Sometimes its good to contrast what you like with something else. It makes you appreciate it even more.

– Darby Conley, Get Fuzzy, 2001.

University of Alberta

CONTRASTING SEQUENCE GROUPS BY EMERGING SEQUENCES

by

Kang Deng

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2009
Edmonton, Alberta

Examining Committee

Osmar R. Zaïane, Computing Science

Scott Dick, Electrical and Computer Engineering

Paul Lu, Computing Science

To my parents and my lovely fiancee Sha, I would have never made it this far without your support and encouragement over the years.

Abstract

Group comparison per se is a fundamental task in many scientific endeavours but is also the basis of any classifier. Comparing groups of sequence data is a relevant task. To contrast sequence groups, we define Emerging Sequences (ESs) as subsequences that are frequent in sequences of one group and less frequent in another, and thus distinguishing sequences of different classes.

There are two challenges to distinguish sequence classes by ESs: the extraction of ESs is not trivially efficient and only exact matches of sequences are considered. In our work we address those problems by a suffix tree-based framework and a sliding window matching mechanism. A classification model based on ESs is also proposed.

Evaluating against several other learning algorithms, the experiments on two datasets show that our similar ESs-based classification model outperforms the baseline approaches. With the ESs' high discriminative power, our proposed model achieves satisfactory F-measures on classifying sequences.

Acknowledgements

There are so many people without whom I would have never advanced this far. I would like to give special thanks to the following people:

My deepest appreciation is dedicated to my parents, who always encourage and support me unconditionally. You are the greatest teachers at the early stage of my life. What you provided me is far more than formal education, more importantly, you taught me how to be a man with integrity, principle and responsibility. No word can express my gratitude to you.

I would like to express my undying gratitude to my supervisor Dr. Osmar R. Zaïane. You taught me the first lesson on research. I cannot imagine how much patience you need to turn an amateur to a master. Thanks for your support, guidance and valuable hints that you have afforded me.

I also would like to thank the members of my defense committee, Dr. Paul Lu and Dr. Scott Dick for carefully reviewing my thesis. Your valuable comments on my dissertation are greatly appreciated.

Appreciations to Mojdeh Jalali Heravi and Vahid Jazayeri who share their code and constructive suggestions. The work is much easier with your input. Wish you have a wonderful future.

I am grateful to all my friends here. You made Edmonton fun. Due to the space limitation, I cannot thank you all here. But I need to mention Jie Gao, Ci Song, Jichuan Shi and Yi Yang. It is my pleasure to start my first journey away from home with you.

Finally, my most special gratitude goes to my lovely fiancee Sha. I will never forget the night at Dongli, the elaborate photo album, and the morning calls over two years, always remember forever and ever. Whatever success I achieve, it would have

never been possible without your encouragement and endless love. May dedicating this thesis to you make up a little for the years I cannot be with you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Any science inevitably calls for comparison. Group comparison has always been a scientific endeavour in Statistics [36] and since the early days of Data Mining with works such as discriminant rule discovery [21]. It also is the basis of any classifier in Machine Learning. Contrast sets (CSs) [8] and Emerging patterns (EPs) [15], which contrast between groups of categorical data, can facilitate the classifiers and thus improving classification performance [16] [34]. At a higher level, data distributions can also be used to contrast databases [48].

Comparing groups of sequence data is a relevant task in many applications, such as comparing amino acid sequences of two protein families, distinguishing harmful operations from normal ones in software management, comparing good customers from churning ones in e-business, or contrasting successful and unsuccessful users (or learners) of software or e-learning environments, are typical examples where contrasting sequence groups is crucial. In this thesis, we focus on the sequence data and try to find the differences between sequence groups.

1.1 Background, Challenges and Approach

To contrast groups of sequences, one fundamental question is:"How do several sequence classes differ?" In categorical data, discriminative patterns, typically conjunctions of attribute value pairs, are extracted to represent multi-dimensional data;

a similar strategy, i.e. the extraction of discriminative patterns, can be adopted in sequence data. However, since the order of items is important in sequences, different from categorical data, the discriminative patterns we are interested in should also consider the order of items in the sequence data. For instance, in a protein family, a protein sequence contains the subsequence $\langle FMVCICDDANEAG \rangle$, where every letter represents an amino acid. If the order of amino acids changes, it is possible that this protein is a member of a different protein family. We opt to use subsequences as discriminative patterns to contrast sequence groups.

In categorical data, learning algorithms based on discriminative patterns, such as CSs and EPs, are proved having satisfactory performance on classification, because they take advantage of the discriminative power between high support and low support patterns. We believe that discriminative subsequences are helpful in sequence classification as well. We give two specific examples to highlight this idea.

Example 1.1. Borrowing an example from Ji, Bailey and Dong [25], for instance, the subsequences "having horns", "faces worship", "stones price" and "ornaments price" appear several times in the Book of Revelation, but never in the Book of Genesis. Biblical scholars might be interested in those subsequences and regard them as fingerprints associated with the Book of Revelation.

Example 1.2. When students collaborate to work on programming course projects, their events are recorded. In this particular case, there are three types of events: T represents the operation about a new task, W means communication by collaborative web page, and S is a source code writing event. After comparing the event sequences of different teams, Kay et al. [27] found that the best student groups had interesting sequential patterns, where W and S occurred in alternation, such as $\langle SWS \rangle$, $\langle WSWS \rangle$. And these patterns were often not present in the other groups. Therefore, those event patterns could be suggested as examples of future teamwork

projects.

Since discriminative subsequences are helpful in classification, two main steps are needed to distinguish one sequence group from another:

- First, discriminative subsequences are extracted from two sequence groups respectively.
- Then, we build a classifier by using the discriminative subsequences.

However, there are two main challenges to contrast sequence groups using subsequences. First, the mining of discriminative subsequences is hard. Wang et al. proved that the complexity of finding emerging patterns is MAX SNP-hard [49]. As a more complex pattern, the number of sequential pattern candidates increases exponentially with the size of the vocabulary; therefore, the mining of discriminative subsequences cannot be done in polynomial time. Another problem is during the classification stage: as subsequences become long, an approximative match is desired instead of an exact match when subsequences are compared against discriminative patterns. For instance, in Example 1.2, subsequences $s_1 = \langle SWS \rangle$ and $s_2 = \langle WSWS \rangle$ are discriminative features to distinguish successful teams from unsuccessful ones. If a new team has an event sequence $s_3 = \langle STWS \rangle$, the problem is whether we should classify it as a unsuccessful team. The new sequence is very similar with s_1 and s_2, so it is more likely that this team will be successful.

In this thesis, we first define Emerging Sequences (ESs) as subsequences that are frequent in sequences of one group and less frequent in the sequences of another, and thus distinguishing or contrasting sequences of different classes. Two hypotheses are proposed:

Hypothesis 1.1. We can compare labeled groups of sequences by extracting subsequences that contrast those groups.

Hypothesis 1.2. We can use Emerging Sequences (ESs) to build classification models to classify sequences.

To verify the hypotheses, an ES-based classification framework is proposed. In this framework, ESs are mined more efficiently by a suffix tree-based approach; and a sliding window matching mechanism is also implemented to consider similar subsequences. Our proposed ES-based learning model can be divided into four stages:

1. Preprocess the sequence datasets and extract Emerging Sequence candidates.
2. Select the most discriminative Emerging Sequences.
3. Transform the sequences into tokenized transactional datasets.
4. Train the classifier by Emerging Sequences.

To validate our 4 stage learning model, we perform experiments on two types of datasets, one from software engineering and another from bioinformatics. We compare our approach to several other techniques, such as $ConSGapMiner$ [25], static feature selection, exact sequence matching, or Association Rule-based Classifier, to illustrate the discriminative power of ESs and the performance of our proposed strategies. The experiments show that ESs are much more discriminative than frequent subsequences and our similar ES-based classification model achieves satisfactory F-measures (as high as 98%) on classifying sequences. When our algorithm is trained by using *jumping emerging sequences* (i.e. subsequences present in a group and totally absent or negligible in others), the best performance can be achieved.

1.2 Dissertation Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce some terminologies and define the problem. In Chapter 3, we review the related works on statistics and data mining. Two sequence mining algorithms and two feature selection strategies are discussed in Chapter 4. Chapter 5 presents the transformation from sequence datasets to transactional datasets and the classification based on ESs. We present the prediction performance of our proposed approach in Chapter 6. Finally, we summarize our research in Chapter 7.

A paper based on this thesis was accepted by The Twelfth International Conference on Discovery Science 2009 [14].

Chapter 2

Preliminaries and Problem Definition

In this chapter, we first explain some notations used throughout this dissertation. Then a formal definition of the problem is given.

2.1 Terminology

Let $I = \{i_1, i_2, \ldots, i_k\}$ be a set of all items, or the alphabet, a sequence is an ordered list of items from I. Given a sequence $S = \langle s_1, s_2, \ldots, s_n \rangle$ and a sequence $T = \langle t_1, t_2, \ldots, t_m \rangle$, we say that S is a subsequence of T or T contains S, denoted as $S \sqsubseteq T$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $s_1 = t_{j_1}$, $s_2 = t_{j_2}$, ..., $s_n = t_{j_n}$.

Definition 2.1 (Subsequence Occurrence)**.** *Given a sequence $S = \langle s_1, s_2, \ldots, s_n \rangle$ and a subsequence $S' = \langle s'_1, s'_2, \ldots, s'_m \rangle$ of S, an occurrence of S' is a sequence of indices $\{i_1, i_2, \ldots, i_m\}$, whose items represent the positions of elements in S.*

For instance, if sequence $S = \langle B, C, B, C, A, C \rangle$, and its subsequence $S' = \langle B, C \rangle$. There are 5 occurrences of S' in S: $\{1, 2\}$, $\{1, 4\}$, $\{1, 6\}$, $\{3, 4\}$ and $\{3, 6\}$.

Definition 2.2 (Gap Constraint)**.** *The gap constraint is specified by a positive integer g. In a subsequence occurrence $o_s = \{i_1, i_2, i_3, \ldots, i_m\}$, the difference of any two adjacent indices is $i_{k+1} - i_k$. If $i_{k+1} - i_k \leq g + 1$, we say the occurrence o_s fulfills the g-gap constraint.*

Table 2.1: A sequence dataset example.

sequence ID	sequences	labels
1	$abcac$	pos
2	cab	pos
3	$bcab$	pos
4	$acabd$	neg
5	bda	neg

For example, if $g = 1$, the occurrences of S' $\{1, 2\}$ and $\{3, 4\}$ fulfill the 1-gap constraint (also 0-gap) but $\{1, 4\}$, $\{1, 6\}$ and $\{3, 6\}$ do not.

Definition 2.3 (Count and Support)**.** *Given a sequence dataset $\mathcal{D}_c$, where c is a class label, $\mathcal{D}_c$ consists of a set of sequences. The count of a sequence α, denoted as $count(\alpha, \mathcal{D}_c)$, is the number of sequences in $\mathcal{D}_c$ containing α; while the support $support(\alpha, \mathcal{D}_c)$ is the ratio between its count and the number of sequences in $\mathcal{D}_c$.*

For example, in Table 2.1, if the gap constraint is 0, the count of the sequence $\alpha = \langle a, b\rangle$ in $\mathcal{D}_{pos}$ is 3, while $support(\alpha, \mathcal{D}_{pos}) = \frac{count(\alpha, \mathcal{D}_{pos})}{3} = 100\%$, meaning all pos sequences contain α.

The notion of Emerging Sequences (ESs) was introduced by Zaïane et al. [51], here we generalize this notion and define:

Definition 2.4 (Emerging Sequences)**.** *Given two contrasting sequence classes, Emerging Sequences (ESs) are subsequences that are frequent in sequences of one group and less frequent in the sequences of another, and thus distinguishing or contrasting sequences of different classes.*

Given two contrasting sequence datasets $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$ and a sequence α, if $support(\alpha, \mathcal{D}_{pos}) - support(\alpha, \mathcal{D}_{neg}) > \delta$, where δ is the minimum difference threshold, α is an Emerging Sequence distinguishing $\mathcal{D}_{pos}$ from $\mathcal{D}_{neg}$. For instance, in Table 2.1, subsequence $\alpha = \langle a, b\rangle$ is an emerging sequence distinguishing $\mathcal{D}_{pos}$

from $\mathcal{D}_{neg}$, when the minimum difference $\delta = 40\%$. Because $support(\alpha, \mathcal{D}_{pos}) - support(\alpha, \mathcal{D}_{neg}) = 100\% - 50\% > \delta$.

Definition 2.5 (Edit Distance)**.** *Edit Distance [30] between two sequences is given by the minimum number of operations needed to transform one sequence into the other, where an operation is an insertion, deletion, or substitution of a single item.*

For instance, given $s_1 = \langle kitten \rangle$ and $s_2 = \langle sitting \rangle$, three operations are needed to convert s_1 into s_2 (substitute k with s, substitute e with i, insert g). So the edit distance between them is $distance(s_1, s_2) = 3$. Edit distance is used to measure the similarity between sequences.

2.2 Problem Definition

Given two contrasting sequence groups $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$ as the training sets, the target of our research is to build an unified model based on discriminative subsequences. When classifying new unknown sequences, we expect to distinguish sequences in one group from another. We believe the Emerging Sequences can facilitate the classification and thus improve the prediction accuracy.

Chapter 3

Related Works

This chapter is a review of the background related to data comparison. Section 3.1 reviews the statistical approaches to handle univariate problems. Section 3.2 presents different data mining solutions for pattern and rule based contrasts. In section 3.3, sequence based contrasts are presented.

3.1 Univariate Contrasts

For univariate contrasts, since there is only one feature, emphasis is less on finding the contrast and more on evaluating its power [26]. For instance, Table 3.1 shows the heights and genders of six people, where the height is the feature and the gender is the class label. In statistics, the discriminative power of height feature can be evaluated by different approaches, e.g. t-test, χ^2 test and Wilcoxon rank sum. Based on those methods, the following questions can be answered: are the means of males' heights and females' heights the same? Do the samples for each class come from the same distribution? How well the dataset fits a hypothesis? Therefore, we can examine whether two classes of data are dissimilar and whether the difference is statistically significant.

Besides those statistical works, Odds Ratio and Risk Ratio are often used in comparative studies between two groups of subjects. The odds ratio is the ratio of the odds of an event occurring in one group to the odds of it occurring in an-

Table 3.1: Univariate data example: height vs gender. This table has the heights and genders of 6 persons. Every column represents one person; the rows represent ID, height and gender respectively.

ID	1	2	3	4	5	6
Height (cm)	182	173	165	181	162	177
Gender	male	male	female	male	female	female

other group. Take the table 3.2 as the example, the odds of disease for males is $322/1173 = 0.275$; while the odds of disease for females is $135/1325 = 0.102$. Therefore, the odds ratio for males to have disease over that of females is $\frac{322/1173}{135/1325} = 2.694$. The risk ratio is a ratio of the probability of the event occurring in the exposed group versus a non-exposed group. The risk ratio for males to have disease over that of females is $\frac{322/1495}{135/1460} = 2.329$.

Table 3.2: Binary feature table: the relation between gender and health

	disease	healthy	total
male	322	1173	1495
female	135	1325	1460

Odds ratio and risk ratio are widely used in univariate contrasts. They can also be used for quality evaluation of multivariate contrasts.

3.2 Pattern and Rule based Contrasts

Compared to univariate contrasts, data miners focus on the multivariate problems. Patterns and rules, typically conjunctions of attribute value pairs, are extracted from datasets and used to distinguish classes.

As the pervasive application of relational database system, researchers started to learn knowledge in the form of rules from relational databases. Characteristic rules and discriminant rules was introduced by Cai et al. [11]. A characteristic rule is

an assertion which characterizes a concept satisfied by all or most of the examples in the class undergoing examination (called the target class), while a discriminant rule is an assertion which discriminates the target class from other classes (called contrasting classes). The attribute-oriented induction method [11] was also applied to extract these two rules from relational databases. Han et al. designed a data mining query language (DMQL) [20], which can find rules to discriminate the target class from contrasting classes.

Based on the association rule mining [1], Liu et al. integrated classification and association rule mining and proposed a new algorithm: Association Rule-based Classifier (ARC) [38] [4]. An association rule is of the form:

$$X \rightarrow C(s, c)$$

where X is a pattern, C is the class label, s represents the support of the pattern in the dataset, and c is the confidence. ARC extracts classification rules with high confidences and supports, and classifies new instances based on those rules. After compared with C4.5 [44], ARC surpasses C4.5 on prediction accuracy. This work demonstrated that the rule based contrasts can improve the classification accuracy in multivariate problems. In our research, ARC is used as a classifier to validate the discriminative power of Emerging Sequences.

To define the discriminative patterns in categorical data, the idea of Contrast Sets (CSs) [8] was introduced by Bay et al. Contrast sets are conjunctions of attributes and values, whose supports differ meaningfully across groups. Suppose cs is a contrast set, and G represents a group; the support of a contrast set $support(cs, G)$ with respect to a group G is the percentage of examples in G. cs is a valid contrast set only if it is significant and large. Significant means $\exists i, j$ such that $support(cs, G_i) \neq support(cs, G_j)$; and a large contrast set cs is one such $max_{ij}|support(cs, G_i) - support(cs, G_j)| > \delta$, where δ is the minimum deviation. Bay et al. also proposed an algorithm STUCCO (Search and Testing for

Understandable Consistent Contrasts) based on Max-Miner [9] to extract contrast sets. By using Chi square to measure the statistical significance of contrast patterns and other pruning techniques, STUCCO can find contrast sets that discriminate one class from others efficiently. CIGAR [23], a variation of STUCCO, is developed by Hilderman et al. While STUCCO pruning the search space by controlling the Type I error, CIGAR seeks to control Type II error. Another algorithm proposed by Satsangi et al. [46] also concentrates on contrast set mining: their association rule based approach finds all contrast sets that are found by STUCCO and other interesting CSs that STUCCO fails to discover.

At the same time, Emerging Patterns (EPs) [15], which are similar with contrast sets are introduced by Dong et al. Different from contrast sets, which are extracted based on the support differences, EPs are patterns fulfilling the support ratio threshold. Thus, $growthRate$ was introduced to represent the ratio of the supports:

$$growthRate = \frac{support(pattern, G_i)}{support(pattern, G_j)}$$

the $pattern$ is an $\rho - emerging\ pattern$ when its $growthRate \geq \rho$. When $growthRate = \infty$, the pattern is called jump emerging pattern, which means this pattern appears in one class but never occurs in the other class. Given a minimum ratio threshold ρ, the extraction of emerging patterns is not efficient for two reasons. One is the threshold constraint is neither monotonic nor anti-monotonic, thus, the Apriori property no longer holds for emerging patterns and the searching space cannot be pruned. Another difficulty results from the fact that emerging patterns with low supports (as low as 1%) can give very useful new insights [15]. However, as the decrease of support ratio threshold, the number of candidates increase exponentially and the mining of EPs is inefficient. Wang et al. [49] proved the complexity of finding emerging patterns is MAX SNP-hard [42], which means that polynomial time approximation schemes to extract EPs do not exist for the problem unless P=NP.

Dong et al. also developed a border differential algorithm [15]: as the large

collections of itemsets are interval-closed, it can be efficiently represented by the maximal and minimum borders discovered by Max-Miner [9]. Based on the border differential algorithm [15], Mao et al. proposed some methods for efficiently discovering highly differentiative gene groups (HDGG) [41]. The discovering of HDGG is challengeable because of the high dimensions of microarrays. With the implementation of the border differential algorithm [15], 75 genes can be handled by current generation PCs. This algorithm surpasses the top-k method [32] on the discovering of HDGG.

Bailey et al. concentrated on the jumping emerging patterns (JEP) [7]. By building a similar tree with Frequent Pattern Tree (FP-Tree) [22], the mining performance is typically around 5 times faster than earlier approaches.

Based on the contrast sets and emerging patterns, some classification algorithms were developed. The first classifier specially designed for EPs was proposed by Dong et al [16], and it is called Classification by Aggregating Emerging Patterns (CAEP). By mining emerging patterns and calculating the aggregate scores, the instance can be classified according to the largest normalized score. Experiments show that CAEP has consistent good accuracy on predication, and it almost always outperforms C4.5 [44] and ARC [38]. Unlike CAEP [16], DeEPs (Decision-making by Emerging Patterns) [31] developed by Li et al. is implemented by the lazy learning strategy, i.e. the generalization beyond the training data is delayed until a query is made to the system. The motivation of DeEPs is that, the eager-learning classifiers, which try to generalize the training data before receiving queries, may ignore some emerging patterns relevant to a specific testing instance. On the contrary, for a testing instance T, DeEPs projects each training instance to contain only items in T, then discovers all emerging patterns from projected data. Since no relevant emerging patterns are missing for testing instances, DeEPs is superior to other classifiers. Another application of emerging patterns is the Weighted Decision Trees

(WDTs) [2] proposed by Alhammady et al. To enhance the noise tolerance, the weights are added to the decision tree. Emerging patterns can help to construct the WDTs, which has better performance than C4.5 [44].

Among classifiers using contrast sets and emerging patterns, most classification algorithms are based on EPs. Nevertheless, compound-risk factors Naïve Bayes (crf-NB) [34] proposed by Li et al. applied both Emerging Patterns and Contrast Sets. Besides EPs and CSs, the features with high Odds Risk values is also an important factor of the classifier. crf-NB has three thresholds for emerging patterns, contrast sets and odds risk patterns respectively; the pattern that meets any condition is a strong pattern. Unlike Naïve Bayes [28], crf-NB only takes strong patterns into account, and ignores the independence of attributes. The experiments show that, crf-NB surpasses Naïve Bayes greatly on classification accuracy. Inspired by crf-NB [34], we propose the Emerging Sequences Naïve Bayes classifier (es-NB), which only considers ESs. More information about es-NB is provided in Chapter 5.

Some other classification algorithms, i.e. CMAR [35], PCL [33] and CPAR [50], also adopt pattern and rule based contrast. Those are not discussed here specifically.

3.3 Sequence based Contrasts

Unlike categorical data, there is very little work on contrasting groups of sequences. Sequence data are very different from relational data, because the order of items is important. As a classic topic, frequent subsequences mining is well developed, and many practical algorithms had been proposed, i.e. GSP [47], SPADE [52], PrefixSpan [43], and SPAM [6]. Those algorithms can be used to extract discriminative subsequences. Given two sequence groups, to extract discriminative subsequences, a simplistic approach is to mine frequent subsequences in each group respectively, then remove those that are also frequent in the other group. However, there are two problems by extracting contrasting subsequences with those algorithms. One

challenge is the low efficiency: the support thresholds in mining distinguishing patterns need to be lower than those used for mining frequent patterns [15], which means the minimum supports offer very weak pruning power on the large search spaces [26]. Another problem of previous algorithms is that items do not have to be appearing closely with each other in the original sequence, while the gaps between items are significant in comparing sequences. An example is the relations between words in a long sentence. A verb probably serves as the predicate of a subject if they are close to each other. Therefore, gap constraints need to be considered in subsequence mining.

The gap requirement between items in the original sequence is applied in alignment of genome or protein sequences [19], e.g. BLAST (Basic Local Alignment Search Tool) [3]. When computing the optimal alignment of two sequences, scores are deducted for insertions or deletions. Under this regime, the sequence with lower gaps is likely to be chosen. To mine contrasting subsequences with gap constraint, an algorithm $ConSGapMiner$ was proposed by Ji et al. [25]. $ConSGapMiner$ introduces maximum-gap constraint to the subsequence mining. There are two main steps of $ConSGapMiner$ algorithm: first, a DFS (Depth First Search) tree is built to enumerate all possible subsequences candidates; then, for each candidate, the bitset operations are applied to prune nodes which cannot fulfill the g-gap constraint, i.e. any two adjacent items in the subsequence have a larger distance than $g + 1$ in the original sequence.

Suppose the vocabulary is $\{A, B, C\}$, Figure 3.1 is the Depth First Search tree for candidate generation. In this DFS tree, each node represents a sequence, and the numbers in the node are the supports of the sequence in the target class and the contrasting class. It starts with an empty sequence; each node in the tree is extended by adding a single item from the alphabet, e.g. AA's children are AAA, AAB, and AAC. Once a new node is created, the supports in both classes are calculated

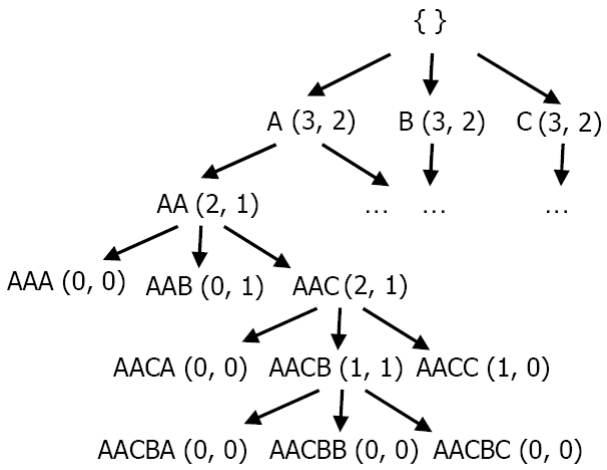


Figure 3.1: DFS tree for candidates generation. [25]

by bitset operations. Bitset operation is a smart algorithm to calculate g-gap constraint supports, however, the number of candidates increases exponentially with the growth of the tree. As a result, $ConSGapMiner$ is not an efficient algorithm because of the nature of its data structure.

Another related work is emerging substrings proposed by Chan et al. [12]. Emerging substrings, which are motivated by Emerging Patterns, occur more frequently in one class rather than in other classes. Based on the Generalized Suffix Tree (GST) [19], emerging substrings can be extracted in linear time. Compared with $ConSGapMiner$ [25], this algorithm can only mine substrings in which items have to be appearing immediately next to each other in the original sequence, i.e. only 0-gap constraint subsequences are extracted. The problem of this algorithm is that, subsequences with larger gaps are ignored, while our approach solves this problem in the classification stage by the similar matching mechanism.

Generally, the contrast sequences are always based on the frequencies of subsequences or substrings. Lin et al. implemented the Contrast Sets on time series data [37]. Instead of calculating the supports of subsequences, they compared subsequences by Euclidean Distance. Given two time series T and S, the subsequence

in T, which distinguishes T from S the most, has the largest distance to its closest match in S. Lin et al. also proposed an algorithm called Group SAX, which can make the extraction more efficient by reducing the distance calculation times. This strategy cannot be implemented on sequence data directly, because the frequencies of subsequences are ignored.

A recent work, which is similar to our framework, concentrates on the software behaviours application [39]. They mine iterative patterns from software behaviours, and distinguish events that generate failures by an SVM classifier. One primary difference between this framework and ours is that, in our algorithm, items in subsequences have to be close to each other in the original sequence, while it is not important in the software behaviours application. Indeed the gap they use is undetermined and arbitrarily large.

Chapter 4

Sequence Mining and Feature Selection

The data on which we focus are sequence data. To distinguish one group of sequence data from another, representative features must be extracted. However, in the domain of sequence data, the number of useful features is exponential in the size of the data. To refine numerous features, Lesh et al. demonstrated that subsequences can reduce the size of features, meanwhile improve the accuracy of classifiers [29]. In their approach, however, they did not consider any gap constraint and apply exact matches only. In this chapter, discriminative subsequences are extracted as emerging sequences, and the classification based on ESs is described in Chapter 5.

In Section 4.1, we explain how we first preprocess the datasets and extract the emerging sequence candidates, two strategies based on preprocessing with and without gap constraint are adopted in this stage. In Section 4.2, both static and dynamic feature selections are proposed to extract the most discriminative and representative subsequences as emerging sequences.

4.1 ES Candidates Mining

To find the representative subsequences, the following domain-and-classifier-independent heuristics are useful for selecting sequences to serve as features [29]:

- Features should be relatively frequent at least in one class.
- Features should be distinctive of at least one class.

The reason for the first heuristic is that the features have to be common in one group; the rare features do not help in the classification. The intuition behind the second heuristic is, if a feature is frequent in both original classes, it cannot improve the accuracy of classifiers. In other words, the emerging sequence candidates should be common in one group, and exceptional in another.

Let $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$ to be two classes of sequences, the supports of a ES candidate α in both classes, denoted as $support(\alpha, \mathcal{D}_{pos})$ and $support(\alpha, \mathcal{D}_{neg})$, need to meet the following conditions:

$$support(\alpha, \mathcal{D}_{pos}) > \theta \tag{4.1}$$

$$support(\alpha, \mathcal{D}_{neg}) \leq \theta \tag{4.2}$$

where θ is the minimum support threshold. Therefore, any subsequence fulfilling the conditions is discriminative.

As sequence mining topic is well developed, many existing algorithms, such as GSP [47], SPADE [52], PrefixSpan [43], and SPAM [6] can extract frequent subsequences easily. However, there are two problems with ESs extraction using those algorithms. One challenge is the low efficiency: the support thresholds in mining distinguishing patterns need to be lower than those used for mining frequent patterns [15], which means the minimum support offers very weak pruning power on the large search spaces [26]. Another problem of those algorithms is that, items do not have to be appearing closely with each other in the original sequence, while the gaps between items are significant in comparing sequences, e.g. in the UNIX command sequence, only if a “ls” is close to a “lpr”, it is possible that they have a

relation because they may operate on the same file. Therefore, the gap constraint is needed when preprocessing, i.e. items in subsequences cannot be too far apart in the original sequences. Two strategies are adopted in ES candidates mining to solve these problems: one is based on $ConSGapMiner$ [25], which can control the gap constraint; another is a suffix tree-based framework, which can extract ES candidates in linear time.

4.1.1 ES Candidates with Gap Constraint

When extracting ES candidates with gap contraint, the base algorithm we used is $ConSGapMiner$ [25]. It assumes the subsequence exists in both groups and uses frequency constraints to prune the search space while counting supports of subsequence occurrences in groups. Other than meeting the requirement above, this algorithm also considers the gap constraint g, i.e. in an subsequence occurrence, if the indices difference of any two adjacent items is lower than or equal to $g + 1$, we say the subsequence fulfills the g-gap constraint.

Given two classes of sequences $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, the original $ConSGapMiner$ extracts subsequences if and only if the following conditions are true:

1. **Frequency Condition:** $support(\alpha, \mathcal{D}_{pos}) > \delta$
2. **Infrequency Condition:** $support(\alpha, \mathcal{D}_{neg}) \leq \beta$
3. **Minimality Condition:** There is no subsequence of α satisfying 1 and 2.

In our application, δ and β are the same; and sequences do not need to fulfill the minimality condition, because subsequences with the highest supports can facilitate the classification while the independence of them is not important. The revised algorithm is presented in Algorithm 1.

The preprocessing algorithm is a recursive algorithm, which builds a DFS tree as illustrated in Figure 3.1. For a sequence s fulfilling the discriminative conditions,

Input: sequence datasets $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, the current sequence s, maximum gap g, support threshold θ, vocabulary I, the global variable R containing all emerging sequence candidate

```
1  foreach i ∈ I do
2      ns ← s + i;
3      support(ns, D_pos) ← support_count(ns, g, pos);
4      support(ns, D_neg) ← support_count(ns, g, neg);
5      if support(ns, D_pos) ≥ θ then
6          if support(ns, D_neg) < θ then
7              R ← R ∪ ns ;
8          end
9          preprocess(ns, g, θ);
10     end
11 end
```

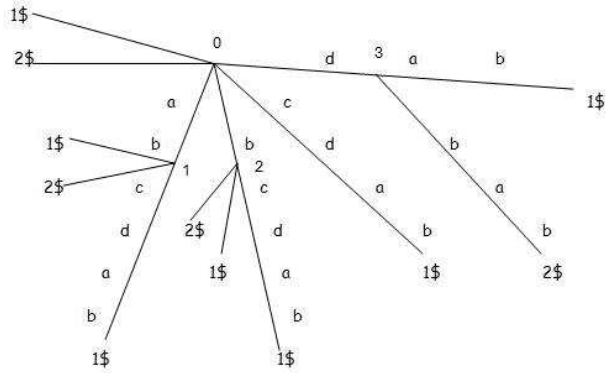
Algorithm 1: Preprocessing with Gap Constraint: $preprocess(s, g, \theta)$. It preprocesses the sequence dataset $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, and saves all ES candidates in the result R.

one item is chosen from the vocabulary at a time (line 1). A new sequence ns is created by appending the item at the end of the original sequence s (line 2). In line 3-4, we calculate the supports of the new sequence in both $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, and test if it can meet the g-gap constraint. If the new sequence ns fulfills the discriminative conditions, we add ns to R; as long as ns is frequent enough in $\mathcal{D}_{pos}$, we continue to search, by creating new sequences from the current one (Line 5-10). Finally, all sequences in R are frequent in $\mathcal{D}_{pos}$ and infrequent in $\mathcal{D}_{neg}$.

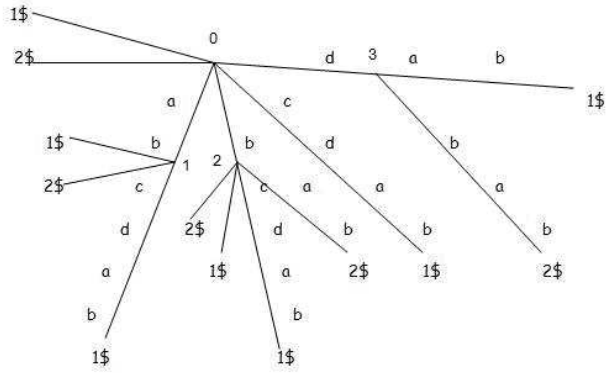
This algorithm uses bitset operations to validate g-gap constraint, and prunes the search space by checking the supports in both sequence groups. However, as the tree grows, the number of candidates increases exponentially. Compared with previous sequence mining algorithms, the revised $ConSGapMiner$ can control the gap constraint, nevertheless, it is still not efficient due to the nature of its data structure (DFS).

4.1.2 ES Candidates with 0-Gap Constraint

Another preprocessing strategy is to mine subsequences without gap, i.e. the items in the subsequence are adjacent in the original sequence (0-gap constraint). The revised $ConSGapMiner$ can still be used to extract subsequences fulfilling 0-gap constraint. However, without considering the gaps, a faster algorithm based on generalized suffix tree (GST) is implemented, which can extract subsequences in linear time [19].



(a) The GST before bab is inserted.



(b) The GST after bab is inserted.

Figure 4.1: An example of the Generalized Suffix Tree (GST). This GST is built by two sequences $s_1 = \langle abcdab \rangle$ and $s_2 = \langle dbab \rangle$. It has 4 internal nodes, and 0 is the root. Figure 4.1(a) is the GST before $\langle bab \rangle$ is inserted; and Figure 4.1(b) is the complete GST.

This tree is called generalized suffix tree (GST) because it is built by the suffixes of sequences. Figure 4.1 is a GST containing sequences $s_1 = \langle abcdab \rangle$ and $s_2 = \langle dbab \rangle$. Every edge starting from the root has a suffix attached with it, and the indices at the leaves indicate the original sequence ID of the suffix, e.g. the edge $\langle cdab\ 1\$ \rangle$ is a suffix of the sequence s_1 because it ends with $1\$$. Given a sequence $s_1 = \langle abcdab \rangle$, its first suffix is extracted starting from the first index, so $suf_1 = \langle abcdab \rangle$; and the second suffix $suf_2 = \langle bcdab \rangle$ starts from the second index. Every suffix is added to the tree after extracted, and the strategy is as follows: suffixes with the same prefixes share the edge. Starting from the root, if an edge has the same prefix with the current suffix, no extra edge is created; once the suffix has a different item, a new internal node and an new edge are created. For example, Figure 4.1(a) is the GST before the suffix $\langle bab \rangle$ is inserted. When inserting $\langle bab \rangle$, an edge from the root starts with b, so no edge is created; however, the next item of this edge is c, while that of the suffix is a, so a new edge is generated from node 2, and the new edge has items $\langle ab\ 2\$ \rangle$ (Figure 4.1(b)).

By Ukkonen's Algorithm [19], the GST can be built in linear time, i.e. if the sum length of sequences is n, the GST is built in $O(n)$ time complexity. The support counting of subsequences is also fast. For instance, to find the count of subsequences ab, we start from the root, and find node 2; the edge $(0, 2)$ has the subsequence $\langle ab \rangle$ attached. So the count of $\langle ab \rangle$ is the number of leaves from node 2, and $count(ab) = 3$. One problem of the current algorithm is that, GST counts the total occurrences of subsequences, while some subsequences appear more than once in one transaction, e.g. $count(ab) = 3$ when there are only 2 transactions. If the count is higher than the transaction number, the support of the subsequence becomes higher than 100%, which does not make sense in classification. To solve this problem, we store the leaves IDs in internal nodes and remove the redundant IDs when counting supports. So the support of a subsequences is the

number of IDs. For example, when building the tree, the transaction IDs in node 1 are { 1$ 2$ 1$}; only { 1$ 2$} are kept, so the count of ab is 2 and its support is $support(ab) = \frac{2}{2} = 100\%$.

In conclusion, the advantage of the suffix tree-based framework is that ES candidate mining can be done in linear time. However, only subsequences fulfilling the 0-gap constraint are mined, i.e. items have to be appearing immediately next to each other in the original sequence. Information may be lost when mining ES candidates by GST. To handle the low gap constraint subsequences, we propose a sliding window matching mechanism for the distance metric between sequences; more information is provided in Section 5.1.

4.2 Feature Selection

After preprocessing, numerous ES candidates are extracted. In this section, we refine the result and select the most discriminative subsequences as ESs. Existing studies on categorical data demonstrate that, discriminative patterns can improve the prediction accuracy [16] [31]. In our research, the sequence datasets are transformed to transactional datasets according to emerging sequences (See Figure 4.2); and the transformation is described in Section 5.1. To build informative transactional datasets, we have to find the most Emerging Sequences, and thus contrasting sequence groups.

To evaluate the discriminative power of subsequences, a similar mechanism with Contrast Sets [8] is applied. Given two sequence groups $\mathcal{D}_{pos}$ and $\mathcal{D}_{neg}$, the ES candidates are ranked by the supports difference:

$$sup\_diff = support(\alpha, \mathcal{D}_{pos}) - support(\alpha, \mathcal{D}_{neg})$$

The larger sup_diff is, the more discriminative the subsequence. After ES candidates are sorted, one question is: how many subsequences should be chosen as

emerging sequences? In this section, we adopt two approaches to select ESs: one fixes the amount of ESs for each sequence group, while another method expresses each original sequence by the same number of ESs.

4.2.1 Static Feature Selection

One straightforward approach to select features is to fix the number of ESs for each sequence group, i.e. after sorting ES candidates of each sequence group, we choose the top n of them as emerging sequences. Since the number of ESs for each sequence group is fixed, we name this method *static feature selection*.

However, the original sequences are not frequency-balanced, i.e. some sequences contain many highly frequent subsequences, while others do not. If only a small amount of ESs are selected, the problem is the existence of silent transactions, i.e. some original sequences do not contain any emerging sequence, and the transformed transactions are empty. In Table 4.1, if the number of ESs (fulfilling 0-gap constraint) that contrast the positive class to the negative class is set to 1, the subsequence ca is selected because its supports in two classes are 75% and 0% respectively. Meanwhile, the sequence 4 cannot be expressed by ESs and is called silent. Therefore, we cannot expect good performances if many of the corresponding transactions are empty.

Table 4.1: A frequency-imbalanced sequence dataset.

sequence ID	sequences	labels
1	$abcad$	pos
2	aca	pos
3	$cabd$	pos
4	e	pos
5	bda	neg
6	$bdcda$	neg

To eliminate the silent transactions, a direct solution is to increase the num-

ber of ESs. When more candidates are selected, the transformed transactions are less likely to be silent. In the dataset illustrated in Table 4.1, if the number of ESs that contrast pos to neg is set to 4, the subsequences ca ($sup\_diff = 75\%$), ab ($sup\_diff = 50\%$), c ($sup\_diff = 25\%$) and e ($sup\_diff = 25\%$) are selected as ES, and the sequence 4 can be expressed. However, even if a large number of candidates are chosen as ESs, some of them are not distinct enough. The transactional datasets are less informative, if many less discriminative subsequences are chosen as ESs. For instance, the sequence 1 in Table 4.1 is already expressed by ca ($sup\_diff = 75\%$) when the number of ESs is 1; if the number of ESs is set to 4, the subsequence c ($sup\_diff = 25\%$) is also extracted. Since c is less discriminative and contained by ca, the increase of ESs number does not keep the most valuable information for the sequence 1. In other words, more ESs do not ensure better prediction accuracy.

In conclusion, static feature selection suffers from silent transactions when only a small amount of ESs are selected; if many candidates are chosen as ESs, however, the transformed transactional datasets are less representative, and thus hard to be distinguished. It is difficult to choose the number of ESs to optimize the classification performance if a static feature selection strategy is adopted.

4.2.2 Dynamic Feature Selection

The number of emerging sequences should be enough so most original sequences can be covered; on the other hand, our classification model suffers from outliers if low frequent subsequences are selected. To avoid this contradiction, a dynamic feature selection strategy is adopted [24]. Each sequence in the input dataset is to be expressed by the selected features. However, for any sequence, only the top-m subsequences, based on sup_diff, are kept. It guarantees that each sequence can be represented by at least m ESs (the high-ranked ones) and the database does not

become too large due to the possible sheer number of candidate subsequences.

```
   Input: the sequence dataset 𝒟, the sorted set of Emerging Sequence
          candidates ES_c, the minimum subsequence number m
   Output: The set of Emerging Sequences ES
 1 foreach sequence ∈ 𝒟 do
 2 |   count ← 0;
 3 |   foreach candidate ∈ ES_c do
 4 |   |   if candidate ⊑ sequence then
 5 |   |   |   count ← count + 1;
 6 |   |   |   mark the candidate ;
 7 |   |   end
 8 |   |   if count = m then
 9 |   |   |   break;
10 |   |   end
11 |   end
12 end
13 ES ← all marked subsequences in ES_c;
```

Algorithm 2: Dynamic Feature Selection.

The dynamic feature selection algorithm is presented in Algorithm 2. For each sequence in the dataset $\mathcal{D}$, we check inclusion of any subsequence (i.e. candidate feature) sorted by sup_diff. We mark candidate subsequences that are included in the input sequences (Line 6) up to m per sequence (Line 8). Then, we output the union of all marked subsequences (Line 13).

Figure 4.2 presents the 4 stages of our proposed learning model. The minimum support θ in Stage 1 is set to 50% as an example. The numbers in the brackets after ESs are their supports in the positive and the negative class respectively. In Stage 1 and 2, we preprocess the original dataset and emerging sequences are extracted. In Stage 3, the sequence dataset is transformed to a transactional dataset, in which the transactions are simple sets of tokens representing ESs. Finally, we build a classifier based on the transactional dataset. The specific strategies in Stage 3 and 4 are described in Chapter 5.

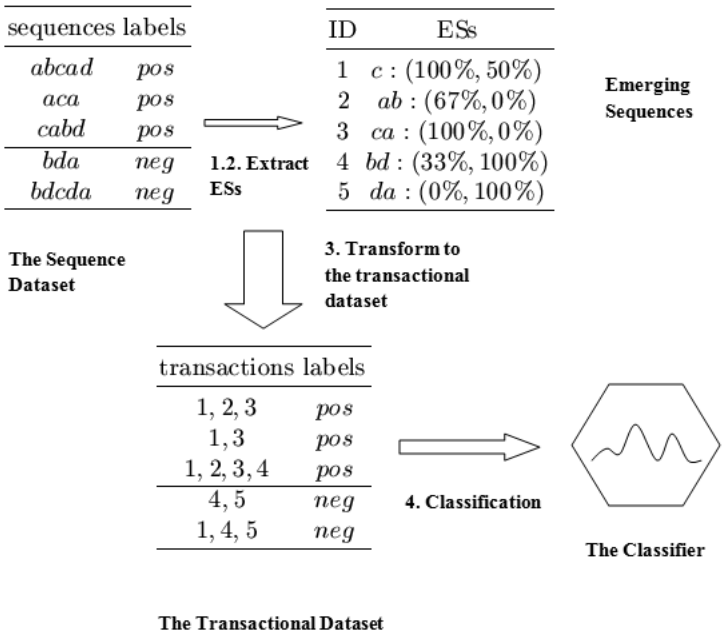


Figure 4.2: Four stages of classification. In Stage 1 and 2, ESs fulfilling 0-gap constraint are extracted. We transform the sequence dataset to a transactional dataset in Stage 3. The classification is performed in Stage 4.

Chapter 5

Transformation and Classification

After preprocessing and feature selection, ESs are extracted to contrast sequence groups. In this chapter, the sequence datasets are transformed to transactional datasets in order to be in a suitable form for learning algorithms. The transactions are simple sets of tokens representing ESs. Each ES is represented by a token (i.e. a simple ID) used within transactions (See Fig 4.2). Then, two classification algorithms trained by ESs are proposed on the transactional datasets.

In section 5.1, we describe the transformation and the sliding window matching mechanism for the distance metric between sequences. In section 5.2, two classification algorithms, Emerging Sequences Naïve Bayes and Association Rule-based Classifier are chosen to validate the discriminative power of emerging sequences.

5.1 Transformation to Transactional Datasets

To transform the sequence datasets to transactional datasets, we tokenize emerging sequences, so transactions are sets of tokens representing ESs. The transformation of training sets is straightforward: a transaction contains the corresponding tokens if the original sequence contain ESs. However, when classifying a new sequence instance, subsequences that are similar with ESs should also be considered.

To consider similar subsequences, we implement a sliding window matching mechanism for the distance metric between sequences. Given a sequence S of

length l_s, an emerging sequence es of length l_e, and $l_s \geq l_e$, we first extract a subsequence S_1 of length l_e starting from the first index of S, whose items are contiguous in S. Then we compare S_1 and es, if they are similar (i.e. not necessarily an exact match), the corresponding transaction should contain the token representing es. If not, we slide the window of length l_e to one position right to extract a new subsequence S_2, and compare it with es again. So there are $l_s - l_e + 1$ subsequences in total.

For example, given a sequence $S = \langle abcde \rangle$ and an emerging sequence $es = \langle bad \rangle$, since the length of es is 3; $5 - 3 + 1 = 3$ subsequences $\langle abc \rangle, \langle bcd \rangle$, and $\langle cde \rangle$ are extracted and compared with the emerging sequence es (see Figure 5.1); no exact matches but $\langle bcd \rangle$ could be similar to $\langle bad \rangle$ (see below).

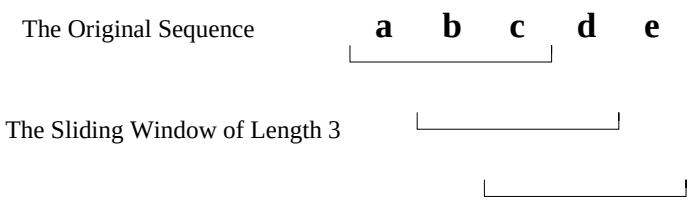


Figure 5.1: Subsequence extraction by the sliding window.

We measure the similarity between the emerging sequence and extracted subsequences by using the edit distance, also known as the Levenshtein distance [30]. The distance measure estimates the minimum needed operations or cost of operations to transform one sequence to the other. For simplification, the cost of all operations are assumed the same being one unit of measure (i.e. deletion, insertion and substitution cost the same one unit of measure). To compare two sequences, we introduce a maximum difference $\gamma \in [0, 1]$. First we calculate the edit distance between sequences. If the distance is equal to or lower than $\gamma \times l_e$, we say they are similar. For instance, when comparing the emerging sequence $es = \langle bad \rangle$ and the subsequence $seq = \{bcd\}$, if $\gamma = 0.4$, as $distance(es, seq) = 1 < (\gamma \times 3)$, they are similar. When $\gamma = 0$, the sequence S has to contain the emerging sequence (i.e.

exact match); when $\gamma = 1$, any subsequence is considered a match regardless. In this research, our proposed classification model has the best performance when γ is between 0.1 and 0.3.

The sliding window mechanism allows us to consider similar matches instead of only exact matches. Its performance is validated in Section 6.4.

5.2 Classification

As discussed in the previous section, sequence datasets are transformed to transactional datasets. To validate the discriminative power of ESs, learning algorithms trained by ESs are used to classify new sequence instances. We implement a Naïve Bayes based classifier and an Association Rule-based Classifier (ARC) [38]. If the features are informative, Naïve Bayes always has good prediction accuracy [34]; ARC is also proven having satisfactory performance when classifying sequence data [24].

5.2.1 Emerging Sequences Naïve Bayes

One reason we choose Naïve Bayes (NB) as the classifier is that no parameter is needed, so the selected features are the only factor associated with the performance of the classification model. Trained by representative features, NB outperforms other state-of-art learning algorithms. In [34], Li et al. compared the performances of the compound-risk factors Naïve Bayes (crf-NB) with SVM [13], C4.5 [44], Bagging [10] and Boosting [18], and found crf-NB had the highest prediction accuracy.

A Naïve Bayes classifier [28] assumes that all features are independent. Given a sequence S and a set of independent subsequences $seq_{indep} = \{seq_1, seq_2, \ldots, seq_n\}$, the sequence S can be represented by a set of subsequence-value pairs: $S = \{seq_1 = v_1, seq_2 = v_2, seq_3 = v_3, ..., seq_n = v_n\}$, where v_i is either $true$ or $false$,

indicating whether S contains seq_i. When C is the class set, according to the Bayes rules, the probability that sequence S is in the class c is:

$$p(c|S) = \frac{p(S|c)p(c)}{p(S)} \tag{5.1}$$

where $p(S|c)$ is the conditional probability of sequence S when class label c is known, and $c \in C$. Due to the independence of subsequences, $p(S|c)$ can be rewritten as:

$$p(S|c) = \prod_i p(seq_i = v_i|c) \tag{5.2}$$

Therefore, the class label predicted by Naïve Bayes is:

$$predict(S) = arg\ max_{c \in C} p(c) \times \prod_i p(seq_i = v_i|c) \tag{5.3}$$

In [45], Rish proved that the class-conditional mutual information is not a good predictor of Naïve Bayes performance, i.e. when features are independent, the Naïve Bayes classifier may not have the best prediction accuracy. Therefore, in the Emerging Sequences Naïve Bayes (es-NB), we do not assume the independence of subsequences. To convert the original Naïve Bayes to es-NB, we simply choose the Emerging Sequences to build the feature set.

$$predict'(S) = arg\ max_{c \in C} p(c) \times \prod_i p(es_i = v_i|c) \tag{5.4}$$

Equation 5.4 is used to predict labels, where $\{es_1, es_2, \ldots, es_m\}$ is the set of Emerging Sequences.

5.2.2 Classification based on Association

Another classifier we choose is Association Rule-based Classifier (ARC) [38]. Unlike the Naïve Bayes classifier, ARC needs two parameters: the minimum support is the threshold to generate rules, and the minimum confidence helps to prune unnecessary rules and select the rules to apply. ARC also proved having satisfactory per-

formance when classifying sequence datasets [24]. There are two stages of ARC: a training stage and a classification stage.

5.2.2.1 Training Stage

Given a transactional dataset $\mathcal{D}$, the minimum support min_sup and the minimum confidence min_conf, the procedures of ARC is as follows. In this dataset, each transaction consists of an antecedent (sequence set) and a label:

$$< seq_1, seq_2, \ldots, seq_i > \rightarrow label$$

Table 5.1: The example of a transactional dataset $\mathcal{D}$. The rows represent transactions; for each row, the first column is the transaction ID, the second column represents the subsequences it contains, and the third column is the class label.

Transaction ID	subsequences	labels
1	seq_1, seq_2, seq_3	A
2	seq_2, seq_3	A
3	seq_1, seq_3, seq_4, seq_5	B
4	seq_3	B
5	seq_1, seq_2, seq_3	A

The support of a rule is the ratio between its occurrence and the number of transactions in this class. For example, given a transactional dataset $\mathcal{D}$ as presented in Table 5.1, the support of rule $r_1 :< seq_1, seq_3 > \rightarrow A$ is $\frac{2}{3} = 67\%$, because this rule appears twice in $\mathcal{D}$ and there are 3 transactions labeled with A. In the rule mining phase, Apriori [1] is used to extract all the item sets with supports higher than min_sup. In our case, the transactional dataset is transformed by a preprocessed sequence dataset, and the ESs used in the transformation fulfill the minimum support threshold. As a result, min_sup is not necessary in the rule mining stage; theoretically, the minimum support in ARC should be 0. However, due to the hardware limitation, the minimum support is between 1% and 4% in

our experiments, thus avoiding too many rules and keeping completeness of the datasets. In conclusion, the extracted rules of this stage are *frequent*.

The second phase is rule pruning, which is based on the confidence. The confidence of a rule is the ratio between the occurrences of the rule and its antecedent. Take $r_1 :< seq_1, seq_3 >\rightarrow A$ as an example, and its occurrence is 2; meanwhile, $< seq_1, seq_3 >$ appears 3 times in the dataset $\mathcal{D}$. So the confidence of r_1 is $conf(r_1) = \frac{2}{3} = 0.67$. For all rules that have the same sequence set, the rule with the highest confidence is chosen as a possible rule. If there are more than one rules having the same confidence, only one is selected randomly. For instance, the following rules have the same sequence set $< seq_1, seq_3 >$:

$$r_1 :< seq_1, seq_3 >\rightarrow A$$

$$r_2 :< seq_1, seq_3 >\rightarrow B$$

The occurrence of the r_1 is 2, while that of r_2 is 1. So, their confidences are 67% and 33% respectively. As a result, only one possible rule r_1 is generated because $conf(r_1) > conf(r_2)$. Finally, if the confidence of the possible rule is higher than the minimum confidence min_conf, we say the rule is *accurate*. Therefore, all the rules in the classifier are both *frequent* and *accurate*.

In our research, two sequence groups are compared at a time. For any two rules r_i and r_j with the same antecedent and different labels, the sum of their confidences is 100%, i.e. $conf(r_i) + conf(r_j) = 100\%$. In our research, we fix min_conf as 51%. If $conf(r_i) \neq conf(r_j)$, the rule with the lower confidence is eliminated; if $conf(r_i) = conf(r_j) = 50\%$, both rules have lower confidences than the threshold, so no rule is selected. Therefore, no useful rule is eliminated by min_conf in our case.

In conclusion, both min_sup and min_conf are not important in the training stage, and the only features can influence the performance of ARC.

5.2.2.2 Classification Stage

In the classification stage, a testing sequence may match more than one rules, which have different labels. There are different strategies to decide the label:

- choose the rule having the highest precedence
- choose the label having the highest average confidence

The first strategy is proposed by Liu et al [38]. Given two rules r_i and r_j, r_i has a higher precedence if:

1. $conf(r_i) > conf(r_j)$, or
2. if $conf(r_i) = conf(r_j)$, $sup(r_i) > sup(r_j)$, or
3. if $conf(r_i) = conf(r_j)$ and $sup(r_i) = sup(r_j)$, r_i is generated earlier than r_j

This strategy makes sense, because it chooses the optimal rule for the testing sequence. However, it ignores the effects of other rules.

Table 5.2: An example of a rule set. The rule set consists of 4 rules, two of them are labeled as pos, while the other two are neg. They are ranked by confidence.

rule ID	rules	confidences
r_1	$< seq_1, seq_3 > \rightarrow pos$	90%
r_2	$< seq_1, seq_2, seq_4 > \rightarrow neg$	85%
r_3	$< seq_3, seq_4 > \rightarrow neg$	80%
r_4	$< seq_1, seq_5 > \rightarrow pos$	60%

Given a testing transaction $s = \{seq_1, seq_2, seq_3, seq_4, seq_5\}$, it matches the antecedents of 4 rules in Table 5.2. If the rule with the highest precedence (r_1) is chosen, the transaction should be labeled pos. However, both r_2 and r_3 also have high confidences, and they should play roles in the final decision. To make a more comprehensive consideration, the alternative strategy is that, the label with

the higher average confidence is selected. In the example above, the label of r_1 and r_4 is pos, and their average confidence is 75%, while that of r_2 and r_3 with label neg is 82.5%. So the transaction s should be labeled as neg. In our ARC, we choose the average confidence strategy, thus avoiding the ignorance of other matched rules.

In very few cases, the testing sequence cannot match any rule in ARC. Therefore, we choose the label having more training data as the default label, and the unknown sequences are classified with the default label directly. As the number of these cases is very low, this simplistic method will not affect the overall accuracy.

Chapter 6

Experimental Results

To evaluate the performance of our proposed classification model, we tested the classifier on two types of datasets in this chapter. As discussed in Chapters 4 and 5, our proposed similar Emerging Sequence-based algorithm (Similar ES) can be divided into four stages (See Figure 4.2):

1. Subsequences, which fulfill the discriminative conditions ($support(\alpha, \mathcal{D}_{pos}) > \theta$ and $support(\alpha, \mathcal{D}_{neg}) \leq \theta$) are chosen as candidates.

2. Subsequences with higher sup_diff are selected as Emerging Sequences.

3. Transform the sequence datasets to the transactional datasets, and subsequences that are similar with ESs are considered as well.

4. Train a classifier by Emerging Sequences.

Figure 6.1 illustrates the approaches on 4 stages. On each stage, different strategies can be applied: on Stage 1, we can mine subsequences with varying gap constraint by $ConSGapMiner$ or without gap (0-gap constraint) by the suffix tree-based framework; on Stage 2, the static or dynamic feature selection methods can be chosen; on Stage 3, we can perform exact matching or similar matching mechanisms; and on Stage 4, two classifiers, Naïve Bayes or Association Rule-based Classifiers can be selected. To build the optimal classification model, we compare different methods on each stage and present the results in this chapter.

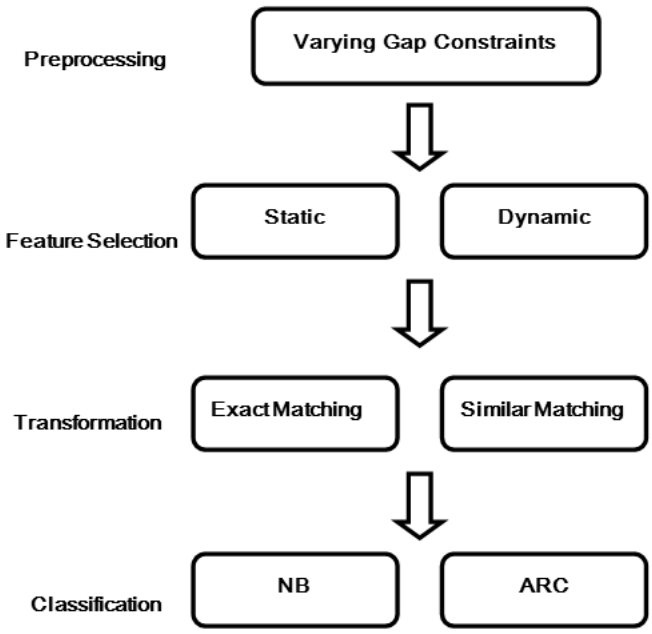


Figure 6.1: 4-Stage Classification Framework

This chapter is organized as follows: we first introduce the evaluation methodology and datasets in Section 6.1. In Section 6.2, we compare the candidates mining algorithms with different gap constraints, and decide the strategy for Stage 1. Section 6.3 presents the comparison between static and dynamic feature selections (Stage 2). Section 6.4 presents the advantage of similar matching mechanism (Stage 3). Before deciding the classifier in Stage 4, we show the discriminative power of emerging sequences, and discuss the most emerging sequences in Section 6.5. Known the most emerging sequences, we compare the performances of our proposed model by different classifiers in Section 6.6.

6.1 Evaluation Methodology and Datasets

To decide the optimal strategy in different stages, we fix the methods of other stages, and only change the method in the current stage. However, when evaluating the strategies in a stage, an issue is how to choose the algorithms in other stages. For

example, when validating the algorithms in Stage 1, there are 8 combinations to choose the algorithms in other stages. Thus we proceed in a top-down fashion, as we select an appropriate strategy at a given stage, we fix the approaches needed at the other stages. To do this, we simply choose the approach that needs simpler parameters or does not need parameter inputs. More specifically, the static feature selection needs a simpler parameter as opposed to the dynamic feature selection which needs a more complex parameter setting, the exact matching does not need parameters, an neither does Naïve Bayes. Therefore, the static feature selection, exact matching mechanism and Emerging Sequences Naïve Bayes classifier are the algorithms of other stages when evaluating the methods of Stage 1.

We apply the F-measure to evaluate the prediction performance. The F-measure is a harmonic average between precision and recall; the relations are illustrated as follows:

Table 6.1: Confusion Matrix.

		Correct Results	
		$E1$	$E2$
Obtained Results	$E1$	TP (true positive)	FP (false positive)
	$E2$	FN (false negative)	TN (true negative)

"Precision is the percentage of slots in the hypothesis that are correct, while recall is the percentage of reference slots for which the hypothesis is correct" [40].

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN}$$

The F-measure can be interpreted as a weighted average of the precision and recall and is typically defined as follows:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Finally, we perform 6-fold cross validation, and the average F-measure of the 6 folds is reported for each dataset.

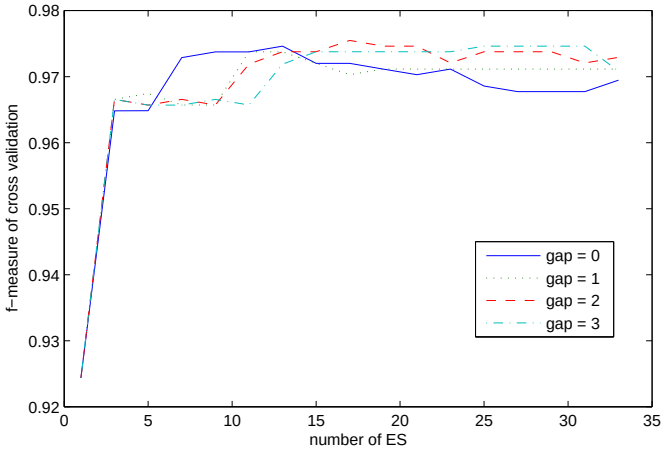
The first type of datasets we use is the UNIX user commands dataset from the UCI Machine Learning Repository [5]. It contains 9 sets of sanitized user data drawn from the command histories of 8 UNIX computer users at Purdue University. This dataset only keeps command names, flags, and shell meta characters, while removing filenames, user names, directory structures etc. The average sequence number in each group is 1011, and the average length of sequences is 27. The size of the vocabulary is 2345. We believe different users have discriminative habits when typing commands.

The second dataset is the epitope data, which are short linear peptides (amino acid sequences) generated by cleavage of antigenic proteins [17]. The identification of epitopes in protein sequences is important for understanding disease pathogenesis, and a major step involves identifying the peptides that bind to a target major histocompatibility complex (MHC) molecule. The average sequence number in each group is 363, and the average length of sequences is 13. The size of the vocabulary is 20. To contrast the binding and non-binding peptides, we perform the test on several groups of the epitope data.

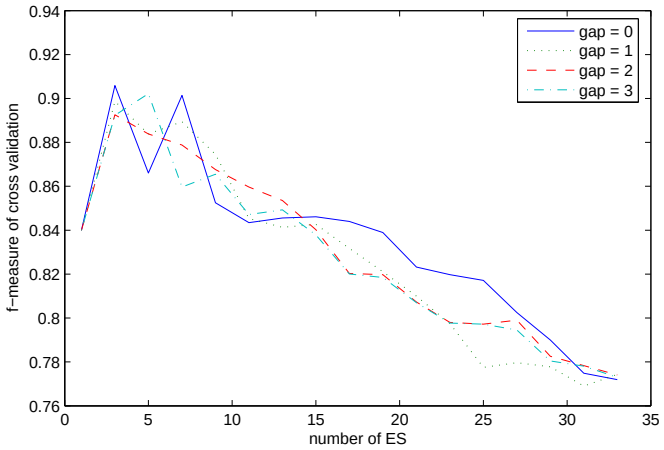
6.2 Candidates Mining by Varying Gap Constraint

As discussed in Section 4.1, the subsequence leaving the gap constraint arbitrary is meaningless to represent the original sequence. So the items of subsequences should be close in the original sequence. We expect the highest prediction accuracy when the gap constraint is close to 0. To find the best gap for Stage 1, we fixed the other stages with algorithms without parameters, thus we adopt the static feature selection, exact matching mechanism and Emerging Sequences Naïve Bayes classifier while changing the gap constraint in Stage 1. In each experiment, two users' commands are chosen from the UNIX command dataset [5], and the highest F-measure indicated the optimal gap constraint needed. To control the gap constraint,

we implement $ConSGapMiner$ [25] as the candidate mining algorithm.



(a) User 2 and User 3.



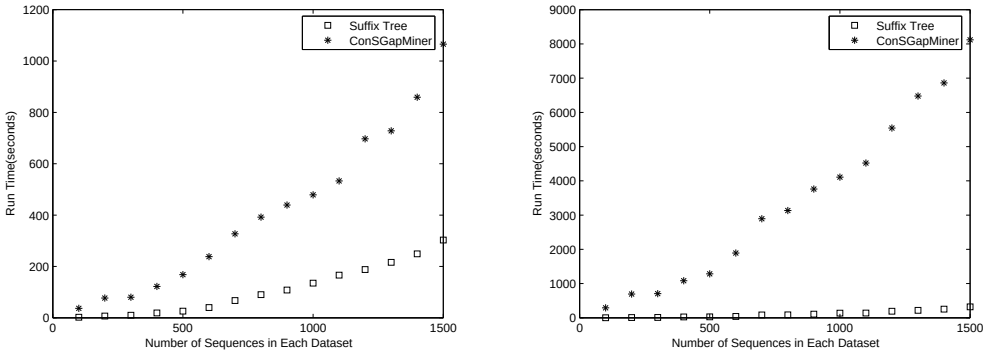
(b) User 6 and User 7.

Figure 6.2: The comparisons of F-measures with different gap constraints. In each figure, a user pair is chosen. X-axis is the number of emerging sequences, y-axis is the average F-measure. There are 4 curves in a figure, each represents the F-measure with a gap constraint (from $gap = 0$ to $gap = 3$). Static feature selection, exact matching mechanism and Emerging Sequences Naïve Bayes classifier are used in this comparison.

Figure 6.2 is the comparisons of F-measures when different gap constraints are chosen. In each figure, one user pair is selected. We find that the curves of different gap constraints are very close. i.e. the gap constraint in $ConSGapMiner$ is not a decisive factor of the classification performance when it is close to 0. Therefore,

we can extract the ES candidates fulfilling 0-gap constraint because it does not influence the prediction accuracy. Note that a 0-gap constraint is different from having no gap constraints at all where any gap is possible. A 0-gap constraint specifically means no gaps are allowed.

Since both $ConSGapMiner$ and our suffix tree-based framework can extract subsequences fulfilling 0-gap constraint, we compare these two algorithms by scalability. As illustrated in Figure 6.3, $ConSGapMiner$ is much slower due to the exponential growth of the data structure used in it. In conclusion, the suffix tree-based framework, if used in Stage 1, has a similar performance with $ConSGapMiner$ on the prediction accuracy while decreasing the run time significantly. All following experiments are performed by adopting the suffix tree-based framework.



(a) Minimum Support $\theta = 0.2$.

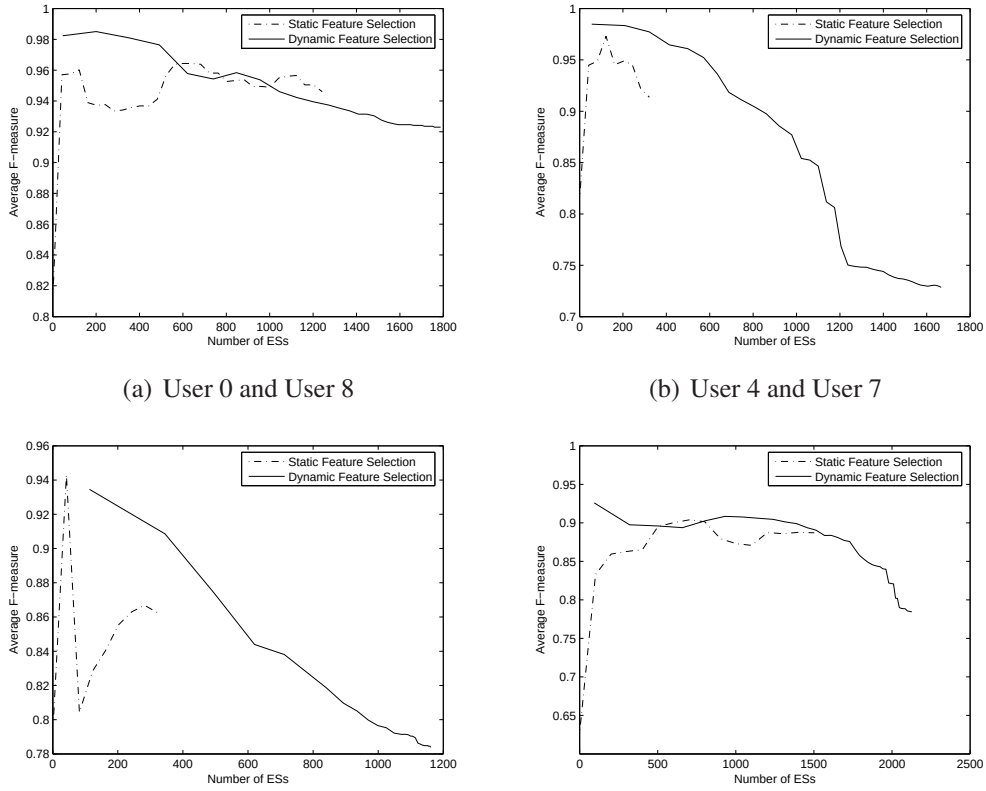
(b) Minimum Support $\theta = 0.05$.

Figure 6.3: Scalability of Similar ES versus ConSGapMiner with increasing size of Unix command dataset.

6.3 Static Feature selection vs Dynamic Feature selection

After ES candidates are extracted, the next issue is the selection of features. How many emerging sequences to keep and which ones to weed out. As discussed in Section 4.2, we implement both the static and the dynamic feature selection meth-

ods. In the static feature selection, we choose the top-k subsequences (ranked by sup_diff) of each class as emerging sequences; while top-m emerging sequences are selected for each original sequence in the dynamic feature selection. As we increase k or m, the sum of ESs in both classes also increases. We present the relation between the sum of ESs and the average F-measure on both the static feature selection and the dynamic feature selection in Figure 6.4.



(a) User 0 and User 8

(b) User 4 and User 7

(c) User 6 and User 7

(d) User 4 and User 6

Figure 6.4: The relation between the sum of ESs and the average F-measure on both the static feature selection and the dynamic feature selection. In each figure, a user pair is chosen. The x-axis is the sum of ESs in two sequence groups, while the y-axis is average F-measure. GST, exact matching mechanism and Emerging Sequences Naïve Bayes classifier are used in this comparison.

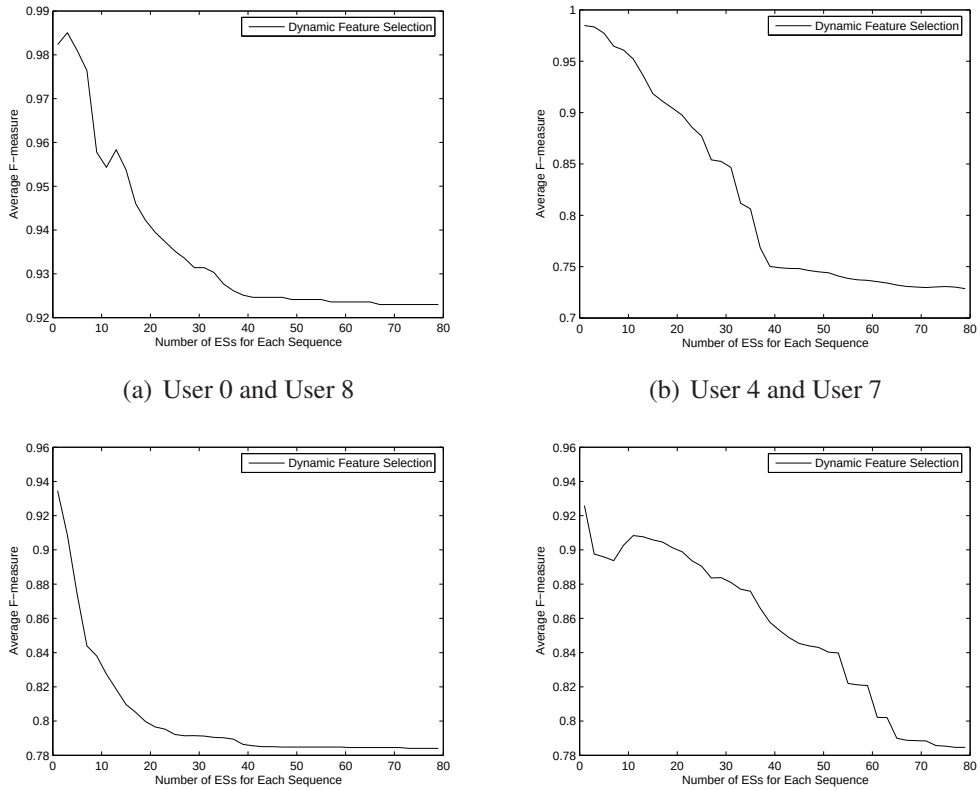
Since the static feature selection method fixes the number of ESs in each sequence class to k, the number of the ESs selected remains always the same ($2k$) regardless of the sizes of classes. Suppose the numbers of ES candidates in two

classes are x and y ($x > y$), at most $2y$ ESs can be selected by the static feature selection. The dynamic feature selection does not suffer from this problem because the number of ESs is fixed by sequence not by class, all sequences have the chance to be expressed. In Figure 6.4(b), for instance, over 1600 ESs are selected by the dynamic approach while the static method can only extract 300 ESs because one class has 150 candidates.

The main reason that the dynamic approach is superior to the static method is that the average F-measure of the dynamic approach is monotonic with respect to the number of ESs. On the contrary, in the static feature selection, the relationship between the number of ESs and the average F-measure is not stable: the highest F-measure is achieved when 600 ESs are selected in Figure 6.4(a); however, in Figure 6.4(b), our algorithm attains the best performance when the number of ESs is about 100. By only comparing the highest classification accuracies of two methods, it is clear that the dynamic feature selection performs better than the static approach in most cases.

Since the dynamic feature selection is easier to control, guarantees that all sequences are expressed (i.e. no silent sequence), and performs better in the classification, the next issue is to determine how-many ESs should be extracted for each sequence, so that the algorithm can achieve the highest prediction accuracy. Figure 6.5 illustrates the relation between the number of ESs for each sequence and the average F-measure. When every original sequence is covered by 1 to 3 ESs, our proposed model has the highest prediction accuracy. In other words, the dynamic feature selection ensures that, every sequence can be expressed by at least 1 to 3 ESs, thus keeping the completeness of the datasets.

In conclusion, the best prediction performance can be achieved when m is set to 1-3. Moreover, in most cases, the dynamic approach outperforms the static one on classification accuracy. Therefore, we choose the dynamic feature selection as the



(a) User 0 and User 8

(b) User 4 and User 7

(c) User 6 and User 7

(d) User 4 and User 6

Figure 6.5: The relation between the number of ESs for each sequence and the average F-measure in the dynamic feature selection. The x-axis is the number of ESs for each sequence, while the y-axis is average F-measure.

method in Stage 2 (see Figure 4.2). The following experiments are all performed based on the dynamic feature selection.

6.4 Similar Matching vs Exact Matching

As we decide the strategies of Stage one and Stage two in our proposed framework, the next issue is the transformation from sequence datasets to transactional datasets. We believe the sliding window matching mechanism considers similar subsequences, thus improving the prediction accuracy. To validate the improvement, we design another algorithm based on exact matching mechanism. Compared with the similar ES-based algorithm, the exact ES-based algorithm also has 4

stages, and both algorithms adopt Emerging Sequences Naïve Bayes as the classifier. The only difference is that the exact ES-based algorithm only considers exact matching, i.e. a transaction contains the tokens if the original sequence contains the corresponding ESs.

In Table 6.2, we present the F-measures and standard deviations of the UNIX user command dataset [5] in Row 1-5, where the minimum support θ is set to 0.01; the result of epitope data [17] is presented in Row 6-11, and θ is set to 0.05. As discussed in Section 6.3, the parameter m is set to 2 to achieve the best classification performance. One more parameter γ, which is specially designed for similar ESs-based algorithm (See Section 5.1), is used to measure the distance difference between two sequences. It is set to 0.1 for the UNIX command dataset, and 0.2 for the epitope dataset.

Table 6.2: Classification performances of exact ESs-based algorithm and similar ESs-based algorithm. For the prediction accuracies of the Similar ESs-based algorithm on all UNIX user pairs, please refer to Appendix A.

Datasets	Exact ESs-based	Similar ESs-based	Difference
user 0 and 3	0.963133 ± 0.0189591	0.962192 ± 0.00864717	1%
user 0 and 5	0.939128 ± 0.0107074	0.940028 ± 0.0118862	0%
user 2 and 7	0.971946 ± 0.0128228	0.969818 ± 0.00861439	0%
user 7 and 8	0.852766 ± 0.0192199	0.853639 ± 0.0233047	0%
user 2 and 3	0.984494 ± 0.00798139	0.984516 ± 0.00600006	0%
I-Ek	0.859395 ± 0.0237948	0.862556 ± 0.023571	0%
HLA-DR1	0.72197 ± 0.0382523	0.741143 ± 0.0228598	2%
HLA-DQ2	0.811726 ± 0.0936836	0.864592 ± 0.0300073	5%
HLA-DQ4	0.789487 ± 0.0922259	0.821227 ± 0.055203	3%
HLA-DR3	0.757886 ± 0.0329568	0.777372 ± 0.0662611	2%
HLA-DR7	0.770727 ± 0.0323399	0.790606 ± 0.035915	2%

From Table 6.2, we observe that, the sliding window matching mechanism enhances the classification accuracy: the F-measures are improved by up to 5%. However, its improvement also depends on the datasets. An extreme example is the

result of user 2 and 3 (Row 5), where the exact matching and similar matching algorithms have similar F-measures. The reason for that is that users 2 and 3 have one length-1 emerging sequence respectively. When the maximum difference γ is set to 0.1, our framework always seeks exact matching subsequences, in other words, both approaches become literally identical.

We notice that the performances on the UNIX command dataset (in the order of 94%) is much better than those on the epitope dataset (in the order of 81%). One explanation is that, the epitope dataset is already preprocessed by removing short, unnatural, and duplicated peptides [17], while the frequencies of peptides are important for our algorithm. Therefore, our preprocessing-embedded model works better on raw data.

6.5 Discriminative Power of Emerging Sequences

Before deciding the classifier in Stage 4, one important task is to find the most emerging sequences. With different minimum support θ, emerging sequences have different discriminative power. Trained by the most discriminative features, classifiers have the best performances; meanwhile, the comparison between classifiers is meaningful. In this section, we first verify the discriminative power of emerging sequences, then the most emerging sequences can be found by varying the minimum support θ.

6.5.1 Emerging Sequences vs Frequent Subsequences

Given two sequence classes, to distinguish the target class from the contrasting class, we select emerging sequences with the largest support differences. To validate the discriminative power of ESs, we design another two 4-stage classification frameworks, one based on the frequent subsequences in the target class, while another based on the ESs we define. The first two stages of these algorithms are

different:

- Frequency-based Algorithm: In Stage 1, subsequences that are frequent constitute the feature set ($support(\alpha, \mathcal{D}_{pos}) > \theta$); in Stage 2, subsequences are ranked by their frequencies.
- ESs-based Algorithm: In Stages 1, subsequences fulfilling the discriminative conditions are selected ($support(\alpha, \mathcal{D}_{pos}) > \theta$ and $support(\alpha, \mathcal{D}_{neg}) \leq \theta$); in Stage 2, subsequences with higher sup_diff are selected as Emerging Sequences..

The motivation for the frequency-based algorithm is that if we rank subsequences according to frequency, those discriminative ones usually have high ranks [39]. We can evaluate the effect of ESs according to the comparison between this approach and the ESs-based Algorithm.

Table 6.3: Classification performances of Frequency-based algorithm and ESs-based algorithm.

Datasets	Frequency-based	ESs-based	Difference
user 0 and 3	0.891992 ± 0.0280118	0.953464 ± 0.0121005	6%
user 0 and 5	0.869967 ± 0.0250203	0.939128 ± 0.0107074	7%
user 2 and 7	0.94854 ± 0.0110985	0.969787 ± 0.00733708	2%
user 7 and 8	0.818205 ± 0.0154953	0.852122 ± 0.0180784	3%
user 2 and 3	0.965973 ± 0.0225253	0.984494 ± 0.00798139	2%
I-Ek	0.760296 ± 0.0299356	0.859395 ± 0.0237948	10%
HLA-DR1	0.63103 ± 0.0218163	0.72197 ± 0.0382523	9%
HLA-DQ2	0.661909 ± 0.0620348	0.811726 ± 0.0936836	15%
HLA-DQ4	0.712941 ± 0.0581559	0.789487 ± 0.0922259	8%
HLA-DR3	0.607697 ± 0.051646	0.757886 ± 0.0329568	15%
HLA-DR7	0.696771 ± 0.0255892	0.770727 ± 0.0323399	7%

Table 6.3 is the comparison between the frequency-based algorithm and the ESs-based algorithm. Both algorithms adopt Emerging Sequence Naïve Bayes as the classifier. We perform the experiments on the same datasets with Section 6.4.

By comparing two approaches (based on frequent subsequences versus based on emerging sequences), we find that Emerging Sequences play a significant role in classification: the F-measures are improved by up to 15%. Therefore, the Emerging Sequences we define are much more discriminative than frequent subsequences.

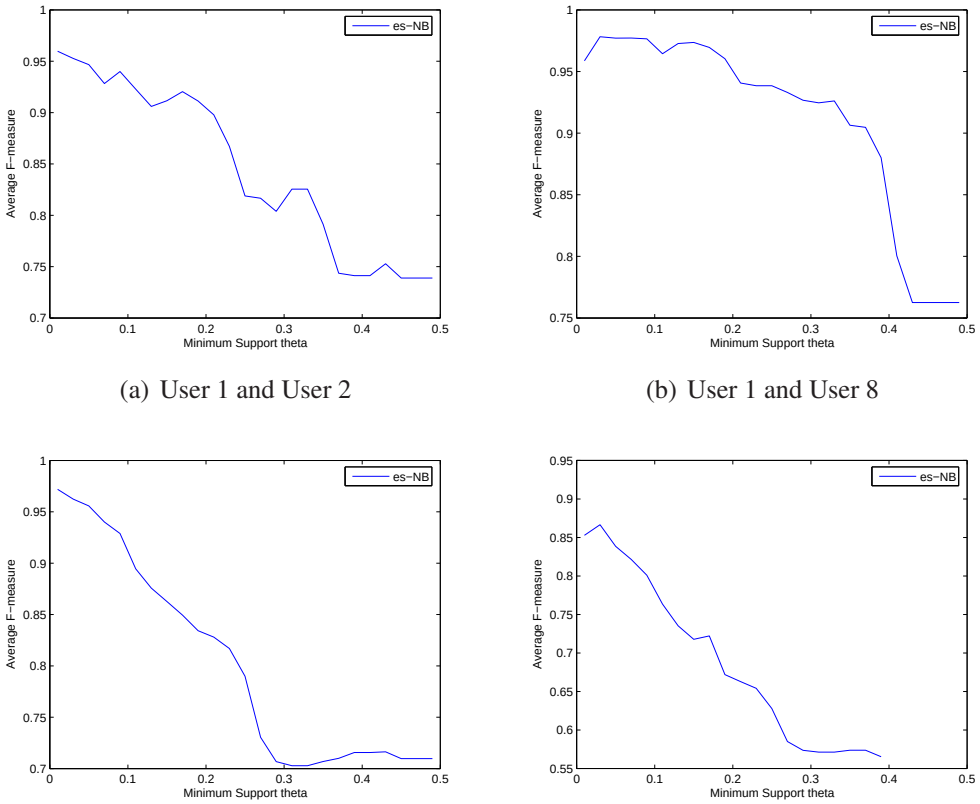
6.5.2 Emerging Sequences of Varying Minimum Support

As we verify the discriminative power of ESs, the next issue is to find the most emerging sequences. In our research, emerging sequences are subsequences that fulfill the discriminative conditions ($support(\alpha, \mathcal{D}_{pos}) > \theta$ and $support(\alpha, \mathcal{D}_{neg}) \leq \theta$). Generally, as the increase of the minimum support θ, less ESs are extracted. In this section, we seek the relationship between θ and ESs; we want to find the optimal value for θ, so ESs have the highest discriminative power.

We test the emerging sequences in the 4-stage classification framework. As ESs are used to train the classifier, if the classifier can achieve a better prediction performance, the ESs have the higher discriminative power. In our proposed framework, the approaches from Stage 1 to 3 are decided. As the classifier of Stage 4 is unknown, we perform the experiments on both classifiers (es-NB and ARC). We choose four user pairs from the UNIX command dataset [5]. The parameter m in Stage 2 is set to 2, and the maximum difference γ in Stage 3 is set to 0.1.

As shown in Figure 6.6, we observe that, as the increase of the minimum support, the curve of the average F-measure goes down rapidly. The es-NB classifier has the best prediction performance when the minimum support is close to 0. In other words, the emerging sequences are the most discriminative if $\theta \approx 0$. To verify this result, we perform the same experiments on the Association Rule-based Classifier, and present the result in Figure 6.7.

From Figure 6.7, the conclusion is the same: with the increase of the minimum support, the classification accuracy degrades. The reason for that is that, if θ is too



(a) User 1 and User 2

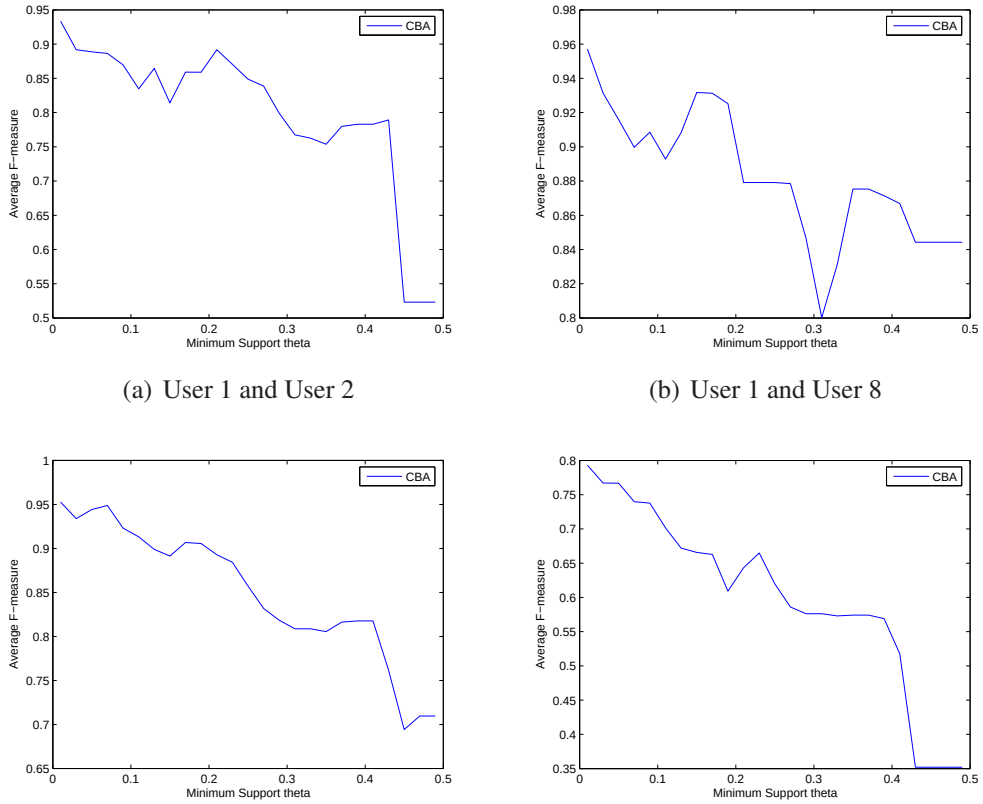
(b) User 1 and User 8

(c) User 2 and User 7

(d) User 7 and User 8

Figure 6.6: The classification performance of es-NB with varying minimum support. Four user pairs are chosen. In each figure, x-axis represents the minimum support θ in preprocessing, while y-axis represents the average F-measure.

high, the minimum support threshold eliminates some emerging sequences of high discriminative power. When θ is set to 0.01, our Similar ESs-based model achieve the highest accuracy. Given two groups of sequences (the target group and the contrasting group), Stage 1 of our algorithm ensures that the ESs candidates hardly appear in the contrasting group, while Stage 2 selects the high-frequent candidates in the target group. In other words, the most emerging sequences are frequent in the target group, while they (almost) cannot be found in the contrasting group. We name this type of subsequences *jumping emerging sequences* (JESs). In conclusion, our proposed algorithm achieves the best performance when the classifier is trained by JESs.



(a) User 1 and User 2

(b) User 1 and User 8

(c) User 2 and User 7

(d) User 7 and User 8

Figure 6.7: The classification performance of ARC with varying minimum support. Four user pairs are chosen. In each figure, x-axis represents the minimum support θ in preprocessing, while y-axis represents the average F-measure.

6.6 Naïve Bayes vs Association Rule-based Classifier

For both Emerging Sequence Naïve Bayes and Association Rule-based Classifier (ARC), the best prediction accuracy can be achieved when the minimum support θ is close to 0. Therefore, instead of comparing the curves in Figure 6.6 and Figure 6.7, we just need to compare their average F-measures when $\theta = 0.01$. We perform the experiments on five user pairs from the UNIX command dataset [5]. We design two algorithms based on our 4-stage classification framework, while the first 3 stages are the same. The only difference is that, one algorithm adopts es-NB, while the other implements ARC.

The average F-measures and standard deviations are presented in Table 6.4. We

Table 6.4: Classification performances of es-NB and ARC. For the prediction accuracies of es-NB on all UNIX user pairs, please refer to Appendix A.

Datasets	es-NB	ARC	Difference
user 0 and 3	0.963133 ± 0.0189591	0.947097 ± 0.0229249	2%
user 1 and 2	0.959736 ± 0.0149326	0.93348 ± 0.0124664	3%
user 1 and 8	0.958649 ± 0.0113836	0.957143 ± 0.0142829	0%
user 2 and 7	0.971946 ± 0.0128228	0.952679 ± 0.00812296	2%
user 7 and 8	0.852766 ± 0.0192199	0.793003 ± 0.0101451	6%

observe that, es-NB outperforms ARC in most cases by up to 6% in average F-measure. One possible explanation is that, the minimum support and minimum confidence thresholds of ARC eliminate some potential rules, while es-NB keeps all discriminative information in the form of conditional probabilities. Therefore, es-NB is chosen as the classifier in our proposed prediction model.

Chapter 7

Conclusion

In this dissertation, we define Emerging Sequences (ESs) as subsequences that are frequent in sequences of one group and less frequent in the sequences of another, and thus distinguishing or contrasting sequences of different classes. ESs can be used to contrast sequence groups. However, there are two challenges to distinguish sequence classes: the extraction of ESs is not trivially efficient and only exact matches of sequences are considered. In our work we address those problems by a suffix tree-based framework and a sliding window matching mechanism for the distance metric between sequences. We propose a 4-stage classification framework for sequence data based on Emerging Sequences.

To decide the approaches and evaluate the current strategies in 4 stages, we design several other algorithms, which adopt $ConSGapMiner$, static feature selection, exact sequence matching, or Association Rule-based Classifier in each stage, and perform the comparisons with the baseline algorithm. We find that the combination of GST, dynamic feature selection, similar matching mechanism and Emerging Sequence Naïve Bayes has a better prediction accuracy when classifying sequence groups. To demonstrate the discriminative power of Emerging Sequences, we performed the experiments on the UNIX command dataset [5] and epitope dataset [17]. The experiments show that ESs are much more discriminative than frequent subsequences on these datasets. Our similar ESs-based classification model achieves

satisfactory F-measures (as high as 98%) on the UNIX command dataset. The best performance can be achieved when our algorithm is trained using *jumping emerging sequences*.

Our proposed algorithm is based on a Naïve Bayes classifier since it gave better results with our Emerging Sequence patterns on those datasets, than other classifiers such as an Associative Classifier. The Associative classifier, however, gave better results using our Emerging Sequence patterns on other protein data, not reported in this thesis. One interesting question which remains an open problem is how to select the best classifier given a sequence dataset or given properties of a set of discovered discriminative sequences.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] Hamad Alhammady and Kotagiri Ramamohanarao. Using emerging patterns to construct weighted decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):865–876, 2006.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[4] Maria-Luiza Antonie, Osmar R. Zaïane, and Alexandru Coman. Application of data mining techniques for medical image classification. In *in Proc. of Second Intl. Workshop on Multimedia Data Mining (MDM/KDD'2001) in conjunction with Seventh ACM SIGKDD*, pages 94–101, San Francisco, CA, august 2001.

[5] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[6] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Knowledge Discovery and Data Mining Conference (KDD)*, pages 429–435, 2002.

[7] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. Fast algorithms for mining emerging patterns. In *PKDD*, pages 39–50, 2002.

[8] Stephen D. Bay and Michael J. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Knowledge Discovery and Data Mining Conference (KDD99)*, pages 302–306, 1999.

[9] Roberto J. Bayardo, Jr. Efficiently mining long patterns from databases. *SIGMOD Record*, 27(2):85–93, 1998.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[11] Yandong Cai, Nick Cercone, and Jiawei Han. Attribute-oriented induction in relational databases. In *Knowledge Discovery in Databases*, pages 213–228. AAAI/MIT Press, 1991.

[12] Sarah Chan, Ben Kao, Chi Lap Yip, and Michael Tang. Mining emerging substrings. In *Database Systems for Advanced Applications (DASFAA)*, page 119, 2003.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[14] Kang Deng and Osmar R. Zaïane. Contrasting sequence groups by emerging sequences. In *Discovery Science*, 2009.

[15] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: discovering trends and differences. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52, New York, NY, USA, 1999. ACM.

[16] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Discovery Science*, pages 30–42, 1999.

[17] Yasser EL-Manzalawy, Drena Dobbs, and Vasant Honavar. On evaluating mhc-ii binding peptide prediction methods. *PLoS ONE*, 3(9):e3268, 09 2008.

[18] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[19] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, January 1997.

[20] Jiawei Han, Yongjian Fu, Wei Wang, Krzysztof Koperski, and Osmar Zaïane. DMQL: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.

[21] Jiawei Han and Micheline Kamber. *Data Mining, Concepts and Techniques*. Morgan Kaufmann, 2001.

[22] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12, 2000.

[23] Robert J. Hilderman and Terry Peckham. A statistically sound alternative approach to mining contrast sets. In *In Proceedings of the 4th Australasian Data Mining Conference (AusDM*, pages 157–172, 2005.

[24] S. Vahid Jazayeri and Osmar R. Zaïane. Plant protein localization using discriminative and frequent partition-based subsequences. In *ICDM Workshops*, pages 228–237, 2008.

[25] Xiaonan Ji, James Bailey, and Guozhu Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowl. Inf. Syst.*, 11(3):259–286, 2007.

[26] J. Bailey K. Ramamohanarao and G. Dong. tutorial Contrast Data Mining: Methods and Applications. International Conference on Data Mining (ICDM), 2007.

[27] J. Kay, N. Maisonneuve, K. Yacef, and O. Zaïane. Mining patterns of events in students teamwork data. pages 1–8, 2006.

[28] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[29] Neal Lesh, Mohammed Javeed Zaki, and Mitsunori Ogihara. Mining features for sequence classification. In *Knowledge Discovery and Data Mining Conference (KDD)*, pages 342–346, 1999.

[30] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[31] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *PKDD*, pages 191–200, 2000.

[32] Jinyan Li and Limsoon Wong. Emerging patterns and gene expression data. In *Proceedings of 12th Workshop on Genome Informatics*, pages 3–13, 2001.

[33] Jinyan Li and Limsoon Wong. Geography of differences between two classes of data. In *PKDD*, pages 325–337, 2002.

[34] Jinyan Li and Qiang Yang. Strong compound-risk factors: Efficient discovery through emerging patterns and contrast sets. *IEEE Transactions on Information Technology in Biomedicine*, 11(5):544–552, 2007.

[35] Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *International Conference on Data Mining (ICDM)*, pages 369–376, 2001.

[36] Tim Funting Liao. *Statoistical Group Comparison*. Wiley's Series in probability and Statistics, 2002.

[37] Jessica Lin and Eamonn J. Keogh. Group sax: Extending the notion of contrast sets to time series and multimedia data. In *In proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 284–296, 2006.

[38] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining Conference (KDD98)*, pages 80–86, 1998.

[39] David Lo, Hong Cheng, Jiawei Han, and Siau-Cheng Khoo. Classification of software behaviors for failure detection: A discriminative pattern mining approach. In *Knowledge Discovery and Data Mining Conference (KDD)*, 2009.

[40] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction, 1999.

[41] Shihong Mao and Guozhu Dong. Discovery of highly differentiative gene groups from microarray gene expression data using the gene club approach. *J. Bioinformatics and Computational Biology*, 3(6):1263–1280, 2005.

[42] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

[43] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *International Conference on Data Engineering (ICDE)*, pages 215–224, 2001.

[44] Ross J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.

[45] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAIInternational Joint Conferences on Artificial Intelligence-01 workshop on "Empirical Methods in AI"*.

[46] Amit Satsangi and Osmar R. Zaiane. Contrasting the contrast sets: An alternative approach. In *IDEAS '07: Proceedings of the 11th International Database Engineering and Applications Symposium*, pages 114–119, 2007.

[47] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.

[48] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Characterising the difference. In *KDD*, pages 765–774, 2007.

[49] Lusheng Wang, Hao Zhao, Guozhu Dong, and Jianping Li. On the complexity of finding emerging patterns. *Theor. Comput. Sci.*, 335(1):15–27, 2005.

[50] Xiaoxin Yin and Jiawei Han. CPAR: Classification based on predictive association rules. In *SDM*, 2003.

[51] Osmar R. Zaïane, Kalina Yacef, and Judy Kay. Finding top-n emerging sequences to contrast sequence sets. February 2007. Technical Report TR07-03, Department of Computing Science, University of Alberta, Edmonton, AB, Canada.

[52] Mohammed Javeed Zaki. Efficient enumeration of frequent sequences. In *CIKM*, pages 68–75, 1998.

Appendix A

Prediction accuracies on all user pairs of the UNIX command dataset

Table A.1: Prediction accuracies on all user pairs of the UNIX command dataset [5]. The experiments are performed on the Similar ES-based classification framework.

	user 0	user 1	user 2	user 3	user 4	user 5	user 6	user 7
user 1	0.788 ± 0.046	—	—	—	—	—	—	—
user 2	0.974 ± 0.008	0.960 ± 0.015	—	—	—	—	—	—
user 3	0.963 ± 0.019	0.977 ± 0.015	0.985 ± 0.008	—	—	—	—	—
user 4	0.943 ± 0.019	0.925 ± 0.013	0.924 ± 0.019	0.961 ± 0.009	—	—	—	—
user 5	0.940 ± 0.012	0.914 ± 0.025	0.957 ± 0.014	0.970 ± 0.015	0.923 ± 0.018	—	—	—
user 6	0.788 ± 0.016	0.838 ± 0.025	0.942 ± 0.007	0.967 ± 0.009	0.922 ± 0.007	0.744 ± 0.031	—	—
user 7	0.973 ± 0.014	0.936 ± 0.014	0.972 ± 0.013	0.973 ± 0.009	0.981 ± 0.004	0.792 ± 0.051	0.936 ± 0.009	—
user 8	0.982 ± 0.009	0.959 ± 0.011	0.982 ± 0.006	0.966 ± 0.008	0.985 ± 0.008	0.805 ± 0.038	0.974 ± 0.005	0.853 ± 0.019