Time Series Classification Method Based on Longest Common Subsequence and Textual Approximation

Abdulla-Al-Maruf
Ritsumeikan University
Kusatsu, Japan

Hung-Hsuan Huang
Ritsumeikan University
Kusatsu, Japan

Kyoji Kawagoe
Ritsumeikan University
Kusatsu, Japan

***Abstract*—Many symbolic representations of time series have been proposed by researchers over past decades. However, it is still not enough to classify time series with high accuracy in such applications as ubiquitous systems or sensor systems. In this paper, we propose a new symbolic representation of time series called l-TAX to increase the accuracy of time series classification. A time series can be represented by term sequences in l-TAX. l-TAX is based on a document like symbolic representation of time series called TAX. We use longest common subsequence as our distance measure between textually approximated time series. During time series classification, consideration of symbol sequences increases the accuracy significantly. In our evaluation, we have demonstrated that l-TAX is effective for classification as well as searching time series data set.**

I. Introduction

Symbolic representation based time series similarity has drawn the attention of researchers for many years ago. The motivation for time series symbolic representation is to include the rich data structures and algorithms from text processing and bioinformatics. It also includes dimension reduction methods to deal with the scalability problem. The prime goal of all these researches are to achieve a high accuracy in time series similarity search. Time series data are available almost everywhere nowadays because of its usability as well as falling price of equipments and sensors by which time series data can be gathered easily. Thus, it is very important to achieve high accuracy in similarity search so that it can be used in mission critical applications confidently.

In time series similarity search technique the system maintains a time series database. The system searches for one or more time series in the database comparing the stored time series with a given query time series [1], [2], [3], [4], [5], [6], [7], [8]. These methods work in two folds. Firstly, time series dimension reduction is done by transforming the time series into its features in a feature space using some transformation functions. It improves the search efficiency because the number of dimensions in the feature space is usually less than the original time space. The transformation function varies with the similarity search method. Secondly, a distance definition (Euclidean distance, Manhattan distance, Dynamic Time Warping etc.) is used as the similarity measure.

The string sequence based distance is also introduced as a time series similarity metric in many researches, such as [9], [10], [11]. In these researches, the time series data are transformed to a set of text symbols. This creates the opportunity to use the text retrieval algorithms for time series. Text operations are very sensitive about assigned symbols and symbol sequences. Time point's features must be considered before assigning symbols for time series to achieve good accuracy. Consideration of symbol sequence in text retrieval system increases the potentiality of the system.

In this paper, we propose, a new and novel method of lcs based time series modeling using text document structure and its symbol sequences. The significant point of the method is to use lcs as distance definition, which is widely used in DNA and string search. However, it is difficult to apply lcs in time series because time series data are not a sequence of terms, but a sequence of values over time. It is not easy to extract terms from such time series data as in a document. We use TAX[4] to convert a time series to a new structure, like a text document. TAX extracts temporal terms (T-term) from time series data and stores them like words in text documents. Our main target application is human movement classifications. Owing to sensor technology development and dissemination, many time series of human movement are available. Especially, we are considering an automated self training system for multi-player sport activities. In this kind of applications, it is important to classify such huge amount of human movement data into meaningful categories dynamically and with high accuracy. The major contribution of this paper is that we attained the most accurate classification result over TAX as well as the existing dominant methods, by introducing the term sequence based time series approximations.

The paper is structured as follows: Section II shows the problem that current method has and how term sequence consideration increases the accuracy. Section III describes how sequences are generated from time series. Section IV shows evaluation of our method. Section V presents some related works. Finally, we conclude our paper in section VI.

II. Time Series Symbolic Representation

In this paper, we focus on time series classification. Our motivation is to increase the classification accuracy. To do so, we consider the symbol sequences in the time series symbolic representation. In the symbolic representation, time series are represented by symbols. From our point of view, symbol's order is very important to achieve high classification accuracy. We closely examine the method TAX[4], which uses time series symbolic representations and does not consider term sequences during classification. We show consideration of term sequence in TAX increases the classification accuracy significantly.

A. TAX

TAX is a novel method which uses existing document retrieval models to retrieve similar time series. It represents time series like a text document. A text document contains series of words or terms. TAX extracts terms from time series data, which is called T-terms and presents them in a document. It is a bag-of-words model based method. Bag-of-words model is widely used in natural language processing and information retrieval. In this model, a text or document is presented as an unordered collection of words without considering grammar or even word order.

TAX finds the important points from the time series and use them to construct terms. TAX considers the issue that not all points in a time series are equally important. Some time points carry special interests. Modern information retrieval systems use this type of heuristics. Different terms carry different interests to the search engines. Modern search engines are aware about the term sequences.

B. Problem of TAX

TAX does not consider term's sequence during its construction or retrieval phases although it uses text retrieval methods. For example, Figure 1 (a) shows TAX approach and (b) shows l-TAX approach. For the query, *the lazy brown dog jumped over the quick fox*, TAX matches the query terms to its bag and gets positive result because those are the same terms without order but l-TAX checks the term sequences and finds the query is not similar to its bag although the terms are same. In this case, l-TAX reduces the false positives of TAX.

So, consideration of term's sequence in TAX increases the classification accuracy. We demonstrate it with the following examples using synthetic and real time series.

C. Examples

In the following examples, we show how TAX fails in some cases and how term sequence consideration can improve the accuracy.

(1) In this example, we use synthetic time series where key points are selected uniformly. We have two time series (O_1 and O_2) and a query time series (Q) as shown in Figure 2. Our goal is to find a time series between O_1 and O_2, which is most similar to Q. From observation it is clear that $O_2 = Q$. So, TAX must find O_2 as Q's most similar time series.

We show the key-points for all the time series using circles. We define a set of T-terms, $T_{term} = \{flat, up, down\}$. For each key point, we assign a term from T_{term}. TAX uses term frequency and inverse document frequency($tf - idf$) to create document vectors for all the time series. For simplicity,

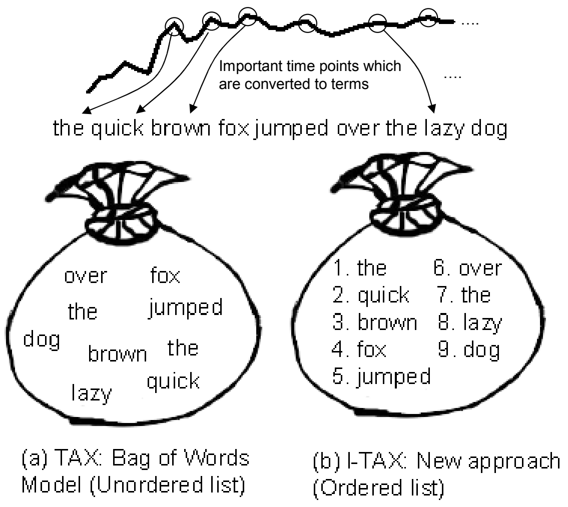
Important time points which are converted to terms
....
....
the quick brown fox jumped over the lazy dog
over
fox
the
jumped
dog
brown
the
lazy
quick
1. the
2. quick
3. brown
4. fox
5. jumped
6. over
7. the
8. lazy
9. dog
(a) TAX: Bag of Words Model (Unordered list)
(b) l-TAX: New approach (Ordered list)

Fig. 1. Difference between TAX and l-TAX.

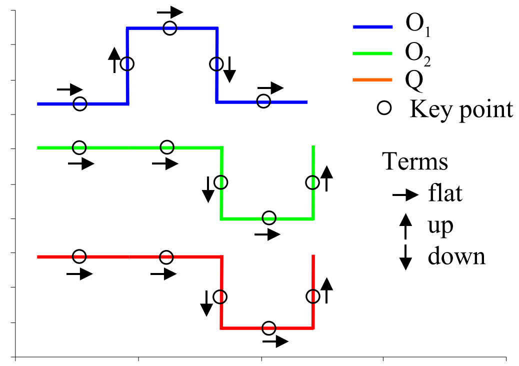

Fig. 2. Example of TAX failure

we consider $idf = 1$. Table I shows the document vectors by counting tf for O_1, O_2 and Q. It also shows the cosine similarity among the time series and the query.

TABLE I
SIMILARITY USING TAX

Time series	Term Frequency			Document vec.	Similarity with Q
	flat	up	down		
O_1	3	1	1	(3, 1, 1)	1
O_2	3	1	1	(3, 1, 1)	1
Q	3	1	1	(3, 1, 1)	

TAX finds both O_1 and O_2 are equally similar to Q. It fails to distinguish that O_1 is different from Q. It happens because both O_1 and Q contain equal numbers of same terms although the term sequences are different for them.

Consideration of term sequence solves the problem. We represent the time series using their terms in Table II. The lcs length between O_2 and Q is higher than O_1 and Q. So, the new method selects O_2 as most similar time series to Q. It successfully identifies that O_1 has a different shape than Q

although both of them have equal number of same terms.

TABLE II
SIMILARITY USING L-TAX

Time series	Term Sequence	lcs length
O_1	(flat, up, flat, down, flat)	4
O_2	(flat, flat, down, flat, up)	5
Q	(flat, flat, down, flat, up)	

(2)Figure 3 shows a more complex example than the previous one using a real time series.

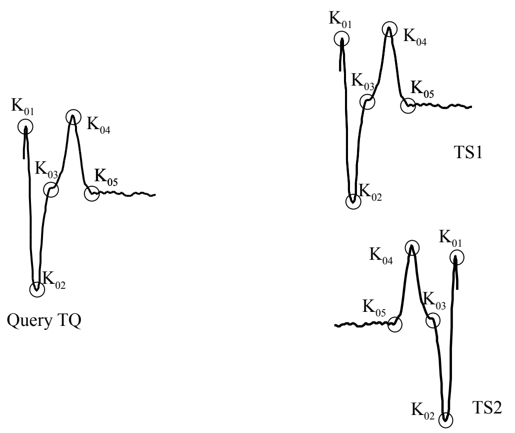

Fig. 3. Example of textual approximation(2).

In the figure, there are three time series data set like ECG data, labeled as TS1, TS2 and TQ. The first one (TS1) indicates normal ECG data, while the second time series (TS2) is just the opposite (mirror view) of TS1. As they are opposite to each other, we assume, they do not belong to same class(for classification). The query time series (TQ) is unclassified and same as first time series (it is the same data as TS1). Our goal is to predict the class label of TQ by using the similarity. Similar time series are classified under same classes. So, the query time series's label should be same as it's most similar time series. This is the nearest neighbor classification technique [12], [13], [14], where we assign a class label to an unlabeled time series by finding its nearest neighbor's class label. We assume that the key points, important points to characterize a time series, are extracted and their feature vectors (the difference between the previous point and the key point, the difference between the key point and the next point) are calculated as shown in Table III. In Figure 3, key points are shown by circles and labeled by Knn such as K_{01} and K_{11}.

TAX applies k-means clustering method on key-points feature vectors to assign T-terms to key points. But any other method could be used in this purpose. To keep this example simple, we apply some thresholds to categorize the key points feature vectors instead of using a clustering method. We use the uniformly selected threshold 0.01 and 0.2 to quantize each of the vector entries. The key points get their T-terms depending on these two thresholds. If any of the

TABLE III
EXAMPLE OF KEY-POINT FEATURE VECTOR (1)

Time series	Key-point	Feature vec.	Tterm
TS1, TS2, TQ	K_{01}	{0.32,-0.33}	$Tterm_{HH}$
	K_{02}	{0.12,0.31}	$Tterm_{LH}$
	K_{03}	{0.04,0.0}	$Tterm_{L0}$
	K_{04}	{0.054,-0.08}	$Tterm_{LL}$
	K_{05}	{-0.04,0.01}	$Tterm_{L0}$

vector entries absolute value is less than or equal to 0.01, we assign "0". All values greater than 0.01 and less than or equal to 0.2 get the symbol "L". Lastly, for all values greater than 0.2, the symbol "H" is assigned. For each of the vector entries, it gets an assignment. By combining two symbol assignments, the corresponding T-term is assigned such as Tterm$_{LH}$ and Tterm$_{HH}$. The set of the used T-term is $\{Tterm_{HH}, Tterm_{LH}, Tterm_{L0}, Tterm_{LL}, Tterm_{L0}\}$.

After term generation, TAX uses term frequency and inverse document frequency to calculate the document vector. We calculate the document vector for given time series TS1, TS2 and TQ using $tf - idf$. For simplicity, we consider idf = 1. The result is shown in Table IV.

TABLE IV
TAX DOCUMENT VECTORS

Time series	Document vec.	Similarity with TQ
TS1	(1, 1, 1, 1, 1)	1
TS2	(1, 1, 1, 1, 1)	1
TQ	(1, 1, 1, 1, 1)	

TQ gets equal match with TS1 and TS2 although they are opposite to each other. The key point sequences for TS1 and TS2 are (K_{01}, K_{02}, K_{03}, K_{04}, K_{05}) and (K_{05}, K_{04}, K_{03}, K_{02}, K_{01}s). It does not make any difference in the tf-idf based approach. So, in this case, TAX classification fails. TAX gets equal match with two different time series, whose labels are different. The only solution of this problem is to consider the term sequences during similarity measure. We introduce lcs with TAX instead of $tf - idf$ to take the term sequences under consideration during similarity search. The similarities among TS1, TS2 and TQ are calculated from term sequences shown in Table V. We use lcs length to measure the similarity between time series. The similarity of $TS1$ and TQ is larger than the similarity of $TS2$ and TQ, which means $TS1$ is more similar to TQ than $TS2$. So, the predicted label for TQ is the same label as $TS1$.

TABLE V
TERM SEQUENCES OF THE EXAMPLE TIME SERIES

Time Series	Term Sequences (Tterm$_{xx}$)	Similarity with TQ
$TS1$	HH, LH, $L0$, LL, $L0$	5
$TS2$	$L0$, LL, $L0$, LH, HH	1
TQ	HH, LH, $L0$, LL, $L0$	

From these examples, it is clear that consideration of term sequences are a mandatory factor to increase the classification accuracy. In section III, we describe the method to represent a time series as a sequence.

III. Sequence construction and classification

In this section, we explain the methods to achieve the term sequences from time series. The construction phase starts by selecting time series key points that characterize the source time series. These key points are the training key points. Considering the key-point's surrounded points, we calculate the key point feature vector for all key points. Then these key-point feature vectors are clustered. Each cluster works as a term and identified by a symbol. For each of the time series, we construct a term-sequence by considering the key points appearing order in the original time series. During retrieval phase, for the query time series, we calculate the key points and key-point's feature vectors using the same procedure. Then these vectors are assigned to the nearest clusters. Now, the clusters contain both training key points and query key points. We construct another term sequence after the assignment by considering the key point's appearing order in the query time series. We find the most similar time series's class from the training time series comparing the query sequence against training sequences using longest common subsequence. In next section, we show the algorithm and describe the process in detail.

A. Algorithm Overview

The whole process is divided in training and search phase. Algorithm 1 and 2 show the training and retrieval phase respectively. Algorithm 1 runs only once during the training phase. If new training data are available only then it runs again. Algorithm 2 runs for every query time series.

1) Key-point Detection: l-TAX uses two different techniques to extract key points from the original time series. The first method is to take the absolute value of 2nd difference of the original time series. The second method is to use the absolute value of the difference between P-points weighted moving average and the original time series data. These filtered values are considered as key point candidates. Two predefined thresholds ϵ_1 and ϵ_2 are used to find the key points for first and second method respectively. In both cases, if a candidate exceeds the threshold at a time, then the value of that time from the original time series is detected as a key point. The final set of key points is constructed by taking the union of these two sets of key points.

2) Key-point Features: We represent all the key points by their feature vectors. These vectors contains key point features by considering their surrounding points. A key point is a point of interest among many points. So, a key point must be influenced by its surrounding points. For each of the key points, we take the differences between the key point and the wf surrounding points in both sides of the key points. wf is a predefined natural number. We take the average of two consecutive differences and construct the feature vector using these averages and the key points. Assume, y_x is a key point. If wf = 2, we compute the differences d_1=$y_x - y_{x-1}$, d_2=$y_x - y_{x-2}$ and d_3=$y_x - y_{x+1}$, d_4=$y_x - y_{x+2}$. We take the averages, avg_1 = $avg(d_1, d_2)$ and avg_2 = $avg(d_3, d_4)$. Then, the feature vector is constructed as (agv_1, y_x, avg_2).

Algorithm 1 l-TAX Training phase

Input: d_{train}: `2D Array of All training data`, ϵ_1: `key point threshold`, ϵ_2: `key point threshold`, wf: `neighborhood size`, t_{num}: `numbers of terms`
Output: s_{doc}: `a document that works like a database`
Variables: s_{train}: `contains the training term sequences`, $key\_feature$: contains feature vectors, $labels_{train}$: contains the training data class labels

```
1: $s_{train} \leftarrow$ empty array
2: $key\_feature \leftarrow$ empty array
3: $s_{doc} \leftarrow$ empty file
4: $d_{test} \leftarrow$ n data from training set as test case
5: $labels_{train} \leftarrow$ load training class labels
6: for $i = 1 \rightarrow length(d_{train}$ except $d_{test})$ do
7:     $key\_points \leftarrow get\_key\_points(d_{train}[i], \epsilon_1, \epsilon_2)$
8:     $key\_feature[i] \leftarrow get\_features(key\_points, wf)$
9: end for
10: $terms \leftarrow get\_clusters(key\_feature, T_{num})$
11: for $i = 1 \rightarrow length(key\_feature)$ do
12:     $j \leftarrow get\_source\_time\_series(key\_feature[i])$
13:     $belongs\_to \leftarrow get\_term\_id(terms, key\_feature[i])$
14:     $S_{train}[j] \leftarrow append(s_{train}[j], belongs\_to)$
15: end for
16: $paremeters \leftarrow optimise\_parameters(S_{train}, d_{test})$
17: Store $paremeters, labels_{train}, s_{train}$ and $terms$ in $S_{doc}$
```

3) T-term Construction: All key points are now represented by their feature vectors. We apply k-means to cluster these feature vectors. The number of cluster is predefined. These clusters work like terms. By checking the key-point feature vectors that belongs to clusters, we can easily determine which key-point belong to which cluster. We assign unique symbols to all the clusters to identify them. We call these symbols T-term id.

4) Term Sequence Construction: We use all the T-terms to construct Term-sequence. We do it from the set of T-terms and its belonging key points. For each key point, we find it's T-term id to obtain the sequence for a particular time series.

Assume that, $K = \{kp_1, kp_2, ..., kp_n\}$ is a set of key points extracted from a time series and ordered by their appearing time in the original time series. T = $\{T_1, T_2, ..., T_N\}$ is a set of T-term's ids. For, each key point from K, we find its belonging term. The corresponding T_{id} from T is used to get the term sequence, $Term - sequence$ = $\{T_{id,kp_1}, T_{id,kp_2}, ..., T_{id,kp_n}\}$. Figure 4 shows the term sequence construction.

We store the training data class labels, terms and the term sequences in a database. In retrieval phase, we load these

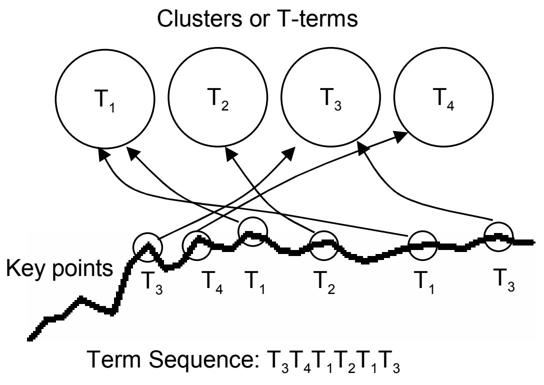

Fig. 4. Term sequence construction

values and use them to predict the class label of a unlabeled query time series.

Algorithm 2 l-TAX Search Phase

Input: d_{query}: A time series, ϵ_1: key point threshold, ϵ_2: key point threshold, wf: neighborhood size
Output: predicted class label for d_{query}
Variable: s_{query}: contains the query term sequence

```
1: Load $s_{doc}$ //Loading $labels_{train}$, $s_{train}$ and $terms$
2: $key\_points \leftarrow get\_key\_points(d_{query}, \epsilon_1, \epsilon_2)$
3: $key\_feature \leftarrow get\_features(key\_points, wf)$
4: for $i = 1 \rightarrow length(key\_feature)$ do
5:     belongs_to $\leftarrow get\_nearest\_term\_id(terms, key\_feature)$
6:     $s_{query} \leftarrow append(s_{query}, belongs\_to)$
7: end for
8: $best\_lcs \leftarrow -1$
9: $best\_matches\_time\_series\_index \leftarrow -1$
10: for $i = 1 \rightarrow length(s_{train})$ do
11:     $lcs \leftarrow get\_lcs(s_{query}, s_{train}[i])$
12:     if $lcs > best\_lcs$ then
13:         $best\_lcs \leftarrow lcs$
14:         $best\_matches\_time\_series\_index \leftarrow i$
15:     end if
16: end for
17: return $labels_{train}[best\_matches\_time\_series\_index]$
```

5) T-term Assignment: In retrieval phase, we load the training sequences, training data labels and clusters from database. When a query is given, we convert the query time series to its sequence representation using the same method as in construction phase. We extract key points from query time series and calculate their feature vectors. Then, we assign the feature vectors to the nearest clusters by finding their nearest cluster's center. It is done using following method:

Assume that, $QK = \{qk_1, ..., qk_n\}$ is a set of query key points and $F_{QK} = \{f_{QK,1}, .., f_{QK,n}\}$ is the set of their feature vectors. For each $f_{QK,x} \in F_{QK}$, we obtain a term T_y from the training terms such that,

$$minimize(distance(f_{QK,x}, center(T_y))).$$

We assign, the closest term T_y to the key-point feature vector $f_{QK,x}$.

6) Classification Using lcs length: Again, after query feature vector assignment, we use term sequences to prepare a query term sequence, s_{query}. Comparing this query sequence with the training sequences using lcs length, we find the best matched sequence and its class label and predict the label as the query time series label.

Assume that, $X = \{x_1, x_2, ..., x_m\}$ is a set of term sequences of a time series obtained during training session. $Y = \{y_1, y_2, ..., y_n\}$ is another sequence obtained from the query time series. Let, $lcs(X_i, Y_j)$ represents the longest common subsequence of prefixes X_i and Y_j. We find the longest sequence as follows:

$$lcs(X_i, Y_j) = \begin{cases} 0 & \text{if i = 0 or j = 0} \\ (lcs(X_{i-1}, Y_{j-1}), x_i) + 1 & \text{if } x_i = y_j \\ longest(lcs(X_i, Y_{j-1}), & \\ \quad lcs(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases} \tag{1}$$

We use dynamic programming to calculate the lcs length. One query can get multiple nearest neighbor with same magnitude. It is because of lcs length definition. We use whole match to calculate the lcs length. If two symbol matches then we increase the length by 1 otherwise 0. For this reason, more than one sequence can get maximum lcs length with the query. we consider all such sequence's class labels. The system predicts the query class that more than 50% of the best matched sequences are labeled.

B. Algorithmic Complexity Analysis

The system is divided into two phases. In both phases the key point detection and their feature calculation take linear time. We maintain a map between key point feature and source time series. It takes linear time too. During training session, we apply k-means clustering to the feature vectors. K-means takes $O(Imnk)$ time, where I is the number of iterations, m is the dimension of the feature vector, n is the number of the key points in training data set and k is the number of terms. The term id assignment and term sequence generation take linear time as we keep the maps for key points to time series and key points to clusters. In retrieval phase, the nearest cluster center search and assignment takes $O(nm)$ time, where n is the number of query key points and m is the number of terms. Longest common subsequence uses dynamic programming, which takes polynomial time. The query sequence matches against all training sequences. Each of the matches takes $O(nm)$ time, where n and m are the lengths of the sequences.

TABLE VI

ACCURACY RATIO OF DIFFERENT TIME SERIES CLASSIFICATION METHODS

Dataset	Euclidean [15]	DTW [16]	ODTW [16]	OTWED [17]	TAX	l-TAX	Remarks for l-TAX		
							No. of T-terms	Avg. No. of Key-points per Training data	Avg. No. of Key-points per Test data
Two Patterns	91.00	100.0	99.85	99.90	80.20	99.88	85	14.00	14.00
Wafer	99.50	98.00	99.50	99.60	98.70	98.46	90	15.70	15.67
Gun-point	91.30	90.70	91.30	98.70	94.70	98.00	100	14.50	14.36
CBF	85.20	99.70	99.60	99.30	96.30	97.78	45	24.80	24.69
Fish	78.30	73.30	76.70	94.90	84.60	94.29	80	131.6	131.4
Synthetic Cntl	88.00	99.30	98.30	97.70	90.33	93.67	75	5.70	5.80
Beef	53.30	50.00	53.30	46.70	80.00	93.33	55	23.30	23.70
Trace	76.00	100.0	99.00	95.00	90.00	93.00	45	25.80	25.74
Coffee	75.00	82.10	82.10	78.60	92.86	92.86	30	16.00	16.00
Yoga	83.00	83.60	84.50	87.00	84.40	90.30	130	159.9	160.0
ECG	88.00	77.00	88.00	90.00	86.00	90.00	28	8.90	9.00
Swedish Leaf	78.70	79.00	84.30	89.80	79.80	86.72	70	36.10	36.21
OliveOil	86.70	86.70	83.30	83.30	83.30	86.67	100	46.90	46.80
Lighting-2	75.40	86.90	86.90	78.70	77.00	85.25	95	52.50	53.43
Face(four)	78.40	83.00	88.60	96.60	58.00	80.68	85	138.9	136.28
OSU Leaf	51.70	59.10	61.60	75.20	69.80	74.38	65	154.7	154.77
Face(all)	71.40	80.80	80.80	81.10	70.70	72.96	850	33.40	32.92
Lighting-7	57.50	72.60	71.20	75.30	64.40	67.12	90	27.50	27.38
50 Words	63.10	69.00	75.80	81.30	42.90	60.44	40	64.60	64.45
Adiac	61.10	60.40	60.90	62.40	57.30	59.85	90	48.60	48.74

IV. EVALUATION

In this section, we evaluate the method to show the effectiveness. We use the method to classify data, where the results of the classification are known. This method has been used extensively in previous researches [16], [18], [12], [19], [20]. We compare l-TAX against other existing methods like Euclidean [7], DTW [16], TAX, OTWED [17].

A. Experimental Setup

1) Data sets: We collect our test data sets from UCR Time Series Data Mining Archive [16]. It contains different data sets with varying lengths and different class cardinalities. These data sets are taken form different sources like motion capture, word recognition, electrocardiogram etc. Data sets are divided in training and test parts. In both cases, the class labels are given. We train the system using the training data. Then for each test data, the system predicts the class label depending on the similarity with the training data.

2) Parameter settings and Accuracy measure: We use the training data set to optimize parameter settings especially the number of T-terms. K-means clustering uses random center selection. To get the same cluster every time, we fix the random seed to 1. So, for a particular number of terms, we always get same clusters for same key point feature vectors. We select the key point parameters ϵ_1 and ϵ_2 in such a way, so that we get variable numbers (5%, 10%, 15%) of key points for different data sets. We set neighborhood parameter, wf = 10. Key point thresholds and neighborhood parameter settings are same as TAX. For term number, we try different values on training data and keep the best one for which the accuracy is maximum.

For our classification evaluations, we define accuracy as:

$$Accuracy = \frac{|KNN(q) \cap std\_set(q)|}{K} \times 100\% \qquad (2)$$

where $KNN(q)$, is the K nearest neighbor for the query, found by the method. $std\_set(q)$, is the correct class for the query. In our evaluation, we use K = 1.

TABLE VII

AVERAGE ACCURACY OF DIFFERENT TIME SERIES CLASSIFICATION METHODS

	Euclidean [15]	DTW [16]	ODTW [16]	OTWED	TAX [17]	l-TAX
Avg. Accuracy	76.63	81.56	83.28	85.56	79.07	85.78
Std. Div.	13.41	14.55	13.63	13.86	14.68	12.50

B. Results

Table VI shows the classification accuracy comparison with other existing methods. l-TAX performs significantly well than other methods. Compare to TAX, l-TAX increases the accuracy for all data sets. Although in case of Wafer data set, it decreases little bit but that is very insignificant. For Face (four) and 50 Words data sets, l-TAX accuracies increase by 29% and 16.22% respectively. TAX performance was very poor for these data sets. l-TAX achieves its maximum accuracy of 99.88% for Two Pattern data set. Other methods also performed well for this data set except TAX. In this case, l-TAX increases the accuracy by 19.68% compare to TAX. l-TAX performs extremely well for some data sets where other methods fail to achieve a higher accuracy. For example, l-TAX gets more than 90% accuracy for Beef and Coffee data sets. Except TAX, all other methods get an accuracy of below 80%. Our l-TAX classification accuracy level seems stable on various kinds of time series.

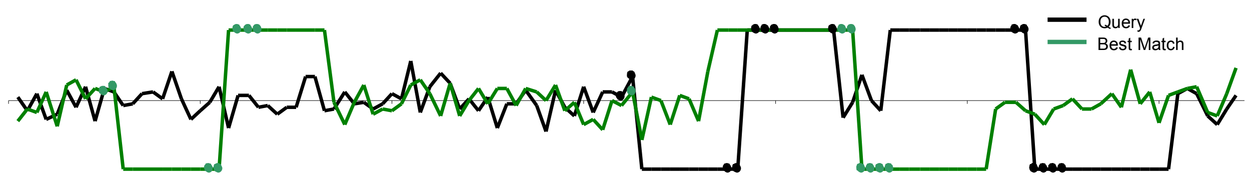

Fig. 5. A Two Pattern data query and the best matched time series by l-TAX

Table VII shows the average accuracies and standard deviations of some methods. Average accuracy and standard deviation shows a method's stability and applicability on different domains. The average accuracies of Euclidean, ODTW, DTW, TAX, OTWED, l-TAX methods are 76.63%, 83.28%, 81.56%, 79.07%, 85.56%, 85.78% respectively. The average accuracy of l-TAX has increased by 6.51% compare to TAX. It gets higher average accuracy than the other methods too. l-TAX ended up with 85.78% accuracy. The average accuracy of l-TAX has increased by 6.51% compare to TAX. The standard deviation of Euclidean, ODTW, DTW, TAX, OTWED, l-TAX methods are 13.41, 13.63, 14.55, 14.68, 13.86, 12.50 respectively. l-TAX gets the minimum standard deviation of 12.5, which is 14.68 for TAX. High accuracy and less standard deviation shows that l-TAX is stable and applicable on different data sets.

In Figure 5, we show a case of Two Pattern data set, where it shows a query and its closest time series found by l-TAX. Both the query and matched data are from same class although they look different by observation. We marked the key points using circles for both the time series. TAX uses the same key points and feature vectors for the same data. But TAX accuracy is 19.68% lower than l-TAX for this data set because it does not consider the key point's term sequences, which is very important feature for time series classification and search. l-TAX considers the term sequences and performs reasonably well.

From the average accuracy and standard deviation view point, l-TAX outperforms DTW, ODTW and OTWED. These methods use time warp as their base. Our method performs significantly well than Euclidean [15]. Euclidean [15] uses euclidean distance to calculate the similarity. l-TAX has got the highest average accuracy and lowest standard deviation than other methods, which uses both time series symbolic representations or non symbolic representations to classify data.

V. Related Work

Many techniques have been invented to find the similarity between two time series [1], [2], [3], [5], [6], [7], [8]. Non symbolic similarity search method like Euclidean distance sums the euclidean distance between two values to corresponding time points. Feature space transformation such as DFT and DWT are the most typical method for similarity search. These feature transformations can be applied when there is no knowledge on internal models. However, they can not be applied effectively when time series data are an output from a certain model like our model-based dynamical systems.

Textual approximation based time series similarities are also investigated although most of them are kinds of symbolic approximation methods. One of the great works was done by H. Shatkay and S. B. Zdonik in [21]. They proposed a general approximate data representation for time series with development of the breaking algorithm. Another symbolic approximation based method is called SAX [22], [7]. SAX is one of the novel symbolic aggregate approximation method to approximate time series with a sequence of symbols. Other methods including some of these existing approximation are based on the value-to-textual conversion of time series, which means that a value at a time is approximated with some predefined symbols calculated from their value [23].

Many methods such as ERP[31] or EDR [18] are proposed, which uses edit distance as their base. A Longest Common Subsequence based work is presented in citeicde:trajectories. Recently another method that combines LCS and EDR is presented in [19]. AMSS [32] is another longest common subsequence based technique that uses their own distance definition to find the distance between two symbols. The other textual approximation methods including [33], [34], [35], [36], [37] are based on application domain dependent symbolic representation of time series from much knowledge on features of time series.

A survey of longest common subsequence algorithms is shown in the paper [24]. The most important lcs methods are described in books on text algorithms [25], [26], [27] and on molecular biology [28], [29], [30].

A novel dissimilarity method between two time sequences is introduced in the paper [9]. In the proposed method, the time series are converted to SAX [7] sequence and then used the dissimilarity method. A new measure of distance between two time series symbolic sequences is proposed in [11]. Paper [10] introduces an extended version of longest common subsequences. It considers not only the longest common subsequences but also the 2nd longest subsequence, 3rd longest sub-sequences and so on to find the time series similarity.

VI. Conclusions

In this paper, we have proposed a time series approximation representation for time series classification, called l-TAX, using the text document structure and longest common subsequence. The main idea of the proposed l-TAX method is to construct a set of temporal terms, called T-terms, from a large amount of time series database and use their sequences to make string sequence to find the time series similarity using Longest Common Subsequence. With the l-TAX, a user can obtain the desirable time series data set with higher accuracy. From the experiments, we have showed that our l-TAX is effective and can be used for a large amount of time series classification.

During training, we have conducted our experiment with different parameter values. Due to page limitation, in this paper, we show the optimized values for which our method performs well. With more proper parameter settings, l-TAX can even get better results than the one that has shown here.

As part of our future work, we plan to evaluate l-TAX based on computation time, explore additional features for automatic parameter selection and to develop pruning and indexing techniques for faster classification.

Acknowledgment

This work is partially supported by KAKENHI #24300039.

References

[1] J. Afalg, H.-P. Kriegel, P. Krer, P. Kunath, A. Pryakhin, and M. Renz, "Similarity search on time series based on threshold queries," in *EDBT*, 2006.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," *FODO*, pp. 69–84, 1993.

[3] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," *ICDE*, pp. 126–133, 1999.

[4] K. Kawagoe, Abdulla-Al-Maruf, K. Deng, and X. Zhou, "Searching time series using textual approximation," in *The Third International Conference on Emerging Databases*, 2011, pp. 121–132.

[5] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *SIGMOD*, pp. 151–162, 2001.

[6] E. J. Keogh and M. J. Pazzani, "A simple dimensionality reduction technique for fast similarity search in large time series databases," *PAKDD*, pp. 122–133, 2000.

[7] J. Lin, E. J. Keogh, S. Lonardi, and B. Y.-C. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," *DMKD*, pp. 2–11, 2003.

[8] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," *VLDB*, pp. 385–394, 2000.

[9] J. Lin, E. Keogh, and S. Lonardi, "Visualizing and discovering non-trivial patterns in large time series databases," *Information Visualization*, vol. 4, no. 2, pp. 61–82, 2005.

[10] H. Wang, "All common subsequences," in *International Joint Conference on Artificial Intelligence*, 2007, pp. 635–640.

[11] A. C.-C. Yang, S.-S. Hseu, H.-W. Yien, A. L. Goldberger, and C.-K. Peng, "Linguistic analysis of the human heartbeat using frequency and rank order statistics," *Physical Review Letters*, vol. 90, no. 10, 2003.

[12] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *International Conference on Knowledge Discovery and Data Mining*, 2004.

[13] ——, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," in *ICDM*, 2006, pp. 623–632.

[14] J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in *Twenty-ninth Annual ACM Symposium on Theory of Computing*, 1997.

[15] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: A novel symbolic representation of time series," *DMKD*, vol. 15, no. 2, pp. 107–144, 2007.

[16] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. (2006) The ucr time series classification/clustering. [Online]. Available: http://www.cs.ucr.edu/ eamonn/time_series_data/

[17] P. F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," in *Pattern Analysis and Machine Intelligence*, 2009, pp. 306–318.

[18] L. Chen, M. T. zsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *SIGMOD*, 2005, pp. 491–502.

[19] M. D. Morse and J. M. Patel, "An efficient and accurate method for evaluating time series similarity," in *SIGMOD*, 2007, pp. 569–580.

[20] V. M., K. G., and G. D, "Discovering similar multidimensional trajectories," in *ICDE*, 2002, pp. 673–684.

[21] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *ICDE*, 1996, pp. 536–545.

[22] E. J. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, 2003.

[23] J. Lin and Y. Li, "Finding structural similarity in time series data using bag-of-patterns representation," *Lecture Notes in COmputer Science*, vol. 5566/2009, pp. 461–477, 2009.

[24] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *International Symposium on String Processing Information Retrieval*, 2000, pp. 39–48.

[25] A. Apostolico and Z. Galil, *Pattern matching algorithms*. Oxford University Press, 1997.

[26] M. Crochemore and W. Rytter, *Text algorithms*. Oxford University Press, 1994.

[27] G. A. Stephen, *String Searching Algorithms*. World Scientific, 1994.

[28] D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997.

[29] E. Myers, *An overview of sequence comparison algorithms in molecular biology*. University of Arizona, 1991.

[30] D. Sankoff and J. B. Kruskal, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley Pub. Co, 1983.

[31] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *VLDB*, 2004, pp. 792–803.

[32] T. Nakamura, K. Taki, H. Nomiya, and K. Uehara, "Amss: A similarity measure for time series data," *Journal of the Institute of Electronics, Information and Communication Engineers*, vol. J91-D, no. 11, pp. 2579–2588, 2008.

[33] M. Baumert, V. Baier, S. Truebner, A. Schirdewan, and A. Voss, "Short- and long- term joint symbolic dynamics of heart rate and blood pressure in dilated cardiomyopathy," *IEEE Trnas. on BME*, vol. 52, no. 12, pp. 2112–2115, 2005.

[34] C. S. Daw, C. E. A. Finney, and E. R. Tracy, "Symbolic analysis of experimental data," *Review of Scientific Instruments*, vol. 74, no. 2, pp. 915–930, 2003.

[35] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[36] J. Kurths, A. Voss, A. Witt, P. Saparin, H. J. Kleiner, and N. Wessel, "Quantitative analysis of heart rate variability," *Chaos*, vol. 5, pp. 88–94, 1995.

[37] F. Wendling, J.-J. Bellanger, J.-M. Badier, and J.-L. Coatrieux, "Extraction of spatio-temporal signatures from depth eeg seizure signals based on objective matching in warped vectorial observations," *IEEE Trnas. on BME*, vol. 43, no. 10, pp. 990–1000, 1996.

[38] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, 1999, pp. 1150–1157.

[39] ——, "Distinctive image features from scale-invariant key points," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[40] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," *International Conference on Computer Vision*, vol. 2, pp. 1470–1477, 2003.

[41] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li, "Descriptive visual words and visual phrases for image applications," in *ACM International Conference on Multimedia*, 2009, pp. 75–84.

[42] Q.-F. Zheng, W.-Q. Wang, and W. Gao, "Effective and efficient object-based image retrieval using visual phrases," in *ACM International Conference on Multimedia*, 2006, pp. 77–80.