to just hold the sequence datasets and a set of header tables and pseudo projection table.

2) Comparison on the basis of time complexity of the Algorithms: the pattern growth algorithm PrefixSpan are time efficient than Apriori based algorithms. They needs less execution time than GSP and SPADE algorithm.

3) Comparison on the basis of ability to scale-up property: PrefixSpan has better performance than the other algorithms and it scales with the database size linearly.

The SPADE and GSP algorithms needs memory space to hold the candidate sequence pattern as well as the sequence database instead for the case of PrefixSpan it needs memory space to fit the sequence database and a set of header table and pseudo projection table. From the above performance analysis it is found that the PrefixSpan is the best among all the other tested algorithms. The PrefixSpan performs better than the other algorithms that the reasons discussed below:

1) Use of Pattern-growth approach without candidate generation: As like the traditional Apriori based approach in which generates the candidate and test is used, the PrefixSpan does not perform the candidate generation and test. It only calculates the frequency of local 1-itemset.

2) Partitioning-based approach as best mean for data reduction: The PrefixSpan algorithm creates the longer sequential pattern from the smaller or shorter one by partitioning the search space and concentrating only on the subspace after supporting the pattern-growth. The search space of this algorithm is concentrate and continued to only a set of projected databases. So the projected dataset for the subsequences α contains all and only the required information for mining the superpattern that can grow up from α. The size of the projected database goes on decreasing as per the mining of longer sequential patterns. In other hand the algorithm which is based on the Apriori algorithms works on the entire or whole database once for all iterations during the mining process. Many insignificant results have to be examined and checked which leads to increase the overhead; this may results in the performance degradation.

3) PrefixSpan requires comparatively stable and less memory space: For the algorithm based on the Apriori approach, they require the candidate generation and test method as well. For both GSP and SPADE requires a huge amount of memory when the support qualify value goes low, since it need to hold a huge number of candidate sets. Instead for the PrefixSpan, it doesn't generate any candidates and explores the divide-and-conquer methodology, so it requires the constant memory space over the mining process.

We illustrate in Table 1 comprehensive study of existing Apriori-based algorithms (GSP, SPADE) and Prefix-Project Pattern-Growth algorithm (PrefixSpan).

<table>
<thead>
<tr>
<th>Algorithm / Purpose</th>
<th>Apriori-based Algorithm</th>
<th>SPADE(Sequential Pattern Discovery using Equivalent Class) algorithm is used for fast (faster than GSP) discovery of sequential pattern and is proposed by MOHAMMED J.ZAKI [5]</th>
<th>PrefixSpan (Prefix-Projected Sequential Pattern Mining) algorithm is used for discovery of sequential pattern and performs better in mining large sequences. It is proposed by Jian Peiet.al [6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key features</td>
<td>Generate &amp; Test candidate.</td>
<td>-A vertical format. -Reduce the costs for computing support counts.</td>
<td>Work on projected prefix database (Less projection).</td>
</tr>
<tr>
<td>Performance</td>
<td>GSP is an iterative algorithm. It scans the database number of times depending on the length of the longest frequent sequences in</td>
<td>SPADE outperforms the GSP by a factor two, and by an order of magnitude with precomputed support of 2-sequences. It has</td>
<td>PrefixSpan mines the complete set of patterns but greatly reduces the efforts of candidate sequence generation. Prefix-projection substantially reduces the size of projected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: COMPARISON BETWEEN GSP, SPADE AND PREFIXSPAN
Databases and leads to efficient processing [6] by minimize search space. A comprehensive performance study shows that PrefixSpan, in most cases, outperforms the a priori-based algorithm GSP and SPADE [6].

**Location Memory wise**
- Not a main-memory algorithm, candidate sequences are stored in a hash-tree.
- ID-List completely stored in the main memory by Hash-tree.
- The memory required by GSP and SPADE is same.
- The memory required by PrefixSpan is much less than GSP and SPADE.

**Limitation**
- Multiple scanning
- Multiple passes over the database
- Same pair is recorded more times when it/they appear(s) more than once in the same customer sequence
- repeatedly merge the ID-list

**Format of database**
- Uses Horizontal Format Database[2]
- Uses Vertical Format Database [5]
- Uses Vertical Format Database[6]

**Candidate Sequence**
- Candidate Sequence generation required [2]
- Candidate Sequence generation required [5]
- Candidate Sequence generation is not required.

It uses prefix projection.[6]

### VI. CONCLUSIONS

Since the search space that we want to mine is very large and data volume is huge, it has made many problems for mining sequential patterns. We discussed in this paper and comparison of mining methods of Sequential Pattern also we propose some technique that can be used to increase the efficiency of these methods. The sequential pattern mining is categorized into two main groups, Apriori approach based algorithms such as GSP and SPADE. And pattern growth approach based algorithms like PrefixSpan. From our comparative survey and previous some studies by various researchers on sequential pattern mining algorithms it is found that the algorithm which are based on the approach of pattern growth like PrefixSpan are better in terms of scalability, time-complexity and space-complexity.

### REFERENCES


