Kafrelsheikh Journal of Information Sciences ISSN (Online): 2535-1478, ISSN (Print): 2537-0677 Volume 3, Issue 2, 2022, PP. 1–9

Classification Event Sequences via Compact Big Sequence

Mosab Hassaan
Faculty of Science, Benha University, Egypt
E-mail: mosab.hassaan@fsc.bu.edu.eg

Abstract: The sequence classification is considered as one of the important data mining tasks. It has a broad range of real-world applications such as bioinformatics, medicine, finance, and abnormal detection. In the literature, several algorithms have been proposed for sequence classification from different aspects. Existing algorithms can be partitioned into three types feature-based, distance-based, and model-based algorithms. In particular, the feature-based algorithms are widely applied for the sequence classification in the literature. In this paper, we propose a new event sequence classification method that based on the idea of the compact big sequence (BigSeq). Our classification method called CBigSeq. It is feature-based method where the features are the used Big Sequences in our model. To evaluate CBigSeq, we compare it with the feature-based method, SeqDT (the state-of-the-art sequence classification algorithm). Our performance study shows that CBigSeq can achieve better performance than SeqDT with respect to classification accuracy, total response time, and count of utilized patterns.

Keywords: Event Sequences, Sequence Classification, Big Sequence, Classification Accuracy

1. INTRODUCTION

The sequence classification is considered an essential data mining task. The sequence classification problem is defined as learning a sequence model to get a class label for the new sequence [1]. Sequence classification has many applications such as bioinformatics, health sciences, medicine, finance, and abnormal detection. For example, in finance, we classify sequence data in a bank to combat money laundering [2]. In bioinformatics, protein sequences are added in large repositories daily. To classify these sequences, we search for sequences that have a similar function. When novel sequence added to these repositories, we compared it with current sequences to predict the category of the novel sequence. In information retrieval, we want to classify the documents into distinct topic categories [3].

There are many challenges should be copied to develop a new algorithms for sequence classification. Now we discuss these challenges. In the first challenge, many algorithms take the input sequence data as a vector of features. Unfortunately, non explicit features in sequence data are existed. In the second challenge, the feature selection step is not trivial task. Finally, in the third challenge, due to the lack of explicit features, we cannot construct an interpretable models.

In the literature, several algorithms have been proposed for sequence classification to address the previous challenges. Existing algorithms can be partitioned into three types (feature-based, distance-based, and model-based algorithms). In this paper, we focus on the feature-based algorithms. In feature-based algorithms, the input data sequence is represented by a vector of features. Thereafter, the classification methods such as Naive Bayes, k-nearest neighbours, decision trees can be used. More details about the feature-based algorithms are listed in Section 3.

In this paper, we propose a novel algorithm called CBigSeq for event sequence Classification that based on our big sequence method (BigSeq) [4]. First, to construct our model, we divide the training sequences into subsets based on the class label. In other words, if we have *m* distinct class label then we divide the training sequences into m subsets. Thereafter, we construct BigSeq for each subset of sequences. Now, we have *m* big sequences (BigSeqs) which will represent our model. Finally, we classify the new coming sequence based on the current *m* BigSeqs. Our

experiments on three real datasets show that CBigSeq has the best performance in most cases compared to the state-of-the-art algorithm, SeqDT [5], in terms of classification accuracy, total response time, and count of utilized patterns.

Organization. This paper is organized as follows. Section 2 reports the preliminary concepts. Section 3 presents the related work. Section 4 discusses the proposed algorithm. Section 5 lists the experimental results. Finally, Section 6 concludes the paper.

2. PRELIMINARY CONCEPTS

Let σ be a set of n distinct events. Event sequence $V = < v1, v2, ..., vh >$ over σ is ordered list such that $v_i \in \sigma$. Event sequence $U = \{u_1, u_2, ..., u_g\}$ is subsequence of the event sequence V if there are g integers $(j_1, j_2,..., j_g)$ such that $1 \leq j_1 < j_2 < ... < j_g \leq h$ and $u_1 = v_{j1}$, $u2 = v_{j2}$,... ,$u_g = v_{jg}$. Event sequence with length h is called an h-sequence. Let $C = \{c_1, c_2, ..., c_m\}$ be the set of m distinct classes. A labelled sequence database D is a set of rows. Each row contains a sequence s and its class label c_i. As an example, let we have Table 1. This table contains the labelled event sequence database D with $C = \{c_1, c_2\}$. The sequence $S_4 = CBC$ has class label c_1 and it is subsequence of the sequence $S_2 = ACBC$ ($S_4 \sqsubseteq S_2$). Also we can said S_2 is supersequence of S_4. The set of all sequences in D which have the same class label c_k is denoted by D_k. As an example, we have $D_1 = \{S_1, S_2, S_3, S_4\}$ and $D_2 = \{S_5, S_6, S_7, S_8\}$.

Table 1. Labeled Event Sequence Database, *D*

Sid	Sequence	Class
S_1	$ABCC$	c_1
S_2	$ACBC$	c_1
S_3	CBC	c_1
S_4	BCC	c_1
S_5	$ABCB$	c_2
S_6	$BCBA$	c_2
S_7	ABB	c_2
S_8	BCB	c_2

For sequence classification, D is divided into two subsets. The first set is the training dataset, D_{train}, to build model (Table 2) and the second set is the test dataset, D_{test}, to test the model in terms of accuracy (Table 3). The set of all sequences in D_{train} which have the same class label c_k is denoted by D^k_{train}. As an example, we have $D^1_{train} = \{S_1, S_2, S_3\}$ and $D^2_{train} = \{S_5, S_6, S_7\}$.

Table 2. The Training Dataset of *D*, D_{train}

Sid	Sequence	Class
S_1	$ABCC$	c_1
S_2	$ACBC$	c_1
S_3	CBC	c_1
S_5	$ABCB$	c_2
S_6	$BCBA$	c_2
S_7	ABB	c_2

Table 3. The Test Dataset of *D*, D_{test}

Sid	Sequence	Class
S_4	BCC	c_1
S_8	BCB	c_2

Problem Definition: Given the training sequence dataset D_{train} and a new sequence with unknown class label S_{new}. The objective is to construct an efficient classifier (model) to classify S_{new} in which the classification accuracy is high, the total response time is responsible, and the count of utilized patterns is small as possible.

3. RELATED WORK

The sequence classification is considered an essential data mining task. In the literature, several algorithms have been proposed for this problem. Recall, based on the used strategy for designing the model (classifier), current algorithms of sequence classification can be partitioned into three types as follows. The first type is the feature-based algorithms, the second type is the distance-based algorithms, and finally, the third type is the model-based algorithms. In this paper, we focus in the feature-based algorithms. More details about them are listed as follows.

In feature-based algorithms, each sequence is converted to a vector therefore the current algorithms of vector data classification methods will be used such as Naive Bayes, k-nearest neighbours, decision trees, hidden Markov models, support vector machines, etc. [6].

To classify sequences, the authors of [7] used the sequential patterns with Naive Bayes classification. They proposed the FeatureMine algorithm (called BayesFM) to efficiently construct the features from the sequence dataset. The main disadvantage of this method, in large feature space, the algorithm could not effectively construct the discriminative features. The Classify-By-Sequence algorithm (CBS) [8] was proposed to classify large sequences. This algorithm mines classifiable sequential patterns (CSPs) from the sequences. Thereafter, based on a scoring function, it assign a score to the new object for each class. Authors of [9] proposed a direct sequential pattern algorithm (BIDE-Discriminative) which uses class information for direct mining of predictive sequential patterns.

SeqDT [5] is a new feature-based classification algorithm which is a tree-based sequence. This algorithm. It constructs a decision tree over the feature space of all subsequences on the training set. In this algorithm, two data mining tasks are combined namely, sequential pattern mining and decision tree. In other words, it uses the methods of sequential pattern mining to mine features for the tree-based sequence classification. There are many disadvantages of SeqDT as follows. Firstly, SeqDT consumes a lot of time because it is two-phases procedure in which the feature construction and decision tree construction are two consequent phases. Secondly, many parameters are determined as input for SeqDT such as g (gap constraint), d (tree depth), maxL (maximum pattern length), minS (minimum value of decreased impurity generated by segmentation), threshold ϵ (maximum value of Gini index in each node), and minN (minimum number of sequences in each node). More details about these parameters are listed in [5]. Unfortunately, setting the values of these parameters is not simple task since each application domain may need a specific setting. In contrast, as we will see our proposed efficient method is a free-parameter method.

4. EVENT SEQUENCE CLASSIFICATION USING BIGSEQ

First, we discuss the construction of Big Sequence (BigSeq) as follows. The BigSeq method was proposed in our previous paper [4] to summarize the event sequence database. The main idea of BigSeq is to merge all sequences into compact big sequence. The construction of BigSeq is based on two definitions (the longest common subsequence and the novel compatible event set). We review the construction of BigSeq using the following running example.

Example 4.1. Given the labeled event sequence dataset $D_{train} = D^1_{train} \cup D^2_{train}$ in Table 2. To construct BigSeq according to $D^1_{train} = \{S_1, S_2, S_3\}$ (Recall, all sequences in D^i_{train} have class label equals to c_i). First, we randomly select any sequence $S \in D^1_{train}$ as initial value of BigSeq. Suppose $S = S_1$ then the initial BigSeq is $S_1 = ABCC$. After that, for each remaining sequence $S' \in D^1_{train} - S_1$, we compute $LCS(S', BigSeq)$ and store every remaining event e where $e \in S'$ and $e \notin LCS(S', BigSeq)$. For instance, the longest common subsequences $LCS(S_2, BigSeq)$ and $LCS(S_3, BigSeq)$ are ACC and CC respectively. Also, for the sequences S_2 and S_3, the set of remaining events are $\{B\}$ and $\{B\}$ respectively. Based on the remaining events of all remaining sequences, we compute the compatible event sets (*core*). Here, in this example, we have only one compatible event set $core_1$, i.e. $core = \{core_1\} = \{\{B,B\}\}$. Thereafter, we compute the representative event, e_{rep}, for $core_1$ and insert it in BigSeq according to the expected range of positions. Here, e_{rep} is the event B. As result, the final Big Sequence of D^1_{train}, $BigSeq(D^1_{train})$, is $ABCBC$. With the same steps, we can find the final Big Sequence of D^2_{train}, $BigSeq(D^2_{train})$. We have $BigSeq(D^2_{train}) = ABCBA$. For more details about the compatible event sets, the expected range of positions, and the construction of BigSeq, please read our previous paper [4].

Before the discussion of sequence classification using BigSeq, we illustrate how to query BigSeq.

4.1. Querying BigSeq

To check that a given subsequence s belongs to a sequence S ($S \in D$) or not, we can do this by checking the Big sequence of D ($BigSeq(D)$) againt the subsequence s with respect to the next two theorems.

Theorem 4.1 Given event sequence database D, its Big Sequence $BigSeq(D)$, and subsequence s. If $s \not\subseteq BigSeq(D)$ then $s \not\subseteq S \; \forall \; S \in D$.

Example 4.2 Given the training dataset D^1_{train} in Table 2, its Big Sequence $BigSeq(D^1_{train})$, $= ABCBC$, and the

subsequence $s = BBB$. Based on Theorem 4.1, we have $s = BBB \not\sqsubseteq ABCBC = BigSeq(D^1_{train})$, then $s \not\sqsubseteq S \; \forall \; S \in D^1_{train}$.

Theorem 4.2 Given event sequence database D, its Big Sequence $BigSeq(D)$, and subsequence s. If $s \sqsubseteq BigSeq(D)$ then either $s \sqsubseteq S$ where $S \in D$ or $s \not\sqsubseteq S \; \forall \; S \in D$.

Example 4.3 Given the training dataset D^1_{train} in Table 2, its Big Sequence $BigSeq(D^1_{train})$, $= ABCBC$, and the subsequence $s = ABC$. Based on Theorem 4.2, we have $s = ABC \sqsubseteq ABCBC = BigSeq(D^1_{train})$ *with* $s = ABC \sqsubseteq ABC = S_1$ and $s = ABC \sqsubseteq ACBC = S_2$. In other hand, if we have another subsequence $s' = ABB \sqsubseteq ABCBC = BigSeq(D^1_{train})$ but $s' \not\sqsubseteq S \; \forall \; S \in D^1_{train}$.

If the subsequence $s \not\sqsubseteq BigSeq(D)$ then we sure that $s \not\sqsubseteq S$ for each $S \in D$ (Theorem 4.1). In the other hand, if $s \sqsubseteq BigSeq(D)$ then either $s \sqsubseteq S$ where $S \in D$ or $s \not\sqsubseteq S \; \forall \; S \in D$ (Theorem 4.2). In this case to prove that $s \sqsubseteq S$ where $S \in D$ or not, we can check the bit-vectors of s event's in $BigSeq(D)$ by performing the Anding operation on the corresponding bits in these bit-vectors. If the result contains one bit equals to one then $s \sqsubseteq S$ where $S \in D$ otherwise $s \not\sqsubseteq S \; \forall \; S \in D$. See next example for more details.

Example 4.4 Given the training dataset D^1_{train} in Table 2, its Big Sequence $BigSeq(D^1_{train})$, $= ABCBC$, and the two subsequences $s = ABC$ and $s' = ABB$.

For the subsequence $s = ABC$, we have $s \sqsubseteq BigSeq(D^1_{train})$ and it has three occurrences in $BigSeq(D^1_{train})$ with positions {1, 2, 3}, {1, 2, 5}, and {1, 4, 5}. Next we separately discuss each occurrence.

1- For the first occurrence, we perform the Anding operation on bit-vectors that fall on positions 1, 2, and 3 in $BigSeq(D^1_{train})$. The output is 00000011 & 00000001 & 00000111 = 00000001. The output bit-vector has one at position 1 (i.e. it is not NULL). This means that s is contained in the sequence, S_1 ($s \sqsubseteq S_1$).

2- For the second occurrence, we perform the Anding operation on bit-vectors that fall on positions 1, 2, and 5 in $BigSeq(D^1_{train})$. The output is 00000011 & 00000001 & 00000111 = 00000001. The output bit-vector has one at position 1. This means that s is contained in the sequence, S_1 ($s \sqsubseteq S_1$). From the first and the second occurrences of s in $BigSeq(D^1_{train})$, we have also two occurrences of s in the sequence S_1 which are {1, 2, 3} and {1, 2, 4}.

3- For the third occurrence, we perform the Anding operation on bit-vectors that fall on positions 1, 4, and 5 in $BigSeq(D^1_{train})$. The output is 00000011 & 00000110 & 00000111 = 00000010. The output bit-vector has one at position 2. This means that s is contained in the sequence, S_2 ($s \sqsubseteq S_2$).

Note that from the first and the second occurrences of s in $BigSeq(D^1_{train})$, we have $s \sqsubseteq S_1$. Therefore, *s has two occurrences in S_1. But, from the third occurrence, s has only one occurrence in S_2.*

Algorithm 1: Querying $BigSeq(D)$

Input: The corresponding $BigSeq(D)$ with bit vectors sets of D and the query subsequence s_q.
Output: $W \subseteq D$ such that $s_q \sqsubseteq S$ for each $S \in W$ otherwise NULL.

1. **if** $s_q \not\sqsubseteq BigSeq(D)$ **then** // Theorem 4.1
2. **return** NULL
3. **end if**
4. **else** // Theorem 4.2
5. **Find** the set of all occurrences of s_q in $BigSeq(D)$, $O(s_q)$.
6. **for** each occurrence $o \in O(s_q)$ **do**
7. Suppose $p_1, p_2, ..., p_{|s_q|}$ be the positions of o in $BigSeq(D)$.
8. Compute $B\_output = B(ep_1)$ & $B(ep_2)$ & ... & $B(ep_{|s_q|})$ // ep_i is the event e at position p_i in $BigSeq(D)$.
9. **for** each $B_i \in B\_output$ **do**
10. **for** j = 1 to 8
11. **if** $B_i[\,j\,] = 1$
12. $W = W \cup S_{8\times(i-1)+j}$
13. **end if**
14. **end for**
15. **end for**
16. **end for**
17. **return** W

For the subsequence $s' = ABB$, we have $s' \sqsubseteq BigSeq(D^1_{train})$ and s' has only one occurrence in $BigSeq(D^1_{train})$ with position {1, 2, 4}. Therefore, we perform the Anding operation on bit-vectors that fall on positions 1, 2, and 4 in $BigSeq(D^1_{train})$. The output is 00000011 & 00000001 & 00000110 = 00000000.

The output bit-vector has no ones (i.e. it is NULL). This means that $s' \not\sqsubseteq S \; \forall \; S \in D$ and at the same time, we have $s' \sqsubseteq BigSeq(D^1_{train})$.

Algorithm 1 outlines the process of querying $BigSeq(D)$. The input of this algorithm is the big sequence of D, $BigSeq(D)$, and the query subsequence s_q while the output is the set of sequences in D that contains s_q denoted as W. Lines 1-2 apply Theorem 4.1 that is if $s_q \not\sqsubseteq BigSeq(D)$ then returns NULL. Lines 4-16 apply Theorem 4.2 as follows. First, we find the set of all occurrences of s_q in $BigSeq(D)$. denoted as $O(s_q)$. Second, for each occurrence $o = \{p_1, p_2, ..., p_{|s_q|}\}$ in $O(s_q)$, we Anding the bit vectors of the events ep_1 , ep_2 , ..., $ep_{|s_q|}$ in $BigSeq(D)$, where ep_i is the event e at position p_i. We denote the output of the Anding operation as B_output. Note that B_output is a set of bit-vectors, B_output = $\{B_1, B_2, ..., B_m\}$, where the length of B_i is 8 bits and $m = D/8$. Finally, for each block $B_i \in B$_output, if there exist any bit in any B_i equals to 1, we add its corresponding sequence to W.

4.2. Classification Method

In this section, we discuss the event sequence classification using BigSeq method. Recall, for event sequence classification, D is divided into two subsets. The first set is the training set D_{train} (to build the model) and the second set is the test set D_{test} (to test the model in terms of accuracy). To build the proposed model using BigSeq, we divide D_{train} into m subsets where m is the number of classes in the dataset ($m = |C|$) such that each subset will have the same class label. For each subset D^i_{train} (Note that each sequence $S \in D^i_{train}$ has class label ci such that $i = 1, ..., m$), we construct its Big Sequence, $BigSeq(D^i_{train})$. As a result, our model will be represented by m BigSeqs. Based on our model, to classify a new sequence S_{new} with unknown class label, we have two cases as follows. In the first case, if S_{new} is not subsequence of $BigSeq(D^i_{train})$. for all $i = 1, ..., m$ (Theorem 4.1), we compute $LCS(S_{new}, BigSeq(D^i_{train}))$ for each $BigSeq(D^i_{train})$ in our model where $i = 1, ..., m$. Thereafter, we search for the maximum length of LCS among all BigSeqs. Suppose $LCS(S_{new}, BigSeq(D^k_{train}))$ has the maximum value then we label S_{new} by c_k. In the second case, if there exist index k such that S_{new} is subsequence of $BigSeq(D^k_{train})$ then we simply label S_{new} by c_k and we do not search for the occurrences of S_{new} in $BigSeq(D^k_{train})$ to save a time. In Experimental Evaluation section, we test the performance of our classification method on real datasets with respect to classification accuracy, total response time, and count of utilized patterns. Algorithm 2 outlines the construction of our model and the classification of a new sequence S_{new} with unknown class label.

Algorithm 2: Model Construction & Sequence Classification

Input: The training dataset D_{train} and the new sequence S_{new} with unknow class label.
Output: The class label of S_{new}

1. **Divide** D_{train} into m subsets based on the class label c_i ($D_{train} = \bigcup_{i=1}^{m} D^i_{train}$)
2. **for** each $i = 1$ to m **do**
3. **Construct** the Big Sequence $BigSeq(D^i_{train})$
4. $Arr\_W[i]$ = Querying($BigSeq(D^i_{train})$, S_{new}) // Call Algorithm 1
5. **end for**
6. **if** $Arr\_W[i]$ = NULL $\forall$ $i = 1, ..., m$ // Theorem 4.1
7. **for** each i = 1 to m do
8. **Compute** $LCS^i = LCS(S_{new}, BigSeq(D^i_{train}))$
9. **end for**
10. $k = \text{argmax}_{i=1...m} LCS^i$
11. **Label** S_{new} by c_k
12. **end if**
13. **else**
14. **Find** the index k where $Arr\_W[k] \neq$ NULL
15. **Label** the sequence S_{new} by c_k

5 EXPERIMENTAL EVALUATION

This section shows the results of experiments on three real datasets. We compare our proposed algorithm, CBigSeq, with SeqDT algorithm [5] (the state-of-the-art algorithm). Note that, we used SeqDT in our experiments beacuse it outperforms the four algorithms SEQL [10], MiSeRe [11], Sqn2Vec [12], and SCIP [13].

CBigSeq was implemented in C++ with STL library and compiled with GNU GCC while the code of SeqDT was downloaded from https://github.com/ZiyaoWu/SeqDT. Experiments were run on laptop (Intel i3 2.4 GHz and 8G memory) running Linux. The three datasets are reported in the next section.

Table 4. Summary Statistics of the Real Datasets used in Our Experiments

Dataset	$\|\mathcal{D}\|$	$\|E\|$	min_L	max_L	avg_L	$\#classes$
Question	1731	3612	4	29	10.17	2
Poineer	160	178	4	100	40.14	3
Gene	2942	5	41	216	86.53	2

5.1 Datasets

To evaluate our proposed algorithm, CBigSeq, we used three real datasets namely, Question [14], Poineer [9], and Gene [15]. These datasets are summarized in Table 4, where |*D*| is the number of event sequences, |*E*| represents the count of the events, min_*L*, max_*L*, avg_*L*, and #classes represent the minimum length, the maximum length, the average length, and the number of distinct classes of the event sequences respectively. Also, Table 5 reports URLs of the used datasets.

Table 5. URLs of the Real datasets Used in Our Experiments

Dataset	URL
Question	http://cogcomp.cs.illinois.edu/Data/QA/QC/
Poineer	http://archive.ics.uci.edu/ml/datasets/Pioneer-1+Mobile+Robot+Data
Gene	http://datamining.xmu.edu.cn:8080/miRNApre-

5.2 Performance of CBigSeq against SeqDT

CBigSeq algorithm is evaluated based on the following three criteria:

1. ***Accuracy:*** To measure the accuracy of our model.
2. ***Total Response Time:*** To measure the efficiency of CBigSeq.
3. ***The Count of Utilized Patterns:*** To show the count of utilized patterns for classification.

5.2.1 Accuracy

In this section, we discuss the accuracy (%) of the two algorithms on the three datasets. For each dataset, we select a number of sequences for training and other number of sequences for testing as in Table 6.

Table 6. Train and Test Sequences for each Dataset

Dataset	Train Sequences	Test Sequences
Question	1493	238
Poineer	129	31
Gene	2354	588

Table 7 reports the accuracy (%) of the two algorithms. From this table, CBigSeq shows higher accuracy than SeqDT Question dataset whereas the opposite occurs on the Pioneer dataset. On Gene dataset, the two algorithms have the same accuracy (100%).

Table 7. Accuracy (%) of the two algorithms (CBigSeq against SeqDT)

Dataset	CBigSeq	SeqDT
Question	99.5%	96.6%
Poineer	80.6%	100%
Gene	100%	100%

Figure 1 reports the scalability test (the number of training sequences) of CBigSeq against SeqDT in terms of accuracy. This scalability test was done on the Question dataset with 238 sequences for testing. From this Figure, CBigSeq algorithm shows a better accuracy when the number of train sequences are 1000 and 1200. For example, with 1200 sequences for training, the accuracy of CBigSeq is 96.6% while the accuracy of SeqDT is 95.3%. On the other hand, SeqDT has a better accuracy when the number of train sequences are 900 and 1100. For example, with 900 sequences for training, the accuracy of SeqDT is 94.5% while the accuracy of CBigSeq is 94.1%.

Recall, many parameters are determined as input for SeqDT such as the two parameters *g* (the gap constraint) and *d* (the tree depth) whereas our proposed algorithm CBigSeq is free-parameters. Therefore, in the next experiment, we show the parameter sensitivity of SeqDT in terms of accuracy while our proposed algorithm, CBigSeq, is stable. This experiment was done on Question dataset with 1200 sequences for training and 238 sequences for testing. We selected values for *g* and *d* as follows. The values of *g* are 1, 2, 3, 4, and 5 whereas the values of *d* are 2, 5, 7, 9, and 11. In Figure 2, for gap constraint (*g*), the accuracy of CBigSeq is stable and equals to 96.6% while the accuracy of SeqDT is not stable and falls on the range from 93.69% to 95.37%. In Figure 3, for tree depth (*d*), the accuracy of CBigSeq is also stable and equals to 96.6% while the accuracy of SeqDT is not stable and falls on the range from 83.19% to 95.37%.

Figure 1. Scalability Test (Accuracy)

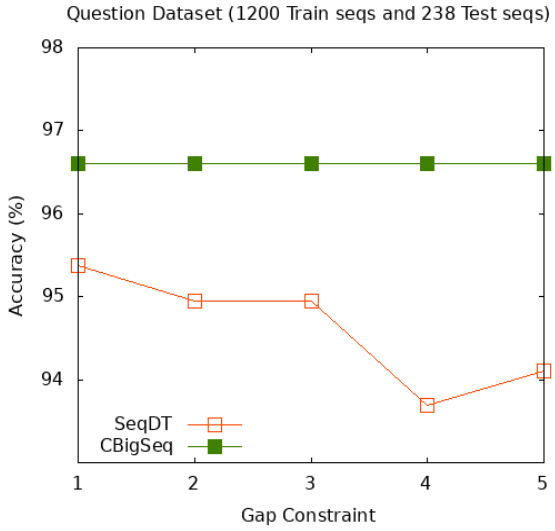

Figure 2. Gap Constraint (Accuracy)

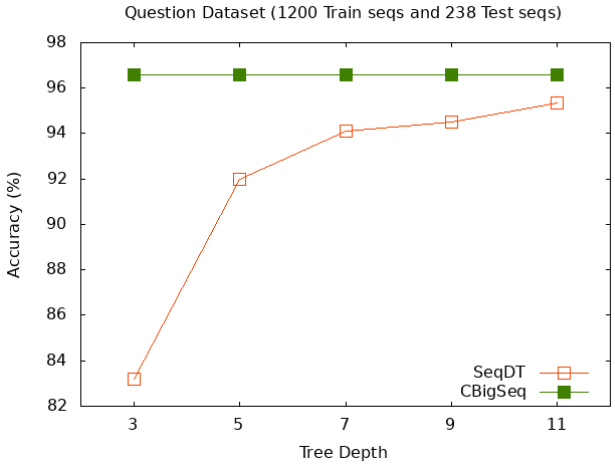

Figure 3. Tree Depth (Accuracy)

5.2.2 Total Response Time

In this experiment, the number of sequences for training and testing is the same as in the previous section (see Table 6). Table 8 reports the total response time (MSec) of the two algorithms on the three datasets. From this table, on Question dataset, CBigSeq outperforms SeqDT by three factors. On Pioneer dataset, the two algorithms have approximately the same time. While, on Gene dataset, SeqDT outperforms CBigSeq by more than two factors.

Table 8. Total Response Time (MSec) of the two algorithms (CBigSeq against SeqDT)

Dataset	CBigSeq	SeqDT
Question	343	1085
Poineer	215	223
Gene	1523	593

Figure 4 reports the scalability test (the number of training sequences) of CBigSeq against SeqDT with respect to total response time (MSec). This scalability test was done on the Question dataset with 238 sequences for testing. From this Figure, CBigSeq algorithm shows the best performance. In other words, BigSeq outperforms SeqDT by more than three factors.

Now, we test the parameter sensitivity of SeqDT in terms of total response time while the proposed algorithm, CBigSeq (free-parameters) is stable. This experiment was done on Question dataset with 1200 sequences for training and 238 sequences for testing. As in previous section, we also used the two parameters g (the gap constraint) and d (the tree depth) with the same selected values. In Figure 5, for gap constraint (g), the total response time of CBigSeq is stable and equals to 343 MSec while the total response time of SeqDT is not stable and falls on the range from 1027 MSec to 1684 MSec. In other words, BigSeq outperforms SeqDT by more than five factors. In Figure 6, for tree depth (d), the total response time of
CBigSeq is also stable and equals to 343 MSec while the total response time of SeqDT is not stable and falls on the range from 810 MSec to 1027 MSec. In other words, BigSeq outperforms SeqDT by more than three factors.

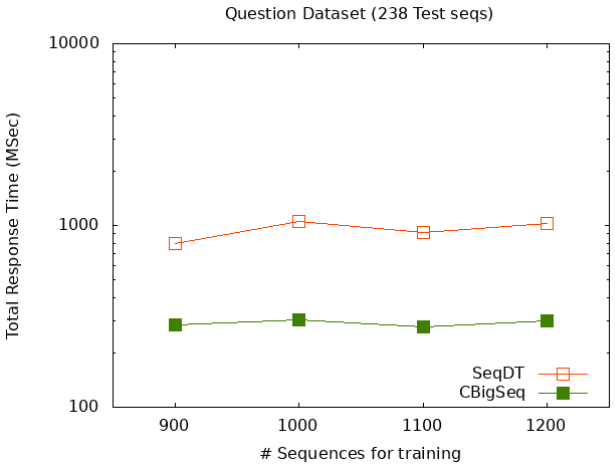

Figure 4. Scalability Test (Total Response Time)

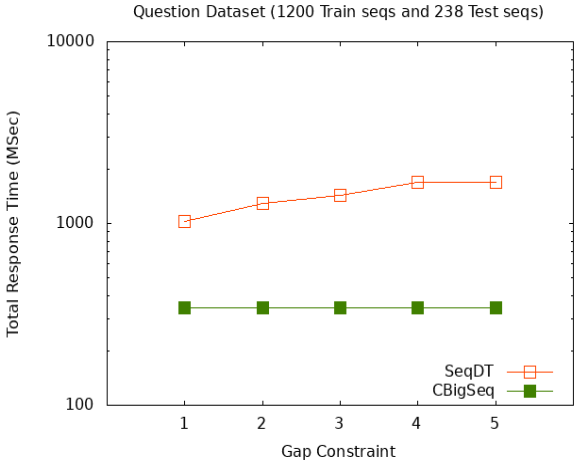

Figure 5. Gap Constraint (Total Response Time)

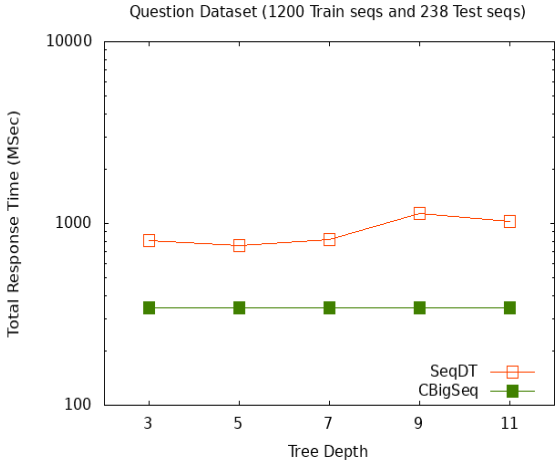

Figure 6. Tree Depth (Total Response Time)

5.2.3 The Count of Utilized Patterns

Table 9 shows the count of utilized patterns for classification by the two algorithms on the three datasets. Note that CBigSeq utilized only $|C|$ patterns i.e. $|C|$ BigSeqs (Recall, $|C|$ is the number of classes in the running dataset) for all datasets. In the Question dataset, the count of utilized patterns of CBigSeq is smaller than SeqDT. This means that our proposed method constructs an efficient model for classification with only a small number of patterns. In the other two datasets, Pionner and Gene, the two algorithms CBigSeq and SeqDT construct their models by approximately the same count of utilized patterns.

Table 9. The Count of Utilized Patterns of the two algorithms (CBigSeq against SeqDT)

Dataset	CBigSeq	SeqDT
Question	2	35
Poineer	3	3
Gene	2	1

6 CONCLUSION

In this paper, we focus on classification event sequence dataset. Existing algorithms of sequence classification are partitioned into three types (feature-based, distance-based, and model-based algorithms). Here, we proposed a feature-based algorithm called CBigSeq. CBigSeq is based on the idea of compact big sequence (BigSeq). Our model contains a set of BigSeqs such that each BigSeq in our model represents a subset of training sequences that have the same class label. Via an extensive evaluation on three real datasets, we show that CBigSeq outperforms the state-of-the-art competitor, SeqDT, in most cases in terms of classification accuracy, total response time, and count of utilized patterns.

Acknowledgments

I wish to express my deep gratitude to my mentor Prof Dr. Karam Gouda. I am very grateful to my parents, my wife, my brother, and my sisters for their continuous moral support and encouragement.

Conflicts of Interest

The author declares that I don't have any conflict of interest regarding this article.

REFERENCES

[1] Han, J., Pei, J. & Kamber M. (2011), Data mining: concepts and techniques. Elsevier.

[2] Liu, X., Zhang, P. & Zeng, D. (2008) Sequence matching for suspicious activity detection in anti-money laundering. Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics.

[3] Sebastiani, F.(2002) Machine learning in automated text categorization. ACM Comput. Surv., 34(1).

[4] Hassaan, M. (2022). Summarizing Event Sequence Database into Compact Big Sequence. International Journal of Advanced Computer Science and Applications, Vol. 13, No. 8.

[5] He, Z., Wu, Z., Xu, G., Liu, Y. & Zou Q. (2023), Decision Tree for Sequences. In IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 1, pp. 251-263.

[6] Xing, Z., Pei, J. & Keogh E. (2010) A brief survey on sequence classification. ACM SIGKDD Explorations Newsletter, vol. 12, no. 1, pp. 40–48.

[7] Lesh, N., Zaki, M. & Ogihara M. (2000). Scalable feature mining for sequential data. IEEE Intell. Syst., vol. 15, no. 2, pp. 48–56.

[8] Tseng, V. & Lee, C. (2009) Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. Expert Systems with Applications, vol. 36, no. 5, pp. 9524–9532.

[9] Fradkin, D. & M¨orchen, F. (2015) Mining sequential patterns for classification. Knowledge and Information Systems, vol. 45, no. 3, pp. 731–749.

[10] Ifrim G. & Wiuf C. (2011) Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 708–716.

[11] Egho, E., Gay, D., Boull´e, M., Voisine, N., & Cl´erot, F. (2017) A user parameter-free approach for mining robust sequential classification rules. Knowledge and Information Systems, vol. 52, pp. 53–81.

[12] Nguyen, D., Luo, W., Nguyen, T., Venkatesh, S. & Phung, D. (2018) Sqn2vec: Learning sequence representation via sequential patterns with a gap constraint. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 569–584.

[13] Zhou, C., Cule, B. & Goethals, B. (2016) Pattern based sequence classification. IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 5, pp. 1285–1298.

[14] Kim, Y. (2014) Convolutional neural networks for sentence classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751.

[15] Wei, L., Liao, M., Gao, Y., Ji, R., He, Z. & Zou, Q. (2014) Improved and promising identification of human micrornas by incorporating a high-quality negative set. IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 11, no. 1, pp. 192–201.