Non-redundant Sequential Association Rule Mining
based on Closed Sequential Patterns

By

Hao Zang

A thesis submitted for the degree of
Master by Research

Faculty of Science and Technology
Queensland University of Technology

November 2010
Abstract

In many applications, e.g., bioinformatics, web access traces, system utilisation logs, etc., the data is naturally in the form of sequences. People have taken great interest in analysing the sequential data and finding the inherent characteristics or relationships within the data. Sequential association rule mining is one of the possible methods used to analyse this data. As conventional sequential association rule mining very often generates a huge number of association rules, of which many are redundant, it is desirable to find a solution to get rid of those unnecessary association rules.

Because of the complexity and temporal ordered characteristics of sequential data, current research on sequential association rule mining is limited. Although several sequential association rule prediction models using either sequence constraints or temporal constraints have been proposed, none of them considered the redundancy problem in rule mining. The main contribution of this research is to propose a non-redundant association rule mining method based on closed frequent sequences and minimal sequential generators. We also give a definition for the non-redundant sequential rules, which are sequential rules with minimal antecedents but maximal consequents. A new algorithm called CSGM (closed sequential and generator mining) for generating closed sequences and minimal sequential generators is also introduced. A further experiment has been done to compare the performance of generating non-redundant sequential rules and full sequential rules, meanwhile, performance evaluation of our CSGM and other closed sequential pattern mining or generator mining algorithms has also been conducted. We also use generated non-redundant
sequential rules for query expansion in order to improve recommendations for infrequently purchased products.
Contents

ABSTRACT ................................................................................................. 1

LIST OF TABLES ....................................................................................... VI

LIST OF FIGURES ....................................................................................... VII

STATEMENT OF ORIGINAL AUTHORSHIP .............................................. VIII

ACKNOWLEDGEMENT .............................................................................. IX

RELATED PUBLICATIONS ......................................................................... X

CHAPTER 1 INTRODUCTION .................................................................... 1

1.1 STATEMENT OF RESEARCH PROBLEMS ........................................ 2

1.2 CONTRIBUTIONS ................................................................................. 4

1.3 STRUCTURE OF THE THESIS ............................................................... 5

CHAPTER 2 LITERATURE REVIEW .......................................................... 7

2.1 DATA MINING IN GENERAL ................................................................. 7

2.2 ASSOCIATION RULE MINING ............................................................. 8

2.3 SEQUENTIAL PATTERN MINING ......................................................... 9

2.3.1 Apriori-like methods and GSP Algorithm ...................................... 11

2.3.2 PrefixSpan ..................................................................................... 13

2.3.3 SPAM ............................................................................................ 14

2.3.4 CloSpan ........................................................................................ 15

2.3.5 TSP: Top-k closed sequence mining ............................................ 16

2.3.6 Sequential Generator Mining ......................................................... 17

2.4 NON-REDUNDANT ASSOCIATION RULE MINING ................................ 18

2.4.1 Non-redundant rules from closed patterns ................................... 19

2.4.2 Min-Max basis .............................................................................. 20
6.5 Efficiency of the Proposed CSGM Rule Mining Method.......................... 70
6.6 Efficiency Comparison Between CSGM and Other Algorithms ............... 73
6.7 Evaluation the Quality of Non-Redundant Sequential Rules .................... 75
6.8 Summary ........................................................................................................ 79

CHAPTER 7 CONCLUSION..................................................................................... 80

7.1 Summary ......................................................................................................... 80
7.2 Limitations .................................................................................................... 81
7.3 Future Work .................................................................................................. 82

REFERENCES ....................................................................................................... 83
List of Tables

Table 3.1 Sequence Database S .................................................................33
Table 3.2 Projected databases ........................................................................34
Table 5.1 An example of sessions ..................................................................57
Table 5.2 User navigation session ....................................................................59
Table 5.3 Search results for each query ..........................................................61
Table 6.1 Parameter description of the synthetic dataset ...............................64
Table 6.2 D5C20N10S20 dataset rule numbers generated ...........................67
Table 6.3 MSNBC dataset rule numbers generated .......................................68
Table 6.4 Car domain dataset rule numbers generated ..................................69
Table 6.5 Rule generation time cost for dataset D5C20N10S20 .................70
Table 6.6 Rule generation time cost for dataset MSNBC ..............................71
Table 6.7 Car domain dataset rule generation time cost ...............................72
List of Figures

Figure 2.1 Prefix Search Tree(a) and Prefix Search Lattice(b)\textbf{Error! Bookmark not defined.}.

Figure 3.1 Lexicographic Tree....................................................................................35

Figure 3.2 Prefix Search Tree ....................................................................................36

Figure 3.3 Pattern Merging .........................................................................................39

Figure 3.4 Sequence-generator pair in a prefix-search tree/lattice .........................42

Figure 5.1 Query Expansions for Car Search System .............................................56

Figure 6.1 Rule numbers for Synthetic dataset D5C20N10S20 .........................67

Figure 6.2 Rule numbers for MSNBC data (10699 sessions) .................................68

Figure 6.3 Rule numbers for Car domain dataset .................................................69

Figure 6.4 Rule generation time cost for dataset D5C20N10S20 .......................71

Figure 6.5 Rule generation time cost for dataset MSNBC ...................................72

Figure 6.6 Rule generation time cost for Car Domain dataset ..............................73

Figure 6.7 Runtime Comparison of CSGM and CloSpan .....................................75

Figure 6.8 Recall of expanded inputs and regular inputs .......................................76

Figure 6.9 Precision of expanded inputs and regular inputs .................................77

Figure 6.10 Recall: Both redundant rules and frequent rules ...............................78

Figure 6.11 Precision: Both redundant rules and frequent rules ...........................78
Statement of Original Authorship

The work contained in this thesis has not been previously submitted; the research to which it refers is the product of my own work. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.
Acknowledgement

First and foremost, I would like to thank my principal supervisor and mentor Dr Yue Xu, who always made time to discuss research, provide guidance, support, help and encouragement in the last two and half years. I would like to thank my associate supervisor Associate Professor Yuefeng Li for his support and guidance for my research work.

I also wish to thank the School of Science and Technology, Queensland University of Technology, which has provided me with a comfortable research environment and financial support, including my scholarship over the period of my candidature. I would also like to thank every member of our research group for offering valuable comments and advice during my research. Finally, I thank my parents for encouraging me to learn all throughout my life and for supporting me in every possible way.

Also, thanks to the CRC (Cooperative Research Centre) Smart Service, Australia for providing the experiment data.
Related Publications

The following paper has been accepted during the course of this research:

Hao Zang, Yue Xu, Yuefeng Li; “Non Redundant Sequential Rule Mining and Application” accepted by WI/IAT 10 (Web intelligence 2010) Workshop, York University, Canada
Chapter 1 Introduction

In many applications, data typically is represented in the form of sequences. Sequence data can be either a consequence of employing a natural temporal ordering among individual data (e.g., web log data, system traces, purchase histories, financial market data) or it can be a result of some inherent physical structure (e.g., genes and DNA sequences). As an important research direction in the field of data mining, recently mining sequential data has drawn more and more attention to researchers in the data mining community (Agrawal & Srikant, 1995; Ayres, Flannick, Gehrke & Yiu, 2002; Pei et al., 2001; Zaki, 2001).

Typical useful knowledge that could be discovered from sequential data is in the form of sequential patterns. A sequential pattern is a sequence of itemsets that frequently occurs in a specific order (Wang, Gao & Li, 2000). Because of the unique nature of ordering, by studying the sequential patterns, people could find user behaviour patterns and other valuable information (Srivastava, Cooley, Deshpande & Tan, 2000) for example, mining user access patterns for Web sites, using the history of symptoms to predict certain kinds of diseases, historic selling patterns for retailers to make their inventory control more efficient (Zhao & Bhowmick, 2003). In the last decade, many algorithms and techniques have been proposed to deal with the problem of sequential pattern mining including Apriori-based approaches such as GSP (Srikant & Agrawal, 1996) and SPADE (Zaki, 2001) and pattern-growth based approaches such as PrefixSpan (Pei et al., 2001) and Spam (Ayres et al., 2002). Even though many
approaches for generating sequential patterns have been proposed, not much work has been done to discover sequential association rules. These existing approaches mainly discuss how to efficiently generate sequential patterns, and do not pay much attention to the quality of the discovered patterns, in particular, all of these approaches suffer from the problem that the volume of the discovered patterns and association rules could be exceedingly large, but many of the patterns and rules are actually redundant and thus need to be pruned (Wang, Li & Zhang Yang, 2005). Mining quality sequential patterns and rules from sequential datasets is a challenge that still needs to be worked on. This research aims to develop new techniques based on the closure concept for effectively and efficiently discovering non-redundant sequential association rules from sequential datasets.

1.1 Statement of Research Problems

Sequential pattern mining, which is the process of extracting certain sequential patterns whose support exceeds a predefined minimal support threshold, has been studied widely in the last decade in the data mining community (Zhao & Bhowmick, 2003). However, less work has been done on sequential association rule mining. Only in recent years, several prediction models which introduced the concept of sequential association rule mining have been proposed, most of which use sequence and temporal constraints in generating association rules (Wang et al., 2005). Similar to classical association rule mining in non-sequential datasets, the existing sequential association rule mining models suffer as a result of the huge number of discovered rules (Guo, Liang, Zhang & Liu, 2007). In classical association rule mining, the resulting rule set can easily contain thousands of rules of which many are redundant and thus useless in practice. While in the case of sequential association rule mining, things get even worse. This is because
the same set of items with different ordering yields different sequential patterns in sequential pattern mining which makes the number of frequent sequential patterns usually much larger than the number of frequent itemsets generated from a dataset of a similar size.

The overwhelming number of extracted association rules and the redundancy existing in the extracted rules are two obstacles that seriously affect the effective use of the extracted knowledge in solving real world problems. This is still an open research question.

Some approaches for generating non-redundant association rules have been proposed in the classical association rule mining area, among which the closure-based approaches (Pasquier, Taouil, Bastide, Stumme & Lakhal, 2005; Xu & Li, 2007, 2008; Zaki, 2004) have proved very effective in eliminating redundancy. In this work we propose to apply the concept of the closure of the Galois connection (Gamter, 1999) to extract non-redundant rules from frequent closed itemsets and minimal generators instead of from frequent itemsets. The approaches of using frequent closed itemsets and generators have been theoretically and also experimentally proved very effective in generating non-redundant association rules from non-sequential datasets (Xu & Li, 2007). However, so far, little work has been done in generating non-redundant sequential association rules. In this Masters thesis work, we propose to apply the idea of using closed patterns and generators to generate non-redundant sequential association rules. For this purpose, new methods to generate frequent closed sequential patterns and sequential generators, and also new methods to generate non-redundant sequential association rules from the closed sequential patterns, must be developed. Thus the main problem this research is aiming to solve is to develop new techniques based on the
closure concept for effectively and efficiently discovering non-redundant sequential association rules from sequential datasets.

1.2 Contributions

The existing classical non-redundant association rule mining techniques could not directly be utilised in sequential association rule mining. This is because sequential pattern mining requires order matching in subsequences (Gaul & Schmidt-Thieme, 2001). This thesis work proposes novel techniques to generate non-redundant sequential association rules from sequential datasets. This study makes important contributions to the field of knowledge discovery as we extend techniques to extract non-redundant sequential association rules. The success of this research project could result in achieving steps towards the goal of extracting high quality sequential association rules for sequential data mining.

In this thesis, we expand the idea of non-redundant rule mining using closed items and generators into sequential mining area. We introduce a new algorithm for mining closed sequential patterns and generators at the same time and a method for composing closed sequence patterns and generators into sequential association rules in order to reduce the number of redundant sequential rules. Our method is much more efficient than the frequent pattern mining algorithm and could result in the reduction of a significant number of redundant sequential rules. We also conduct experiments towards the application of the sequential rules to the area of recommendation making.

The contributions are briefly summarised as follows:
1) Introduce the definition of non-redundant sequential association rules

2) Propose a new algorithm called CSGM for mining closed sequential patterns and sequential generators at the same time

3) Propose a method called SEQ_MinMax that extends the MinMax non-redundant association rule mining method to the sequential mining area. The proposed method generates non-redundant sequential rules based on sequential generators and closed sequential patterns.

4) Apply association rules to recommender systems to improve the accuracy of recommendations.

1.3 Structure of the Thesis

Chapter 1: This chapter states the main issues discussed in the thesis, the problems and the aims of this research project.

Chapter 2: This chapter provides a detailed literature review of related disciplines including association rule mining, sequential pattern mining and web log mining.

Chapter 3: This chapter introduces the CSGM algorithm which is used to mine closed sequences and sequential generators.

Chapter 4: This chapter presents the algorithms of non-redundant sequential rule mining.
Chapter 5: This chapter presents the application of association rules to improve product recommendation.

Chapter 6: This chapter provides experiment results and evaluation to show the effectiveness achieved by the proposed approaches.

Chapter 7: This chapter concludes the thesis and proposes some research problems for future work.
Chapter 2 Literature Review

2.1 Data Mining in General

Data mining is the process of automatically discovering useful information in large data repositories. It could not only find novel and useful patterns but also provide capabilities to predict the outcome of a future observation. Several commonly used mining techniques including classification, clustering and association rule mining.

Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications. In standard classification procedure, individual items are placed into groups based on quantitative information associated with one or more characteristics inherent in the items.

Cluster analysis divides data into several clusters (groups); each cluster has meaningful, useful characteristics. Cluster analysis provides an abstraction from individual data objects to the clusters in which those objects reside, the goal of clustering is that the objects within a group must be similar or related to each other and must be different or unrelated to those objects in other groups. Homogeneity within a group and differences between groups must occur to a certain level in order to achieve distinct clusters.

Association analysis is useful for discovering interesting relationships in the datasets, such relationships could be represented in the form of association rules and sets of frequent patterns.
2.2 Association Rule Mining

Association rule mining, which aims to extract interesting correlations and associations among sets of items in large datasets, has two phases: extracting frequent itemsets and generating association rules from the frequent itemsets with the constraints of minimal support and minimal confidence. The rules with a confidence value larger than a user-specified minimum confidence threshold are considered interesting or useful. This is known as a frequent based approach.

Traditionally, there are two basic measures used in association rule mining, support and confidence. Support is a measure that defines the percentage/fraction of records/entries in the dataset that contain \( X \cup Y \) to the total number of records/entries. The basic method for determining the support does not take into account the quantity of the item with a record and the support value serves as a statistical significance of the rule (Zhao & Bhowmick, 2003). Confidence is a measure that defines the percentage/fraction of records/entries in the dataset that contain \( X \cup Y \) to the total number of records/entries that contain just \( X \). The confidence value serves as a measure of the strength or precision of the rule (Zhao & Bhowmick, 2003).

Apriori is the first association rule mining algorithm that pioneered the use of support-based pruning to systematically control the exponential growth of candidate itemsets (Agrawal, Imielinski & Swami, 1993).

The Apriori algorithm is a level-wised algorithm which employs a generate-and-test strategy to find frequent itemsets. It is one of the most influential association rule mining algorithms. However, nowadays, the transaction datasets have become far
larger than they were 10 years ago. The Apriori algorithm nows faces two problems in dealing with large datasets, first of all it requires multiple scans of the transaction database, incurring a major time cost; in addition, it generates too many candidate sets which take up quite a lot of memory space.

Since all of the Apriori-based mining algorithms have time or space costing problems when handling a huge number of candidate sets and a large database, a new method which avoids candidate generation-and-test and utilises a new data structure to reduce cost was introduced by Han, Pei and Yin (2000). It is FP-Tree algorithm.

Compared to the Apriori-like mining algorithm, the FP-Tree algorithm avoids the candidate generating step, and it only need to scan the whole dataset/transaction database twice. So it is much faster than the Apriori-like algorithm. It is also claimed that the pattern-growth approach is more efficient and scalable than other approaches, such as Apriori, (Han, Pei & Yin, 2000) and is effective in mining dense databases/datasets (because of the way the structure encapsulates concisely valid item information) (Ceglar & Roddick, 2006) which are becoming more common.

### 2.3 Sequential Pattern Mining

The sequential pattern is a sequence of itemsets that frequently occurred in a specific order, all items in the same itemset are supposed to have the same transaction-time value or within a time-gap. Sequential patterns indicate the correlation between transactions while association rules represent intra-transaction relationships.
Sequential pattern mining was first introduced by Agrawal and Srikant (1995). It is the process of extracting certain sequential patterns whose support exceeds a predefined minimal support threshold. Since the number of sequences can be very large, and users have different interests and requirements, to get the most interesting sequential patterns usually a minimum support is predefined by the users. By using the minimum support we can prune out those sequential patterns of no interest, consequently making the mining process more efficient. Obviously a higher support of a sequential pattern is desired for more useful and interesting sequential patterns. However some sequential patterns that do not satisfy the support threshold are still interesting. Yang, Wang and Yu (2001) introduced another metric called surprise to measure the interestingness of sequences. A sequence $s$ is a surprising pattern if its occurrence differs greatly from the expected occurrence, when all items are treated equally. In the surprise metric the information gain was proposed to measure the overall degree of surprise, as detailed by Yang et al. (2001).

Sequential pattern mining is used in a great spectrum of areas. In computational biology, sequential pattern mining is used to analyse the mutation patterns of different amino acids. Business organisations use sequential pattern mining to study customer behaviours. Sequential pattern mining is also used in system performance analysis and telecommunication network analysis.

General algorithms for sequential pattern mining have been proposed recently. Most of the basic and earlier algorithms for sequential pattern mining are based on the Apriori property proposed by Agrawal and Srikant (1994). The property states that any sub-pattern of a frequent pattern must be frequent. Based on this heuristic, a series of Apriori-like algorithms have been proposed: AprioriAll, AprioriSome, DynamicSome
Later, a series of data projection based algorithms were proposed, which included FreeSpan (Han, Pei, Mortazavi-Asl et al., 2000) and PrefixSpan (Pei et al., 2001). Several closed sequence mining algorithm were also introduced, such as CloSpan (Yan, Han & Afshar, 2003) and TSP (Tzvetkov, Yan, & Han, 2005). Most of these projection-based algorithms will be briefly explained in the next subsections.

2.3.1 Apriori-like methods and GSP Algorithm

Classical Apriori-based sequential pattern mining algorithms were first introduced by Agrawal and Srikant (1995). Given the transaction database containing customer sequences, each of which has three attributes: customer-id, transaction-time and purchased-items ordered according to purchase time, the mining process was decomposed into five phases:

1 Sort Phase: the original transaction database is sorted with customer-id as the major key and transaction time as the minor key, the result is a set of customer sequences.

2 $L$-itemsets Phase: the sorted database is scanned to obtain large 1-itemsets according to the predefined support threshold.

3 Transformation Phase: the sequences are replaced by those large itemsets they contain, all the large itemsets are mapped into a series of integers to make the mining more efficient. At the end of this phase the original database...
is transformed into set of customer sequences represented by those large itemsets.

4 Sequence Phase: all frequent sequential patterns are generated from the transformed sequential database.

5 Maximal Phase: those sequential patterns that are contained in other super sequential patterns are pruned in this phase, since we are only interested in maximum sequential patterns.

The Apriori-like algorithm is not so efficient, but it became the basis of many efficient algorithms developed later.

GSP (Generalized Sequential Pattern) was introduced by Srikant and Agrawal (1996), it is also an Apriori-based pattern mining algorithm. The whole algorithm has two subprocesses: candidate pattern generation and frequent pattern generation.

In the candidate generation process, candidate k-sequences are generated based on the large (k-1) – sequences using the same method described by Agrawal and Srikant (1994).

Given a sequence \( s = < S_1, S_2, ..., S_n > \) and subsequence \( c \), \( c \) is a contiguous subsequence of \( S \) if any of the following conditions hold:

1). \( c \) is derived from \( s \) by dropping an item from either \( S_1 \) or \( S_n \).

2). \( c \) is derived from \( s \) by dropping an item from an element \( S_j \) that has at least 2 items.

3). \( c \) is a contiguous subsequence of \( c' \) and \( c' \) is a contiguous subsequence of \( s \).
The candidate sequences are generated in two steps: joining phase and pruning phase.

In the joining phase, candidate k-sequences are generated by joining two 
\((k-1)\)-sequences that have the same contiguous subsequences. When joining the two sequences the item can be inserted as a part of the element or as a separate element. For example, \(<(a,b)(c)>\) and \(<(a,b)(d)>\) have the same contiguous subsequence \(<(a,b)>\), based on those candidate 4-sequence \(<(a,b)(c,d)>,<(a,b),(c)(d)>\) and \(<(a,b)(d)(c)>\) can be generated.

While in the pruning phase, those candidate sequences that have a contiguous subsequence whose support count is less than the minimal support are deleted. It also uses the hash-tree structure to reduce the number of candidates to be checked in the next phase.

2.3.2 PrefixSpan

PrefixSpan (Pei et al., 2001) is a more efficient algorithm for mining sequential patterns compared with Aprioriall. PrefixSpan is also capable of dealing with very large databases. PrefixSpan mainly employs the method of database projection to make the database for next pass much smaller and consequently increasing the speed of the algorithm. Also in PrefixSpan there is no need for candidate generation, this step is instead by recursively generating projected database according to the sequence prefix.

The first step of PrefixSpan is to scan the sequential database to get the length-1 sequence, which is in fact the large 1-itemsets. Then the sequential database is divided into different partitions according to the number of length-1 sequences, where each partition is the projection of the sequential database that takes the corresponding length-1 sequences as Prefix. The projected database only contains the postfix of these
sequences. By scanning the projected database, all the length-2 sequential patterns that have the parent length-1 sequential patterns as Prefix can be generated. Then the projected database is partitioned again by those length-2 sequential patterns. The same process is executed recursively until the projected database is empty or no more frequent length-k sequential patterns can be generated.

PrefixSpan mainly avoids generating and counting candidate sequences, which is the most time-consuming part of Apriori-like sequential mining methods. By using projection, the database that PrefixSpan scans each subsequent time is much smaller than the original database. The main cost of PrefixSpan is the projected database generation process, and in order to improve the performance a bi-level projection method that uses the triangle S-Matrix is introduced.

### 2.3.3 SPAM

SPAM (Sequential PAttern Mining) is a typical algorithm which integrates a variety of old and new algorithmic contributions. It is introduced by Ayres et al. (2002), a lexicographic tree has been used to store all the sequences. SPAM traverses the sequence tree in a standard depth-first search (DFS) manner. At each node n, the support of each sequence-extended child is tested. If the support of a generated sequence s is greater than or equal to minimum support, SPAM stores that sequence and repeats the DFS recursively on s. (Note that the maximum length of any sequence is limited since the input database is finite.) If the support of s is less than minimum support, then SPAM does not need to repeat the DFS on s by the Apriori principle, since
any child sequence generated from s will not be frequent. If none of the generated children are frequent, then the node is a leaf and user can backtrack up the tree.

SPAM uses a vertical bitmap representation of the data. A vertical bitmap is created for each item in the dataset, and each bitmap has a bit corresponding to each transaction in the dataset. If item i appears in transaction j, then the bit corresponding to transaction j of the bitmap for item i is set to one; otherwise, the bit is set to zero.

To enable efficient counting and candidate generation, the bitmap should be partitioned in order to make sure all transaction sequences in the database appear together in the bitmap.

SPAM uses depth-first traversal, and its bitmap representation idea requires quite a lot of memory. So it is a space-inefficient algorithm, however, it is very efficient for those databases which have very long sequential patterns.

2.3.4 CloSpan

In order to reduce the time and space cost when generating explosive numbers of frequent sequence patterns, CloSpan (Closed Sequential Pattern Mining) was developed. Instead of mining the complete set of frequent subsequences, it mines only frequent closed subsequences, which are the sequences containing no super sequence with the same support (occurrence frequency) (Yan et al., 2003).

CloSpan divides the mining process into two stages. In the first stage, a candidate set is generated. Usually this candidate set is larger than the final closed sequence set. This set is called a suspicious closed sequence set (a super set of the closed sequence set). In the second stage, a pruning method is called to eliminate non-closed sequences.
One major difference between CloSpan and PrefixSpan is that CloSpan implements an early termination mechanism to avoiding unnecessary traversing of search space.

By using both backward sub-pattern and backward super-pattern methods, some patterns will be absorbed or merged, and the search space growth can be reduced.

2.3.5 TSP: Top-k closed sequence mining

The TSP algorithm is a typical sequential mining algorithm; it could find the top-k closed sequential patterns from the sequence database. It only selects the top-k wanted sequences, which could avoid the users having to set the minimal sequence support value (considered as a trivial and very difficult task).

The TSP algorithm also uses the concepts of prefix projection-based sequential pattern mining and the PrefixSpan algorithm. It uses a hash collection, called SID-Hash, as a key in the phase of verifying the closed sequential pattern or minimal generator (Tzvetkov et al., 2005).

The algorithm first scans the whole database to find length-1 frequent items, which could either item-step extend or sequence-step extend to the given sequence. Having found all the length-1 frequent sequential patterns, the algorithm divides the search spaces into several parts. The next step finds all the subsets of sequential patterns. The subsets of sequential patterns could be mined by constructing corresponding projected databases and each of which would be mined recursively. The algorithm runs recursively until all the frequent sequences that have prefix are discovered. The algorithm termination condition is: the number of sequences in the projected database is less than the minimum support (Tzvetkov et al., 2005).
Another advantage of the TSP algorithm is that it could mine top-k closed patterns without minimum support threshold (setting minimum support is a subtle task: neither too small nor too big is suitable). The TSP mining process starts with minimum support equals 1, raises it progressively during the mining process, and then uses the raised minimum support to prune the search space.

2.3.6 Sequential Generator Mining

In a sequential database, the sequential generator refers to patterns without any subsequence with the same support. Sequential generators used together with closed sequential patterns can bring additional information, which closed sequential patterns alone are not able to provide. According to the Minimum Description Length (MDL) principle (Li, Li, Wong, Pei & Dong, 2006), generator are the minimal member and preferable over closed patterns in terms of association rule induction and classification because the length of generator is shorter than closed patterns. However, unlike closed sequential pattern mining which has several proposed methods, only one sequential generator mining algorithm has been proposed. The GenMiner method proposed by Lo, Khoo and Liy (2008), is the first sequential generator mining algorithm; it fills the research gap in sequential generator mining.

The GenMiner method can extract sequential generators in a three-step compact-generate-and-filter approach. GenMiner first discovers all the frequent sequential patterns and produces a compact representation of the space of those frequent patterns in a lattice format (Yan et al., 2003). The second step retrieves all candidate generators from the lattice and prunes sub-search spaces containing the non-generators by the unique characteristic of sequential generator: the extension of a
non-generator sequential pattern is also not a sequential generator. The second step could ensure that the candidate generator set is not too large. In the final step, all non-generators from the candidate set are filtered out.

The lattice in GenMiner is called PSL (Prefix Search Lattice) which is a compact form of Prefix Search Tree (PST) described in 2.3.2. The PSL is meant to avoid space explosion by identifying and grouping common sub trees occurring in a PST.

![Figure 2.1 Prefix Search Tree(a) and Prefix Search Lattice(b)](image)

The performance study shows that GenMiner can run a lot faster than a full frequent sequential pattern mining method, such as PrefixSpan, and its speed can be on par with or at times faster than that of closed sequential pattern mining algorithms, such as CloSpan.

### 2.4 Non-redundant association rule mining

Association rule mining finds hidden correlations among different items by using two thresholds known as support and confidence. However, it is always a difficult problem to choose the support and confidence values, and the improper selection of these
thresholds could lead association rule mining algorithms to produce many redundant rules (Ashrafi, Taniar & Smith, 2007). The support-confidence approach has a number of deficiencies. For instance, high confidence and large support rules are often trivial or well-known and thus of little use as they do not convey to the user any new information. Also, many variants of a rule might be produced thus giving the same or similar information repeatedly (this is known as redundancy), and some rules are just incidental. Redundant rules give very little, if any, new information or knowledge to the user and often make it more difficult to find new knowledge (and perhaps they even contribute to overwhelming the user when it comes to finding interesting or important rules). And such rules could be a main obstacle to efficiently discovering association rules. As a result, it becomes obvious of the need to find an alternative approach that could reduce the number of redundant rules.

Eliminating redundant association rules has attracted a great deal of attention from researchers during recent years. One approach is using the frequent closed itemset instead of the frequent itemset to generate rules. The notion of a closed frequent itemset has its origins in the mathematical theory of Formal Concept Analysis (FCA). Closed itemsets generate association rules in the same manner as for frequent itemsets.

### 2.4.1 Non-redundant rules from closed patterns

Zaki presented a rule generation framework based on the frequent closed itemset (Zaki, 2000). The number of association rules produced by Zaki is exponentially smaller than the rule set from the traditional approach (generating rules from full frequent itemsets). This is because the association rule generating procedure requires all elements in the
frequent itemset as a rule antecedent, since the size of frequent closed itemset is normally much smaller than the full frequent itemset, the cost of rule generation of his framework is much lower. Zaki also mentioned the framework does not lose any information when deducing the large number of association rules, he considered the closed itemset uniquely determines the set of all frequent itemsets and their exact frequency.

In his another work (Zaki, 2004), the non-redundant association rules are defined as the more general rules, which means those rules are having minimal antecedent and consequent, in terms of subset relation. The idea of the non-redundant association rule generating method is based on the Galois closure operation. The non-redundant rules are generated by the minimal generators, and frequent closed itemsets. The minimal generators are those itemsets without any subsets sharing the same support, while the closed itemsets are those itemsets without any supersets sharing the same support.

### 2.4.2 Min-Max basis

Pasquier et al. introduced the theory of a condensed representation of association rules (Pasquier et al., 2005). This representation is also characterised by frequent closed itemsets and their generators. In this representation, non-redundant association rules are defined as those having minimal antecedent and maximal consequent, therefore these rules are also called min-max association rules.

In this work, an association rule is considered redundant if it brings the same information or less information than is brought by another rule of the same support and
confidence. Pasquier et al defined two basis called min-max exact and min-max approximate which contain all min-max association rules.

In the min-max exact rule basis, antecedent and consequent itemsets that belong to the same closure also have the same support. In a word, rules in the min-max exact rule basis have the confidence set at 100%. All exact association rules and their supports can be deduced from the min-max exact basis.

In the min-max approximate basis, confidence of the rule is less than 100%, the support of the antecedent is greater than that of the consequent. All approximate association rules with their support and confidence can be deduced from the min-max approximate basis. A further reduction work can be put on the min-max approximate basis, which is to reduce those transitive min-max approximate rules, the non-transitive min-max rules are in the form of \( g \rightarrow (f \setminus g) \), where \( f \) is a frequent closed itemset and \( g \) a frequent generator, such that the closure of \( g \) is an immediate predecessor of \( f \).

Pasquier’s work also provides the method that can generate the full set of association rules, as well as their support and confidence, based on the two basis.

2.4.3 Concise representation of association rules

Xu and Li enhanced the redundant association rule reduction by using another concise representation of association rules. Both reliable exact basis (Xu & Li, 2007) and reliable approximate basis (Xu & Li, 2008) have been introduced.

The non-redundant rules in their work are those rules having minimal antecedent and maximal consequent, this concept is similar to Pasquier et al. However, in Pasquier’s
definition, a rule is redundant to another rule if the rule has longer antecedent, shorter consequent, identical confidence and identical support, while in the reliable basis, the threshold of redundancy does not require identical support. Normally, the reliable basis is the subset of Pasquier’s min-max basis.

Xu and Li use of certainty factor theory (Tsadiras & Margaritis, 1998) proved that the elimination of redundant rules defined in the reliable basis will not reduce the belief to the extracted non-redundant rules, which means the elimination of redundancy will not cause information loss problems.

The experiment performance shows that by using reliable basis, the number of extracted rules could be significantly reduced. Reliable basis is more concise than min-max basis, and can generate higher quality association rules than min-max basis. In addition, from the reliable basis, all association rules can be deduced.

2.5 Sequential Association Rule Mining

The idea of sequential association rule is a general concept. Since the application of sequential patterns is wide ranging, the diversity suggests that it may not be possible to apply a single sequential pattern model to solve all of these problems. Recent research has established several meaningful sequential pattern models for rule mining, they are frequent pattern based models, regular or periodical pattern based models, statistically significant based models and approximate patterns. This research is mainly focused on frequent pattern based sequential rule mining.
For frequent pattern based sequential rule mining, one important task is to discover the frequent patterns. After finding these frequent patterns, then relationships between the sets of the sequences can be extracted.

The definition of a sequential association rule derived from frequent sequence patterns is almost the same as a conventional association rule:

Given s and s’ representing two different frequent sequences in sequential datasets, \( s \rightarrow s' \) constitutes a sequential association rule if its confidence is larger than a given threshold. The confidence of \( s \rightarrow s' \) is defined as \( \text{confidence}(s, s') = \frac{\text{support}(s, s')}{\text{support}(s)} \), where \( \text{support}(s, s') \) means the numbers of sequential transactions containing both s and s’ in sequential transaction sets.

The sequential association rule mining method is introduced in Agrawal & Srikant, (1995). Sequential rules are extracted from the frequent sequential pattern set. For each frequent pattern, every combination of prefix is chosen as antecedent, while the rest of pattern is selected as consequent. The detail of the algorithm is listed below.

**Algorithm: FullRulesMining** \((\text{min}\_\text{sup}, \text{min}\_\text{confidence}, F)\)

**Input:** \(\text{min}\_\text{sup}, \text{min}\_\text{confidence}, \) frequent sequential pattern set \(F\)

**Output:** The complete Rule set \(FR\) (full-sequential rules)

1: Let \(F = \) Frequent Sequential Patterns

2: **For each** pattern \(f \in F\)

3: \hspace{1em} **For each** prefix \(pre\) of pattern \(f\)

4: \hspace{2em} Let \(post = f / pre\) (remove prefix \(pre\) from pattern \(f\))

5: \hspace{2em} Let rule \(r = pre \rightarrow post\), \(\text{support}(r) = \text{support}(f)\)

\hspace{2em} And \(\text{confidence}(r) = \frac{\text{support}(f)}{\text{support}(pre)}\)

6: \hspace{2em} **IF** \(\text{confidence}(r) \geq \text{min}\_\text{confidence}\)

7: \hspace{2em} Add rule \(r\) into rule set \(FR\)
It is a time and space cost work to generate all the sequential association rules, especially in the situation where the number of frequent sequential patterns is huge and the average length of a sequential pattern is quite long.

2.6 Web Mining

Buried in the enormous, heterogeneous and distributed information on the Web is knowledge with great potential value. With the rapid development of the Web, it is urgent and important to provide users with tools for efficient and effective resource discovery and knowledge discovery on the Web. Web mining is the use of data mining techniques to automatically discover and extract information from web documents and services. It is an integrated technology in which several research fields are involved, such as data mining, computational linguistics, statistics, informatics and so on.

According to the kinds of data to be mined, Web mining is divided into three distinct categories (Desikan, Pathak, Srivastava & Kumar, 2005; Kosala & Blockeel, 2000; Srivastava et al., 2000): Web Content Mining, Web Structure Mining, and Web Usage Mining. In our research, we focused on Web usage mining.

For web applications, where users’ requests are satisfied by downloading pages to their local machines, the use of mining techniques to predict access behaviours and hence help with prefetching of the most appropriate pages to the local machine cache can dramatically increase the runtime performance of those applications. These mining techniques analyse web log files composed of listings of page accesses (references) organised typically into sessions. These techniques are part of what is called Web Usage Mining, a term first introduced by Cooley (Cooley, 1997).
Typically, Web usage mining techniques rely on a Markov assumption with depth n. This means that it is assumed that the next request page depends only on the last n pages visited. A study conducted by Jespersen, Pedersen and Thorhauge (2003) showed that Markov-based structures for Web usage mining are best suited for prefetching, targeted ads, and personalisation.

Web usage mining approaches can be classified, based on the type of patterns they produce, into three categories: association rules, frequent sequences, and frequent generalised sequences. With association rules, the problem of finding web pages visited together is similar to finding association among itemsets in transaction databases. Frequent sequences can be thought of as an ordered (by time) list of non-empty itemsets, and hence frequent sequence mining can be thought of as association rule mining over temporal datasets. A frequent generalised sequence is a frequent sequence that allows wildcards in order to reflect the user’s navigation in a flexible way. Among the three approaches, the frequent sequence approach gives better results than the association rules and the frequent generalised sequence approaches when we need to find the correct predictions within the first predictions.
2.7 Summary

In this chapter, the background of association rule mining, sequential mining and web log mining has been discussed and reviewed. We reviewed the algorithms for sequential pattern mining, especially addressed the techniques such as projected database, frequent or closed sequential pattern and minimal sequential generator retrieval. We then focused on the definition of association rule redundancy and discussed several presentation models for non-redundant association rules. In addition, we briefly reviewed the literature regarding Web usage mining.
Chapter 3 Sequential Pattern Mining

Sequential pattern mining is the procedure to extract frequent sequences from sequential transaction databases. Many algorithms have been introduced to mine frequent sequences, closed sequences and sequence generators such as FreeSpan (Han, Pei, Mortazavi-Asl et al., 2000), PrefixSpan (Pei et al., 2001), CloSpan (Yan et al., 2003), TSP (Tzvetkov et al., 2005), GenMiner (Lo et al., 2008). But these algorithms generate these three different types of patterns separately, which is time consuming. In this chapter, we propose a new algorithm called Closed Sequence and Generator Mining (CSGM) to mine sequential generators and closed sequential patterns together with a single dataset scan. The proposed CSGM algorithm greatly improves the efficiency of the pattern mining. CSGM is an extension of CloSpan and PrefixSpan algorithms, which we will describe in detail before we introduce CSGM in this chapter.

3.1 Definitions

Definition 1 [Itemset and Sequence]

Let $I = \{I_1, I_2, ... I_n\}$ be the set of items. An itemset is a subset of $I$.

A sequence is an ordered list $S=< s_1, ..., s_m >$, where each $s_i$ is a single itemset, called an element of $S$; $m$ is referred to as the length of $S$. The length of a sequence is the number of itemsets in the sequence, i.e. $|S|$. Each number between 1 and $|S|$ is a position of $S$.

In this paper, we only consider simplified sequences whose elements are 1-itemsets, i.e., each element in a sequence contains only one single item in $I$. For a sequence $X$, itemset($X$) is defined as a set of the items that appear in $X$, i.e., itemset($X$) $\subseteq I$. 
Definition 2 [Subsequence and Super-sequence]

A sequence \( \alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle \) is a subsequence of another sequence \( \beta = \langle \beta_1, \beta_2, \ldots, \beta_m \rangle \) denoted as \( \alpha \sqsubseteq \beta \), if and only if \( \exists i_1, i_2, \ldots, i_n \), such that

\[
1 \leq i_1 < i_2 < i_3 < \cdots < i_n \leq m \quad (i_k \text{ are integers}), \quad \text{and} \quad \alpha_1 = \beta_{i_1}, \alpha_2 = \beta_{i_2}, \ldots, \alpha_n = \beta_{i_n}.
\]

We call \( \beta \) is the super-sequence of \( \alpha \) and \( \beta \) contains \( \alpha \), i.e. \( \alpha \sqsubseteq \beta \).

Definition 3 [Sequence Database and Sequence support]

A sequence database, \( D = \{s_1, s_2, \ldots, s_m\} \), is a set of sequences. Each sequence in \( D \), often called a transaction, is associated with an ID. The absolute support of a sequence \( a \) in a sequence database \( D \) is the number of sequences in \( D \) that contain \( a \). The relative support of a sequence is the ratio of its absolute support and the total number of sequences in the dataset.

In this thesis, whenever support is mentioned, we refer to the absolute support, denoted as \( \text{support}(a) \), \( a \) is a sequence.

Definition 4 [Frequent sequences]

Given a minimum support threshold, a sequence is called frequent if its support is greater than or equal to the minimum support.

For traditional association rule mining (i.e., non-sequential association rule mining), association rules are generated from frequent itemsets whose occurrences exceed a predefined threshold. Recently, many efforts have been made to improve association rule mining by exploiting frequent closed itemsets. The definition of closed patterns...
comes from the closure operation of the Galois connection (Ganter & Wille, 1999). Two mappings are defined below, where $X \subseteq I$, $Y \subseteq D$, and $D$ is the collection of transactions:

\[ \tau : 2^I \rightarrow 2^D, \tau (X) = \{ t \in D | \forall i \in X, i \in t \} \]

\[ \gamma : 2^D \rightarrow 2^I, \gamma (Y) = \{ i \in I | \forall t \in Y, i \in t \} \]

In the case of sequential datasets, for a sequence $X = <x_1, ..., x_m>$, itemset($X$) $\subseteq I$, and a transaction $t = <a_1, ..., a_n>$ in a sequential dataset $D$, if all items in $X$ appear in $t$ (not necessarily consecutively) and in that order, as defined above, $X$ is said a subsequence of $t$, denoted as $X \sqsubseteq t$. Accordingly, two mappings are defined below, where $P^I$ is the set of all possible permutations of $I$ from size 2 to $n$:

\[ \tau^s : P^I \rightarrow 2^D, \tau^s (X) = \{ t \in D | X \sqsubseteq t \} \]

\[ \gamma^s : 2^D \rightarrow 2^{P^I}, \gamma^s (Y) = \{ X \in P^I | \forall t \in Y, X \sqsubseteq t, \exists X', X \sqsubseteq X', X' \sqsubseteq t \} \]

$\tau^s (X)$, called the transaction mapping of $X$, is a set of transactions each of which contains $X$ (i.e., $X$ is a subsequence of each transaction in $\tau^s (X)$). $\gamma^s (Y)$, called the sequence mapping of $Y$, is a set of sequences each of which is a subsequence of all the transactions in $Y$. The composition $\gamma^s \circ \tau^s$ is called a sequential closure operator on sequences. $\gamma^s \circ \tau^s (X) = \gamma^s (\tau^s (X))$ is called the set of sequential closures of $X$, each of the closures is a maximum common subsequence among the transactions which contain $X$. A very distinctive feature of the sequential closure is that a sequence can have a set of closures rather than just one single closure as in the case of conventional itemsets. For example, a sequence transaction dataset contains 4 sequences $\{ <abc>, <acb>, <abc>, <bc> \}$, sequence $<bc>$ is the subsequence of sequence $<abc>$,
sequence <ac> is the subsequence of sequence <abc> and <acb>. Unlike the conventional itemsets closure (which treats <abc> and <acb> as same), sequence <ac> has two different closures.

**Definition 5** [Frequent Closed sequences]

Let $X$ be a sequence, $\text{itemset}(X) \subseteq I$. $X$ is a frequent closed sequence iff $X \in \gamma' \circ \tau' (X)$ and $X$ is a frequent sequence whose support is larger than or equal to a pre-defined threshold.

Generally, for a sequence $X$, $\text{itemset}(X) \subseteq I$, $X \subset Y$, $Y \in \gamma' \circ \tau' (X)$. An important and useful property of closure is that the support of $X$ is the same as its closure, i.e., $\text{support}(X) = \text{support}(Y)$, $Y \in \gamma' \circ \tau' (X)$. From the definitions above, we can see that a frequent sequence $s$ is a **frequent closed sequence** if there exists no sequence $s'$ such that (1) $s \sqsubseteq s'$ (2) $\text{support}(s) = \text{support} (s')$. For example, a sequence transaction dataset contains 4 sequences \{<abc>,<acb>,<abc>,<bc>\}, giving a minimum support count = 2, <abc> is the frequent closed sequence.

**Definition 6** [Sequential Generators]

A sequence $g$, $\text{itemset}(g) \subseteq I$, is a generator of a closed sequence $c$, $\text{itemset}(c) \subseteq I$, iff $c \in \gamma' \circ \tau' (g)$ and $g \sqsubseteq c$. $g$ is said to be a generator of the closed sequence $c$ if \( \exists g' \sqsubset g \text{ such that } \exists c' \in \gamma' \circ \tau' (g') \text{ and } c' = c \). For example, a sequence transaction dataset contains 4 sequences \{<abc>,<acb>,<abc>,<bc>\}, sequence <a>,<bc>, has support equal to 3, and there is no sequence that contains <a>,<bc> and shares the same support equal to 3, so <a>,<bc> both are sequential generators.
**Definition 7 [Prefix, project, postfix]**

Given a sequence $A = \langle e_1, e_2, \ldots, e_n \rangle$, a sequence $B = \langle e'_1, e'_2, \ldots, e'_m \rangle$ ($m \leq n$) is called a prefix of $A$ if and only if $e'_i = e_i$ for ($i \leq m$).

Given sequence $A$ and $B$ such that $B$ is a subsequence of $A$, i.e. $B \subseteq A$, a subsequence $A'$ of sequence $A$ is called a projection of $A$ with respect to prefix $B$ if and only if (1) $A'$ has prefix $B$ and (2) there exists no proper super-sequence $A''$ of $A'$ ($A' \subseteq A''$, but $A' \neq A''$) such that $A''$ is a subsequence of $A$ and also has prefix $B$.

Let $A = \langle e_1, e_2, \ldots, e_n \rangle$ and $B = \langle e_1 e_2 \ldots e_m \rangle$ be two sequences, $(m \leq n)$, sequence $C = \langle e_{m+1} \ldots e_n \rangle$ is called the postfix(suffix) of $A$ with respect to prefix $B$. denoted as $C = A/B$.

For example, $\langle a \rangle$, $\langle aa \rangle$, $\langle aab \rangle$ are prefixes of sequence $\langle aabcacdc \rangle$ but $\langle abc \rangle$ and $\langle ac \rangle$ are not. $\langle aabcacdc \rangle$ is the postfix of the same sequence with respect to prefix $\langle a \rangle$, $\langle bcacdc \rangle$ is the postfix with respect to prefix $\langle aa \rangle$, and $\langle cacdc \rangle$ is the postfix with respect to prefix $\langle aab \rangle$.

**Definition 8 [Projected database]**

Let $\alpha$ be a sequential pattern in sequence database $S$. The $\alpha$-projected database, denoted as $S|\alpha$ is the collection of postfixed sequences in $S$ with the prefix $\alpha$.

Definition 7 and Definition 8 are the key concepts for the Algorithm PrefixSpan, which is described in the following section.
3.2 PrefixSpan algorithm

PrefixSpan is an algorithm to mine for frequent sequences or sequential patterns. It is a pattern-growth, depth-first search method. The major idea of PrefixSpan is that, instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences, the projection is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix (Pei et al., 2004).

The PrefixSpan method is given below.

Algorithm PrefixSpan($\alpha$, $l$, $D_{\alpha}$)

Input: $D_{\alpha}$: the $\alpha$-projected database. If $\alpha = \emptyset$ it is the sequence database $S$

$\alpha$: sequential pattern; $l$: the length of $\alpha$;

Output: The complete set of sequential patterns

1: If $D_{\alpha}$ is empty, return, otherwise scan $D_{\alpha}$ once, find the set of frequent items $\beta$ such that $\beta$ can be appended to $\alpha$ to form a sequential pattern. Return if there is no more frequent items.

2: For each frequent item $\beta$, append it to $\alpha$ to form a sequential pattern $\alpha'$, and output $\alpha'$

3: For each $\alpha'$, construct $\alpha'$-projected database $D_{\alpha'}$, and Call PrefixSpan($\alpha'$, $l+1$, $D_{\alpha'}$)
Here we use the sequence database shown in Table 3.1 (TID represents the transaction ID, each TID follows a sequence) as an example to explain the method. The initial call to the method is PrefixSpan (< >, 0, S). Firstly, the algorithm is to find all length-1 sequential patterns. Then it scans S once to find all frequent items in sequences. Each of these items is a length-1 sequential pattern. In Table 3.1, they are <a>:4, <b>:2, <c>:3, <d>:2, <e>:4, where <pattern>:count represents the pattern and its support. The complete set of sequential patterns can be partitioned into several subsets according to the length-1 prefix, i.e., the ones having prefix <a>; the ones having prefix <b>;... the ones having prefix <e>. To do this, we construct a projected postfix database for each length-1 pattern, then recursively call the PrefixSpan algorithm to find subsets of sequential patterns. For the database in Table 3.1, the sequences having prefix <a> are projected with respect to <a> to form the <a>-projected database, which consists of four postfix sequences: <cde>, <bcfe>, <eb> and <ce> as shown in the middle column in Table 3.2. By scanning the <a>-projected database once, all the length-2 sequential patterns having prefix <a> can be found which are: <ab>:2, <ac>:2, <ae>:2, as shown in the right column of Table 3.2. Similarly, these sequences having

<table>
<thead>
<tr>
<th>TID</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a c d e</td>
</tr>
<tr>
<td>20</td>
<td>e a b c f e</td>
</tr>
<tr>
<td>30</td>
<td>a e b</td>
</tr>
<tr>
<td>40</td>
<td>d a c e</td>
</tr>
</tbody>
</table>

Table 3.1 Sequence Database S
prefix <a> can be partitioned into subsets, i.e., those having prefix <ab>; those having prefix <ac>; and those having prefix <ae>. These subsets can be mined by constructing respective projected databases and mining each one recursively until the algorithm terminates (when the remaining sequences are unable to construct a projected database).

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Postfix Database</th>
<th>Sequential Patterns : Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>cde, bcfe, eb, ce</td>
<td>a:4, ab:2, ac:2, ae:2, ace:2</td>
</tr>
<tr>
<td>b</td>
<td>cfe, ace</td>
<td>b:2</td>
</tr>
<tr>
<td>c</td>
<td>de, fe, e</td>
<td>c:3, ce:3</td>
</tr>
<tr>
<td>d</td>
<td>e, ace</td>
<td>d:2, de:2</td>
</tr>
<tr>
<td>e</td>
<td>abcfe, b</td>
<td>e:4, eb:2</td>
</tr>
</tbody>
</table>

Table 3.2 Projected databases

Unlike Apriori-like algorithms, PrefixSpan only grows longer sequential patterns from the shorter frequent ones. It mainly employs the method of database projection to make the database for next pass much smaller, and there is no need for candidate generation in PrefixSpan, only recursively generate the projected database according to their prefix. As compared to an Apriori-like algorithm such as GSP (Srikant & Agrawal, 1996), which generates and tests a substantial number of candidate sequences, PrefixSpan searches a much smaller space.

The major cost of PrefixSpan is the construction of the projected databases. In the worst case, PrefixSpan constructs a projected database for every sequential pattern. If there are a large number of sequential patterns, the cost is non-trivial. A pseudo-projection
technique could reduce the cost of projection substantially when a projected database can be held in main memory. The method goes as follows. When the database can be held in main memory, instead of constructing a physical projection by collecting all the postfixes, one can use pointers referring to the sequences in the database as a pseudo-projection. By examining a set of projected databases, one can observe that postfixes of a sequence often appear repeatedly in recursive, projected databases.

A lexicographic tree data structure has been introduced for sequence enumeration. For a finite database, all the sequences could be arranged in the tree. The root of the tree is labeled with a null sequence, each node in the tree corresponds to a sequence, if a parent node corresponds to a sequence $s$, then its child is an extension of $s$. The left sibling is less than the right sibling in sequence lexicographic order.

![Lexicographical Tree: each node represents a sequential pattern, numbers are their support](image)

Figure 3.1 Lexicographic Tree
PrefixSpan actually performs a depth-first search in the prefix search tree (an example is shown in Figure 3.2) to find all the frequent sequences. For each sequence $s$ and its projected database $D_s$, it performs the sequence extension recursively until all the frequent sequences which have the prefix $s$ are discovered (it terminates the extension when the number of sequences in the $s$-projected database is less than min-support).

Figure 3.1 represents the frequent sequence enumeration for the example in Table 3.1. It provides a visual representation of the partitioning process of PrefixSpan. The first level of the tree stores all the length-1 frequent sequences, the second and third level represents the length-2 and length-3 frequent sequences, etc.
This typical data structure has also been used in the CloSpan mining algorithm, which is introduced below.

### 3.3 CloSpan algorithm

CloSpan uses a similar projection database concept and it is an extension of the PrefixSpan algorithm. Instead of mining all frequent sequences, CloSpan mines only closed sequences. The Clospan algorithm consists of two stages.

For the first stage, the candidate set is generated using the same pruning technique as PrefixSpan (Pei et al., 2001) while incorporating an early termination condition to eliminate sequences which are unlikely closed sequences. For a dataset \( D = \{ s_1, s_2, ..., s_n \} \),

\[
L(D) = \sum_{i=1}^{n} l(s_i)
\]

is the total number of item appearances in \( D \), where \( l(s_i) \) is the length of the sequence \( s_i \). For example, for the sequence database \( M = \{ <a,b,c,d,e>, <a,c,d>, <d,f> \} \), \( L(M) = 10 \). Given two sequences \( S \) and \( S' \), let \( S \) be a subsequence of \( S' \), \( D_s \) represent the projected postfix database with respect to sequence \( S \). Pei et al. (2004) have proved that, if \( L(D_s) = L(D_s') \), then all the descendants of \( S \) and \( S' \) in the prefix search tree should have the same support. So if \( S \) and all its descendants have been discovered, it is unnecessary to search the descendants of \( S' \). The \( L(D_s) = L(D_s') \) is called the early termination condition.

Given a sequence \( s \) and its projected database \( D_s \), if among all the sequences in \( D_s \), an item \( a \) does always occur before an item \( b \), then \( D_{s \triangleleft a \triangleright b} = D_{s \triangleleft b} \cdot (\triangleleft \) is the append operation, \( s \triangleleft a \) represents item \( a \) is appended to \( s \) to form a new sequential pattern.)
Therefore, \( \forall \gamma, s \diamond b \diamond \gamma \) is not closed. We need not search any sequence in the branch of \( s \diamond b \).

Given a subsequence \( s \), and its projected database \( D_s \), if \( \exists a, a \) is a common prefix for all the sequences in \( D_s \), then \( \forall b, s \diamond b \) is closed, \( a \) must be a prefix of \( b \). That means \( \forall b \sqsubseteq a \), we need not search \( s \diamond b \) and its descendants expect the branch of \( s \diamond a \).

According to the early termination condition described above, if a sequence \( S < S' \) where \( < \) indicates that \( S \) is discovered before \( S' \), and \( S \sqsubseteq S' \), the condition of \( L(D_s) = L(D_{s'}) \) is sufficient to stop searching any descendants of \( S' \). For example, if there are sequence \( <f> \) and sequence \( <af> \), if \( L(D_{<f>}) = L(D_{<af>}) \), then \( D_{<f>} = D_{<af>} \).

As shown in Figure 3.3(a), sub trees (descendant branches) of sequence \( <f> \) and \( <af> \) can be merged into one. This means, we do not need to search the sub tree under sequence \( <f> \). By merging nodes in the original prefix search tree, the original prefix search tree becomes a prefix search lattice. This situation is called ‘backward sub-pattern’.
Another situation, which is called ‘backward super-pattern’, is that, sequence $S < S'$, and $S \subseteq S'$, if the condition of $L(D_s) = L(D_{s'})$ holds, it is sufficient to transplant the descendants of $s$ to $s'$. For example, if we know $L(D_{<b>}) = L(D_{<eb>})$, we can conclude that $D_{<eb>}=D_{<eb>}$. There is no need to grow sequence $<eb>$, since all the children of $<b>$ are the same as that of $<eb>$ and vice versa. Therefore, the sequence beginning with $<eb>$ can absorb any sequence beginning with $<b>$. Figure 3.3(b) shows that their sub-trees can be merged into one.

For the second stage, each sequence $s$ needs to be checked to determine whether there exists a super sequence $s'$ sharing the same support as $s$ ($\text{support}(s') = \text{support}(s)$).
CloSpan first finds all the sequences that have the same support as s, than it checks whether there is a super-sequence containing s. This method adopts the fast subsumption checking algorithm (Hashtable structure) introduced by Zaki (Zaki & Hsiao, 1999). The CloSpan algorithm only generates closed sequential patterns and not generators.

The algorithm of CloSpan can be described as follows:

**Algorithm** Closed Mining \((D, \text{min\_sup}, L)\)

**Input**: A database \(D\), and \(\text{min\_sup}\)

**Output**: The complete closed sequence set \(L\)

1: Remove infrequent items and empty sequences from \(D\)
2: \(S^l \leftarrow \) all frequent 1-item sequences
3: For each sequence \(s \in S^l\)
4: do CloSpan \((s, D_s, \text{min\_sup}, L)\)
5: Eliminate non-closed sequences from \(L\);

**Algorithm** CloSpan \((s, D_s, \text{min\_sup}, L)\)

**Input**: A sequence \(s\), a projected database \(D_s\), and \(\text{min\_sup}\)

**Output**: The prefix search lattice \(L\).

1: Check whether a discovered sequence \(s'\) exists such that either \(s \sqsubseteq s'\) or \(s' \sqsubseteq s\), and database size \(L(D_s) = L(D_{s'})\)
2: **If** such super-pattern or sub-pattern exists **then**
3: modify the link in \(L\), **Return**;
4: **else** insert \(s\) into \(L\)
5: Scan \(D_s\) once, find the set of frequent items \(\alpha\) such that \(\alpha\) can be appended to form a sequential pattern \(s \hat{\rightarrow} \alpha\).
6: **IF** no valid \(\alpha\) available **then**
7: **Return**;
8: For each valid $\alpha$ do
   Call CloSpan($s \circ \alpha$, $D_s \circ \alpha$, min_sup, $L$)
9: Return

3.4 CSGM Algorithm

One important task of our research is to generate both closed sequential patterns and sequential generators. Although several closed sequential pattern mining algorithms have been introduced in sequential pattern mining, none of them can generate minimal sequential generators. The only existing sequential generator mining algorithm GenMiner (Lo et al., 2008) does not extract closed sequential patterns. Both of these methods require a separate algorithm and a separate step to find either sequential generators or closed sequential patterns.

In this subsection, we propose a novel method to mine sequential generators and closed sequential patterns together, which is called Closed sequential and sequence generator mining algorithm (CSGM)

The CSGM algorithm is an extension of the closed sequence mining algorithm CloSpan. The purpose of CSGM is to generate both sequential generators and closed sequential patterns together by scanning sequential database only once, with the hope of reducing time cost compared to generating closed sequences and generators separately. The CSGM algorithm uses a similar prefix-search-lattice data structure and the ‘projected database’ concept as for CloSpan. The detailed procedure of this algorithm consists of two major steps: scanning database and mining the entire candidate set of closed
sequential patterns and their corresponding generators; then eliminating all non-closed sequential patterns.

The CSGM algorithm first scans the sequential database once, and finds all frequent length-1 sequences. These length-1 sequences are those patterns containing only one item. Since the generators of length-1 sequences are themselves, we put these sequences and a set of their corresponding generators together as sequence-generator pairs, and we also find the corresponding project databases for these sequences.

![Sequence-generator pair in a prefix-search tree/lattice](image)

**Figure 3.4 Sequence-generator pair in a prefix-search tree/lattice**

For a sequence-generator pair \((s, s_g)\), \(s\) is a potential closed sequence and \(s_g\) is a set of generators of \(s\). Each sequence-generator pair \((s, s_g)\) and its project database \(D_s\) will be used as the inputs to algorithm CGSpan, which is given below. CSGM will call the recursive subprocedure CGSpan until all the candidate closed sequences and their corresponding generators are discovered. Algorithm CSGM and algorithm CGSpan are described below.
Algorithm **CSGM** \((D, \text{min\_sup}, L)\)

**Input:** a database \(D\), and \(\text{min\_sup}\)

**Output:** The complete closed sequence-generator pair set \(L\)

1. Remove infrequent and empty sequences from \(D\)
2. \(S' \leftarrow \text{a set of all sequence-generator pairs of } (s, s_g)\),
   
   \(S\) is a frequent 1-item sequence
3. For each sequence-generator pair \((s, s_g) \in S1\)
4. \quad Do \text{CGSpan}\((s, s_g), D_s, \text{min\_sup}, L\)
5. Eliminate non-closed sequences

Algorithm **CGSpan\((s, s_g), D_s, \text{min\_sup}, L)\)

**Input:** A sequence \(s\), \(s_g\) is the set of minimal generator of \(s\),

\(s\) is a projected database \(D_s\), and \(\text{min\_sup}\)

**Output:** The prefix search lattice \(L\)

1. Check whether a discovered sequence \(s'\) exists such that
   
either \(s \subseteq s'\) or \(s' \subseteq s\), and database size \(\text{Size}(D_s) = \text{Size}(D_{s'})\)
2. IF such super-pattern or sub-pattern exists then
3. \quad Modify the link in \(L\), \textbf{Return};
4. Else insert \(s\) into \(L\)
5. \quad Scan \(D_s\) once, find the set of frequent items \(\alpha\) such that
   
   \(\alpha\) can be appended to \(s\) to form a sequential pattern \(s \diamond \alpha\)
   
   \textbf{IF} \(\text{support}(s) = \text{support}(s \diamond \alpha)\), then \((s \diamond \alpha)_g = s_g\)
   
   \textbf{ELSE} \((s \diamond \alpha)_g = \text{FindGenerator}(s \diamond \alpha, D_s \diamond \alpha)\)
6. IF no valid \(\alpha\) available then \textbf{Return}
7. \quad For each valid \(\alpha\) do
   
   Call \text{CGSpan}\((s \diamond \alpha, (s \diamond \alpha)_g), D_s \diamond \alpha, \text{min\_sup}, L)\)
8. \textbf{Return}
CGSpan is a recursive method which generates the whole prefix search lattice. The input parameter of this method is a sequence-generator pair, its corresponding project database, minimum support and initialised prefix search lattice. The first step of CGSpan is to check if any subsequences and super sequences that share the same support of the input sequence exist on the search lattice. If such a sequence exists, then it will perform a similar procedure as for CloSpan to modify the search lattice (subsequence absorb). Else, the sequence-generator pair is inserted into $L$. Then CGSpan scans the projected database, to find the set of frequent items, these items can be appended to the sequence to form a new sequence pattern. If no such item exists, the method will be stopped. Else, if the new formed pattern has the same support as the original one, the sequential generator set will inherit the original one’s generator set. Otherwise, it will call the FindGenerator method (algorithm 3) to mine the sequential generator sets for the newly formed pattern. For each valid item, CGSpan will recursively call itself, but replace the input sequence and project database with the newly formed pattern sequence. The method will modify the prefix search lattice until all the closed patterns are discovered.

**Algorithm FindGenerator** ($s, D_s$)

**Input:** a sequence $s$, and projected database $D_s$

**Output:** Sequence generator set $G$, initially $G$ is empty

1: **For each** subsequence $s'$ of $s$ which is appeared in $D_s$

2: **IF** support($s'$) = support($s$) and $s'$ is not a super sequence

   Of any sequence in $G$, then insert $s'$ into $G$

3: Eliminate all super sequences of $s'$ from $G$

4: **Return**
The FindGenerator method mines sequence generators for the input sequence, the input parameter of this method is a sequence and its projected database. The method initialises an empty sequence generator set $G$ at first, for each subsequence $s'$ of $s$, the method checks the support of $s'$, if the support of $s'$ is equal to $s$, and $s'$ does not have any super sequence in the generator set $G$, then $s'$ is selected as the sequence generator candidate and inserted into the generator set $G$. Finally the method returns only the sequence’s generators if those sequences do not have any subsequence sharing the same support.

The recursively performed CGSpan algorithm discovers the entire candidate set of closed sequential-generator pairs. At this stage, a simple pruning strategy will be performed to remove all non-closed sequential-generator pairs.

The pruning technique here is simple. It checks all the sequences in the candidate set, if there exists a sequence which is not only a subsequence but also sharing the same support of another sequence, the CSGM algorithm will remove such subsequence and its generator according to the definition of a closed sequential pattern.

Finally, a complete set of closed-sequences and minimal generators in the format of sequence-generator pairs is discovered from the original sequential database.
3.5 Summary

We introduced our CSGM method of generating closed sequences and sequence generators by one scan of the transaction database. The CSGM algorithm is an extension of the CloSpan algorithm. It utilises the similar concept of the projection database and prefix search lattice data structure as CloSpan and PrefixSpan. The CSGM method uses sequential-generator pairs as the minimum dataset elements, and keeps recording closed sequential patterns and their corresponding generators during the mining procedure.
Chapter 4 Non-redundant Sequential Rule Mining

From sequential patterns, we can extract relations between the sets of sequences in the format of sequential association rules. However, the huge number of sequential rules will cause several problems. The first issue is the large quantity of low quality rules that are almost meaningless and will not give people any useful information. The second issue is that the rule generation cost of the full sequential rules is also quite high, even for a sparse dataset. Moreover, sometimes it is even impossible to use for mining full sequential rules for dense datasets. So it is quite reasonable to seek a concise and non-redundant representation for sequential rules.

In this chapter, we extend the non-redundant association rule theory into the sequential mining area. We first give the definitions of the redundant sequential rules, then we introduce the min-max sequential basis for non-redundant rule representation which requires both sequential generators and closed sequential patterns. Finally we provide an algorithm which can efficiently extract non-redundant sequential rules.

4.1 Definitions

Definition 4.1.1 [Confidence of sequential rules]

Let $X, Y$ be two different sequences in a transaction database. The confidence of rule $X \rightarrow Y$ is defined as $\text{support}(X, Y)/\text{support}(X)$. Note that $\text{support}(X, Y)$ represents the number of sequences that contain both $X$ and $Y$ in transaction database.
**Definition 4.1.2 [Redundant Sequential Association Rules]**

Let $X \rightarrow Y$ and $X' \rightarrow Y'$ be two sequential rules with confidence $cf$ and $cf'$, respectively. $X \rightarrow Y$ is said a redundant rule to $X' \rightarrow Y'$ if $X \sqsubseteq X'$, $Y \sqsubseteq Y'$, and $cf \leq cf'$. 

For example, among the four rules: (1) $A \rightarrow BC$ ($cf=2/3$), (2) $AB \rightarrow CE$ ($cf=2/3$), (3) $AB \rightarrow BC$ ($cf=2/3$), (4) $A \rightarrow BCE$ ($cf=1$), according to the definition, rules (1), (2), (3) are considered redundant to rule (4) because rule (1) has a shorter consequent than that of rule (4) and rules (2) and (3) have a larger antecedent and a shorter consequent than rule (4).

For rules (1) and (4), with the same antecedent $A$, rule (4) can bring more information ($BCE$) than that concluded by rule (1) which is ($BC$). Therefore, rule (1) does not contribute new information if we have rule (4), and is thus redundant. Similarly, rules (2) and (3) also do not bring more information than rule (4) but require more information to be fired. Our aim is focused on mining sequential rules with small antecedent but large consequent.

**4.2 Pasquier Min-Max rules**

Pasquier et al introduced the non-redundant association rule called min-max association rules (Pasquier et al., 2005). Rule $r$ is considered a min-max association rule if no other association rule $r'$ has the same support and confidence, and if it has an antecedent that is a subset of the antecedent of $r$ and a consequent that is a superset of the consequent of $r$. Min-max association rules are those rules having minimal antecedent but with maximal consequent.
Pasquier et al’s work also introduced the concept of min-max exact and approximate basis to represent the set of non-redundant rules in non-sequential datasets which are defined as below:

**Definition (Min-Max Exact Basis)**

Let $Closed$ be the set of closed itemsets, and for each closed item $c$, $Gen_c$ be the set of generators of $c$. The Min-Max exact basis is:

$$\text{MinMaxExact} = \{ r: g \rightarrow (c \setminus g) \mid c \in Closed \land g \in Gen_c \land g \neq c \}$$

**Definition (Min-Max Approximate Basis)**

Let $Closed$ be the set of closed itemsets, and for each closed item $c$, $Gen$ be the set of all generators of closed items in $Closed$. The Min-Max approximate basis is:

$$\text{MinMaxApprox} = \{ r: g \rightarrow (c \setminus g) \mid c \in Closed \land g \in Gen \land \text{Closure}(g) \subseteq c \}$$

The Min-max exact basis contains non-redundant rules whose confidence is 1 and the Min-max approximate basis contains non-redundant rules whose confidence is less than 1.

### 4.3 Sequential Min-max rules

As we defined in 4.1, sequential association rules are similar to the non-sequential association rules. Sequential rules share the same characteristics or key concepts as non-sequential rules, such as transitivity, support, and confidence. The only difference is that sequential associations describe sequential relationships between items.
With the redundant rule definition in 4.1, we extend the min-max non-redundant sequential rule theory (Pasquier et al., 2005) from association rule mining to the sequential association rule mining area.

**Definition (Min-max sequential association rules)**

Let $R$ be the set of sequential association rules. A sequential rule $r: s_1 \rightarrow s_2 \in R$ is a min-max sequential rule iff not $\exists r': s_1' \rightarrow s_2' \in R$ with $\text{support}(s') = \text{support}(s)$; $\text{confidence}(s') = \text{confidence}(s)$, $s \sqsubseteq s_1$, and $s_2 ' \sqsubseteq s_2$.

The min-max sequential association rules are the non-redundant sequential rules having minimal antecedent and maximal consequent. $r$ is a min-max sequential rule if no other sequential rule $r'$ has the same support and confidence, and it has an antecedent that is the subsequence of the antecedent of $s$ and a consequent that is a super sequence of the consequent of $r$.

We define two new bases which suit sequential data. They are sequential min-max exact basis and sequential min-max approximate basis. These two bases also formed a concise representation for sequential association rules.

Suppose that $g, s_1, s_2$ are sequences, $g \sqsubseteq s_1 \sqsubseteq s_2$, closure $(g) = \text{closure}(s_1) = \text{closures}(s_2)$, then the two rules $r_1: g \rightarrow (s_2 \backslash g)$ and $r_2: s_1 \rightarrow (s_2 \backslash s_1)$, where $s_2 \backslash s_1$ denotes a subsequence of $s_2$ by removing the sequence $s_1$, will have the same confidence, the antecedent of $r_1$ is shorter than that of $r_2$ and the consequent of $r_1$ is longer than that of $r_2$. If $s_2$ is a closed sequence and $g$ is a generator, i.e., $s_2 = \text{closure}(s_2)$ and $g$ is the minimal sequence which has the same closure as $s_2$, $g \rightarrow (s_2 \backslash g)$ will have the shortest antecedent and longest consequent among the rules $s_1 \rightarrow (s_2 \backslash s_1)$ where $g \sqsubseteq s_1 \sqsubseteq s_2$. Therefore, similar to Min-max exact rules which are generated using a closed itemset and its generator, the sequential
exact rules can be generated using a closed sequence and its sequential generators. Since closure$(g) \subseteq$ closure$(s_2)$, the confidence of $g \rightarrow (s_2 \setminus g)$ is 1. The sequential Min-max exact rules are defined as follows.

**Definition (Sequential Min-Max Exact Basis)** Let $Closed$ be the set of closed sequential patterns, and for each closed sequential pattern $c$, let $Gen_c$ be the set of sequential generators of $c$. The sequential min-max exact basis is:

$$SeqMinMaxExact = \{ r: g \rightarrow (c \setminus g) \mid c \in Closed \land g \in Gen_c \land g \neq c \}$$

Suppose that $g_1, s_1, c_1, g_2, s_2$ and $c_2$ are sequences, $g_1 \subseteq s_1 \subseteq c_1$, $g_2 \subseteq s_2 \subseteq c_2$, $g_1$ and $g_2$ are generators, and $c_1$ and $c_2$ are closed sequences, then the two rules $r_1: g_1 \rightarrow (c_2 \setminus g_1)$ and $r_2: s_1 \rightarrow (c_2 \setminus s_1)$ will have the same confidence, the antecedent of $r_1$ is shorter than that of $r_2$ and the consequent of $r_1$ is longer than that of $r_2$, which means that $r_2$ is redundant to $r_1$ and should not be generated. Given that $g_1$ is a generator and $c_2$ is a closed sequence, it is impossible to find another sequence $g$ with $g \subseteq g_1$, $g$ and $g_1$ have the same support. Therefore, $g_1 \rightarrow (c_2 \setminus g_1)$ is considered a non-redundant rule, and all rules $s_1 \rightarrow (c_2 \setminus s_1)$ with $g_1 \subseteq s_1 \subseteq c_1$ are considered to be redundant. This means that from a generator and a closed sequence, we can generate a non-redundant rule. The confidence of the rule is less than 1 because the support of $g_1$ is larger than that of $c_2$.

The sequential min-max approximate basis is defined below.

**Definition (Sequential Min-Max Approximate Basis)**

Let $Closed$ be the set of closed sequential patterns, $Gen$ be the set of all sequential generators of closed sequential patterns in $Closed$. The sequential min-max approximate basis is:
SeqMinMaxApprox = \{ r: g \to (c\setminus g) \mid c \in \text{Closed} \land g \in \text{Gen} \land \text{Closure}(g) \subseteq c \} 

4.4 Rule generation based on closed sequences and sequence generators

The non-redundant sequential rules are generated from the frequent closed sequences and the sequential generators defined in section 3.4.

According to the definition of sequential min-max rules, we use the sequential generators as antecedents of rules. For each generator, we generate its consequent in the form of removing the same prefix parts from a closed sequence which the generator has, the remaining part of this closed sequence is considered as the correspondent consequent of the rule.

The algorithm of non-redundant rule mining is given as follows.

Algorithm 4.4 NRRuleMining(L,\text{min}_\text{sup}, \text{min}_\text{confidence})

Input: \text{min}_\text{sup}, \text{min}_\text{confidence}, Closed sequence set L including corresponding generators

Output: Rule set R (non-redundant sequential rules)

1: Let \( C = \text{closed sequence set of } L, \ G = \text{generator set of } L \)
2: For each sequence pattern \( g \in G \)
3: For each closed sequence pattern \( c \in L \)
4: \( b := c \setminus g \)
5: \( r : g \to b, \ \text{support}(r) = \text{support}(c), \ \text{and confidence}(r) = \frac{\text{support}(c)}{\text{support}(g)} \)
6: If ( confidence(r) \geq \text{min}_\text{confidence} )
7: Add rule r into rule set R

For example, if sequence \( AB \) is in a generator, while sequence \( ABDF \) is a closed sequence, then \( AB \to DF \) is considered to be a non-redundant sequential rule.
4.5 Summary

In this chapter, we proposed the definition of a non-redundant sequential association rule, and gave a sequential min-max rule basis in order to form a compact representation of a sequential rule set. According to the sequential min-max rule basis, the non-redundant sequential rules can be extracted from sequential generators and closed sequential patterns. We introduced a novel algorithm called CSGM, which can generate closed sequential patterns and their corresponding sequential generators together. We also gave a detailed method for generating non-redundant sequential rules from sequential generators and closed sequential patterns.
Chapter 5 An Application of Sequential rules in Product Online Search

Nowadays, the Internet has become an important source of information of any kind and also a big marketplace for online product searches. Search engines are the common tool for information seekers or shoppers to search for desired information or products. What users need to do is to provide the search engine with details of the features of the product they are looking for. However, usually the user query is short and may not accurately reflect the user’s needs. For information retrieval, it has been widely recognised that query expansion can improve retrieval performance of information retrieval. In this project, we propose to generate the associations between products or products attributes by mining users’ online navigation log data. By using the association rules, we can expand the user’s initial query so that more relevant products can be retrieved and thus the accuracy of the product search can be improved. A user’s query $Q$ is represented by a set of terms $Q = \{q_1, q_2, \ldots q_n\}$ that a user provides to the search engine. In the product search, the terms in the query are attribute values of the product the user is looking for. When searching for products online using a product search engine, a user needs to provide the search engine with a query which consists of the products or product attributes in which the user is interested. The query, generally, is very short and insufficient to present the user’s actual preferences or needs. In this thesis, we propose to expand the user’s query by using association rules among products.
For each \( q \) in \( Q \), \( q \) is a product or a product attribute. If \( q \) is a product, by using the sequential rules which have \( q \) as the antecedent, we can derive more products which are given in the consequent of the rules. These derived products can be added to the query. The rationale of using sequential rules to expand a user query is based on the interpretation of the associations. The association between the antecedent and the consequent of a rule indicates that the consequent could happen if the antecedent happens. Therefore, the derived products, which are in the consequent of a rule, could be interesting to the user if the user has shown interest in the product in the antecedent of the rule.

We developed a car search system that uses the associations generated from user navigation data to expand a user’s query with additional products or product attributes. Figure 5.1 illustrates the structure of the system. The car search system can return products with similar attributes as the initial user input query. With query expansion, all products that have attribute values matching any attribute values of the expanded query are selected as the candidate products. A candidate product will be recommended to the user if the similarity between the candidate product’s attributes and the attributes in the query is high. The system will recommend top-N candidate products based on the similarity.
This system first processes the web logs to generate user navigation sessions, then generates sequential association rules by using the proposed methods described in Chapter 4. Finally those sequential rules are used for query expansion.

5.1 Generating User Navigation Sessions

In this subsection, we describe the procedures and web log mining methods to generate user navigation sessions, which are represented by product sequences viewed by users consecutively.

The web log data from a car sale website contains 3.5 million user click records. It records all users’ access data, i.e. the webpage links the user clicked during their navigation of the car sale website. Each record contains a user’s IP address, requested URL, requested web page type, navigation date and time and other web related information. Based on the URL field of the record, we can retrieve the web page on which the user clicked, and extract useful information, such as the product (car) description which the web page directs to. The raw web log data always contains
erroneous and incomplete records which need to be filtered out at first. In our work, those records whose URLs are not related to products such as image, video links and external website URLs are removed. The cleaned web log data contains 899581 records.

We extracted user navigation sessions by processing the original web logs. Assume that $M = \{e_1, e_2, \ldots\}$ is a set of unique items appearing on the Website, such as cars in the car search website. In this thesis, a user navigation session $S = <e_{i1}, e_{i2}, \ldots, e_{in}>$ is defined as a sequence of items consecutively visited or viewed by a user, $e_{ij} \in M$. Suppose that $M = \{e_1, e_2, e_3, e_4, e_5, e_6\}$, the following table gives an example of a set of sessions.

<table>
<thead>
<tr>
<th>Session ID</th>
<th>User navigation sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$e_2 \ e_1 \ e_5 \ e_2 \ e_6$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$e_3 \ e_5 \ e_2$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$e_2 \ e_1 \ e_2 \ e_4$</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$e_3 \ e_2 \ e_5$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$e_6 \ e_4 \ e_1 \ e_2$</td>
</tr>
</tbody>
</table>

Table 5.1 An example of sessions

For analysing user access behaviour, unique users must be identified. Since users are treated as anonymous by the web server, we simply use client IP to identify users. In
other words, requests from the same IP address will be treated as from the same user. The web log file is an ordered set of web page requests made by users. The requests are stored in the order in which the server receives them. If multiple users are browsing the site concurrently, their requests are intermingled in the log file. The page requests made by one user can be extracted from the log file. Since it is expected that a user may visit a web site more than once, a user navigation session is usually defined as a sequence of pages visited by the same user, such that no two consecutive pages are separated by more than a certain amount of time. Many websites have adopted the value of 30 minutes to define a session (Zhuang, Kou & Zhang, 2005).

For a session containing web page URLs \(<U_1, U_2, \ldots, U_n>\), we convert it into a sequence of products \(<P_1, P_2, \ldots, P_n>\).

For example, the URL: http://www.domainname.com/product/productid=1234567 can be simply replaced by the term “product-1234567” since “product-1234567” is the product which appears on that page.

The processed user navigation sessions contain lists of products (i.e., cars in this case) viewed by users during their online surfing. An example of user sessions for the car search system is shown in Table 5.2.
5.2 Generating Sequential Association Rules from Session Data

We generate non-redundant sequential association rules by using our proposed Sequential min-max algorithm described in Chapter 4. In order for comparison, we also generate the full sequential association rules by using the classic rule mining method which was introduced in Chapter 2.5.

An example of such association rules is given below:

\[
\text{BMW}_525I \quad \text{MINI\_COOPER} \rightarrow \text{TOYOTA\_LANDCRUISER} \quad \text{HSV\_CLUBSPORT}
\]

Support (absolute) = 340, confidence = 60%

The antecedent of this rule consists of two items, BMW_525I and MINI_COOPER; the corresponding consequent consists of two items as well, TOYOTA_LANDCRUISER and HSV_CLUBSPORT. This rule describes that the user who navigated to
BMW_525I and MINI_COOPER may be also interested in TOYOTA_LANDCRUISER and HSV_CLUBSPORT.

5.3 Query Expansion by Sequential Rules

Finally, we expand the user query by applying the association rules. For a user query, we search the set of association rules for any rules that have the query as its antecedent. If such a rule exists, we can then take its consequent and add the items in the consequent into the query.

For example, a customer’s original query (initial input) is “MAZDA_6”, but instead of retrieving the results based on the original query, we first find the rule starting with the query as its antecedent, e.g., “MAZDA_6 → Toyota_Camry, Honda_Civic, BMW_3”. The items in the consequent are used to expand the query so that the expanded query becomes MAZDA_6, Toyota_Camry, Honda_Civic, BMW_3.

For each item in the expanded query, we conduct an individual search to retrieve products that are relevant to that item, then merge the results from all individual searches by using the round robin merge method introduced by Steidinger (2000). Finally, TOP-N recommender products are chosen and returned to the user from the merged set.

For the example above, query items MAZDA_6, Toyota_Camry, Honda_Civic and BMW_3 are put into the car search system individually to get the search results as shown in table 5.3.
Then a round robin merge will be performed on those search results in Table 5.3 in order to get the candidate set.

**Candidate Set** \( C = \{ M_1, T_1, H_1, B_1, M_2, T_2, H_2, B_2, ..., M_i, T_i, H_i, B_i \} \)

Finally we select the TOP-N elements from the candidate set \( C \) as the final search results for the expanded query.

### 5.3 Summary

In this chapter we introduce the concept of initial user query expansion, which can help improve the precision of search results. We also introduce the application structure of our car search system, and the steps involved in this work: Web usage mining and sequential rules generation from the user navigation session data. Both non-redundant

<table>
<thead>
<tr>
<th>Query</th>
<th>Search Results in Car Search System</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAZDA_6</td>
<td>( M_1, M_2, ..., M_i )</td>
</tr>
<tr>
<td>Toyota_Camry</td>
<td>( T_1, T_2, ..., T_i )</td>
</tr>
<tr>
<td>Honda_Civic</td>
<td>( H_1, H_2, ..., H_i )</td>
</tr>
<tr>
<td>BMW_3</td>
<td>( B_1, B_2, ..., B_i )</td>
</tr>
</tbody>
</table>

**Table 5.3 Search results for each query**
sequential rules and full sequential rules are extracted and adapted into the query expansion stage.
Chapter 6 Experiments and Evaluation

We have conducted several experiments to evaluate the effectiveness and efficiency of the proposed non-redundant sequential association rule mining method and its performance in a real world application. The experiments can be divided into four groups according to the evaluation purpose:

- The first group includes the experiments for evaluating the efficiency of the proposed non-redundant sequential association rule mining method.
- The second group is to evaluate the effectiveness of the proposed method to find non-redundant rules.
- The third group is the time-cost comparison of our CSGM algorithm with the CloSpan algorithm.
- The fourth group is the rule quality test for the non-redundant sequential rules generated by our method.

All the experiments are conducted on a Dell OptiPlex 785 desktop with 2.33GHz Intel Core2 CPU, running Windows XP Professional. Our proposed system is developed as a Java application and running on the JVM (Java virtual machine) with a maximum memory allocated at 512MB.

6.1 Data Collections

We use three different datasets for evaluating our non-redundant rule mining method.
The D5C20N10S20 synthetic dataset is generated by IBM synthetic data generator which was also used in AprioriAll (Agrawal & Srikant, 1995) and CloSpan (Yan et al., 2003). The symbol D,C,N,S corresponding to the parameters when generating the data. The D5C20N10S20 dataset contains a total of 3402 sequences.

<table>
<thead>
<tr>
<th>D</th>
<th>Number of sequences (in 1000’s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Average number of events (items) per sequence</td>
</tr>
<tr>
<td>N</td>
<td>The number of different events (in 1000’s)</td>
</tr>
<tr>
<td>S</td>
<td>The average number of events (items) in the maximal sequences</td>
</tr>
</tbody>
</table>

Table 6.1 Parameter description of the synthetic dataset

The MSNBC dataset is a web log dataset which comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. It contains almost one million sequences; however, there are only 17 distinct events (pages). It is considered a very dense sequential dataset, we only collect the first 10,000 sequential records during experiment.
The web log click-through dataset from an online car sale/advertising domain is used in this experiment. The click-through log of this web domain records the user’s navigation history, such as the car or dealer information that the user has viewed. Since the original log data could not be directly used for the experiment, we also did a series of Web usage mining works (described in Chapter 5) to generate a suitable dataset which contains sequential patterns in the format of sessions, with each session consisting of several ordered webpages (or car products) viewed by one user in a specific time period.

The processed click-through log dataset contains a total of 153491 sessions, each session is a sequence composed of several events (the event is represented to be either a user-clicked webpage or a car product).

### 6.2 Hypothesis

We have conducted experiments to verify the following hypotheses:

1) The number of non-redundant sequential rules is much less than that of the full sequential rules.

2) Mining for non-redundant sequential rules using the proposed methods requires less time than mining for full sequential rules.

3) By using non-redundant sequential rules for query expansion in a car search system, the precision and recall of the car search will be higher than using the set of full rules.
4) Our proposed method for mining closed sequential pattern and sequential generators together is faster than mining them separately using different algorithms such as CloSpan and GenMiner.

We have divided our experiments into three groups: efficiency analysis, effectiveness analysis and query expansion effectiveness analysis.

6.3 Evaluation Metrics

The efficiency and effectiveness of our proposed non-redundant sequential rule mining method are evaluated, based on the number of rules generated and the time cost of generating the rules. The widely used measures Recall and Precision are used as the metrics for evaluating the effectiveness of query expansion by using sequential rules.

The recall and precision were defined as below:

\[
\text{Precision} = \frac{|T \cap R|}{|R|}
\]

\[
\text{Recall} = \frac{|T \cap R|}{|T|}
\]

Where \( T \) is the set of relevant products in the testing dataset, \( R \) is the set of recommended products.
6.4 Effectiveness of the Proposed CSGM Rule Mining Method

In this set of experiments, we verified the first hypothesis in chapter 6.2. We evaluated the effectiveness of eliminating redundant rules by comparing the number of rules generated by using our proposed method based on closed patterns and generators and the traditional method based on frequent patterns.

<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Number</th>
<th>NR Rules Number</th>
<th>Reduction Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>756146</td>
<td>24187</td>
<td>96.8%</td>
</tr>
<tr>
<td>22</td>
<td>185697</td>
<td>10460</td>
<td>94.4%</td>
</tr>
<tr>
<td>24</td>
<td>46448</td>
<td>4106</td>
<td>91.2%</td>
</tr>
<tr>
<td>26</td>
<td>10430</td>
<td>1421</td>
<td>86.4%</td>
</tr>
<tr>
<td>28</td>
<td>2493</td>
<td>568</td>
<td>77.2%</td>
</tr>
</tbody>
</table>

Table 6.2 D5C20N10S20 dataset rule numbers generated

Figure 6.1 Rule numbers for Synthetic Dataset D5C20N10S20
<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Number</th>
<th>NR Rules Number</th>
<th>Reduction Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>375891</td>
<td>263212</td>
<td>30%</td>
</tr>
<tr>
<td>8</td>
<td>94842</td>
<td>67705</td>
<td>28.6%</td>
</tr>
<tr>
<td>10</td>
<td>40329</td>
<td>34283</td>
<td>15%</td>
</tr>
<tr>
<td>12</td>
<td>22316</td>
<td>20399</td>
<td>8.6%</td>
</tr>
<tr>
<td>14</td>
<td>14186</td>
<td>13483</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 6.3 MSNBC dataset rule numbers generated

Figure 6.2 Rule numbers for MSNBC data (10699 sessions)
### Table 6.4 Car domain dataset rule numbers generated

<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Number</th>
<th>NR Rules Number</th>
<th>Reduction Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>260</td>
<td>124769</td>
<td>57906</td>
<td>53.6%</td>
</tr>
<tr>
<td>270</td>
<td>104142</td>
<td>43060</td>
<td>58.7%</td>
</tr>
<tr>
<td>280</td>
<td>89170</td>
<td>31866</td>
<td>64.3%</td>
</tr>
<tr>
<td>290</td>
<td>78232</td>
<td>23759</td>
<td>69.3%</td>
</tr>
<tr>
<td>300</td>
<td>28391</td>
<td>18094</td>
<td>36.3%</td>
</tr>
</tbody>
</table>

Figure 6.3 Rule numbers for Car domain dataset

The results from the Figures 6.1-6.3 and Tables 6.2-6.4 showed that the non-redundant rule mining method proposed in this thesis can significantly reduce the number of the discovered rules. All the tables show the reduction ratio, which is defined as below:

\[
\text{reduction ratio} = \frac{\text{Redundant Sequential rule number}}{\text{Full Sequential rule number}}
\]
The average redundancy reduction ratio is considerably high, especially in those circumstances with lower minimum support settings. The only exception is in the MSNBC dataset, the difference was not distinct between the two methods unless using extremely low support settings, this is because the dataset contains a very small number of distinct events (web pages).

6.5 Efficiency of the Proposed CSGM Rule Mining Method

The first group of experiments were used to verify the second hypothesis in chapter 6.2. We compared our CSGM-based non-redundant sequential rule mining method with the PrefixSpan-based (Pei et al., 2001) full sequential rule mining method. The full sequential rules were extracted from full frequent sequential patterns generated by the PrefixSpan algorithm. We recorded the running time and extracted rule numbers for both methods with a variety minimum support settings but the same minimum confidence settings (at 50%). Experiments were conducted on all of the three datasets, with the following results:

<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Generation Time Cost (in seconds)</th>
<th>NR Rules Generation Time Cost (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>34200</td>
<td>599</td>
</tr>
<tr>
<td>22</td>
<td>10010</td>
<td>180</td>
</tr>
<tr>
<td>24</td>
<td>4130</td>
<td>115</td>
</tr>
<tr>
<td>26</td>
<td>2200</td>
<td>69</td>
</tr>
</tbody>
</table>

Table 6.5 Rule generation time cost for dataset D5C20N10S20
Figure 6.4 Rule generation time cost for dataset D5C20N10S20

<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Generation Time Cost (in seconds)</th>
<th>NR Rules Generation Time Cost (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>97443</td>
<td>55432</td>
</tr>
<tr>
<td>8</td>
<td>66847</td>
<td>36854</td>
</tr>
<tr>
<td>10</td>
<td>33729</td>
<td>24138</td>
</tr>
<tr>
<td>12</td>
<td>13989</td>
<td>11537</td>
</tr>
<tr>
<td>14</td>
<td>9795</td>
<td>8725</td>
</tr>
</tbody>
</table>

Table 6.6 Rule generation time cost for dataset MSNBC
Figure 6.5 Rule generation time cost for dataset MSNBC

<table>
<thead>
<tr>
<th>Support</th>
<th>Full Rules Generation</th>
<th>NR Rules Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time Cost (in seconds)</td>
<td>Time Cost (in seconds)</td>
</tr>
<tr>
<td>260</td>
<td>91320</td>
<td>3290</td>
</tr>
<tr>
<td>270</td>
<td>36560</td>
<td>2970</td>
</tr>
<tr>
<td>280</td>
<td>18530</td>
<td>2510</td>
</tr>
<tr>
<td>290</td>
<td>7870</td>
<td>1690</td>
</tr>
<tr>
<td>300</td>
<td>6100</td>
<td>1150</td>
</tr>
</tbody>
</table>

Table 6.7 Car domain dataset rule generation time cost
The results from Figures 6.4-6.6 and Tables 6.5-6.7 showed that the proposed non-redundant rule mining method greatly improved the efficiency of mining for sequential association rules.

6.6 Efficiency Comparison Between CSGM and Other Algorithms

As described in 3.4 and 2.3.2, CSGM has a computational complexity similar to CloSpan. The only difference between these two methods is that in some cases when performing CSGM, when a new appended sequence does not have the same support as the previous subsequence, it will call the find generator function for mining the sequential generators in its own closure, and this step will cost some extra time resources. However, in most cases, the appended sequence could inherit the previous subsequence’s generator directly, without any computation.
Another generator mining algorithm GenMiner (Lo et al., 2008) also uses a similar data structure and pruning method as CloSpan. GenMiner mines all sequential generators but not any closed sequences and has a similar performance to CloSpan. However, in order to mine non-redundant sequential association rules which require both generators and closed sequences, GenMiner should work together with CloSpan, which means that an additional procedure should be added, and the accumulated runtime would be considerably longer, up to twice the length of time required to perform an individual CloSpan or CSGM algorithm.

We compared the time-cost of CSGM with the alternative method which combines CloSpan and GenMiner, since the GenMiner algorithm can be on par with or at times faster than the CloSpan algorithm (Lo et al., 2008). To simplify the experiment, we estimated the time cost of the alternative method to be two times that of performing CloSpan algorithm. We selected the car sale domain dataset in this set of experiments. The same support settings as for the experiment conducted in Section 6.4 are used, this time we evaluate the efficiency of using CSGM to generate all closed sequence-generator pairs and using CloSpan to generate all closed sequences.

From the performance comparison in Figure 6.7, we found that the time-cost of generating all closed sequence-generator pairs is almost the twice as fast as generating all closed sequences and all sequence generators separately.

Therefore, using CSGM for non-redundant sequential rule mining is more efficient than the alternative method combining CloSpan and GenMiner.
6.7 Evaluation the quality of Non-Redundant Sequential Rules

These experiments were conducted to verify the third hypothesis in Chapter 6.2. We used both non-redundant sequential rules and full sequential rules extracted from the car sale domain dataset for query expansion in the recommender system mentioned in Chapter 5. In order to evaluate recommendation improvement of the proposed approach, we compared the recommendation results generated with and without using query expansion. User session records were used as testing data in this experiment. They contained sequences of products viewed by each user. Each session was split into two parts, the first part is used as the query, the second part is used as the testing data to
evaluate the accuracy of the recommended products. If a recommended product is contained in the testing data, this is a hit. Each time, we chose the first several candidate items from one of the sessions, and these candidate items were used as the query. If a rule could be matched by the candidate items, its consequent was used to expand the query. With the expanded query, the TOP-N recommender results were returned. We compared the recommendation results generated with and without using query expansion. We got the following results as shown in Figures 6.8 and 6.9.

Figure 6.8 Recall of expanded inputs and regular inputs
We found that using non-redundant sequential rules for query expansion can improve both recall and precision for the product search.

Another experiment was also carried out, in which we used non-redundant sequential rules or full sequential rules for initial query expansion. From the experiment results in Figure 6.10 and Figure 6.11, we found that using non-redundant sequential rules to expand the query has the same, or even slightly better, recommendation effectiveness than using full frequent sequential rules. Since the cost of generating full frequent sequential rules is far more expensive than producing the non-redundant sequential rules, it is more efficient using non-redundant sequential rules than using the whole rule set.

Figure 6.9 Precision of expanded inputs and regular inputs
Figure 6.10 Recall: Both redundant rules and frequent rules

Figure 6.11 Precision: Both redundant rules and frequent rules
6.8 Summary

In this chapter, we conducted experiments on our proposed non-redundant sequential rule mining method. The experiment results showed our method performed significant redundancy and time-cost reduction as compared to the full sequential rule mining method. During the recommendation effectiveness test, we found that using non-redundant sequential rules of user navigation history for initial query expansion could improve the recommendation results. By using the sequential rules we can find users’ interests more accurately and thus improve the quality of car recommendation compared to the standard matching-based search. Moreover, by only using the non-redundant rules, the same or even better recommendations can be generated as compared to using the whole set of rules.
Chapter 7 Conclusion

In this chapter we summarise our results and draw conclusions from our experiments. We also outline future research that the work in this thesis may lead to.

7.1 Summary

We presented the definition of the redundancy of sequential association rule and proposed the Sequential Min-Max basis for concise representation of non-redundant sequential association rules. According to this basis, we introduced a method for mining non-redundant sequential rules based on sequential generators and closed sequential patterns. Our method guaranteed the generated non-redundant sequential rules have the minimal antecedent and the maximal consequent.

We also proposed our novel sequential pattern mining algorithm CSGM (Closed sequential and generator mining) which generates closed sequential patterns and sequential generators simultaneously with one scan of the dataset. According to the experiments, CSGM has a similar complexity to the closed sequence mining algorithm CloSpan (Yan et al., 2003) and the sequential generator mining method GenMiner (Lo et al., 2008), but it is considerably faster than conducting sequential generator mining and closed sequential pattern mining separately.

Our experiments conducted on three different datasets show that generating non-redundant sequential rules is far more efficient than generating full sequential rules. And the quantity of the non-redundant sequential rules is much smaller than the full ones.
We also utilise both non-redundant rules and full sequential rules for infrequent purchase products (car) recommendation. From the experiment results, we found non-redundant sequential rules have the same recommendation effectiveness as full sequential rules. In another words, after removing a large number of redundancies, the non-redundant sequential rules do not have any information loss.

We conclude that non-redundant sequential rules have a similar quality as full sequential rules but with much more concise representations and a much lower rule generation cost.

### 7.2 Limitations

Our proposed methods have some aspects that need improvement. In some datasets, e.g. the MSNBC dataset, which has few distinct events/items, the size of the generated frequent sequences sets and closed sequences sets are similar at certain minimum support settings. In this case, the non-redundant rule mining method does not have a large reduction ratio compared to the full sequential rules. Our algorithm is not suitable for datasets that are too dense or too sparse (i.e. those datasets either with only a few events or that have too many distinct events).

In addition, although our Sequential Min-Max basis contains rules with minimal antecedent and maximal consequent, we believe that some redundancy still exists in our method.
7.3 Future Work

In our future work, we will consider how to improve our CSGM algorithm for dense and sparse datasets. One possible solution is using TOP-N closed sequential patterns and sequential generators instead of all closed sequential patterns generators generated with predefined minimum support threshold.

Theoretical proof that non-redundant sequential rules do not result in information loss will be conducted in the future. Further redundancy elimination in Sequential Min-Max basis will also be considered.
References


Conference on Computational Intelligence and Multimedia Applications, 2005.


