Contrasting Sequence Groups by Emerging Sequences

Kang Deng and Osmar R. Zaïane

Department of Computing Science, University of Alberta
Edmonton, Alberta, T6G 2E8
{kdeng2,zaiane}@cs.ualberta.ca

Abstract. Group comparison per se is a fundamental task in many scientific endeavours but is also the basis of any classifier. Contrast sets and emerging patterns contrast between groups of categorical data. Comparing groups of sequence data is a relevant task in many applications. 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. 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 classifier for sequence data based on Emerging Sequences. Evaluating against two learning algorithms based on frequent subsequences and exact matching subsequences, the experiments on two datasets show that our similar ESs-based classification model outperforms the baseline approaches by up to 20% in prediction accuracy.

Key words: Emerging Sequences, Classification, Sequence Similarity

1 Introduction

Any science inevitably calls for comparison. Group comparison has always been a scientific endeavour in Statistics [20] and since the early days of Data Mining such as discriminant rule discovery [12]. It also is the basis of any classifier in Machine Learning. Contrast sets [3] and emerging patterns [7] contrast between groups of categorical data. 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.

To contrast groups of sequences, one fundamental question is: "How do several sequence classes differ?" In categorical data, discriminative patterns, typically a conjunction 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 considers the order of items in the sequence data. We opt to use subsequences as discriminative patterns to contrast sequence groups. Discriminative subsequences are helpful in classification as well. Borrowing an example from Ji, Bailey and Dong, 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 [14]. Biblical scholars might be interested in those subsequences and regard them as fingerprints associated with the Book of Revelation.

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 [27]. As a more complex pattern, the mining of 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.

In this paper, 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. Then, a similar 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 similar 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 two other techniques, one based on frequent subsequences and another doing exact match, to illustrate the discriminative power of ESs and the performance of the sliding window matching mechanism. The experiments show that our similar ES-based classification model outperforms the other two approaches by up to 20% in prediction accuracy. 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.

In the next section, we introduce some terminologies and define the problem. In Section 3, we describe the sequence mining algorithm and the feature selection strategy. Section 4 presents the classification based on ESs. We present the prediction performance of our proposed approach in Section 5. Related work is discussed in Section 6. Finally, Section 7 presents our conclusions.

2 Preliminaries and Problem Definition

In this section, we first explain some notations used throughout this paper. 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 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 (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.*

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 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 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 sequences contain α.

The notion of Emerging Sequences (ESs) was introduced by Zaïane et al. [28], here we generalize this notion and define:

Definition 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 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 $support(\alpha, \mathcal{D}_{pos}) - support(\alpha, \mathcal{D}_{neg}) = 100\% - 50\% > \delta$.

Table 1. A sequence dataset example.

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

Definition 5 (Edit Distance). *Edit Distance 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 a 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.

3 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 [17]. In their approach, however, they did not consider any gap constraint and apply exact matches only. In this section, we explain how we first preprocess the datasets and extract the ESs candidates; then implement a dynamic feature selection to mine the most discriminative subsequences.

3.1 ES candidates Mining

To find the Emerging Sequence candidates, the following domain-and-classifier-independent heuristics are useful for selecting sequences to serve as features [17]:

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

In other words, the ES 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 $support(\alpha, \mathcal{D}_{neg})$, needs to meet the following conditions:

$$support(\alpha, \mathcal{D}_{pos}) > \theta \tag{1}$$

$$support(\alpha, \mathcal{D}_{neg}) \leq \theta \tag{2}$$

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

As sequence mining is well developed, many existing algorithms, such as GSP [26], SPADE [29], PrefixSpan [23], and SPAM [2] can extract frequent subsequences easily. However, there are two problems by extracting ESs 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 [7], which means the minimum support offers very weak pruning power on the large search spaces [15]. 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. Hence, we implement a Generalized Suffix Tree (GST) [11] based algorithm to extract ES candidates.

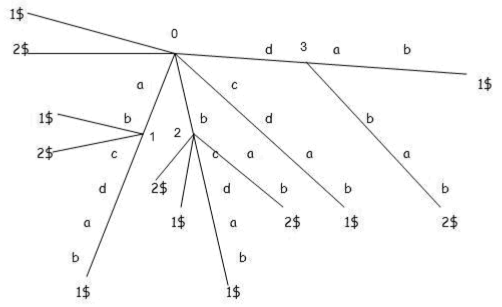


Fig. 1. An example of the Generalized Suffix Tree (GST). This GST is built by two sequences $s_1 = \langle abcdab \rangle$ and $s_2 = \langle dbab \rangle$. It has 4 internal nodes, and 0 is the root.

Figure 1 shows an example of a GST built by two sequences. 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\$. Since the supports of subsequences are stored in the internal nodes of GST, ES candidates can be extracted efficiently.

The advantage of the suffix tree-based framework is that ES candidates 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. 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 4.2.

3.2 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 [8] [18]. Therefore, we believe the most Emerging Sequences can help to contrast sequence groups as well.

To evaluate the discriminative power of subsequences, a similar mechanism with Contrast Sets [3] 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.

The selected features should be representative enough so that every original sequence can be covered - i.e. all sequences should be expressed by the selected features. To avoid numerous emerging sequences, a dynamic feature selection strategy is adopted [13]. 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.

The dynamic feature selection algorithm is presented in Algorithm 1. 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 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.

4 Transformation and Classification

After preprocessing and feature selection, ESs are extracted to contrast sequence groups. In this section, the sequence datasets are transformed to transactional datasets in order to be in a suitable form for learning algorithms. Then, a classification algorithm trained by ESs is proposed. The transactions are simple sets

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

Algorithm 1: Dynamic Feature Selection.

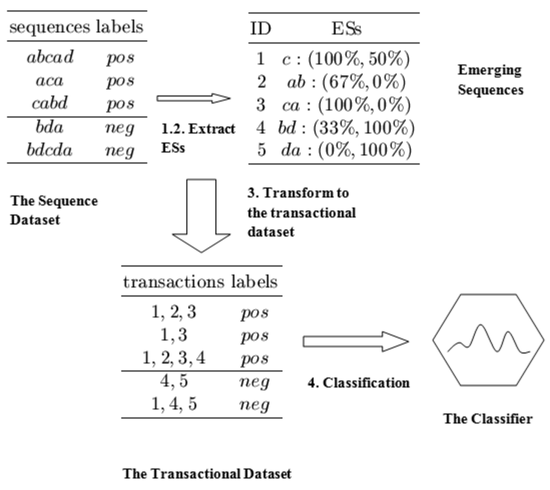
of tokens representing ESs. Each ES is represented by a token (i.e. a simple ID) used within transactions (See Fig 2).

4.1 Transformation to transactional datasets

To transform a sequence using the Emerging Sequence set representation, we implement a sliding window matching mechanism to consider similar subsequences. 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; no exact matches but $\langle bcd \rangle$ could be similar to $\langle bad \rangle$ (see below).

To compare the emerging sequence and the extracted subsequences, 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.



sequences	labels
$abcad$	pos
aca	pos
$cabd$	pos
bda	neg
$bdcda$	neg

ID	ESs
1	c : (100%, 50%)
2	ab : (67%, 0%)
3	ca : (100%, 0%)
4	bd : (33%, 100%)
5	da : (0%, 100%)

transactions	labels
1, 2, 3	pos
1, 3	pos
1, 2, 3, 4	pos
4, 5	neg
1, 4, 5	neg

Fig. 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.

The sliding window mechanism allows us to consider similar matches instead of only exact matches. This strategy improves the prediction accuracy.

4.2 Classification

On the classification stage, we implement a Naïve Bayes (NB) classifier based on Emerging Sequences. Trained by representative features, NB outperforms other state-of-art learning algorithms. In [19], Li et al. compared the performances of crf-NB with SVM [6], C4.5 [24], Bagging [4] and Boosting [10], and found crf-NB had the highest prediction accuracy.

A Naïve Bayes classifier [16] 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 $false$. When C is the class set, according to the Bayes rules, the probability that sequence S is in the class c is:

$$p(c|S) = \frac{p(S|c)p(c)}{p(S)} \tag{3}$$

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:

$$p(S|c) = \prod_i p(seq_i = v_i|c) \tag{4}$$

Therefore, the class label predicted by Naïve Bayes is:

$$predict(S) = arg\ max_{c \in C} p(c) \times \prod_i p(seq_i = v_i|c) \tag{5}$$

In [25], 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.

$$predict'(S) = arg\ max_{c \in C} p(c) \times \prod_i p(es_i = v_i|c) \tag{6}$$

Equation 6 is used to predict labels, where $\{es_1, es_2, \ldots, es_m\}$ is the set of Emerging Sequences.

5 Experimental Results

To evaluate the performance of our proposed classification model, we test the classifier on two types of datasets. The results are presented in this section.

5.1 Evaluation Methodology

Our proposed similar Emerging Sequence-based algorithm (Similar ES) can be divided into four stages (See Figure 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. Emerging Sequences are selected, so each sequence can be covered by the top-m ESs.
3. Transform the sequence datasets to transactional datasets, and subsequences that are similar with ESs are considered as well.
4. Train es-NB classifier.

For comparison, we design two other models, one based on frequencies, where frequent subsequences in the positive class are considered discriminant, and one identical to our approach but doing exact matches (Exact ES):

– 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; and in Stage 3, only exact matching subsequences are considered in transformation.
– exact ESs-based Algorithm: Stages 1 and 2 are the same with our Similar ESs-based algorithm, while only exact matching subsequences are considered in Stage 3.

The motivation for the frequency-based algorithm is that if we rank subsequences according to frequency, those discriminative ones usually have high ranks [22]. We can evaluate the effect of ESs according to the comparison between this approach and the exact ESs-based Algorithm. Exact ESs-based Algorithm is used to test the performance of the sliding window mechanism. For fair comparison, we choose the same parameters when applicable, e.g. the parameter m in Section 3.2 is set to 2 in all experiments.

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 in Table 2:

Table 2. Precision and Recall.

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

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

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.

5.2 Comparisons on Two Types of Datasets

The first type of datasets we use is the UNIX user commands dataset from UCI Machine Learning Repository [1]. 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 number of each sequence group is 1011, and the average length of sequences is 27. The size of the alphabet is 2345. We believe different users have discriminative habits when typing commands. In each experiment, two users' commands are chosen, and the F-measures and standard deviations are presented in Row 1-5 of Table 3. The minimum support θ is set to 0.01, and the maximum difference γ is set to 0.1 for this dataset.

The second dataset is the epitope data, which are short linear peptides (amino acid sequences) generated by cleavage of antigenic proteins [9]. 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 number of each sequence group is 363, and the average length of sequences is 13. The size of the alphabet is 20. To contrast the binding and non-binding peptides, we perform the test on six groups of the epitope data, and the results are presented in Row 6-11 of Table 3. The minimum support θ is set to 0.05, and the maximum difference γ is set to 0.2.

Table 3. Classification performances of three algorithms.

Datasets	Frequency-based	exact ESs-based	similar ESs-based
user 0 and 3	0.891992 ± 0.0280118	0.953464 ± 0.0121005	0.962192 ± 0.00864717
user 0 and 5	0.869967 ± 0.0250203	0.939128 ± 0.0107074	0.940028 ± 0.0118862
user 2 and 7	0.94854 ± 0.0110985	0.969787 ± 0.00733708	0.969818 ± 0.00861439
user 7 and 8	0.818205 ± 0.0154953	0.852122 ± 0.0180784	0.853639 ± 0.0233047
user 2 and 3	0.965973 ± 0.0225253	0.984494 ± 0.00798139	0.984516 ± 0.00600006
I-Ek	0.760296 ± 0.0299356	0.859395 ± 0.0237948	0.862556 ± 0.023571
HLA-DR1	0.63103 ± 0.0218163	0.72197 ± 0.0382523	0.741143 ± 0.0228598
HLA-DQ2	0.661909 ± 0.0620348	0.811726 ± 0.0936836	0.864592 ± 0.0300073
HLA-DQ4	0.712941 ± 0.0581559	0.789487 ± 0.0922259	0.821227 ± 0.055203
HLA-DR3	0.607697 ± 0.051646	0.757886 ± 0.0329568	0.777372 ± 0.0662611
HLA-DR7	0.696771 ± 0.0255892	0.770727 ± 0.0323399	0.790606 ± 0.035915

From Table 3, we observe that, our proposed similar ESs-based algorithm achieves satisfactory accuracies, comparing with the other two simpler approaches. By comparing the first two approaches (frequent subsequences versus emerging sequences), we find that Emerging Sequences play a significant role in classification: the F-measures are improved by up to 15%. The sliding window mechanism enhances the classification as well: 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 second and third algorithms have similar F-measures. The reason for that is that users 2 and 3 have one length-1 ES 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.

A problem we notice is that, the performances on the UNIX command dataset is much better than those on the epitope dataset. One explanation is that, the epitope dataset is already preprocessed by removing short, unnatural, and duplicated peptides, while the frequencies of peptides are important for our algorithm. Therefore, our preprocessing-embedded model works better on raw data.

5.3 Performances of Varying Minimum Support

Our Similar ESs-based algorithm achieves very high accuracy on the UNIX user command dataset. In this sub-section, we test the performance on UNIX command dataset by varying the minimum support threshold.

Table 4. Classification performances of different minimum supports.

θ	user 0 and 3	user 7 and 8	user 2 and 7
0.01	0.962192 ± 0.00864717	0.853639 ± 0.0233047	0.969818 ± 0.00861439
0.05	0.934213 ± 0.0104158	0.84044 ± 0.0210007	0.954692 ± 0.0125252
0.09	0.920798 ± 0.0184558	0.799181 ± 0.0159464	0.923499 ± 0.0162301
0.13	0.896651 ± 0.0226247	0.727808 ± 0.013115	0.876666 ± 0.00943212
0.17	0.865009 ± 0.0182603	0.718475 ± 0.0192607	0.851387 ± 0.0126434
0.21	0.917635 ± 0.0127357	0.663929 ± 0.00603273	0.828419 ± 0.0110561
0.25	0.84761 ± 0.0222	0.629819 ± 0.0100243	0.777707 ± 0.031677
0.29	0.801563 ± 0.028182	0.573211 ± 0.0103331	0.706538 ± 0.0177003
0.33	0.785189 ± 0.0258754	0.573587 ± 0.0102116	0.703857 ± 0.0172083

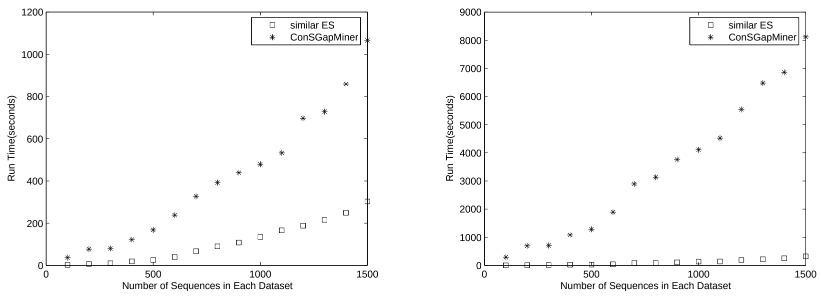
Table 4 presents the F-measures for different minimum supports on three datasets. With the increase of the minimum support, the classification accuracy degrades. The reason for that is, the minimum support threshold eliminates some emerging sequences of high discriminative power. When θ is set to 0.01, the Similar ESs-based model achieves 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.

6 Related Works

In many domains, comparing groups is significant and the fundamental purpose is to learn the differences between contrasting classes. Contrast sets (CSs) [3] and emerging patterns (EPs) [7] contrast between groups of categorical data. Some classification algorithms were developed by implementing CSs and EPs [8] [18].

Those classifiers achieve satisfactory performances because of the discriminative power of the features.

To mine contrasting subsequences, an algorithm *ConSGapMiner* was proposed by Ji et al. [14]. There are two steps of their algorithm: first, a DFS (Depth First Search) tree is built to enumerate all possible subsequence candidates; then, for each candidate, the bitset operations are applied to prune nodes which cannot fulfill the g-gap constraint. However, *ConSGapMiner* is not an efficient algorithm because of the nature of its data structure.



(a) Minimum Support $\theta = 0.2$. (b) Minimum Support $\theta = 0.05$.

Fig. 3. Scalability of Similar ES versus ConSGapMiner with increasing size of Unix command dataset.

Another related work is emerging substrings [5] proposed by Chan et al. Emerging substrings, which are motivated by EPs, occur more frequently in one class rather than in other classes. Based on the Generalized Suffix Tree (GST) [11], emerging substrings can be extracted in linear time. Comparing with *ConSGapMiner* [14], this algorithm can only mine substrings, i.e. items have to be appearing immediately next to each other in the original sequence. The problem of this algorithm is that, only exact matching substrings are considered in classification, while our approach solves this problem by the sliding window mechanism. *ConSGapMiner*, if used in Stage 1 of our framework would give a similar accuracy in general. However, the execution time would be slower as illustrated in Figure 3 due to the exponential growth of the data structure used in *ConSGapMiner*.

Generally, the contrasting sequences are always based on the frequencies of subsequences or substrings. Lin et al. implemented the Contrast Sets on time series data [21]. Instead of calculating the supports of subsequences, they compared subsequences by Euclidean Distance. This strategy cannot be implemented on sequence data directly, because the frequencies of subsequences are ignored.

A recently work, which is similar to our framework, concentrates on the software behaviours application [22]. 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.

7 Conclusion

In this paper, 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. 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 classifier for sequence data based on Emerging Sequences.

Evaluating against two learning algorithms based on frequent subsequences and exact matching subsequences, the experiments on two datasets show that our similar ESs-based classification model outperforms the baseline approaches by up to 20% in prediction accuracy. When our algorithm is trained using *jumping emerging sequences*, the best performance can be achieved.

The shown evaluation 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 paper. 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.

References

1. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.
2. 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.
3. 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.
4. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. Sarah Chan, Ben Kao, Chi Lap Yip, and Michael Tang. Mining emerging substrings. In *Database Systems for Advanced Applications (DASFAA)*, page 119, 2003.
6. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
7. 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.

8. Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Discovery Science*, pages 30–42, 1999.
9. Yasser EL-Manzalawy, Drena Dobbs, and Vasant Honavar. On evaluating mhc-ii binding peptide prediction methods. *PLoS ONE*, 3(9):e3268, 09 2008.
10. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.
11. Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, January 1997.
12. Jiawei Han and Micheline Kamber. *Data Mining, Concepts and Techniques.* Morgan Kaufmann, 2001.
13. S. Vahid Jazayeri and Osmar R. Zaïane. Plant protein localization using discriminative and frequent partition-based subsequences. In *ICDM Workshops*, pages 228–237, 2008.
14. Xiaonan Ji, James Bailey, and Guozhu Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowl. Inf. Syst.*, 11(3):259–286, 2007.
15. J. Bailey K. Ramamohanarao and G. Dong. tutorial Contrast Data Mining: Methods and Applications. International Conference on Data Mining (ICDM), 2007.
16. Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.
17. 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.
18. Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *PKDD*, pages 191–200, 2000.
19. 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.
20. Tim Funting Liao. *Statoistical Group Comparison.* Wiley's Series in probability and Statistics, 2002.
21. 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.
22. 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.
23. 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.
24. Ross J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning).* Morgan Kaufmann, January 1993.
25. Irina Rish. An empirical study of the naive bayes classifier. In *IJCAIInternational Joint Conferences on Artificial Intelligence-01 workshop on "Empirical Methods in AI"*, 2001.
26. Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.
27. Lusheng Wang, Hao Zhao, Guozhu Dong, and Jianping Li. On the complexity of finding emerging patterns. *Theor. Comput. Sci.*, 335(1):15–27, 2005.

28. Osmar R. Zaïane, Kalina Yacef, and Judy Kay. Finding top-n emerging sequences to contrast sequence sets. Technical Report TR07-03, Department of Computing Science, University of Alberta, February 2007.
29. Mohammed Javeed Zaki. Efficient enumeration of frequent sequences. In *CIKM*, pages 68–75, 1998.