DSCo-NG: A Practical Language Modeling Approach for Time Series Classification

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

Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
{daoyuan.li,tegawende.bissyande,jacques.klein,yves.letraon}@uni.lu

Abstract. The abundance of time series data in various domains and their high dimensionality characteristic are challenging for harvesting useful information from them. To tackle storage and processing challenges, compression-based techniques have been proposed. Our previous work, Domain Series Corpus (DSCo), compresses time series into symbolic strings and takes advantage of language modeling techniques to extract from the training set knowledge about different classes. However, this approach was flawed in practice due to its excessive memory usage and the need for *a priori* knowledge about the dataset. In this paper we propose DSCo-NG, which reduces DSCo's complexity and offers an efficient (linear time complexity and low memory footprint), accurate (performance comparable to approaches working on uncompressed data) and generic (so that it can be applied to various domains) approach for time series classification. Our confidence is backed with extensive experimental evaluation against publicly accessible datasets, which also offers insights on when DSCo-NG can be a better choice than others.

1 Introduction

Time series data usually refer to temporally or spatially ordered data, which are abundant in numerous domains including health-care, finance, energy and industry applications. Besides their abundance, time series data are becoming increasingly challenging to efficiently store, process and mine useful information due to their high dimensionality characteristics. In order to tackle these challenges, researchers have proposed many approaches to model time series more efficiently. Compression-based techniques are especially promising and have been adopted in many recent studies, including dimensionality reduction [9, 21, 11] and numerosity reduction [24]. Symbolic Aggregate approXimation (SAX) [15] is an approach that is capable of both dimensionality and numerosity reduction. Among all time series data mining tasks, time series classification (TSC) has received great interests from researchers and practitioners thanks to its wide application scenarios including speech recognition, medical diagnosis, etc.

Our previous work, Domain Series Corpus (DSCo) [13] for TSC, takes advantage of SAX to compress real-valued time series data into text strings and builds per-class language models as a means of extracting representative patterns in the training phase. To classify unlabeled samples, we compute the fitness of each symbolized sample against all per-class models by finding the best way to segment this sample and choose the class represented by the model with the best fitness score. We also prove that although DSCo works with approximated data, it can perform similarly to approaches that work with original uncompressed numeric data. One issue with DSCo, however, lies in its excessive memory usage when calculating the fitness score of one sample against language models, which makes it impractical for real-world applications.

In this paper, we set to improve DSCo's time and space complexity and propose a next generation of DSCo: DSCo-NG . We follow our initial intuition that time series data are similar to *sentences from different languages or dialects*, but apply a more efficient approach to find nuances of difference from these *languages*. Specifically, unlike in DSCo where we try to find the best way to recursively segment time series, DSCo-NG breaks time series into smaller segments of the same size, and this simplification of the classification process also leads to simplified language model inference in the training phase. Overall, the contributions of this paper are summarized as follows:

– We propose a new practical language modeling-based approach for time series classification, which has a linear time complexity and small memory footprint. Previously DSCo works optimally on a High Performance Computing (HPC) platform, e.g. ULHPC [20], while DSCo-NG can virtually run efficiently on any personal computers thanks to its low complexity.
– 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 and first generation DSCo that DSCo-NG is both performant and efficient.
– We investigate the performance of DSCo-NG by scrutinizing the characteristics of datasets and provide insights in application scenarios when DSCo-NG could be a better choice than other approaches.
– We offer a new perspective for TSC: traditional TSC approaches compare instances against instances, which can be computationally inefficient when the training dataset is large, while our approach aggregates training sets into models and compares the fitness of instances to such models, making comparisons more efficient.

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

2 Background

In this section, we briefly introduce time series, TSC, SAX and DSCo. For a more detailed information on DSCo, the readers are encouraged to refer to [13]. Traditionally, time series data refers to temporally ordered data, e.g., data sequences that are related to time. However, data mining community [7] embraces a broader definition, relaxing the time aspect and incorporating any ordered sequences. For instance, images may also be transformed into time series representation [25]. In this paper, we define a time series $T = t_0, t_1, ..., t_{n-1}$, where t_i ($0 \leq i \leq n-1$) is a real-valued number and that T has a length of n, i.e., $|T| = n$.

TSC is a common category of tasks that involves learning from a training dataset and applying the learned knowledge to classify instances from a testing dataset, where instance classes or labels are often unknown or purposefully hidden. TSC tasks are commonly found in various application domains such as image and speech recognition, medical analysis, industrial automation, etc. Many techniques have been proposed for TSC, including k-Nearest Neighbors, shapelets [25], and bag-of-features [2]. In practice, the Nearest Neighbor (1NN) approach has been proven to work very well [1], especially when combined with a good time series distance metrics such as Dynamic Time Warping (DTW) [4] and Time Warp Edit Distance (TWED) [16, 19].

In the literature of time series data mining, real-valued data are sometimes transformed into symbolic representations, so as to potentially benefit from the enormous wealth of data structures and algorithms made available by the text processing and bioinformatics communities. Besides, symbolic representation approaches make it easier to solve problems in a streamed manner [15]. Finally, many algorithms target discrete data represented by strings over floating point numbers. Symbolic Aggregate approXimation (SAX) [15] is one such technique that is popular among the community [14, 17]. It can perform both dimensionality reduction and numeriosity reduction on time series and transforms real valued time series data into a string of alphabets.

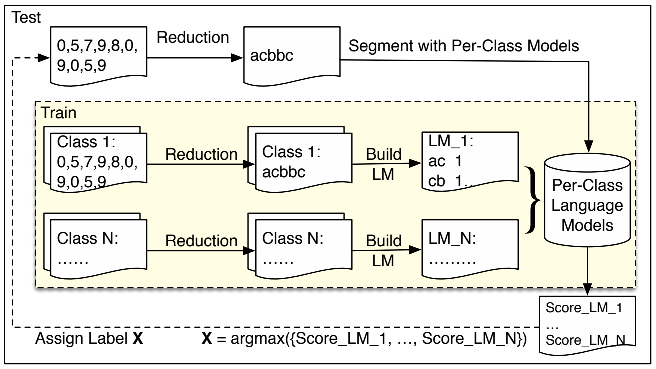

Fig. 1: Flow chart illustration of DSCo's training (in yellow) and testing process.

Once time series data are compressed into strings, DSCo [13] builds per-class unigram and bigram language models which contains artificial words and phrases that have been extracted from the training dataset. Fig. 1 illustrates how DSCo works. When classifying unlabeled instances, DSCo calculates the fitness scores of one instance against all per-class language models, and the fittest model's class label will be assigned to this instance. DSCo builds on the simply intuition that time series signals are comparable to sentences and phrases from natural languages and dialects in the real-world: each dialect have their unique words and patterns, which is similar to distinguishable features in time series. In DSCo, we try to recursively segment an instance using a Viterbi algorithm until we find the best way to divide such instance with a given language model. Due to this intensive process, DSCo has an almost linear time complexity and space complexity of $O(m^2n^2)$, where m is the number of instances in the training set and n is the length of time series.

3 Related Work

Due to TSC's wide application scenarios, there are a plethora of algorithms made available by the research community. An extensive review of time series mining has been done by Fu [7]. Here we only survey the works that are closely related to ours due to space limitation. Since DSCo-NG is a compression-based approach, we introduce related approaches that also takes advantage of time series compression techniques.

There are basically two methods for compressing time series, i.e., dimensionality reduction that works on the time axis and numerosity reduction that works on the value axis. Dimensionality reduction mechanisms include Piecewise Linear Representation (PLR) [8], Piecewise Aggregate Approximation (PAA) [9], and methods that keeps only perceptually important points (PIP) [6]. Our previous work [11] takes advantage of Discrete Wavelet Transform for dimensionality reduction. On the value axis, Xi et al [24] have proposed using numerosity reduction to speed up TSC, and Lin et al have proposed SAX [15], which converts real-valued data into a symbolic form. Note that it is possible to apply both dimensionality reduction and numerosity reduction using SAX.

Symbolic representation of time series has opened a new avenue for TSC since it makes it possible to borrow paradigms from the text mining community. For instance, the *bag-of-words* approach has inspired the *bag-of-features* [2, 22] and SAX-VSM [18] approach for TSC. Furthermore, Representative Pattern Mining (RPM) [23] compresses time series to strings using SAX and then tries to identify the most representative patterns in the training set. These patterns are then used to match against testing instances during classification. Unlike RPM, DSCo does not try to find which patterns are representative or not. Instead, we evaluate testing instances' fitness to each class in an overall perspective.

Note that our compression-based approach is not to be confused with compression-based time series similarity measures [10], which compares the compression ratios of time series under the assumption that compressing similar series would produce higher compression rates than compressing dissimilar ones.

Finally, as a part of our smart building project, a language modeling approach, which inspired the idea of DSCo, was applied for household electric appliance profiling [12], where language models are used to classify and maintain profiles of different appliances.

4 Next Generation Domain Series Corpus for TSC

Since our approach is based on a simple intuition that if we abstract time series classes to *languages*, these *languages* will be descriptive so that it is able to differentiate instances or *sentences*. In practice, we firstly harvest descriptive language models of different patterns from a training corpus. Later in the classification phase, these language models are used to find out which instances are likely to be written in a corresponding language.

The main complexity of original DSCo lies in the classification process, where testing instances are recursively segmented in order to produce the best segmentation result using a language model, in DSCo-NG we try to break the testing instances into sub-sequences of the same length. Then we calculate the product of bigram probability of these sub-sequences. This scheme is inspired by the intuition that when using a sliding widow of size w to iterate over the training set, all possible unigrams and bigrams are already captured within the language model of a specific class. As a result, there is no need to use a sliding window of variable length during the classification process, thus reducing the classification complexity. To better illustrate how DSCo-NG works, we detail it in three steps in the subsections below.

4.1 Compressing Time Series into Texts

There are potentially many approaches that can compress time series data into texts. For instance, one may think of creating a mapping from range of values to alphabets. However, for the benefit of reusing existing mature techniques, we have leveraged SAX for this task. Recall that SAX is capable of both dimensionality and numerosity reduction, as long as the required length and cardinality parameters are specified. Previously with DSCo we arbitrarily reduced the dimensionality of all long (of size larger than 100) time series to 100, for the sake of computational efficiency. Here for DSCo-NG we do not conduct dimensionality reduction since DSCo-NG is efficient enough, allowing us to remove one parameter (or heuristics-based decision) from our processing pipeline. However, as our previous study [11] suggests, conducting dimensionality reduction can potentially increase overall classification accuracy.

4.2 Extracting Language Models

Once time series are compressed to texts, a language model can be extracted to summarize each time series class. Since the text representation does not have word boundaries, we need to create artificial words. To that end, we employ a sliding window mechanism that generates such words. In order to facilitate reader understanding, we reproduce the procedure from [13] in Algorithm 1. This algorithm collects all possible sub-strings of length w within a string, so that no descriptive segment is left uncaptured from the original time series. For example, we can break string `abcde` into the following 2-alphabet words: *ExtractWords(abcde, 2)* produces an output of $[ab, bc, cd, de]$.

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

```
1: procedure ExtractWords($S, w$)
2:     $words \leftarrow \varnothing$
3:     for $i \leftarrow 0, GetLength(S) - w + 1$ do
4:         $word \leftarrow SubString(S, i, w)$          ▷ Sub-string of size $w$
5:         $words \leftarrow words \cup \{word\}$
6:     return $words$
```

Next, we build ngram language models for each time series class in our training set, which is illustrated in Algorithm 2. Unlike DSCo that requires a minimum word length and a maximum word length to capture words, here we use a single length w. Note that the probability of ngrams are calculated independently, since different classes may have different number of training instances.

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

```
1: procedure BUILDLM($SL, w$)
2:     $LMs \leftarrow \varnothing$
3:     for all $(string, label) \in SL$ do
4:         if $NGrams_{label} \notin LMs$ then
5:             $NGrams_{label} \leftarrow \varnothing$
6:         $words \leftarrow ExtractWords(string, w))$
7:         for all $ngram \in GetNGrams(words)$ do
8:             InsertOrIncreaseFreq($NGrams_{label}, ngram$)
9:         $LMs \leftarrow LMs \cup NGrams_{label}$
10:    ConvertFreqToProbability(LMs)
11:    return $LMs$
```

4.3 Classifying Unlabeled Instances

As mentioned earlier, classification in DSCo-NG is performed by checking which language model is the best fit for the tested sample. Specifically, we compare the sample's fitness scores to each model, which is calculated following the ngram statistical language model probability as shown in Equation 1.

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-(n-1)}, ..., w_{i-1}) \quad (1)$$

In practice, bigrams ($n = 2$) and trigrams ($n = 3$) are most prominent [3]. We have opted for the bigram model due to its simplicity for both the language model extraction process and fitness score calculation, which is approximated as $P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i|w_{i-1})$. During the classification process, we need to break time series strings into words. Unlike original DSCo which breaks *sentences* into variable sized *words*, here we adopt the same sliding window size w as the uniform word length. As we shall show later, this simplified process yields similar classification accuracy but greatly reduces the complexity compared with DSCo.

4.4 Time and Space Complexity

During the preprocessing phase, SAX has a linear time and space complexity when transforming real-valued time series into text representation. When extracting language models in the training phase, each training sample is went through once and models are stored to external storage, resulting an $O(n)$ time and space complexity. Finally, the classification process go through testing samples constant times with language models loaded from external storage, yielding linear time complexity. Language models loaded to memory has a theoretic complexity of $O(\alpha^w)$ where α is the alphabet size used when using SAX to compress real-valued data, and w is the length of *artificial words*. In practice, language models seldom exceed a few megabytes, due to the fact that time series in a domain have a very limited number of *words*.

DSCo-NG's real advantage comes when the training set is large. Given a training set of m_1 time series of length n, when classifying a testing set of m_2 instances, traditional kNN approaches have to conduct $m_1 \times m_2$ pairwise comparisons. Even when using a linear similarity measure such as Euclidean distance, the overall time complexity goes up to $O(m_1 m_2 n)$. On the other hand, DSCo-NG would only have a computational complexity of $O(cm_2 n)$ where c is the number of classes and $c \ll m_1$, making DSCo-NG a magnitude faster than kNN. And this is indeed great improvement even compared with DSCo which has a time complexity of $O(cm_2 nw^2)$ and space complexity of $O(m_1^2 n^2)$.

5 Experimental Evaluation

In order to evaluate performance of our new approach, we have implemented DSCo-NG and tested it on an open dataset archive. To facilitate reproducibility, we have open sourced our implementation with full documentation and tutorials on GitHub[1]. We opt for testing with the UCR Time Series Classification Archive [5] for three reasons: 1) this archive has a large number of publicly accessible datasets; 2) these datasets are from a wide range of domains, from environmental monitoring to medical diagnosis; 3) it comes with precomputed classification accuracy rates for DTW-based 1NN, which is the most widely used similarity measure in the research community and has become the *de facto* state-of-the-art benchmark for TSC. In the experiments below we consider 39 datasets from the *Newly Added Datasets* sub-archive because of its uniform file format and structure.

5.1 Implementation and Setup

In theory, when calculating ngram probabilities, the larger n is, the more accurate these probabilities will be. However, in practice it is seldom the case, due to the lack of training data and the rise of complexity when n becomes larger. As a result, our implementation considers the bigram model with unigram fallback as a trade-off between efficiency and accuracy. Note that falling back to unigrams may not always work, when specific unigrams are missing from the training set. In this case it is necessary to employ a penalty mechanism to offset the influence of such unigrams. From our experience, these missing unigrams' probability could be set as a constant of low probability value, so that the missing probabilities do not overwhelm the existing ones and lead to inaccurate classification.

5.2 Parameter Optimization

Ideally, time series mining approaches should have as few parameters as possible, even parameter-free, so as to avoid presumption on data [10]. In reality it is extremely difficult to achieve. For instance, even the popular DTW distance requires a warping window size to be set in order to produce optimal results. In DSCo-NG , we essentially have two parameters: the cardinality of SAX alphabet when compressing real-valued data to text strings and the sliding window size or length of artificial words. Normally, approaches

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

based on SAX have to specify both the cardinality and a PAA size to which time series are reduced. Since DSCo-NG does not necessarily need dimensionality reduction, we only need to fix for a suitable cardinality, i.e., a good alphabet size that keeps sufficient information during time series compression. To that end, we try to reduce time series using different cardinality values from 3 to 20, which is range supported by major SAX implementations. For length of artificial words, we also fix a range to 2 to 20 in order to avoid extremely long words, in order to limit the size of language models.

Fig. 2 presents the classification accuracy from four datasets across different domains. As shown, although these four datasets have different characteristics in terms of training dataset size, time series length and number of classes, there is a clear trend when high classification accuracy is achieved. That is, generally good accuracy is achieved with small to medium SAX alphabet size and the alphabet size has more impact than the word length (imagine projecting the 3D plots to the 2D plane defined by the alphabet size and accuracy axis). This is extremely useful to narrow down the parameter space, even though in fact our parameter space is already small ($18 * 19 = 342$ combinations in total). Note that there are other methods available for finding the optimal parameters. For instance, in [23] the authors have adopted an algorithm named DIRECT. Thanks to the small parameter space and efficiency of DSCo-NG we employ a brute force approach for finding the best parameters for different datasets. Naturally, there is not a single parameter setting that guarantees good performance, since different datasets can be totally different in number of classes, size, time series lengths and variation amplitude. However, it is indeed possible to set the same parameters for datasets with similar characteristics.

5.3 Comparison of Classification Performance

Now that we have fixed the parameters for DSCo-NG , here we set to compare its performance with its predecessor DSCo and the state-of-the-art approach DTW-based 1NN classifier. Fig. 3 presents the classification results. It clearly demonstrates that DSCo-NG outperforms its predecessor. In fact, in *90% (35/39)* of the datasets, DSCo-NG is more or equally accurate compared with DSCo, indeed suggesting performance improvement in accuracy. This is probably due to the fact that DSCo tries to find the best way to segment time series; however, with insufficient training data this segmentation process will result in suboptimal segmentation and thus not as high accuracy. Furthermore, we note that in *72% (28/39)* of the datasets, DSCo-NG also outperforms the state-of-the-art DTW-based 1NN, indicating its superiority in specific datasets. Besides, we would like to remind the readers that DSCo-NG is potentially more scale than 1NN based approaches, especially for datasets with a large training set, e.g., the *ElectricDevices* dataset.

We have demonstrated the performance of DSCo-NG through complexity analysis and extensive experiments. Although DSCo-NG outperforms our previous work in vast majority of tested datasets, it remains unclear why DSCo-NG outperforms DTW-based 1NN in certain datasets while underperforms in other ones. To this end, we investigate in which scenarios DSCo-NG performs better. Obviously the size of training set can be an important factor, because our model-based approach has to capture from different and a large number of instances the representative patterns, while for instance-based approaches – e.g. kNN – one representative instance could potentially help accurately

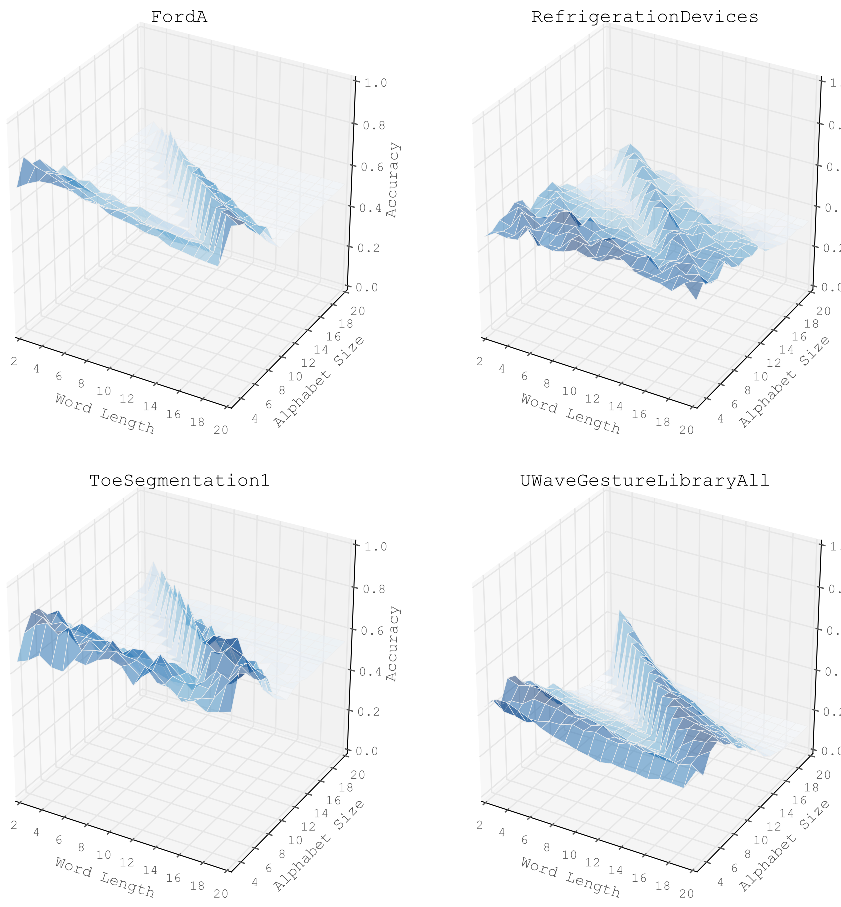

Fig. 2: 3D surface plots of classification accuracy with different parameters, darker blue indicates higher accuracy.

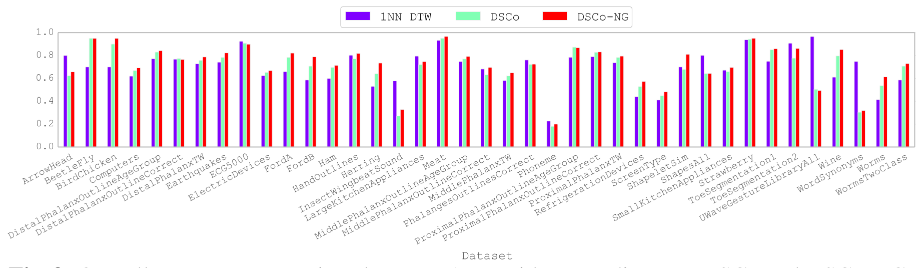

Fig. 3: Overall accuracy comparison between 1NN with DTW distance, DSCo and DSCo-NG.

classifying all similar instances. This is a major reason why DSCo-NG greatly underperforms 1NN for the *WordSynonyms* dataset, which has many (25) classes but very few

(267) training instances. Besides, some classes in this dataset has as few as two instances, making the language model extraction highly inaccurate for DSCo-NG .

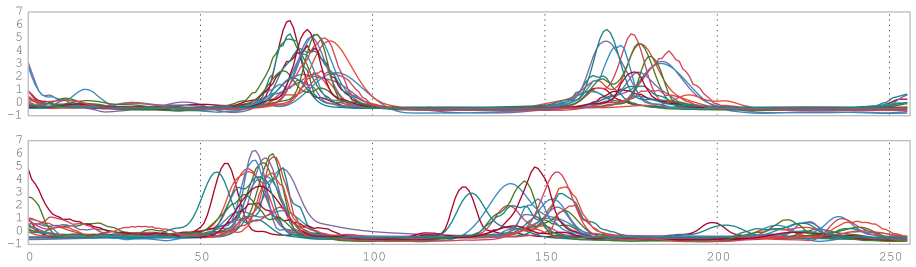
Fig. 4: All instances of two classes (1 and 5) from *InsectWingbeatSound*'s training set.

Besides training set size, in this study we found another important factor that lies in how small segments constitute a time series. Fig. 4 shows why DSCo-NG does not perform well for *InsectWingbeatSound*: these two classes consist of similar segments installed in different positions of time series. Thus DSCo-NG will consider these segments as the same word unless we set an extremely long word length. Similarly, DSCo-NG underperforms for *UWaveGestureLibraryAll* because instances in this dataset are composed of three different segments.

Finally, we demonstrate with one example why DSCo-NG outperforms DTW-based 1NN. Consider the two classes from the *FordA* dataset as shown in Fig. 5. It is obvious that visually it is impossible for a human being to distinguish these two classes, because there are two many samples that are not properly aligned like in Fig. 4. As a result, for 1NN classifier, these samples could be distracting so that it fails to find similar samples given a testing instance. However, DSCo-NG is able to aggregate samples within a class so that it finds the overall descriptive way to differentiate different classes.

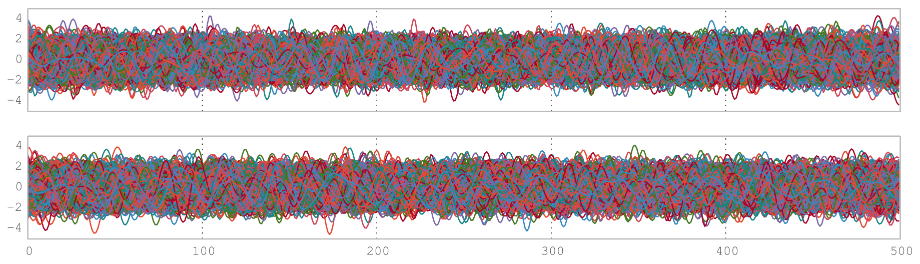
Fig. 5: All instances of two classes (-1 and 1) from *FordA*'s training set.

6 Conclusions and Future Work

In this study we have improved our previous work DSCo and propose a new approach for TSC. Through complexity analysis and extensive experiments, we show that DSCo-NG is both efficient and performant when comparing with DSCo and the state-of-the-art

DTW-based 1NN. Besides, DSCo-NG does not require datasets to be properly aligned, as a result it can save time and efforts preparing for time series data, and result in better classification accuracy with not properly aligned data. Finally, unlike DTW-based 1NN and similar approaches, DSCo-NG can work with data of variable length, which make it suitable for streaming applications.

Since DSCo-NG uses SAX to discretize real-valued time series to text representations, there can be other symbolization techniques to replace SAX and make DSCo-NG parameter-free. In the future, we plan to investigate such opportunities and study the impact of different symbolization techniques on the performance of DSCo-NG.

Acknowledgment

The authors would like to thank Paul Wurth S.A. and Luxembourg Ministry of Economy for sponsoring this research work.

References

1. Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: SDM. vol. 11, pp. 699–710 (2011)
2. 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)
3. Bellegarda, J.R.: Statistical language model adaptation: review and perspectives. Speech communication 42(1), 93–108 (2004)
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, 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/`
6. Chung, F.L., Fu, T.C., Luk, R., Ng, V.: Flexible time series pattern matching based on perceptually important points. In: International joint conference on artificial intelligence workshop on learning from temporal and spatial data. pp. 1–7 (2001)
7. Fu, T.C.: A review on time series data mining. Engineering Applications of Artificial Intelligence 24(1), 164–181 (2011)
8. Keogh, E.: Fast similarity search in the presence of longitudinal scaling in time series databases. In: Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence. pp. 578–584. IEEE (1997)
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., Lonardi, S., Ratanamahatana, C.A.: Towards parameter-free data mining. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 206–215. ACM (2004)
11. Li, D., Bissyande, T.F., Klein, J., Le Traon, Y.: Time Series Classification with Discrete Wavelet Transformed Data: Insights from an Empirical Study. In: The 28th International Conference on Software Engineering and Knowledge Engineering (2016)
12. Li, D., Bissyande, 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. pp. 604–609. IEEE, Taipei (2016)

13. Li, D., Li, L., Bissyande, T.F., Klein, J., Le Traon, Y.: DSCo: A Language Modeling Approach for Time Series Classification. In: The 12th International Conference on Machine Learning and Data Mining. New York (2016)
14. 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)
15. 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)
16. 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)
17. 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)
18. 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)
19. Serrà, J., Arcos, J.L.: An empirical evaluation of similarity measures for time series classification. Knowledge-Based Systems 67, 305–314 (2014)
20. Varrette, S., Bouvry, P., Cartiaux, H., Georgatos, F.: Management of an academic hpc cluster: The ul experience. In: Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014). pp. 959–967. IEEE, Bologna, Italy (July 2014)
21. Wang, Q., Megalooikonomou, V.: A dimensionality reduction technique for efficient time series similarity analysis. Information systems 33(1), 115–132 (2008)
22. 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)
23. Wang, X., Lin, J., Senin, P., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Rpm: Representative pattern mining for efficient time series classification. In: Proceedings of the 19th International Conference on Extending Database Technology (2016)
24. 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)
25. 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)