A Study of Sequential Pattern Mining Algorithms

Mihika Shah
Dwarkadas J. Sanghvi College of Engineering
Mumbai, Maharashtra, India

Prof. Lynette D’mello
Dwarkadas J. Sanghvi College of Engineering
Mumbai, Maharashtra, India

ABSTRACT
Research on mining sequential patterns has been carried out on a large scale. It was first introduced for market analysis, and it has been increasingly used for various other applications. Various algorithms have been proposed for sequential pattern mining, and they can be broadly divided into two categories – apriori based and pattern growth based. This paper presents a survey on some of the existing algorithms in both categories, like GSP, SPADE, SPIRIT, SPAM, FreeSpan, and PrefixSpan.

Keywords
Sequential pattern mining, apriori based algorithms, pattern growth based algorithms

INTRODUCTION
Data mining is the process of extracting interesting, previously unknown and potentially useful information from large repositories of information[1]. Finding frequent patterns in the vast pool of data that is available to us is an essential part in data classification, clustering and other data mining tasks as it helps us find associations, correlations and other interesting patterns in data. Items that appear together frequently is known as a frequent itemset. Sequential patterns are those patterns where there exists a sequence to the items appearing together (for example first X, then Y, and finally Z). Sequential pattern mining was first introduced by [2] for a study of customer purchase sequences. The most important aspect of sequential pattern mining is the order in which the items occur in the itemset. Sequential pattern mining can be used for a wide variety of purposes, like finding user access patterns for websites, using symptoms of a patient to predict a disease and also making inventory control efficient[1]. Sequential pattern mining finds all those frequent subsequences, whose minimum support threshold min_sup is not less than that specified by the user, and prunes out the rest. An important aspect of finding frequent subsequences is that those subsequences need not be consecutive i.e. other items can be placed between them, however, what is important is the order of the items. For example, customers may purchase a notebook first, then a pencil, and finally an eraser.

SEQUENTIAL PATTERN MINING ALGORITHMS
Sequential Pattern Mining has been greatly studied in recent years, and there exist various algorithms exist to mine these patterns. The categories into which sequential pattern mining can be classified are: Apriori based algorithms, and pattern growth based algorithms.

Apriori Based Algorithms
These algorithms are based on the apriori property which has been proposed for association rule learning. The property states that any sub-pattern of a frequent pattern must be frequent. The Apriori algorithm is described as antimonotonic, which means that the support of an itemset never exceeds the support of its subsets. While the Apriori algorithm has been the base of many efficient algorithms developed later, it is not efficient enough. With the Apriori property, the frequent itemsets are mined by identifying the frequent 1-itemsets (first scan of database). Next, the frequent 1-itemsets are used to generate candidate frequent 2-itemsets, and this process is again repeated to obtain the frequent 2-itemsets. This process is iterated until any frequent k-itemsets are generated for some k.

Key features of Apriori-based algorithm[3]
- Breadth-first search: Apriori-based algorithms are breadth-first (level-wise) search algorithms because they build all the k-sequences, in the kth iteration of the algorithm, as they pass through the search space.
- Generate-and-test: Algorithms that use this approach execute inefficient pruning method as they first generate a huge number of candidate sequences and then check for satisfying some user specified constraints. Hence this pruning process
consumes a lot of memory in the initial stages of sequential pattern mining.

Multiple scans of the database: This feature consists of scanning the original database to determine whether a long list of generated candidate sequences is frequent or not. It is an extremely undesirable characteristic of most apriori-based algorithms and requires a lot of processing time and I/O cost.

**GSP (Generalized Sequential Patterns)**

It is an apriori-based algorithms that integrates with time constraints and relaxes the definition of transaction. It also considers the knowledge of taxonomies[1]. GSP uses the downward-closure property of sequential patterns and adopts a multiple-pass, candidate generate-and-test approach[4]. This algorithm is not a main-memory algorithm. If the candidates do not fit in memory, the algorithm generates only those many candidates that will fit in memory. The data is then scanned to count the support of these candidates. The candidates without minimum support are deleted, and the frequent sequences resulting from these candidates are written to disk.

The GSP algorithm makes multiple passes over the sequence database as follows:

- It discovers the frequent sequences that have the minimum support in the first pass.
- Every data sequence is examined at each pass in order to update the occurrence number of the candidates which is contained in this sequence.

**SPADE(Sequential Pattern Discovery using Equivalence classes)**

SPADE utilizes combinatorial properties to decompose the problem into smaller sub-problems, which are then solved independently in main memory using simple join operations and efficient lattice search techniques[5]. All the sequences in SPADE are discovered in only three passes. Moreover, the number of input sequences and other database parameters have linear scalability in SPADE.

SPADE maps a sequence database into vertical data format. It requires one scan to find the frequent 1-sequences. To find the frequent 2-sequences, the vertical to horizontal database is scanned. Finally, during the third scanning of the database, all the longer sequences are enumerated by using temporal join.

The use of vertical data format reduces scans of the sequence database. However, SPADE generates large sets of candidates in breadth first manner in order to grow longer sequences, despite the apriori pruning. Hence, most of the difficulties that occur in GSP occur in SPADE as well[4].

**SPIRIT (Sequential Pattern Mining with Regular expression constraints)**

SPIRIT mines user specified sequential patterns by using regular expression constraints. Regular expressions were chosen because of their simple and natural syntax. It saves computing effort by not mining patterns the users are not interested in. Apart from this, SPIRIT uses constraint-based pruning followed by support-based pruning and then storing the regular expressions in finite state automata, and this improved the performance significantly. There exist 4 algorithms of SPIRIT, the difference between them being the extent to which the constraints are pushed into the candidate generation and pruning processes. SPIRIT(N) uses only the including constraints to prune the candidate sequences, and every other process is the same as GSP. SPIRIT(L) checks every candidate k-sequence according to the regular expression by using legal operator and SPIRIT(V) checks it according to valid operator. In SPRIT(R), the candidates are generated by enumerating the possible combination of path traverse of the regular expression. Experiments have shown that SPIRIT(R) outperforms the other three algorithms when constraints are highly selective, while SPIRIT(V) is the overall winner in most other cases[1].

**SPAM (Sequential Pattern Mining)**

It is an algorithm to find sequential pattern in a transaction database. It is particularly efficient when the sequential patterns in the database are very long. To generate candidate sequences, a depth first strategy is used, and then various pruning mechanisms are employed to reduce search space. The transactional data is stored in a vertical bitmap representation, due to which efficient support counting and significant bitmap compression can be done. Each bitmap contains a bit, which represents each transaction in the dataset. If an item i appears in a transaction j, then the bit relative to transaction j of the bitmap for item i is set to 1, else it is set to 0. Also, an important feature of this algorithm is that it outputs the new frequent itemsets in an online and incremental fashion. Experimental results show that it is more efficient compared to SPADE and PrefixSpan on large datasets, but requires more memory than them.
Pattern Growth Algorithms

The main focus of pattern growth algorithms is to avoid the candidate generation step, and to focus the search on a limited portion of the database. The search space partitioning feature plays an important role in pattern growth. Almost every pattern growth algorithm starts with building a representation of the database to be mined, then proposes a way for the search space to be partitioned, generating as few candidate sequences as possible by growing on the already mined frequent sequences, and finally applying the apriori property as the search space is being traversed recursively looking for frequent sequences.

Key features of pattern growth based algorithms:

- **Search space partitioning:** It allows partitioning of the generated search space of large candidate sequences for efficient memory management. There are various ways to partition the search space. Smaller partitions can be mined on parallel once the search space is partitioned.

- **Tree projection:** In tree projection, the algorithms implement a physical tree data structure representation of the search space, which is then traversed depth-first or breadth-first in search of frequent sequences. Pruning is based on the apriori property.

- **Depth first traversal:** Depth first traversal makes a big difference in performance, and it also helps in the early pruning of candidate sequences as well as mining of closed sequences. This performance is only because depth-first traversal uses far less memory, more directed search space and hence generates less candidate sequences than breadth-first traversal.

- **Candidate sequence pruning:** Pattern growth algorithms try to utilize a data structure that allows them to prune candidate sequences early in the mining process. This results in early display of smaller search space and maintains a more directed and narrower search procedure.

**FreeSpan (Frequent Pattern Projected Sequential Pattern Mining)**

FreeSpan is an algorithm which aims to reduce the candidate subsequences. FreeSpan uses frequent items to project sequence databases into a set of smaller projected databases recursively using the currently mined frequent sets, and generates subsequence fragments in each projected database. This process partitions both the data and the set of frequent patterns to be tested, and confines each test being conducted to the corresponding smaller projected database. FreeSpan scans the original database only thrice, irrespective of the maximal length of the sequence. Experimental results show that it is more efficient and faster as compared to GSP[6]. The drawback of FreeSpan is the considerable amount of sequence duplication that takes place, as the same sequence could appear in more than one projected database. However, the size of each projected database usually (but not necessarily) decreases rapidly with recursion[7].

**PrefixSpan**

It is an algorithm for sequential pattern mining which does not require candidate generation. It uses the method of database projection, and each projected database is associated with one frequent item, and each database is then mined separately[4]. The main idea of PrefixSpan is that, instead of projecting sequence database by considering all possible occurrences of frequent subsequences, the projection is based only on frequent prefixes. Moreover, it is significantly faster than GSP and FreeSpan. The major cost of PrefixSpan is the construction of projected database.

**EXTENSION OF SEQUENTIAL PATTERN MINING**

Sequential pattern mining has been studies extensively during recent years, and there exists a wide variety of algorithms to implement sequential pattern mining. However, because of the vast potential of mining sequential patterns, research on extensions of sequential pattern mining have also been carried out. Some of these extensions are mining closed sequential patterns, multidimensional, constraint based and time based sequential pattern mining.

**Closed Sequential Pattern Mining**

Most of the sequential pattern algorithms developed perform well in databases consisting of short frequencies. However, when mining long frequencies, or using low support thresholds, the performance degrades. A solution for this is to mine closed patterns. In closed sequential pattern mining, only frequent closed sequences are mined, that is, those containing no supersequence with the same support. This will generate a lesser number of discovered sequences, while having the same expressive power. Also, because of their compactness, they may be quicker to find. Some of
the algorithms which are used for mining closed sequential patterns are CloSpan and BIDE [6].

**CloSpan**

CloSpan was proposed to reduce the time and space cost for huge frequent sequential patterns. It mines only the frequent closed subsequences, and not the complete set of frequent subsequences. CloSpan consists of two stages: in the first stage, a candidate set larger than the final closed sequence set is generated. In the second stage, elimination of non-closed sequences is done by pruning them[6]. This means that if there exist two prefix-based projected databases that are exactly the same, one of them can be stopped from growing. Thus backward subpatterns and backward superpatterns can be pruned[4].

**BIDE[6]**

BIDE adopts a closure checking scheme, which mines closed sequential patterns without candidate maintenance. Thus, BIDE avoids the candidate maintenance and test problem suffered by CloSpan. It has linear scalability for the number of sequences in the database. Experimental results show that BIDE is more efficient compared to CloSpan.

**Multidimensional Sequential Pattern Mining**

Single dimensional sequential pattern mining consists of only one attribute with time stamps in the pattern discovery process, while in multidimensional sequential pattern mining, multiple attributes may be considered. This type of pattern mining gives more informative and useful patterns than single dimension pattern mining. The SeqDim, DimSeq and UniSeq algorithms are used for multidimensional sequential pattern mining [1].

**Time-interval Sequential Pattern Mining**

While sequential patterns can tell us the order of items that frequently occur together, there is no indication of the time span between the two items that are frequent together. The solution to this problem is to generalize the mining pattern to discover the time interval between the sequential patterns[7].

**Constraint Based Sequential Pattern Mining**

Mining that is performed without user specific constraints may produce patterns that are of no use to the user. Thus, constraint based sequential pattern mining focuses on user specific constraints to derive only those patterns that are useful to the user. This improves both the efficiency, and the interestingness of the pattern[4].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key Feature</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSP</td>
<td>Uses downward closure property of sequential patterns.</td>
<td>- Only three passes of the database are required. - Input sequences have linear scalability.</td>
<td>- Multiple scans of the database are needed, which generates large candidate sequences. - Inefficient for long sequences. - No incremental mining function.</td>
</tr>
<tr>
<td>SPADE</td>
<td>Uses lattice search techniques.</td>
<td>- Does not mine patterns users are not interested in.</td>
<td>Generates large number of candidate sequences.</td>
</tr>
<tr>
<td>SPIRIT</td>
<td>Uses regular expression constraints.</td>
<td>- Efficient for long sequences. - Efficient support counting.</td>
<td>Requires more memory, as all the data should fit in the main memory.</td>
</tr>
<tr>
<td>SPAM</td>
<td>Data is stored in a vertical bitmap representation.</td>
<td>- Scans original database only thrice. - Reduces search space.</td>
<td>Sequence duplication may take place, as the same sequence could appear in more than one database.</td>
</tr>
<tr>
<td>FreeSpan</td>
<td>Based on projected sequence database.</td>
<td>- Does not require candidate generation. - Reduces search space.</td>
<td>Cost of projected database.</td>
</tr>
<tr>
<td>PrefixSpan</td>
<td>Based on projected prefix sequence database.</td>
<td>-</td>
<td>Cost of projected database.</td>
</tr>
</tbody>
</table>
CONCLUSION
This paper provides an overview of the various algorithms that exist for sequential pattern mining. The algorithms studied in this paper are GSP, SPADE, SPIRIT, SPAM, FreeSpan, and finally PrefixSpan. The different extensions to sequential pattern mining have also been discussed. However, while a lot of work has been done on these algorithms in sequential pattern mining, there exist a few challenges which need to be overcome, like handling large user space, efficiency, scalability, and to find the complete set of patterns which satisfy the minimum threshold [7]. Future research should work on overcoming these challenges, and improve the efficiency of existing algorithms.

ACKNOWLEDGMENT
We sincerely thank Dwarkadas J. Sanghvi College of Engineering and the Computer Engineering department for their support.

REFERENCES