IEICE TRANS. INF. & SYST., VOL.E97–D, NO.4 APRIL 2014

PAPER Special Section on Data Engineering and Information Management

Textual Approximation Methods for Time Series Classification: TAX and l-TAX

Abdulla Al MARUF[†a)], **Hung-Hsuan HUANG**[††b)], *Nonmembers*, *and* **Kyoji KAWAGOE**[††c)], *Member*

SUMMARY A lot of work has been conducted on time series classification and similarity search over the past decades. However, the classification of a time series with high accuracy is still insufficient in applications such as ubiquitous or sensor systems. In this paper, a novel textual approximation of a time series, called TAX, is proposed to achieve high accuracy time series classification. l-TAX, an extended version of TAX that shows promising classification accuracy over TAX and other existing methods, is also proposed. We also provide a comprehensive comparison between TAX and l-TAX, and discuss the benefits of both methods. Both TAX and l-TAX transform a time series into a textual structure using existing document retrieval methods and bioinformatics algorithms. In TAX, a time series is represented as a document like structure, whereas l-TAX used a sequence of textual symbols. This paper provides a comprehensive overview of the textual approximation and techniques used by TAX and l-TAX

key words: *time series, similarity search, textual approximation, document retrievals, key points, longest common subsequence*

1. Introduction

There are many symbolic representations of time series. The motivation of a symbolic time series representation is to apply rich data structures and algorithms from the text processing and bioinformatics domain to the time series. A symbolic representation also solves the curse of dimensionality problem by reducing an original time series to a lower dimension. Researches in this area all have a common goal to achieve a high accuracy in time series classification because of its significance and applicability to various domains.

A time series database is maintained in a time series classification and similarity search system. Such a system takes a query time series as its input and finds the most similar time series from the database [1]–[6]. These methods apply time series dimension reduction techniques to transform the time series into its features in a feature space using certain transformation functions. This 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 methods. Finally, the distance definition (Euclidean distance, Manhattan distance, Dynamic Time Warping etc.) is used to calculate the similarity between the query and stored time series.

TAX [3] is a novel method of time series modeling based on textual approximation that uses a text document-like structure as its core. The significant point of this method is that it uses existing document retrieval techniques, which are widely used in natural language processing and string searches. However, it is difficult to apply such techniques in a time series because time series data are not a sequence of terms, but rather a sequence of numeric values over time. It is not easy to extract the terms from such time series data in the same way as from a text document. TAX converts the time series into a new structure which is similar to a text document.

TAX extracts temporal terms (T-terms) from time series data and stores them like words in a text document by introducing a temporal feature vector constructed from local features of the key-points. TAX then applies techniques from natural language processing. TAX performs time series classification very well. An extended version of TAX, called l-TAX [7], was proposed to reach even higher accuracies than TAX. l-TAX uses techniques from both natural language processing and bioinformatics. The major contribution of l-TAX is that it achieves the most accurate classification accuracies over existing dominant methods.

The remainder of this paper is structured as follows. Some previous works related to our paper are described in Sect. 2. Section 3 provides some definitions used herein as well as a short introduction of TAX and l-TAX. Next, in Sect. 4, a detailed description of TAX using document retrieval methods is given. Section 5 describes l-TAX itself, and an experimental evaluation of both methods is then shown in Sect. 6. Finally, we provide some concluding remarks regarding our paper in Sect. 7.

2. Related Work

Many different techniques have been proposed to find the similarity between two time series [1], [2], [4]–[6], [28]. A threshold-based query technique was introduced in [1]. This technique decomposes a time series into time intervals of the subsequent elements. Similar interval sequences with values above a threshold are retrieved as being similar. The technique considers all data points above the threshold without considering the noise. Our method uses a similar kind of threshold approach to sort out the key points but after noise filtration is applied. In [2], an efficient similarity search for

Manuscript received July 13, 2013.
Manuscript revised October 29, 2013.

[†]The author was with the Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.
[††]The authors are with the Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.
a) E-mail: maruf@coms.ics.ritsumei.ac.jp
b) E-mail: huang@fc.ritsumei.ac.jp
c) E-mail: kawagoe@is.ritsumei.ac.jp

DOI: 10.1587/transinf.E97.D.798

Copyright © 2014 The Institute of Electronics, Information and Communication Engineers

a sequence database, which uses DFT for the feature space transformation, is proposed. This causes a curse of dimensionality. In our proposed model, we use selected key-points to extract different features. As the number of key-points is very low, the dimensionality problem is avoided. In [4]–[6], a time series in equal segments is divided, and the mean of all segment values, or Piecewise Aggregate Approximation values, is used to reduce the dimensionality. As our method filters the data before feature extraction and considers only some of the important data, it obtains better features for representing the time series.

A number of works have been conducted on the textual approximation of a time series, although most of them are a type of symbolic approximation method. One of the pioneer works in this area was conducted by H. Shatkay and S.B. Zdonik [8]. The authors proposed a general approximate data representation for a time series with the development of a breaking algorithm. They focused on representing the sequences using real valued functions. Their approximation is always tightly coupled with the application domain, whereas TAX/l-TAX uses a generic domain-independent approximation. Another symbolic approximation method called SAX [5], [9] was also proposed. SAX is a novel symbolic aggregate approximation method used to approximate a time series with a sequence of symbols. SAX uses Piecewise Aggregate Approximation to approximate a time series into its symbolic form. It divides the time series into equal segments. The mean of each segment is mapped to symbols from a predefined alphabet. By converting a time series into a symbolic sequence, the storage and search costs can be drastically reduced, which is the main point of SAX. From our viewpoint, it is quite inefficient to consider all data points during a symbolic representation.

Wang et al. [18] proposed Piecewise Vector Quantized Approximation (PVQA) for a time series data approximation method. The authors used a piecewise quantization method where each time series is divided into equal sized segments, and each segment is replaced with the most similar codeword.The time series is then approximated as a sequence of codes. The idea of PVQA is much similar to that of APCA [28] and SAX [5] because they are segment-based, which means that time series data are decomposed into small-length segments, where each segment is approximated using the average value in APCA, and encoded using the average value in SAX. In PVQA, a time series is decomposed into equal sized segments. Each segment is encoded and approximated using a so-called codebook. The basic idea of our methods, TAX and l-TAX, is not segment based but key-point based. Key-points for a time series approximation are extracted from the time series. Each key-point is encoded using pre-calculated terms. The neighbor points around a key-point are used as the characteristics of the key-point. Segment decomposition is unnecessary in both l-TAX and TAX.

Seo et al. [12] proposed a method for multivariable stream data classification, which is similar to our own. The main idea behind their method is to transform a raw time series into a symbolic sequence by calculating the difference between two consecutive data points, and then assigning the value to a symbol based on its range values so that the n-gram is considered as a word. To classify the time series, they used the conventional tf-idf method as one of the classifiers. Although their method is quite similar to our own, they significantly differ in terms of the temporal feature definition and term construction. In our method, we calculate the multi-dimensional feature vector from the contiguous values around a key-point. A key-point is extracted from a time series as a characterizing point of that time series. The set of terms is constructed by clustering all feature vectors beforehand.

Edit-distance based methods have also been proposed such as ERP [29] and EDR [21]. ERP [29] can be viewed as a variant of EDR [21] and DTW, where the authors proposed a distance function called Edit Distance with Real Penalty, which deals with a relaxed equality rather than a strict equality between symbols. Our method uses a strict equality and we have a plan to implement a relaxed equality in the future. AMSS [19] is another longest-common subsequence based technique that uses its own distance definition to find the distance between two symbols, which is in contrast to our use of exact symbol matching. Other textual approximation methods including those in [30]–[34] are based on application-domain dependent symbolic representation of a time series from significant knowledge on the time series features, which are quite unlike our method as our goal is to build a domain independent model. Non-metric similarity functions based on the Longest Common Subsequence are presented in [27], where the authors proposed a method that performs significantly well for noisy data. For noisy data exact similarity computation is inefficient. The method in [27] introduces approximate algorithms with provable performance bounds, and is proposed for trajectories, which might contain high noisy data. Our proposed model is based on single dimensional data.

A framework for an uncertain time series similarity search was proposed in DUST [35]. DUST provided a theoretically sound method of computing distances between two time series where individual time stamps may be associated with different error distributions. The distance function of DUST depends on the type of error distribution. Our model uses selected key-points, and thus it does not depend on how the data are distributed. This makes the TAX model generic for various domains.

3. Time Series Textual Approximation

The novelty of our proposed method is that we introduce a new time series representation technique to represent a time series as a text-document-like structure for use in an existing document retrieval model. In our proposed method, a time series is represented as a sequence of temporal terms, called T-terms, which are basically transformed by constructing multi-dimensional feature vectors from contiguous values around the important data points that characterize a time se-

ries, called key-points.

Existing methods focus only on a value-based approximation to represent a time series, which we consider to be the main reason for the application dependencies of such existing methods.

The challenge is to develop a generic approximation model to obtain more precise classification for various types of time series data sets incorporating a document retrieval model for time series documents. Although there have been some previous researches [5], [8]–[12], as described in Sect. 2, in which a time series is effectively represented by a set of symbols, they are only for a few specific areas of applications. None of these methods consider representing a time series in a variety of application areas using one generic method. To incorporate document retrieval models into a time series, we developed two new textual models for a time series representation, i.e., TAX and l-TAX, as described herein.

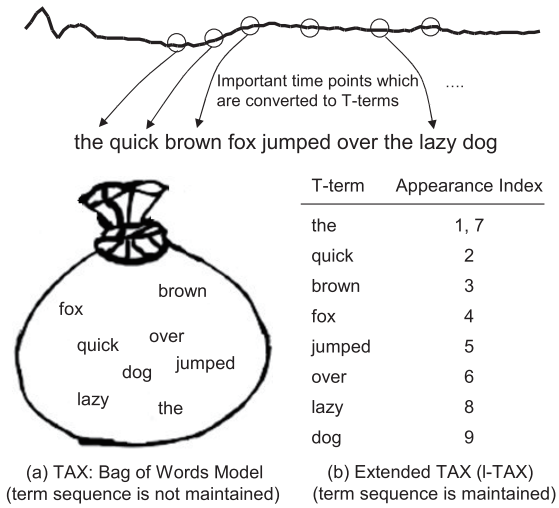


Fig. 1 Difference between TAX and l-TAX.

3.1 TAX[††††]

TAX uses existing document retrieval models to retrieve similar time series. It transforms a time series into a document-like structure. A text document contains a series of words or terms. The TAX process extracts terms, called T-terms, from a time series, and stores them in a document. TAX is a bag-of-words model based method. A bag-of-words model is widely used in natural language processing and information retrieval. This model presents a text or document as an unordered collection of words disregarding grammar and even word order.

TAX uses heuristics as in modern search engines. In modern search engines, not all the words in a document are considered equally important during document retrieval. Similar to search engines, TAX also uses only the important points of a time series to construct the T-terms, which are used to retrieve the time series. These important point selections are very important from the viewpoint of TAX. For all the time series, TAX uses the T-terms to calculate the *term-frequency* and *inverse-document-frequency* (tf-idf) to create a document vector representation of a time series. A query time series, which is also transformed into a TAX document vector, is compared with the stored document vectors using a cosine measure to find any similarities. This process in more detail is discussed in Sect. 4.

3.2 l-TAX[†††††]

Term sequence consideration is a significant factor for correct document retrieval. During the construction or retrieval phase, TAX does not keep any map of the T-terms sequence, despite using document retrieval methods. For example, Fig. 1 (a) shows The TAX approach for a query, *the lazy brown dog jumped over the quick fox*. The original text is "*the quick brown fox jumped over the lazy dog*". TAX matches the query terms to its bag and obtains a positive result because these (texts) are (regarded as) the same set of terms when the word order is ignored. Although the query is different, TAX identifies this match as a success. A set of T-terms can be used to represent many different time series. If the number of T-term occurrences is the same in various time series, then from the viewpoint of TAX, such occurrences will be considered as similar, as their tf-idf count will always be the same. This is a considerable failure for TAX, which increases its false-positive rate.

TAX has therefore been extended to solve this problem. The extended TAX, which is called l-TAX, inherits a significant feature from a modern search engine retrieval model. For this feature, the term's order of appearance in the document is considered. Figure 1 (b) shows the same problem under which TAX fails. l-TAX keeps a map of the terms and their position in the document. During retrieval, a TAX query document is therefore checked based on its terms and their positions. If a position varies for the same term between the query and stored document, l-TAX correctly identifies them as dissimilar. As l-TAX represents a time series as a sequence of T-term appearances, it uses the Longest Common Subsequence (LCS) as its distance measure. l-TAX finds a similarity by checking the LCS length between the query and the stored time series. For a certain query, the time series that obtains the maximum LCS length is considered as the most similar to the query time series.

3.3 Terminology

In this section, we describe some basic TAX concepts and definitions that are needed to understand the whole process.

3.3.1 Time Series

Time series data, T, can be represented by a sequence of pairs $e_i = (v_i, t_i)$, which are composed of a value v_i and time

[††††]The original work of TAX was presented in [3]
[†††††]The original work of l-TAX was presented in [7].

point t_i, i.e., $T = (e_1, e_2, \ldots, e_n)$. For simplicity, we assume that the time interval of the time points are equal and no missing value time points exist. We can then simply write time series data, T, as a sequence of values, such as like $T = (v_1, v_2, \ldots, v_n)$.

3.3.2 T-term

A temporal term is a set of selected and quantized data points that frequently occur in a time series. A T-term is an analogy of the term (word) in a document. Therefore, time series data, T, can be approximated using a set of T-terms contained by T. A large amount of time series data can be approximated using a small number of T-terms. Each of the T-terms is identified using a unique $Tterm$ id.

3.3.3 T-Document

A T-document is an approximation of time series data, as a document, which contains a sequence of T-terms. A T-document, $Tdoc_i$, contains a sequence of T-terms such as $Tdoc_i =< Tterm_{i_1}, Tterm_{i_2}, \ldots, Tterm_{i_j} >$.

As defined above, a set of temporal terms is used to describe a textual approximation of time series data. The terms in the documents and T-terms in the T-documents differ in their manner of term representation. Owing to the nature of time series data, following representation conditions should be taken into consideration.

1. A T-term should be fundamental, which means that the T-term should represent its characteristics, called key-points, in the time series data.
2. A T-term should be minimal. In other words, the number of T-terms should be small enough to represent the key features of a time series. Moreover, this number should be nearly constant even if a large number of newly arrived time series need to be approximated.

To meet the above two conditions, we propose the following concepts of a key-point and its features.

- A key-point of a time series is a point that partially approximates the time series. All key-points of the time series data are an approximation of the data. Therefore, an approximation of related time series data can be obtained by connecting adjacent key-points into a line.
- The characteristic feature of a key-point is defined by a vector, which is called a feature vector. The dimension of a feature vector is $NF = 2N_f + 1$, where NF is the dimension of the vector, and N_f is the number of neighboring data points (in the time series) of the key-point.
  - The middle entry of a feature vector i.e. the N_f-th entry, indicates the difference around the key-point, toward its positive direction.
  - The first to the $(N_f - 1)$th entries, indicate the average difference with the points before a key-point within the related time interval.

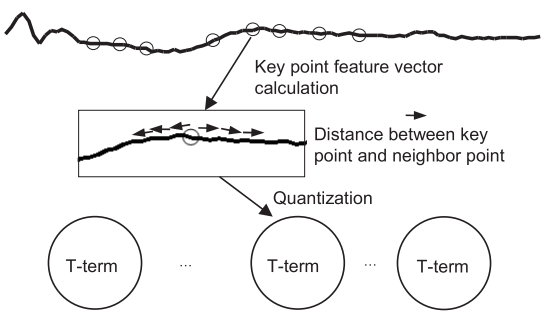


Fig. 2 T-term.

  - The $(N_f + 1)$th to the $(2N_f + 1)$th entries indicate the average difference with the points after a key-point within the related time interval.

Figure 2 shows a time series with many key-points extracted to characterize the time series. In the figure, one key-point is illustrated to exemplify its particular feature. The differences from the key-point to its neighboring data points (in this case, $N_f = 3$) are calculated and aggregated into the bins of its feature vector corresponding to the data points used in difference calculation. From the calculated feature vector of the key-point, an appropriate $T - term$ is assigned to the key-point through a quantization of that feature vector.

From the viewpoint of both TAX and l-TAX, a T-document is a time series representation method. Such a representation needs to satisfy the following conditions.

1. A T-document should represent a temporal document vector for retrieval.
2. A T-document should be comparable.

TAX and l-TAX follow two different methods for representing a T-document. TAX uses a similar vector representation required to represent a text document. T-document vector F_i of T-document $Tdoc_i$, is represented as, $F_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,j})$, where $f_{i,j}$ is a weight of $Tterm_j$ in T-document $Tdoc_i$. l-TAX uses a sequence of symbols to represent a T-document. In l-TAX, T-document vector of the T-document $Tdoc_i$, i.e., S_i, is represented as, $S_i = (s_1, s_2, \ldots, s_j)$, where s_j is a symbol representing a particular $T - term_j$ in T-document $Tdoc_i$.

3.4 Basic TAX/l-TAX Process

We show the TAX and l-TAX processes in Fig. 3. In both cases, the method is divided into two phases. In the construction phase, TAX and l-TAX detect key-points from the time series data and use their feature vectors to prepare T-terms that are used to construct T-documents. The T-documents are stored in database. During the retrieval phase, query time series is approximated by the TAX or l-TAX. Alike the construction phase, both methods (TAX and l-TAX) detect key-points and calculate their feature vectors. These feature vectors are assigned to the T-terms constructed during the construction phase. After the assignment, a query T-document vector is generated and used to

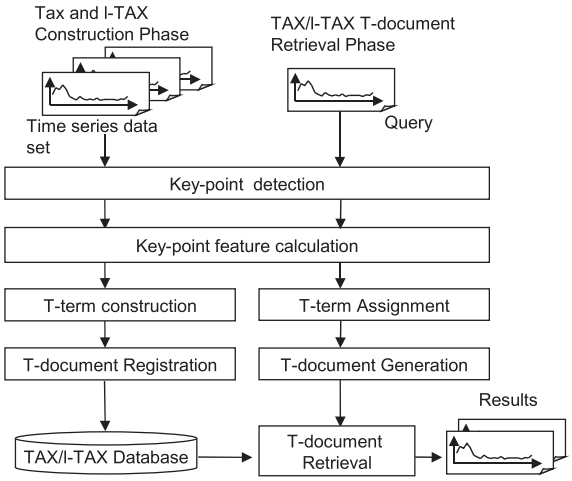


Fig. 3 TAX process.

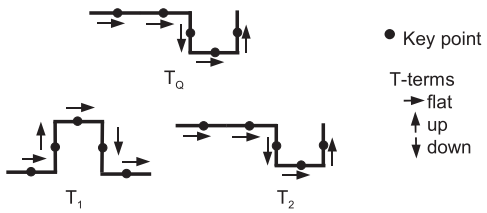


Fig. 4 Example of TAX AND l-TAX (1).

retrieve a set of similar documents from the database.

The details of each of these steps are discussed in Sect. 4.

3.5 Examples

In the following examples, we show how well TAX and l-TAX perform on the same time series. We use only simple time series so that readers can understand the process easily. In both cases, TAX fails whereas l-TAX finds the similar time series successfully. The examples are used only to explain the difference between the two methods as well as the effectiveness of the document retrieval methods. The definition of our multi-dimensional feature vector and the T-term construction method are described in the next section.

(1) This example uses two time series (T_1 and T_2) and a query time series (T_Q) as shown in Fig. 4, which are all synthetic time series. We apply both TAX and l-TAX to find the most similar time series to T_Q from T_1 and T_2. It can be clearly observed that $T_Q = T_2$.

In our model, we first convert all time series into their textual forms. To do so, we assume that some important data points exist that can be used to characterize the time series. We can therefore represent a time series by considering only those points and without losing the important properties. In Fig. 4, we show these points using circles, which were selected uniformly for the sake of simplicity. We assign some textual terms to each of these points to represent the time series textually. To represent the given time series, we define a

Table 1 Time series similarity using TAX.

Time series	Term Frequency			Document Vector	Similarity with T_Q
	flat	up	down		
T_1	3	1	1	(3, 1, 1)	1
T_2	3	1	1	(3, 1, 1)	1
T_Q	3	1	1	(3, 1, 1)	

Table 2 Time series similarity using l-TAX.

Time series	Term Sequence	LCS Length
T_1	(flat, up, flat, down, flat)	4
T_2	(flat, flat, down, flat, up)	5
T_Q	(flat, flat, down, flat, up)	

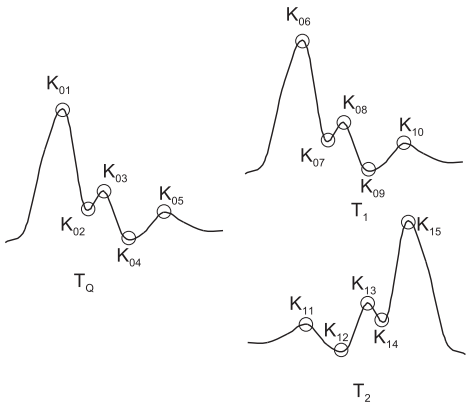


Fig. 5 Example of TAX AND l-tax(2).

set of textual terms, $Term = \{flat, up, down\}$, which can represent all three given time series. Using this set, the textual representation of the time series T_1, T_2 and T_Q become, $T_1 = \{flat, up, flat, down, flat\}$, $T_2 = \{flat, flat, down, flat, up\}$ and $T_Q = \{flat, flat, down, flat, up\}$.

TAX uses the term frequency and inverse document frequency (tf-idf) to create document vectors for all time series. For simplicity, we consider $idf = 1$. Table 1 shows the document vectors by counting tf for T_1, T_2 and T_Q. This table also shows the cosine similarity of T_1, T_2 and T_Q.

TAX shows that both T_1 and T_2 are similar to T_Q, but fails to distinguish that T_1 is different from T_Q because both T_1 and T_Q contain equal numbers of the same terms, although their term sequences are different.

The l-TAX approach is shown in Table 2. We represent each time series as the term sequence. The LCS length between T_2 and T_Q is higher than that between T_1 and T_Q. So, l-TAX therefore selects T_2 as most similar time series to T_Q, and T_1 is successfully identified as having a different shape than T_Q, although both of them have equal number of the same terms.

(2) Fig. 5 shows a more complex example using real time series. Here, we perform a similarity search to classify the given time series.

In the figure, there are three electrocardiography (ECG) time series data, labeled as T_1, T_2 and T_Q. T_1 is a normal ECG data, whereas T_2 is a mirror view (opposite) of T_1. As-

Table 3 Example of key-point feature vector (1).

Time series	Important Points	Vector to Represent important points	Terms Assigned to Imp. points
T_Q	K_{01}	{0.32,-0.33}	$Term_{HH}$
	K_{02}	{0.12,0.31}	$Term_{LH}$
	K_{03}	{0.04,0.0}	$Term_{LO}$
	K_{04}	{0.054,-0.08}	$Term_{LL}$
	K_{05}	{-0.04,0.01}	$Term_{LO}$
T_1	K_{06}	{0.32,-0.33}	$Tterm_{HH}$
	K_{07}	{0.12,0.31}	$Term_{LH}$
	K_{08}	{0.04,0.0}	$Term_{LO}$
	K_{09}	{0.054,-0.08}	$Term_{LL}$
	K_{10}	{-0.04,0.01}	$Term_{LO}$
T_2	K_{11}	{0.01,-0.04}	$Term_{LO}$
	K_{12}	{-0.08,0.054}	$Term_{LL}$
	K_{13}	{0.0,0.04}	$Term_{LO}$
	K_{14}	{0.31,0.12}	$Term_{LH}$
	K_{15}	{-0.33,0.32}	$Term_{HH}$

sume that T_1 and T_2 represent two different categories of cardiac problems, *IrregularPulse* and *HeartPounding* respectively. T_Q is the given ECG data, where the category is unknown. We have to find the category of T_Q. To do so, we find the similarity of T_Q with T_1 and T_2.

From the observation it is clear that T_Q is the same as T_1. Similar time series are classified under the same category. T_Q and T_1 should therefore have the same category because T_Q is more similar to T_1 than to T_2. This is the nearest neighbor classification technique [13]–[15], where we assign a category label to an unlabeled time series by finding the category of its nearest neighbor. We apply both TAX and l-TAX to find the most similar time series of T_Q from T_1 and T_2. We assume that the important points that characterize the time series are extracted, shown in the circles in Fig. 5. We label these points as K_{01}, K_{02}, ..., K_{13} etc. We also assume that these points are directly influenced by their surrounding points which is why they are more important than any other points. To represent these important points we also involve their surrounding points. For each important point, we calculate a vector by taking the difference between the important point and its surrounding points (previous and next time points from the original time series). The vectors that represent the important points are shown in Table 3.

We now represent the time series using some textual terms. To do so, we assign some textual terms to each of the key points. There are many ways to assign a term to an important point. As a simple example, we assign the terms based on the vectors calculated to represent the important points. We uniformly define some thresholds and check the vector values against these thresholds to find the appropriate terms. Depending on the values and thresholds, we map each of the entries of a vector to a symbol from $S = \{O, L, H\}$. If any of the vector entries have an absolute value of less than or equal to 0.01, we assign the symbol O. Any values greater than 0.01, and less than or equal to 0.2, are assigned a symbol L. Finally, all values of greater than 0.2, are assigned the symbol H. For each vector entry, the associated important point receives an assignment. By combining the assigned symbols, the important points receive their terms such as Term_{LH} and Term_{HH}. In this example, $\{Term_{HH}, Term_{LH}, Term_{LO}, Term_{LL}, Term_{LO}\}$ is used as the term set.

Table 4 TAX document vectors.

Time series	Document vector	Similarity with T_Q
T_1	(1, 1, 1, 1, 1)	1
T_2	(1, 1, 1, 1, 1)	1
T_Q	(1, 1, 1, 1, 1)	

Table 5 Term sequences of the example time series.

Time Series	Term Sequences (Tterm_{xx})	Similarity with T_Q
T_1	$HH, LH, L0, LL, L0$	5
T_2	$L0, LL, L0, LH, HH$	3
T_Q	$HH, LH, L0, LL, L0$	

After term generation, TAX uses $tf\text{-}idf$ to calculate the document vector for each time series. We calculate the document vectors for the given time series T_1, T_2 and T_Q using $tf\text{-}idf$. For simplicity, we consider $idf = 1$. The results are shown in Table 4.

TAX finds T_Q to be equally similar to T_1 and T_2, although T_1 and T_2 are not the same, and are rather opposite each other. The term sequences for T_1 and T_2 are HH, LH, $L0$, LL, and $L0$, and $L0$, LL, $L0$, LH, and HH, respectively. This creates no difference in the $tf\text{-}idf$ based calculation because $tf\text{-}ifd$ only counts the number of appearances of a term. In this case, TAX classification therefore fails. TAX obtains an equal match with two different time series whose categories differ. The only solution to this problem is to consider the term sequences during a similarity calculation. l-TAX uses LCS with TAX instead of $tf\text{-}idf$ to take the term sequences into consideration during a similarity search. The similarities among T_1, T_2 and T_Q are calculated from the term sequences shown in Table 5. In this case, the LCS length is used to measure the similarities among the time series. The similarity between T_1 and T_Q is higher than the similarity between T_2 and T_Q, which means that T_1 is more similar to T_Q than to T_2. The predicted category for T_Q is therefore the same as T_1.

These examples show that the term sequence must be considered to obtain an accurate similarity of various kinds of time series and increase the classification accuracy.

4. Textual Approximation

In this section, we discuss the necessary details of TAX and l-TAX, providing the reader with a better understanding of the proposed process. We then explain the process in greater details in the following section.

4.1 Key-Point Detection

Key-points are the important points characterizing the features of a time series. There are many ways to extract such

points. TAX and l-TAX combine two methods for a better approximation of time series data by extracting key-points in multiple ways.

The first method is to consider the second difference of the time series from the viewpoint of the time dimension. For time series data $T =< v_1, \ldots, v_n >$, the second difference, i.e. g_2, is defined as, $g_2 =< g_{2,1}, \ldots, g_{2,n} >$, $g_{2,i} = g_{1,i} - g_{1,i-1}$, where $g_{1,i} = v_i - v_{i-1}$. A set of key-point candidates can be obtained by using g_2. All candidates are verified to check whether they are a key-point using a threshold ϵ_1. If a candidate exceeds this threshold at a particular time, then the value of that time is detected as a key-point $r_{1,k}$ for the time series data. When threshold ϵ_1 is given, several key-points, $K_1 = \{r_{1,1}, \ldots, r_{1,s_1}\}$ can be detected from the time series data using this method, where s_1 is the number of key-points detected.

In the second method, the difference between P-point weighted moving average and the original time series data is used. The P-point weighted moving average is defined as $\sum_{l=1}^{P}(w_l * u_l)$, where w_l is a weight between 0 and 1, $\sum(w_l) = 1$ and u_l is the l-th entry value in the time series sub-sequence to be filtered. Assume that the time series data filtered by the P-point weighted moving average can be represented by $FT_P =< d_{P,1}, \ldots, d_{P,n} >$. Then, given threshold ϵ_2, if a point meets the condition $abs(d_{P,j}) \leq \epsilon_2$, the point is a key-point, $r_{2,k}$. Next, the following key-point set, K_2, can be obtained as $K_2 = \{r_{2,1}, \ldots, r_{2,s_2}\}$, where s_2 is the number of key-points detected.

As described above, key-points from both methods are merged as a set of T-term candidates. The final set of key points is calculated as, $K = \cup_{i=1}^{2}(K_i)$, where K_i indicates the key points received from different methods. In this case, $i = 2$. Using the proper settings of the two parameters, ϵ_1 and ϵ_2, key-point extraction can be well controlled to characterize a set of time series data.

4.2 Key-Point Features

The feature vectors are calculated for all key-points. The dimensionality of a feature vector is $2N_f + 1$. For a key-point v_{t_p}, at $t = t_p$ in the time series data, the i-th entry $f_{t_p,i}$ of the feature vector is calculated using the following equation.

$$f_{t_p,i} = \sum_{j=1}^{W_f}(v_{(t_{kp}-N_f\times W_f-w_f/2+(i-1)\times W_f)+j-1} - v_{(t_{kp}-N_f\times W_f-w_f/2+(i-1)\times W_f)+j})/(2 \times N_f + 1), \quad (1)$$

where W_f, $(= 2 * w_f + 1)$ is related to the number of time series points, around the key point, which are used to calculate the feature vector.

4.3 T-Term Construction

Both TAX and l-TAX quantize the feature vectors to construct a set of T-terms. Quantizing each dimension by splitting the domain into certain intervals because the dimensionality of the feature vector is large and many quantized cells contain fewer T-terms. The quantization method used by TAX and l-TAX is therefore a clustering method known as the bag-of-key-points model, which is frequently used in image processing research. All key-point feature vectors are clustered by using the K-means clustering method. The clustering depends on the distance between the cluster center and a feature vector. The number of clusters, N_{T-term} is predetermined in the case of the K-means. Each cluster is identified by its T-term ID.

4.4 T-Document Vector Construction

All T-terms are used to construct a T-document vector. T-document vector construction is different in the case of l-TAX. Herein, we discuss only TAX T-document vector construction, and l-TAX T-document vector construction is described in Sect. 5.

TAX uses $tf\text{-}idf$ to construct its T-document. The values of $tf\text{-}idf$ is a product of the term frequency (TF) in a document and the inverse of the number of documents (IDF) containing the term. The following equation is used to calculate the $tf\text{-}idf$ of the i-th document of the j-th term:.

$$tfidf(Tterm_j, tdoc_i, N) = tf(Tterm_j, tdoc_i) \times idf(Tterm_j, N), \quad (2)$$

where $tf(Tterm_j, tdoc_i)$ is the term frequency of the T-term $Tterm_j$ in the T-document $tdoc_i$, $idf(Tterm_j, N)$ is the inverted document frequency, and T_N is the number of total time series. $tf(Tterm_j, tdoc_i)$ and $idf(Tterm_j, N)$ are calculated as follow:

$$tf(Tterm_j, tdoc_i) = \sigma_i freq(Tterm_j)_i$$
$$idf(Tterm_j, N) = log(\frac{N}{numTdoc(Tterm_j)})$$

For $tf(Tterm_j, tdoc_i)$, $freq(Tterm_j)_i$ is the frequency of the T-term $Tterm_j$ in the T-document $tdoc_i$, and σ_i is a parameter to normalize $freq(Tterm_j)_i$. For $idf(Tterm_j, N)$, N is the number of T-documents, and $numTdoc(Tterm_j)$ is the number of T-documents that contains the T-term $Tterm_j$.

4.5 T-Term Assignment

During the retrieval phase, when TAX receives a query, it converts the query time series into its T-document representation using the same method used in the construction phase. TAX extracts key-points from the query time series and calculates their feature vectors. It then assigns T-terms to each of the feature vectors in the following manner.

Assume that, $QK = \{q_{k_1}, \ldots, q_{kN_{qk}}\}$ is a set of query key-points and $F_{QK} = \{f_{QK,1}, ..f_{QK,N_{qk}}\}$ is the set of their feature vectors. For each $f_l \in F_{QK}$, TAX determines T-term $Tterm_{min} = Tterm_m$ such that

$$minimize_m(distance(f_l, center(Tterm_m)).$$

This $Tterm_{min}$ is assigned to the key-point f_l.

4.6 T-Document Retrieval

A query T-document vector $Tdoc_q$ is prepared using its tf-idf calculation. TAX uses this query document to search and output the following set of documents, TQ.

$$TQ = \{Tdoc_m | distance_{Tdoc}(Tdoc_q, Tdoc_m) \leq \epsilon_q, Tdoc_m \in TD\}, \quad (3)$$

where TD indicates all T-documents in the TAX database.

It should be noted that neither the sequence nor the temporal information of a time series is directly used in the T-document retrieval phase. All information related to the temporal aspects is absorbed in the T-document representation with TAX.

5. Sequential Representation of TAX

In Sect. 3, we showed a failure case of TAX. TAX does not consider the term sequences, but such consideration can significantly increase the classification accuracy. To solve this problem, TAX was extended into l-TAX. The motivation of l-TAX is to consider the term sequence based on time series sequential representation. l-TAX is prepared to consider the term sequences during classification, and represents a T-document vector as a sequence of symbols instead of tf-idf counts. l-TAX uses the fundamental idea of TAX and differs from TAX in the manner in which it constructs a T-document vector and calculates the distance between T-documents. In the next section, we describe how l-TAX creates its T-document vector and the distance definition it uses for term sequences.

5.1 Term Sequence Construction

l-TAX uses term-sequences to create a T-document vector, which is generated form the set of T-terms and its corresponding key-points. For each key-point, its T-term id is found to obtain the sequence for a T-document.

Assume that, $K = \{kp_1, kp_2, \ldots, kp_n\}$ is a set of key-points extracted from a time series and ordered by the appearance time in the original time series. $T = \{T_1, T_2, \ldots, T_N\}$ is a set of T-term ids. For, each key-point from K, we find its corresponding term. The corresponding T_{id} from T is used to obtain the term sequence, $Term_{sequence} = \{T_{id,kp_1}, T_{id,kp_2}, \ldots, T_{id,kp_n}\}$. Figure 6 shows the term sequence construction.

l-TAX stores the training data class labels, terms, and term sequences in a database. During the retrieval phase, l-TAX loads these values and uses them to predict the class label of an unlabeled query time series.

5.2 Classification Using LCS

l-TAX converts a query time series into a T-document vector using a sequence representation. l-TAX compares this query T-document with the stored T-documents using the LCS length. It finds the best matched sequence and its class label, and predicts the label as the query time series label.

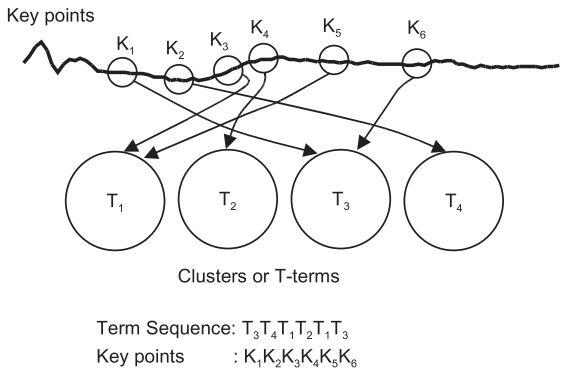


Fig. 6 Term sequence construction.

The LCS problem can be solved using the dynamic programming described in [16], [17]. Assume that $X = (x_1, .., x_m)$ and $Y = (y_1, .., y_n)$ are two term sequences of length of m and n, respectively. A dynamic programming algorithm iteratively builds a $(m + 1) \times (n + 1)$ score matrix LCS, where $LCS[i, j]$, $0 \leq i \leq m$, $0 \leq j \leq n$, is the length of the LCS between two prefixes $X[i] = (x_1, \ldots, x_i)$ and $Y[j] = (y_1, \ldots, y_j)$. The LCS score matrix can be calculated as follows:

$$LCS[i, j] = \begin{cases} 0 & \text{if i = 0 or j = 0} \\ LCS[i-1, j-1] + 1 & \text{if } x_i = y_j \\ max(LCS[i-1, j], & \\ \quad LCS[i, j-1]) & \text{if } x_i \neq y_j \end{cases} \quad (4)$$

The LCS length of two sequences is found in $LCS[m, n]$. In l-TAX, one query can obtain multiple nearest neighbors with the same magnitude. This is because of the distance definition of l-TAX. l-TAX uses a whole match to calculate the LCS length. If two symbols match exactly, we increase the length by 1; otherwise, the length is 0. For this reason, more than one sequence can obtain an equal maximum LCS length with the query. We consider the class labels of all such sequence. The system predicts a query class in which more than 50% of the best matched sequences are labeled.

5.3 Algorithmic Complexity Analysis

The whole process is divided into training and search phases. Algorithms 1 and 2 show the training and retrieval phases of l-TAX [7] respectively. Algorithm 1 is run only once during the training phase. If new training data are available only then it runs again. Algorithm 2 is run for every query time series.

In both phases, the key-point detection and feature calculation take linear time. We maintain a map between the key-point feature and the original time series, which also takes linear time. During a training session, we apply K-means clustering to the feature vectors. K-means clustering

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, w_f: 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}

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** $labels_{train}[best\_matches\_time\_series\_index]$

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 the training data set, and k is the number of terms. The term assignment, document vector generation (TAX), and term sequence generation (l-TAX) also take linear time as we keep the maps for key-points to time series and key-points to the clusters. During the 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. TAX calculates the cosine distance of the query document vector and training document vectors, which is conducted in linear time. The longest common subsequence uses dynamic programming, which takes polynomial time. The query sequence matches against all training sequences. Each match takes $O(nm)$ time, where n and m are the lengths of the sequences.

In this paper, we focus on the accuracy of the proposed method, rather than its efficiency. As explained above, the efficiency of the method is similar to that in other LCSS methods such as PVQA [18] and AMSS [19]. Further details of the efficiency of our method are omitted due owing to the space limitations of this paper.

6. Evaluation

To evaluate the effectiveness of TAX and l-TAX compared to other metrics or similar measures, we performed a simple classification task, which is to assign one of the possible categories to an unknown time series from a known set of categories. This method has been used extensively in previous researches [13], [20]–[23]. We compare TAX and l-TAX with other existing methods such as Euclidean [5], DTW [20], TAX, OTWED [24] and SAX [5]. We chose these methods for a comparison with our own method based on the work in [24]. These methods use the same techniques as TAX and l-TAX. These methods, including TAX/l-TAX use Euclidean distance, dynamic programming, or edit distance in their distance definitions and we performed the same classification task to demonstrate their effectiveness.

6.1 Experimental Setup

6.1.1 Data-Sets

The data-sets were collected from the UCR Time Series Data Mining Archive [20]. These data-sets contain data from twenty different domains. The sources of the time series range from motion capture (GunPoint), to OCR word recognition (50Words), and electrocardiogram measurements (ECG200). Lightning 2 and 7 are lightning data collected from the Fast On-orbit Recording of Transient Events (FORTE) satellite. The data source gives us 5397 training and 18612 test time series data with varying length and different category cardinalities. The smallest time series (Synthetic Control) contains sixty data points, whereas the longest one (Lightning-2) contains 637 data points. Table 6 shows the characteristics of the data-sets.

For each data-set, a training subset is defined along with a testing subset. The classification is performed based on the simple nearest-neighbor decision rule. Initially, we select a training data set containing a time series for which the correct category is given. To assign a category to an unknown time series selected from the testing data-sets, we select its nearest neighbor (using an LCS similarity measure) within the training data-sets, and then assign the associated category to its nearest neighbor.

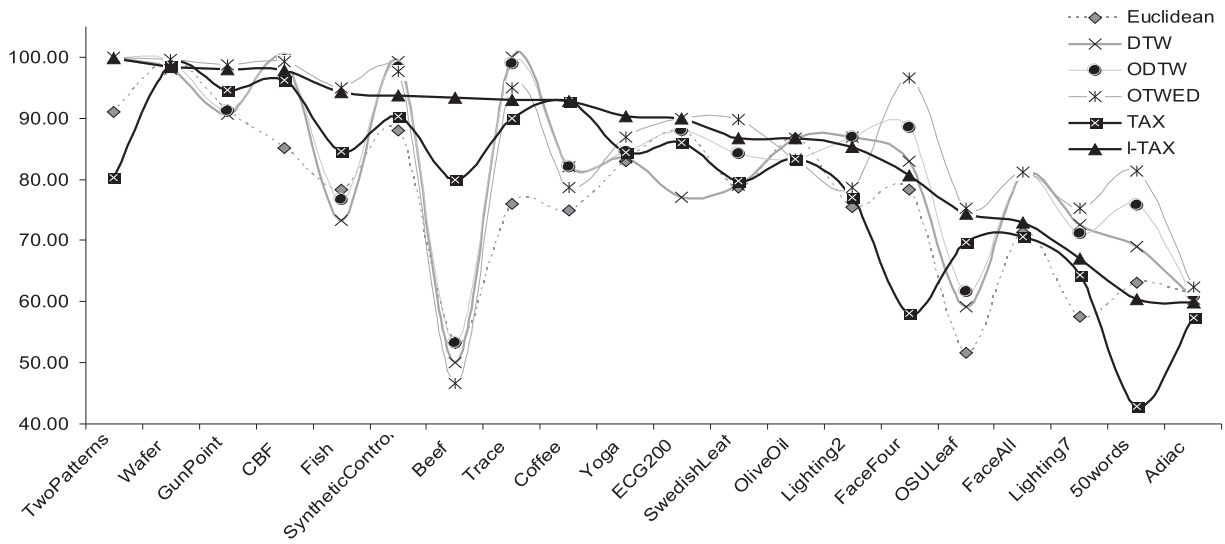


Fig. 7 Accuracy comparison of TAX/l-TAX with other methods (Data for Euclidean, DTW, ODTW, OTWED are taken from [24] and data for SAX are taken from [25]).

Table 6 Characteristics of the short time-series datasets.

Dataset	Categories/Length	|Train|:|Test|
50Words	50/270	450:455
Beef	5/470	30:30
Coffee	2/286	28:28
FaceAll	14/131	560:1690
FISH	7/175	175:175
Lightning2	2/637	60:61
OliveOil	4/570	30:30
SwedishLeaf	15/128	500:625
Trace	4/275	100:100
Wafer	2/152	1000:6174
Adiac	37/176	390:391
CBF	3/128	30:900
ECG200	2/96	100:100
FaceFour	4/350	24:88
GunPoint	2/150	50:150
Lightning7	7/319	70:73
OSULeaf	6/427	200:242
Synthetic	6/60	300:300
TwoPattern	4/128	1000:4000
Yoga	2/426	300:3000

6.1.2 Parameter Settings and Accuracy Measure

Both TAX and l-TAX require certain parameters to be set to produce an optimal output. We train the system using the training data to determine the optimized values for the parameters. We use a leave-one-out cross-validation (in which a time series is selected one at a time from the data-set, the remaining time series data are used as the training set, and then the selected data is used for testing). We arbitrarily set all the parameters and check the accuracy. We perform this test many times on the training data-set taking different parameter sets. We keep the parameters that allows the system to perform better. We use this method to find the optimal values for the key-point threshold parameters ϵ_1 and ϵ_2. We optimize these two thresholds by selecting their values in a manner such that the number of key-points always remains between 5% and 15% of the original time series length. For example, if a time series contains 100 data points then the number of key-points must be at least five and at most fifteen. While training, we found the neighborhood parameter, $wf = 10$ works best for both TAX and l-TAX. The typical weights in the Gaussian averaging operator [26] are used as the weight (w_l) values here to calculate the key points. For the number of terms, we perform the leave-one-out method with different values and keep the best one for obtaining the maximum accuracy. Our evaluations are introduced the same way as proposed in [20]. A simple method for comparing the time series classification methods is based on 1-NN and leave-one-out.

We define our evaluation metric as follows: For a given query q, the category of the query time series (given our prior knowledge) is taken as the correct category ($Correct\_category(q)$), and compared with the category of the K nearest neighbor of q found by TAX or l-TAX ($KNN\_category(q)$). If both categories are equal, then we consider it as a correct classification. For a certain test set, the accuracy is defined as follows:

$$Accuracy = \frac{|Correct\_classification|}{N_T} \times 100\% \quad (5)$$

where, $Correct\_classification = \{q_i | KNN\_category(q_i) = Correct\_category(q_i)\}$, $i = 1..N_T$ and N_T is the number of total time series in the test set. For our experiment, we set $K = 1$. This parameter setting of K is frequently used for comparing the time series classifications [20].

6.2 Results

Figures 7 and 8 show the classification accuracy comparison among TAX/l-TAX and other existing methods. As shown

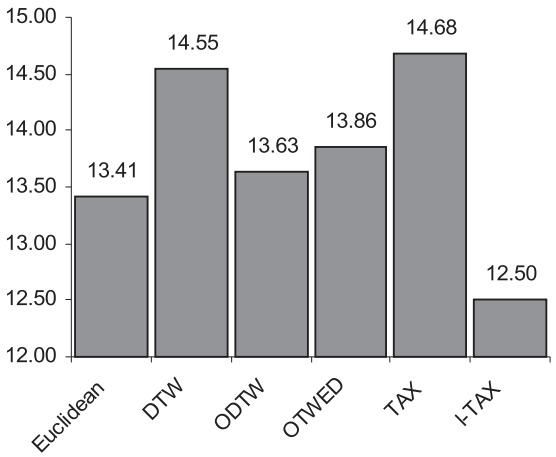


Fig. 8 Standard deviation comparison of TAX/l-TAX with other methods.

in the figures, TAX and l-TAX perform better than other methods. In particular, l-TAX performs significantly better. Compared to TAX, l-TAX increases the accuracy for all data-sets. Although it decreases slightly in the case of the Wafer data-set, this is very insignificant. For the Face (four) and 50 Words data-sets, l-TAX accuracies increase drastically. The performance of TAX was very poor for these data-sets. l-TAX achieves its maximum accuracy for the Two Pattern data-set. Methods other than TAX also performed well for this data-set.l-TAX performs extremely well for some data-sets where the other methods fail to achieve a higher accuracy. For example, l-TAX obtains higher than 90% accuracy for the Beef and Coffee data-sets. Except for TAX, all other methods obtain accuracies of below 80%. The l-TAX classification accuracy seems to be stable on various types of time series.

Figure 8 shows the standard deviation of accuracies for the same data-sets found through different methods. l-TAX receives the lowest standard deviation among them, which is high for TAX compared to the other methods. This lowest standard deviation for l-TAX shows its stability for different types of time series.

l-TAX performs slightly better than OTWED [24], which considers the time warp edit distance with a stiffness adjustment. l-TAX also performs better than the time wrap edit distance and optimized time wrap edit distance on certain data-sets, and significantly better than Euclidean [25] which uses the Euclidean distance to calculate the similarity. The average accuracies of the Euclidean, DTW, ODTW, OTWED, TAX, l-TAX, and SAX methods are 76.63%, 81.56%, 83.28%, 85.56%, 79.07%, 85.78% and 66.04% respectively. SAX evaluation data [25] are shown here only for a comparison of the accuracy results, although this may not be meaningful without considering the storage and processing time. The average results and standard deviations show that l-TAX outperforms all other methods, and uses both time series symbolic representations and non-symbolic representations to classify data.

One of our benchmarks is the Beef data-set. TAX and l-TAX obtain 80.00% and 93.33% accuracy results respectively. All other methods concerned obtain less than 55% accuracy. This is one of the successful cases of the proposed model, which shows that considering all data points of a time series can lead to poor results. Our proposed model uses only the important data points from the time series to build terms similar to the existing document retrieval model, whereas the other methods involve all the data points of a time series without considering the noise. SAX [5] uses a textual approximation of a time series. SAX uses all data points to approximate a time series. Another aspect of this benchmark is a consideration of the term sequence, and the basic difference between TAX and l-TAX lies in their consideration of such a term sequence. The evaluation shows that l-TAX performs better than TAX. The accuracy increases by 13.33% because of the term sequence consideration. TAX uses a tf-idf based approach, where the term sequence does not affect the results. This is because changing the term sequence does not change the magnitude of the document vector. l-TAX uses LCS as its distance measure. LCS is robust to noise and permits some symbols to be unmatched [27] during a distance calculation. l-TAX shows that the term sequence has a direct effect on the accurate retrieval of a time series.

l-TAX shows quite a stable performance for different data-sets, and a generic applicability on various domains. However, the performance of l-TAX on the 50Words and Lightening7 data-sets is worse than for other methods. Moreover, all methods have low levels of accuracy for the Adiac and OSULeaf data-sets. The reason for these poor performances is still undetermined and further investigation is required.

7. Conclusions

In this paper, we proposed both TAX and l-TAX, which use a document retrieval model to classify time series data. To solve some of its fundamental problems, the original method of textual approximation (TAX) was extended, and renamed l-TAX. The main idea of these methods is to construct a set of temporal terms, called T-terms, from a large amount of time series data and use their tf-idf count or sequences to make a tf-idf based document vector or string sequence to find the time series similarity. With l-TAX, a user can obtain desirable time series data sets with higher accuracy than with TAX or any other method. Experimental results show that l-TAX is effective and can be used for a large amount of time series classifications. The evaluation results demonstrate that the proposed methods can perform stably for various types of time series.

As a part of our future work, we plan to explore additional features for automatic parameter selection and to develop pruning and indexing techniques for faster retrieval. We will explore our model on sparse and multidimensional data, particularly in regard to the trajectories. We also plan to compare our methods with other methods in terms of time and space complexity. The performance of l-TAX on cer-

tain data-sets requires further investigation and we therefore plan to extend l-TAX to achieve higher accuracy for these data-sets as well.

Acknowledgment

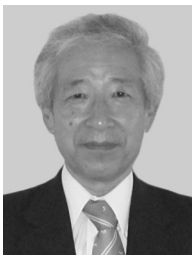
This work was partially supported by KAKENHI #24300039 and by MEXT-Supported Program for the Strategic Research Foundation at Private Universities, 2013-2017.

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," EDBT, pp.276–294, 2006.
[2] R. Agrawal, C. Faloutsos, and A.N. Swami, "Efficient similarity search in sequence databases," FODO, pp.69–84, 1993.
[3] K. Kawagoe, A. Al-Maruf, K. Deng, and X. Zhou, "Searching time series using textual approximation," The Third International Conference on Emerging Databases, pp.121–132, 2011.
[4] 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.
[5] 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.
[6] B.K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," VLDB, pp.385–394, 2000.
[7] A. Al-Maruf, H.H. Huang, and K. Kawagoe, "Time series classification method based on longest common subsequence and textual approximation," The Seventh International Conference on Digital Information Management, pp.130–137, 2012.
[8] H. Shatkay and S.B. Zdonik, "Approximate queries and representations for large data sequences," ICDE, pp.536–545, 1996.
[9] 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.
[10] 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.
[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] S. Seo, J. Kang, and K.H. Ryu, "Multivariable stream data classification using motifs and their temporal relations," Information Sciences, pp.3489–3504, 2009.
[13] E. Keogh, S. Lonardi, and C.A. Ratanamahatana, "Towards parameter-free data mining," International Conference on Knowledge Discovery and Data Mining, pp.206–215, 2004.
[14] E. Keogh, S. Lonardi, and C.A. Ratanamahatana, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," ICDM, pp.623–632, 2006.
[15] J.M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," Twenty-ninth Annual ACM Symposium on Theory of Computing, pp.599–608, 1997.
[16] D. Sankoff, "Matching sequences under deletion/insertion constraints," Proceedings of the National Academy of Sciences, pp.4–6, USA, 1972.
[17] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," International Symposium on String Processing Information Retrieval, pp.39–48, 2000.
[18] W. Qiang and V. Megalooikonomou, "A dimensionality reduction technique for efficient time series similarity analysis," Information systems, pp.115–132, 2008.
[19] T. Nakamura, K. Taki, H. Nomiya, and K. Uehara, "Amss: A similarity measure for time series data," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J91-D, no.11, pp.2579–2588, Nov. 2008.
[20] E. Keogh, Q. Zhou, B. Hu, Y. Hao, X. Xi, L. Wei, and C.A. Ratanamahatana, "The UCR time series classification/clustering homepage," http://www.cs.ucr.edu/~eamonn/time_series_data/, accessed Oct. 29, 2013.
[21] L. Chen, M.T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," SIGMOD, pp.491–502, 2005.
[22] M.D. Morse and J.M. Patel, "An efficient and accurate method for evaluating time series similarity," SIGMOD, pp.569–580, 2007.
[23] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," ICDE, pp.673–684, 2002.
[24] P.F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," Pattern Analysis and Machine Intelligence, pp.306–318, 2009.
[25] 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.
[26] M.S. Nixon and A.S. Aguados, Feature extraction and image processing, Academic Press, 2008.
[27] M. Vlachos, G. Kollios, and D. Gunopulos, "Elastic translation invariant matching of trajectories," Machine Learning, pp.301–334, 2005.
[28] 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.
[29] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," VLDB, pp.792–803, 2004.
[30] 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. Biomed. Eng., vol.52, no.12, pp.2112–2115, 2005.
[31] 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.
[32] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
[33] 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.
[34] 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 Trans. Biomed. Eng., vol.43, no.10, pp.990–1000, 1996.
[35] S.R. Sarangi and K. Murthy, "Dust: A generalized notion of similarity between uncertain time series," Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp.383–392, 2010.

Abdulla Al Maruf received BSc. in Computer Science from North South University, Bangladesh in 2007. He received the M.Eng in Copmputer Engineering at the Ritsumeikan University, Japan in 2012. He has research interests in time series similarity, machine learning and bioinformatics.

Hung-Hsuan Huang received BSc. in Computer Science from National Chen-Chi University, Taiwan in 1998 and MSc. from National Taiwan University, Taiwan. He received his PhD from the Kyoto University in 2009. Currently he is an assistant professor at the Ritsumeikan University, Japan. He has research interest in embodied conversational agent and virtual 3D space. He is a member of JSAI, IPSJ, TAAI, HIS, ACM and IEICE.



Kyoji Kawagoe received B.Eng. and M.Eng in Electronic Engineering from Osaka University in 1975 and 1977, respectively. He also received Ph.D from Tsukuba University in 1992. He joined Ritsumeikan University in 1997, while he had worked for NEC Corporation since 1977. He is currently a full professor of Collage of Information Science and Engineering, Ritsumeikan University. His research interests include multimedia data engineering, ubiquitous computing and network related software research and development. He is a member of IEEE, ACM, ACM SIGMOD, Database Society of Japan, IEICE, and IPSJ.