DSCo: A Language Modeling Approach for Time Series Classification

Daoyuan Li, Li Li, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon

University of Luxembourg
firstname.lastname@uni.lu

Abstract. Time series data are abundant in various domains and are often characterized as large in size and high in dimensionality, leading to storage and processing challenges. Symbolic representation of time series – which transforms numeric time series data into texts – is a promising technique to address these challenges. However, these techniques are essentially lossy compression functions and information are partially lost during transformation. To that end, we bring up a novel approach named Domain Series Corpus (DSCo), which builds per-class language models from the symbolized texts. To classify unlabeled samples, we compute the fitness of each symbolized sample against all per-class models and choose the class represented by the model with the best fitness score. Our work innovatively takes advantage of mature techniques from both time series mining and NLP communities. Through extensive experiments on an open dataset archive, we demonstrate that it performs similarly to approaches working with original uncompressed numeric data.

1 Introduction

Time series data refers to a sequence of data that is ordered either temporally, spatially or in other defined order. Such data are abundant in various domains including healthcare, finance, energy and industry applications. Furthermore, time series data are often characterized as large in size and high in dimensionality [7]. These characteristics of time series data – together with its abundance – has led to various challenges in both storage and analytics. For example, the BLUED non-intrusive load monitoring dataset [1] records voltage and current measurements in a single household for one week with a sampling rate of 12 kHz, leading to a total of tens of billions of numeric readings and making it extremely difficult to mine meaningful patterns in real-time using these raw numeric data. Researchers from EPFL also argue that while smart meter technologies make it possible for utility companies to analyze household energy consumption data in real-time, data acquired by smart meters are often so large that analytic tasks become extremely expensive [28].

Traditionally, researchers have proposed various methodologies to represent time series more efficiently, including dimensionality reduction [9] and numerosity reduction [29] techniques. Another line of research on time series representation focuses on converting numeric values into symbolic form [7], while one of the most prominent approaches is Symbolic Aggregate approXimation (SAX) [14], which adapts both dimensionality and numerosity reduction techniques to transform numeric time series into symbolic representations. Although SAX comes with a distance measure that can be used for nearest neighbor classification, it is still unclear how classification performance will be affected when classifying SAX's symbolic representations of time series.

In this paper, we set to investigate how symbolic representations of time series can tackle time series classification (TSC) challenges. Specifically, we propose a novel TSC approach named Domain Series Corpus (DSCo, pronounced as `disco`), which firstly transforms numeric values into texts and then builds per-class language models from these texts. To classify unlabeled samples, we compute the fitness of each symbolized sample against all per-class models and choose the class represented by the model with the best fitness score. Our work innovatively takes advantage of mature techniques from both time series mining and Natural Language Processing (NLP) communities. Through extensive experiments on an open dataset archive, we demonstrate that our approach not only performs similarly or better than state-of-the-art approaches (which works with original data that possesses much more information), but can also work on reduced data, an essential property to ensure scalability in TSC.

Overall, the contributions of this paper are summarized as follows:

- We bring up a novel method for TSC by leveraging mature techniques from the NLP community. By taking advantage of language modeling techniques we are able to consider both *local* and *global* similarities among time series.
- We have tested our approach extensively on an open archive which contains datasets from various domains, demonstrating by comparison with state-of-the-art approaches that DSCo is performant, efficient and can be generalized.
- We prove that although our approach works with approximated data, DSCo can perform similarly to approaches that work with original uncompressed numeric data.
- We propose a new perspective for TSC: we view time series data as *sentences* (as in natural languages), where some *words* and their *combinations* will define different classes. In this way, we approximate a TSC task to a pseudo language detection problem.

The remainder of this paper is organized as follows. Section 2 briefly surveys related research work to ours. Section 3 provides our intuition and the necessary background information on time series classification as well as preliminaries on language modeling. Section 4 presents the details of our approach, while experiments and evaluation results compared with related work are described in Section 5. Section 6 concludes the paper with directions for future work.

2 Related Work

TSC is a major task in time series mining thanks to its wide application scenarios. As a consequence, there are a plethora of classification algorithms for TSC. Fu [7] has surveyed extensively on time series mining and TSC in his review paper. Due to space limitations, here we only introduce works that are closely related to ours.

Classically, machine learning classification is built by defining two elements: a *distance measure* to compare samples and a *classification algorithm* which implements the method of comparison. For example, in 1NN classification, a given sample is directly compared with samples from the training set. The tested sample will be assigned the class label of the sample from the training set which is the closest to it following a distance measure, for instance, the Euclidean distance. This approach can be expensive if the training dataset is large and thus will lead to a high number of pairwise comparisons.

As stated by Battista et al. “all of the current empirical evidence suggests that simple nearest neighbor classification is very difficult to beat” [2]. The core of nearest neighbor classifiers lies in defining an accurate distance measure. To perform best, kNN classifiers leverage the Dynamic Time Warping (DTW) distance which mitigates problems caused by distortion in the time axis [4, 20]. Thanks to its maturity and performance, DTW based 1NN classification has become one of the most prominent TSC approaches. Although recent empirical comparison [23] reveals that the Time Warp Edit Distance (TWED) [15] performs statistically more accurate than DTW. Other common distance measures include Euclidean distance, Edit Distance on Real sequence (EDR) [5] and Minimum Jump Cost (MJC) [24].

One issue with DTW, however, is that it focuses on finding *global* similarities, i.e., the overall curve *shape* of two time series in the time dimension. As a result, it requires applications to specify a proper warping window size or to properly align data samples. To solve this issue, *shapelets*-based algorithms [30, 16, 19, 8] try to find phase-independent defining subsequences in time series based on *local* similarities, so that such subsequences can be used as discriminatory features (or *primitives*) for classification. Our approach differs from *shapelets* since we consider both *global* and *local* similarities using language modeling techniques.

Some other approaches also borrow paradigms from the text mining community. One of the most prominent ones is the *bag-of-words* approach, which inspired the *bag-of-features* [3, 27] approach for TSC. SAX-VSM [22] also takes advantage of bag-of-words approach and builds term frequency-inverse document frequency (*tf**idf*) vectors in its training phase. It defines a similarity measure of two vectors (that are constructed from original series) based on their inner product.

The closest work related to DSCo are probably compression-based TSC approaches. Xi et. al. proposes numerosity reduction techniques [29] in order to speed up DTW distance calculation, while our approach takes advantage of both numerosity reduction and dimensionality reduction techniques to reduce the size of time series data. Furthermore, DSCo is not based on DTW and it is computationally more efficient than DTW, although there has been efforts to optimize DTW’s time complexity [10, 25]. Finally, this approach is inspired by our previous work [12], where we consider TSC in the household appliance profiling domain and draw an analogy between dialects in natural languages and different patterns of electricity signals each type of electrical appliance emits.

3 Background and Key Intuition

In this section, we briefly introduce the mechanism behind SAX and how SAX is traditionally used for TSC tasks. Then we present language modeling and how it can be used in combination with *SAX-ified* strings. In the meanwhile, we introduce our intuitions on tackling TSC with symbolic representation and language modeling.

3.1 Text Representation with SAX

Time series classification often involves learning what patterns are associated to a specific class of data readings. Consider the case of two samples from the `BirdChicken` [6] dataset illustrated in Fig. 1, where bird/chicken images have been transformed into time series (illustrated in gray curves). By observing the readings in different segments of the time series and which segment succeeds another, one can immediately summarize the characteristics of each class.

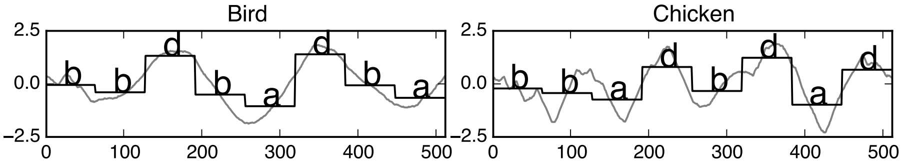

Fig. 1: Illustrative examples of how SAX works. The gray curves represent time series samples from the `BirdChicken` dataset. The black curves illustrate PAA representations of two time series samples where dimension is reduced from 512 to 8 ($n = 512$ and $s = 8$). Alphabets on top of the PAA curves are SAX representations of two time series samples with an alphabet size of four.

However, to better visualize the significant patterns, one must first reduce the dimensionality of the time series by considering only relevant variations. To that end, we leverage Piecewise Aggregate Approximation (PAA) [11] which can reduce the time series from n dimensions to s dimensions by dividing the data into s segments of equal size. The data reduction representation is then a vector of the mean values of the data readings per segment. Let $\bar{V} = \bar{v}_1, ..., \bar{v}_s$ be this vector where each $\bar{v}_i$ is computed by equation 1 (for more information please refer to [11, 14]).

$$\bar{v}_i = \frac{s}{n} \sum_{k=\frac{n}{s}(i-1)+1}^{\frac{n}{s}i} v_k \tag{1}$$

The black curves in Fig. 1 illustrate the PAA representation of the original time series readings. Although very simple, the PAA dimensionality reduction enables to make an analogy with languages: the succession of segments (at different levels) in a time series is comparable to a succession of words and expressions which may define the vocabulary and phrases of a language. The intuition becomes straightforward: if one can collect a dictionary of the segments and their co-occurrence frequencies, pairwise comparison between samples can be effective. In order to benefit from the plethora of algorithms that exist in the NLP field, we must transform the PAA representation into a more symbolic representation with alphabets. To that end, we build on the Symbolic Aggregate approXimation (SAX) [14].

As an example of symbolic representation that fits with our intuition for language modeling using PAA values of time series segments, we consider the Symbolic Aggregate approXimation (SAX) [14]. SAX was initially brought up to transform real valued time series data into a sequence of alphabets, i.e., a string. It has then been proven especially efficient for motif (repeated patterns) discovery tasks. For example, it is advantageous to use SAX in order to find variable-length motifs [13, 21]. Fig. 1 illustrates the SAX representations of samples from the two classes in the `BirdChicken` dataset. The string sequences yielded can now be considered as text with expressions from a specific "language".

3.2 Language Modeling

Given a string representation of time series data, we can apply language modeling to assess whether it fits the model of a class. Language models are used to answer questions such as *"How likely a string of words from a language vocabulary is good phrasing in this language?"*. A statistical language model is a probability distribution over strings of a corpus [18]. Thus, any sequence of words W has a probability score $P(W) = P(w_1, ..., w_n)$ in the language model, indicative of its relative validity within a language.

N-gram language models are common means of language modeling. In the simplest case of *unigram* models (1-gram models), the probability score of the sequence of words W is approximated to the product of the probabilities of each word. Equation 2 provides the formula for computing this score.

$$P(W) = P(w_1, ..., w_n) \approx \prod_{i=1}^{n} P(w_i) \tag{2}$$

Bigram models put conditions on the previous word to account for the likelihood of co-occurrence between two words (for instance, *beer drinkers* appears more often than *beer eaters*). The probability score of the sequence W is then approximately the product of conditional probabilities of words with their previous peer. It is computed by the formula in Equation 3.

$$P(W) = P(w_1, ..., w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1}) \tag{3}$$

Since in a given corpus it is possible that certain bigrams are never observed beforehand, it is reasonable to use a back-off mechanism to take into account only their unigram probabilities. Although N-grams models can be theoretically insufficient because language has long-distance dependencies (for instance, some words may co-occur in a sentence but not directly following each other in the sequence: *"the computer which I just bought and setup in my room crashed"*), these models have been shown to be efficient in practice and used in various fields including speech recognition, author attribution and malware detection.

4 Domain Series Corpora for TSC

Since symbolic representation of time series data is a promising mechanism to tackle the numeriosity and high dimensionality issue in the era of big data, we investigate how symbolic representation and language modeling can be used for TSC. Recall that DSCo builds on the simple intuition that time series patterns in a specific class of a given domain can be differentiated from other class patterns, which is similar to NLP methods that distinguish texts from different languages or dialects. The assumption is thus that the language model extracted from the samples of a specific class will be descriptive and discriminative enough to differentiate it from another language model within the same domain. DSCo therefore consists of building a corpus of *words* representative of time series subsequences (or *segments*) for a given domain and the associated language models for its classes.

Fig. 2 illustrates the steps for building per-class Domain Series Corpora for a specific domain. First of all, data readings of time series samples from each class are transformed into texts. Next, language modeling is applied on these texts to extract the corresponding language models, so that afterwards these models can be used to test and classify unlabeled samples.

4.1 Data Representation as Texts

As described in Section 3, we create symbolic representations for real-valued samples. In DSCo we have leveraged SAX for this task. It is nonetheless possible to leverage another symbolic representation algorithms for time series. The output of this step is a string representation for each time series sample.

4.2 Language Model Inference

DSCo explores a training set of time series to extract meaningful patterns of segments by studying their occurrence frequencies. Once time series are represented as texts, a language model can be built to summarize each time series class. To build a language model for a time series class, DSCo generates its corresponding dictionary by collecting *words* that appear in the training set. A large body of work have been proposed in the NLP literature on how to obtain such dictionaries.However, since the symbolic representations generated by SAX have no word boundaries, we need to break them into smaller pieces first by employing a corpus acquisition mechanism. This is common procedure for some natural languages such as Chinese, which has no obvious word boundaries.

One approach to dictionary acquisition is to break the sequences using an annealing algorithm, which is a probabilistic and non-deterministic algorithm that randomly permutes the possible segmentations and searches the whole solution space with the best segmentation until thresholds are met according to an objective function. Such an approach is able to find word boundaries with reasonable accuracy given a large training set. Unfortunately, this algorithm is highly time-consuming. Besides, time series training sets are seldom sufficiently large to accommodate this algorithm. As a result, we extract *words* from the symbolic representations using a naïve sliding window method described in Algorithm 1. This algorithm collects all possible sub-strings of length w within a string, so that no descriptive segment is left

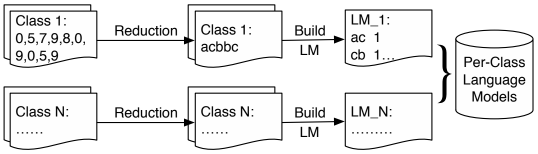

Fig. 2: Process for building language models in DSCo.

uncaptured from the original time series. For example, we can break string `abccc` into the following 2-alphabet segments: `ab`, `bc`, `cc` and `cc`.

Algorithm 1 Extract *words* from a string (*S*) using a sliding window (of length *l*).

```
procedure ExtractWords($S, l$)
    $words \leftarrow \varnothing$
    for $i \leftarrow 0, GetLength(S) - l + 1$ do
        $word \leftarrow SubString(S, i, l)$                ▷ Sub-string of size $l$
        $words \leftarrow words \cup \{word\}$
    return $words$
```

Next, in order to better preserve the descriptive information that time series generally come together in a sequence, we also compute the frequencies of n-grams. We build n-gram language models for each time series class in our training set. This process is illustrated in Algorithm 2. In order to be generic, we define a minimum (`minWL`) word length and a maximum word length (`maxWL`): The intuition behind this is that 2-alphabet segments may be generic but not descriptive, while segments with larger length may be descriptive but not generic enough. These extreme values can be easily determined with enough domain knowledge. For instance, if we assume that electrocardiogram (ECG) patterns generally have similar lengths, the `minWL` and `maxWL` values can be predefined to avoid producing noisy segments.

Algorithm 2 Build language models (*LMs*) from a list (SL) of $(string, label)$ pairs.

```
procedure BuildLM($SL, minWL, maxWL$)
    $LMs \leftarrow \varnothing$
    for all $(string, label) \in SL$ do
        if $NGrams_{label} \notin LMs$ then
            $NGrams_{label} \leftarrow \varnothing$
        for $wl \leftarrow minWL, maxWL$ do
            $words \leftarrow ExtractWords(string, wl))$
            for all $ngram \in GetNGrams(words)$ do
                InsertOrIncreaseFreq($NGrams_{label}, ngram$)
        $LMs \leftarrow LMs \cup NGrams_{label}$
    ConvertFreqToProbability(LMs)
    return $LMs$
```

When all n-grams in every per-class language model are counted, we convert their frequencies into probabilities within each language model. Note that frequencies may need normalization when there are different number of instances in each class. Otherwise n-gram probabilities will be biased and lead to classification errors in the next step.

4.3 Classification

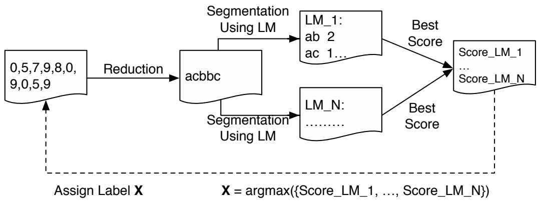

Fig. 3: Illustration of DSCo's classification process.

In DSCo, classification is performed by checking which language model is the best fit for the tested sample, as shown in Fig. 3. First, similar to the training phase, each test sample is reduced to a string using the same algorithms and parameters. Then, in order to test model fitness, each language model – which summarizes the characteristics of all samples from a given class – is used to segment the time series' text representation. To be computationally efficient, we consider a probabilistic language modeling approach with bigrams as introduced in Section 3. We compute the segmentation score as the product of conditional probability of all segmented words following Equation 3. If a bigram is not known in the language model, we just back-off to the unigram probability values for this specific segment. Since there are different ways to segment the text according to a specific language model, we only consider the best segmentation score that can be obtained with each language model. That is, each language model segments the sample and keeps its best segmentation score, then all language models compare their scores and the winning language model's class label will be applied to the sample.

Algorithm 3 Given language models, find the best way (with the maximum probability) to segment a string (*S*).

```
Require: global $minWL$ and $maxWL$
procedure SPLITSTRING(S)
    $P \leftarrow \varnothing$
    $sl \leftarrow GetLength(S)$
    for $l \leftarrow minWL, min(sl, maxWL)$ do
        if $sl - l \geq minWL$ then
            $P \leftarrow P \cup \{(Slice(S, 0, l), Slice(S, l, sl))\}$
    return $P$
procedure SEGMENT(S, prev)
    UpdateViterbiTable()                ▷ Dynamic programming to avoid repetitive computing
    if $GetLength(S) < minWL$ then
        return 0
    $Sgs \leftarrow \varnothing$
    for all $(h, t) \in SplitString(S)$ do
        $Sgs \leftarrow Sgs \cup \{P(S|prev) * Segment(t, h)\}$
    return $max(Sgs)$
```

5 Evaluation

In order to evaluate the feasibility and performance of our approach, we have implemented DSCo and tested on an open dataset archive. We first reduce time series data using SAX and show that SAX distance-based 1NN under-performs DTW. Then we investigate the value added by DSCo on top of SAX. Finally, through comparison with Euclidean- and DTW-based 1NN classification, we show that DSCo is indeed performant. We have open sourced our implementation[1] in order to increase reproducibility.

To explore the performance of the DSCo approach and investigate the extent of its applicability, we consider the Time Series Classification Archive [6] from University of California, Riverside. The datasets contained in this archive are popular within the TSC community, allowing for a reliable comparison baseline. Besides, the archive includes error rates for DTW- and Euclidean-based nearest neighbor classification as a performance benchmark for TSC. The UCR archive is composed of two sub-archives: `Pre_Summer_2015_Datasets` and `Newly Added Datasets` which include datasets from various fields, ranging from electrocardiograms (ECG) to intra-species image recognition data. We tested on the latter (which contains 39 different datasets) because its file format and internal data structures are consistent, making it possible to conduct batch processing in a content-agnostic manner. Furthermore, both sub-archives have similar dataset diversity and some datasets for specific domains (for example, ECG data) appear in both sub-archives. Specifically, these 39 datasets have various number of classes from 2 to 60 and different number of training and testing instances from 20 to 8,926, with time series lengths varying from 80 to 2,079.

5.1 Reducing Data using SAX

In order to validate SAX's data reduction performance, we take those `Newly Added Datasets` from UCR and transform the numeric data records into symbols. We have set the maximum time series length to 100 and varied SAX's alphabet size from 3 to 20, so that we can take advantage of SAX's dimensionality and numeriosity reduction mechanism. Since SAX can be viewed as a lossy compression function, we evaluate how well the original information are kept for TSC using 1NN classification and compare the classification performance between DTW distance and SAX's internal distance measure as defined in [14]. SAX's distance measure is essentially a variance of Euclidean distance except that the distance between two alphabets are predefined in SAX's look-up table. For instance, for an alphabet size of 4, $dist(a, b) = 0$ and $dist(a, c) = 0.67$. Fig. 4 presents the 1NN classification comparison, where the solid lines indicate SAX distance-based 1NN classification accuracy for each of the 39 datasets, and the dashed lines shows DTW distance-based 1NN classification performance.

Here we show the classification accuracy of DTW-based 1NN because it is the most mature and widely used distance measure among the research community. Furthermore, classification performance using DTW is readily available from the UCR archive. Finally, using DTW requires one to explicitly setting its warping size parameter. In Fig. 4 we show DTW's performance with the best warping size (parameter space is 100: warping size varies from 0 to 100 percent of original time series length) that is found in [6]. When comparing SAX distance to DTW's best performance, we believe it is fair to present SAX's best accuracy results as well (parameter space is 18: alphabet size varies from 3 to 20). In this case DTW-based 1NN outperforms SAX in $69.2\%(27/39)$ datasets

[1] `https://github.com/serval-snt-uni-lu/dsco`

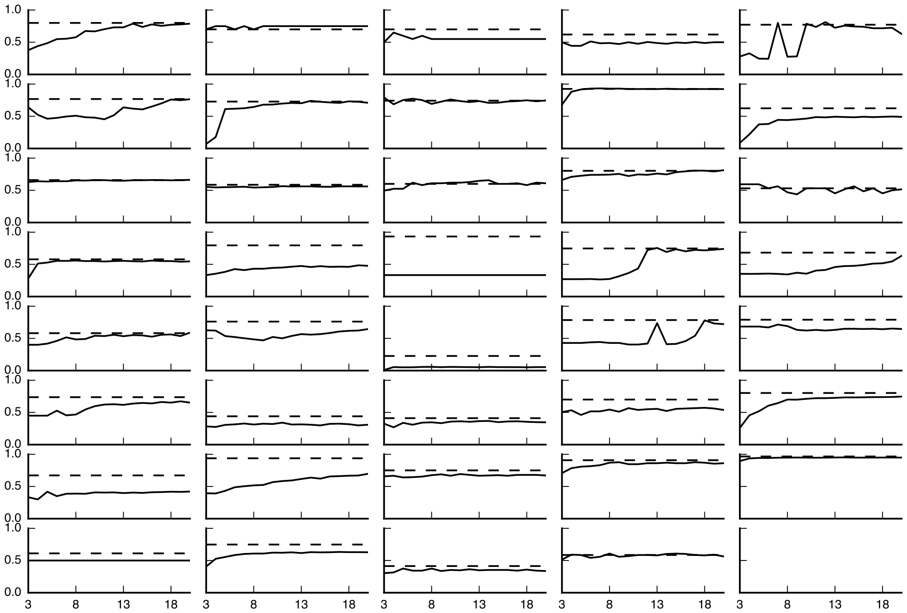

Fig. 4: 1NN classification accuracy comparison between DTW (dashed) and SAX (solid) distance.

indicating that SAX distance-based 1NN classification indeed under-performs DTW, possibly due to two major facts, namely SAX's lossy reduction of the original numeric data and SAX distance's inability to consider time series' global similarities. As a result, we take advantage of DSCo and take into account both global and local similarities of time series to counter SAX's lossy reduction. And as we shall demonstrate later, DSCo indeed classifies time series more accurately than SAX distance-based 1NN classification.

5.2 Implementation and Setup

Normally, DSCo's classification process can be extremely expensive due to the need of recursively dividing strings in to smaller pieces and segmenting these sub-strings. However, a good implementation may take advantage of the Viterbi algorithm [26] – which is in essence a dynamic programming approach – to avoid redundant computation. In addition, the classification process calculates best segmentation scores based on the text segmentation algorithm provided in [17], which works exceptionally well for NLP text segmentation tasks and has a computational complexity of $O(nL^2)$, where n is the length of a testing string, and L is the maximum word length. Finally, DSCo computes segmentation scores using mainly bigram probabilities, only falling back to unigram probabilities when a specific bigram is not found.

Recall that time series longer than 100 have been arbitrarily reduced to 100-alphabet strings during dimensionality reduction, in order to speed up the classification process; and we have varied SAX's alphabet size from 3 to 20 to search for the best text representation. Since DSCo requires two parameters: a $(minWL, maxWL)$ tuple, we experiment with three sets of parameter settings: short segments $(minWL = 2, maxWL = 10)$, long segments $(minWL = 11, maxWL = 20)$ and short-long combined $(minWL = 2, maxWL = 20)$. Results illustrated in Fig. 5 show that DSCo's performance are relatively consistent regardless short or long segments (*words*) are configured. As a result, in the following comparisons, we fix $minWL$ to 2 and $maxWL$ to 20.

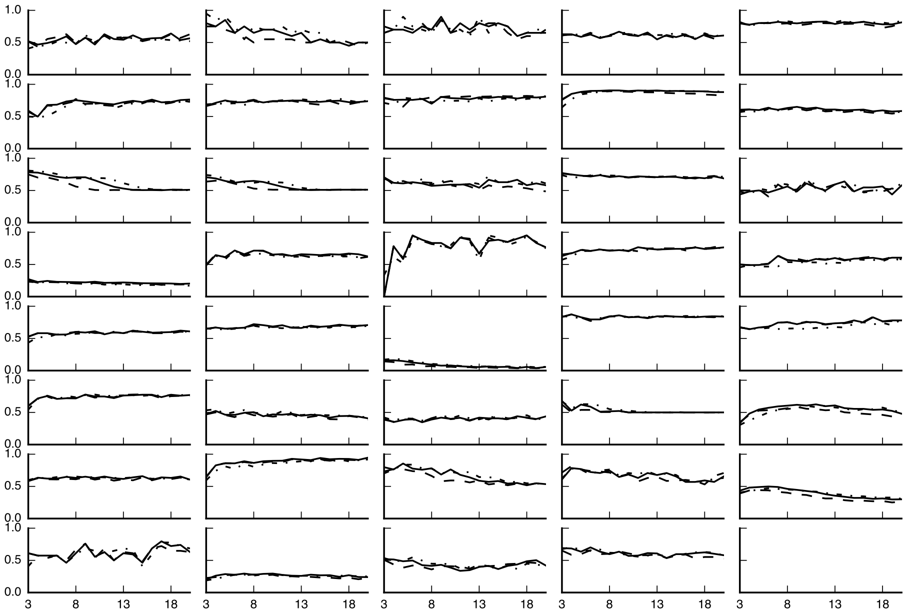

Fig. 5: DSCo's classification accuracy with different parameter settings: short segments (dashed-dotted lines), long segments (dashed lines) and combined (solid lines).

5.3 Comparison of Classification Performance

In the first round of experiments, we investigate the value added by the language modeling to the use of the SAX representations. Since SAX already comes with a distance metric which was demonstrated to yield better performance than the Euclidean distance on real-valued time series data [14], we compare DSCo against SAX-distance-based 1NN classification. Table 1 shows respectively the classification accuracy and alphabet size when each classification approach performs best. DSCo outperforms in *74% (29/39)* datasets. The results further shows no explicit correlation between DSCo and SAX-distance-based 1NN classification performances, both obtained mostly with unrelated alphabet sizes. Since both DSCo and SAX-distance-based 1NN leverage SAX, these results thus suggest that DSCo's performance cannot be directly attributed to the usage of SAX representation, but rather to the language modeling process.

In the second round, we compare the classification results of DSCo against the benchmark 1NN with Euclidean distance. In *74% (29/39)* of the datasets, DSCo performs better in terms of classification accuracy. We also compare the improvement brought by DTW – the state-of-the-art approach – over Euclidean distance: DTW-based 1NN beats

Table 1: Classification accuracy comparison between the best performance of DSCo and 1NN with SAX distance, where $|\alpha|$ is the alphabet size when best performance is achieved.

Data-	SAX's Best		DSCo's Best		Data-	SAX's Best		DSCo's Best		Data-	SAX's Best		DSCo's Best	
set	$\lvert\alpha\rvert$	acc.	$\lvert\alpha\rvert$	acc.	set	$\lvert\alpha\rvert$	acc.	$\lvert\alpha\rvert$	acc.	set	$\lvert\alpha\rvert$	acc.	$\lvert\alpha\rvert$	acc.
1	14	**0.79**	11	0.62	14	20	**0.81**	3	0.77	27	11	0.34	4	**0.53**
2	4	0.75	5	**0.95**	15	3	0.59	14	**0.64**	28	14	0.37	17	**0.45**
3	4	0.65	9	**0.90**	16	8	**0.56**	3	0.27	29	18	0.57	3	**0.68**
4	6	0.51	9	**0.67**	17	19	0.49	6	**0.72**	30	20	**0.75**	11	0.64
5	12	0.81	12	**0.83**	18	3	0.33	6	**0.95**	31	20	0.42	19	**0.66**
6	20	**0.77**	20	**0.77**	19	13	0.76	17	**0.77**	32	20	0.70	15	**0.94**
7	14	0.74	20	**0.76**	20	20	**0.63**	7	**0.63**	33	11	0.69	5	**0.85**
8	3	**0.79**	3	0.78	21	20	0.59	20	**0.62**	34	9	**0.88**	5	0.78
9	7	**0.93**	11	0.91	22	20	0.64	8	**0.72**	35	14	**0.95**	6	0.50
10	19	0.50	9	**0.65**	23	8	0.06	4	**0.18**	36	3	0.50	17	**0.80**
11	20	0.66	3	**0.78**	24	18	0.78	4	**0.87**	37	17	**0.63**	7	0.30
12	11	0.56	3	**0.71**	25	7	0.71	17	**0.83**	38	5	0.38	6	**0.54**
13	14	0.66	3	**0.70**	26	19	0.67	18	**0.78**	39	8	0.61	6	**0.71**

1NN with Euclidean distance in *64% (25/39)* datasets. We further compare directly DSCo with DTW-based 1NN classification. Fig. 6 illustrates the results where we consider the best performance with DTW (i.e., with the best warping window size) and the best performance of DSCo (i.e., with the best SAX alphabet size). As shown, DSCo performs similarly to DTW in most datasets. DSCo appears to have good performance in image recognition tasks (for example, `BeetleFly` and `BirdChicken`), while it performs badly for some datasets where the training set has unbalanced distribution of different classes (e.g., `WordSynonyms`), making it difficult for DSCo to extract discriminatory n-grams. Overall, DSCo performs better than DTW in *64% (25/39)* datasets, indicating good classification performance.

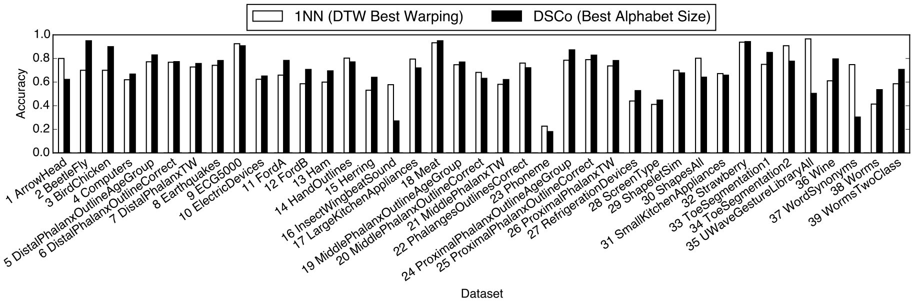

Fig. 6: Overall accuracy comparison between 1NN with DTW distance and DSCo.

Finally, we investigate if pruning bigrams affect classification accuracy. Fig. 7 illustrates the case with the `ECG5000` dataset, where we removed all bigrams that has frequencies lower than the mean value from each language model and use the pruned language mod-

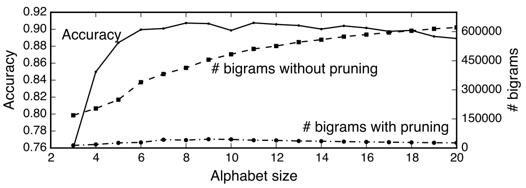

Fig. 7: For the `ECG5000` dataset, classification accuracy remains the same after pruning up to 95.8% bigrams.

els for classification. In some cases the pruned language models has less than 5% bigrams from the original ones. Yet surprisingly, the classification accuracy remains exactly the same. This indicates high redundancy in the language models and pruning techniques can be used in order to reduce memory footprints and speed up the classification process.

5.4 Time and Space Complexity

SAX has a linear time and space complexity when transforming real-valued time series into PAA and then to text representation. DSCo's language model inference step processes each training sample constant times and stores the models to external storage, resulting an $O(n)$ time and space complexity. We have shown that the fitness metric algorithm for classification in DSCo has a computational complexity of $O(nL^2)$ (in our experiments, $n \leq 100$ and $L = 20$). In comparison, the state-of-the-art DTW-based metric has a computational complexity of $O(n^2)$, although with LB_Keogh lower bounding technique this complexity can be reduced to $O(n)$ [25]. However, for complete classification processes, where a tested sample is compared against all samples from the training set, the 1NN approach using DTW with LB_Keogh yields a complexity of of $O(mn)$ where m is the size of training set. DSCo, on the other hand, has a complexity of $O(cnL^2)$ where c is the number of classes ($c \ll m$ for most scenarios).

Next, we compute the space complexity of DSCo . Given a time series sample T_i of length n, Algorithm 1 produces maximum $n - l + 1$ unique *words* of length l. We denote T_i's l-sized words as $W_{i,l}$, then

$$1 \leq |W_{i,l}| \leq n - l + 1 \tag{4}$$

We denote the set of *unigrams* from a specific class of time series consisting of m instances ($T = \{T_1, ..., T_m\}$) as W, then

$$m \leq |W| \leq m \sum_{l=minWL}^{maxWL} (n - l + 1) \tag{5}$$

Since we use *bigrams* in our DSCo implementation, it thus has a space complexity of $O(m^2n^2)$, equivalent to that of shapelet-based algorithms, which also have a time complexity of $O(m^2n^2)$ for their full search procedures [8].

5.5 Limitations

We have demonstrated DSCo's performance through extensive experiments and complexity analysis. However, it also has its own limitations. Since DSCo essentially summarizes training samples into models, it performs best when there are a sufficient number of training samples in each class. Furthermore, our current implementation is built on top of SAX, tweaking SAX's parameters – for example, alphabet size – may require users to look inside data samples in order to achieve best performance. Nevertheless, we believe DSCo has offered a new perspective for TSC tasks and its limitations can be tackled in the near future by employing more advanced symbolization and corpus acquisition algorithms.

6 Conclusions and Future Work

In this work, we have brought up a novel approach named DSCo for time series classification. It works on symbolized time series data and builds per-class language models, against which testing samples are fitted in order to predict their corresponding class labels. Through extensive experiments we are able to prove that DSCo performs similarly or better than some state-of-the-art TSC approaches that works with original numeric data, namely 1NN with Euclidean and DTW distance. By taking advantage of mature algorithms from the NLP community, DSCo is able to achieve close-to-linear time complexity, which will be a great advantage for real-time applications.

Our future work will focus on further improving DSCo's performance, including reducing computation overhead and memory consumption by more effectively pruning n-grams in language models, improving classification accuracy and finding key defining subsequences for better user comprehension. In addition, other symbolization techniques can be taken advantage of to make DSCo more generalized and parameter free.

References

[1] Anderson, K., Ocneanu, A., Benitez, D., Carlson, D., Rowe, A., Berges, M.: Blued: A fully labeled public dataset for event-based non-intrusive load monitoring research. In: Proceedings of the 2nd KDD workshop on data mining applications in sustainability. pp. 1–5 (2012)

[2] Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: SDM. vol. 11, pp. 699–710 (2011)

[3] Baydogan, M.G., Runger, G., Tuv, E.: A bag-of-features framework to classify time series. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(11), 2796–2802 (2013)

[4] Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370 (1994)

[5] Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 491–502. ACM (2005)

[6] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive (July 2015), www.cs.ucr.edu/~eamonn/time_series_data/

[7] Fu, T.c.: A review on time series data mining. Engineering Applications of Artificial Intelligence 24(1), 164–181 (2011)

[8] Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery 28(4), 851–881 (2014)

[9] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems 3(3), 263–286 (2001)

[10] Keogh, E., Wei, L., Xi, X., Lee, S.H., Vlachos, M.: Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In: Proceedings of the 32nd international conference on Very large data bases. pp. 882–893. VLDB Endowment (2006)

[11] Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 285–289. ACM (2000)

[12] Li, D., Bissyandé, T.F., Kubler, S., Klein, J., Le Traon, Y.: Profiling household appliance electricity usage with n-gram language modeling. In: The 2016 IEEE International Conference on Industrial Technology (ICIT 2016). IEEE, Taipei (March 2016)

[13] Li, Y., Lin, J.: Approximate variable-length time series motif discovery using grammar inference. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining. p. 10 (2010)

[14] Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and knowledge discovery 15(2), 107–144 (2007)

[15] Marteau, P.F.: Time warp edit distance with stiffness adjustment for time series matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 306–318 (2009)

[16] Mueen, A., Keogh, E., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1154–1162. ACM (2011)

[17] Norvig, P.: Natural language corpus data. In: Segaran, T., Hammerbacher, J. (eds.) Beautiful data: the stories behind elegant data solutions, pp. 219–242. O'Reilly Media, Inc. (2009)

[18] Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 275–281 (1998)

[19] Rakthanmanon, T., Keogh, E.: Fast shapelets: A scalable algorithm for discovering time series shapelets. In: Proceedings of the thirteenth SIAM conference on data mining (2013)

[20] Ratanamahatana, C.A., Keogh, E.: Three myths about dynamic time warping data mining. In: Proceedings of SIAM International Conference on Data Mining. pp. 506–510 (2005)

[21] Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S., Lerner, M.: Grammarviz 2.0: a tool for grammar-based pattern discovery in time series. In: Machine Learning and Knowledge Discovery in Databases, pp. 468–472. Springer (2014)

[22] Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model. In: IEEE 13th International Conference on Data Mining. pp. 1175–1180. IEEE (2013)

[23] Serrà, J., Arcos, J.L.: An empirical evaluation of similarity measures for time series classification. Knowledge-Based Systems 67, 305–314 (2014)

[24] Serra, J., Arcos, J.L.: A competitive measure to assess the similarity between two time series. In: Case-Based Reasoning Research and Development, pp. 414–427. Springer (2012)

[25] Smith, A.A., Craven, M.: Fast multisegment alignments for temporal expression profiles. In: Proceedings of the 7th Annual International Conference on Computational Systems Bioinformatics. vol. 7, pp. 315–326. World Scientific (2008)

[26] Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2), 260–269 (1967)

[27] Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Mining and Knowledge Discovery 26(2), 275–309 (2013)

[28] Wijaya, T.K., Eberle, J., Aberer, K.: Symbolic representation of smart meter data. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. pp. 242–248. ACM (2013)

[29] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on Machine learning. pp. 1033–1040. ACM (2006)

[30] Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)