





# Text to Time Series Representations: Towards Interpretable Predictive Models

Mattia Poggioli<sup>1</sup>, Francesco Spinnato<sup>2,3</sup>(✉) , and Riccardo Guidotti<sup>1,3</sup> 

<sup>1</sup> University of Pisa, Pisa, Italy

{mattia.poggioli,riccardo.guidotti}@unipi.it

<sup>2</sup> Scuola Normale Superiore, Pisa, Italy

francesco.spinnato@sns.it

<sup>3</sup> ISTI-CNR, Pisa, Italy

francesco.spinnato@isti.cnr.it

**Abstract.** Time Series Analysis (TSA) and Natural Language Processing (NLP) are two domains of research that have seen a surge of interest in recent years. NLP focuses mainly on enabling computers to manipulate and generate human language, whereas TSA identifies patterns or components in time-dependent data. Given their different purposes, there has been limited exploration of combining them. In this study, we present an approach to convert text into time series to exploit TSA for exploring text properties and to make NLP approaches interpretable for humans. We formalize our Text to Time Series framework as a feature extraction and aggregation process, proposing a set of different conversion alternatives for each step. We experiment with our approach on several textual datasets, showing the conversion approach's performance and applying it to the field of interpretable time series classification.

**Keywords:** Time Series Classification · Interpretable Machine Learning · Natural Language Processing · Explainable AI

## 1 Introduction

In recent years, both Time Series Analysis (TSA) and Natural Language Processing (NLP) have seen a surge in popularity [2, 8, 14]. NLP has found numerous applications, including machine translation, email spam detection, information extraction and summarization, and question-answering [14]. Meanwhile, the development of time series classifiers [2] and the increasing availability of time-dependent data such as electrocardiogram records, motion sensor data, climate measurements, and stock indices [8] have fueled interest in TSA. Despite the individual growth of NLP and TSA, there has been limited exploration into combining these two fields, which usually have different goals. NLP focuses mainly on enabling machines to manipulate and generate human language, whereas TSA identifies local patterns or components in time-dependent data. However, they also share similarities since the text, from a human perspective, remains a

sequence of spoken or written expressions rather than a comprehensive machine-readable representation. This work explores the benefits of integrating TSA into NLP to make it more interpretable for humans.

NLP relies on text representation techniques to convert text into machine-readable input for classification, clustering, and sentiment analysis [31]. Typically, text encoding involves transforming text into vectors representing its content. Traditional approaches build an explicit representation of the distributional properties of the text, using raw frequencies like bag-of-words [9], or exploiting tf-idf [12], and n-grams [28]. State-of-the-art techniques are based on transformers [26], which can better capture semantic and syntactic relationships between words and sentences. The vectorial representations generated by these models, known as embeddings, are denser and lower-dimensional than previous models, even if not interpretable [22]. On the other hand, TSA techniques can capture the temporal evolution of sequences of data points measured at regular intervals [6]. By considering temporal dependencies, TSA can be used for various purposes, such as descriptive analysis, clustering, classification, and forecasting [16].

In this paper, we investigate the impact of converting text observations into time series observations to solve interpretable text classification through time series representations. In particular, we exploit interpretable models originally developed for time series [25] as interpretable text classifiers. In the literature, state-of-the-art time series classifiers are mainly black-box models [2], not interpretable from a human standpoint. We instead focus on time series classification through shapelets [30], i.e., subsequences that allow for interpretable predictions based on local similarities in shape. Hence, we propose using shapelets in NLP by turning texts into time series. To perform this transformation, we design and implement TOTS, a framework to turn Text to Time Series. TOTS exploits a range of alternatives for converting a text into multivariate time series, including sentence embeddings [17, 20], sentiment scores [11], and linguistic features [5, 13, 19]. In this way, we can leverage implicit representations of language and concrete linguistics variables to represent language vectorially. Then, TOTS adopts aggregation techniques to compress multivariate time series into univariate ones. By compressing time series, we can identify shapelets on 1-d signals that are easier to analyze and interpret than multivariate ones, shedding light into many domains, such as sentiment analysis over time, event detection, social media trends, etc.

Overall, this work contributes to the fields of TSA and NLP by *(i)* proposing a formalization of text to time series conversion, *(ii)* exploring dimensionality reduction and aggregation techniques that can effectively convert multivariate time series into univariate, *(iii)* testing different text to time series conversions through a novel evaluation metric, and *(iv)*, showing the effectiveness of the approach in the field of interpretable text/time series classification. The rest of this work is structured as follows. Section 2 discusses related works at the intersection between TSA and NLP. Section 3 introduces notions useful for describing the proposed transformations, detailed in Sect. 4. Section 5 presents the experimental results, and Sect. 6 concludes the paper.

## 2 Related Works

A proper intersection between TSA and NLP lies in analyzing texts produced within a time window. Works in this domain build time series by extracting features from each text and condensing them within each timestep to represent a time-dependent phenomenon. In [10], the authors constructed a sentiment scoring rule from the count of positive and negative words in multiple social media texts, resulting in an event-driven, irregularly spaced time series. In [1], the authors combined text mining and time series to analyze sequences of dated documents, such as news articles, and extracted correlations and patterns among frequently used words. Differently from the described approaches, we analyze documents individually and introduce time by splitting each text content.

Other works are focused on the relationship between features (such as market sentiment) extracted from Twitter data and financial trends. In [21], the authors analyzed correlations between stock-market events and features extracted from micro-blogging messages, relying on overall activity measures (e.g., number of posts, re-posts) and graph-related indices (e.g., number of connected components, degree distribution). In [27], the authors used market sentiment and text mining techniques for financial time series, proposing a hybrid model that combines the conventional ARIMA model with a support vector regressor method to extract valuable insights from the market sentiment. Similarly, in [18], it was proposed ST-GAN, combining financial news texts and numerical data to predict stock trends. In [4], the authors used a flexible multiple-output Gaussian process to analyze multimodal statistical causality between cryptocurrency market sentiment and price processes, proposing an NLP framework for interpretable sentiment indices as inputs for time-series models. Differently from these approaches, we convert each text into individual time series representations, moving away from the financial domain and focusing on classification rather than forecasting.

To the best of our knowledge, the only approach for mapping text to time series is  $T^3$  [29].  $T^3$  uses combinations of granularity, n-grams, and different space-filling curves to assign appropriate numeric values to each character. When applied to the “record linkage” problem,  $T^3$  achieved good accuracy with considerable speed-ups. Our study goes beyond this work by focusing on mapping text at the sentence level, allowing for incorporating multiple types of features, including advanced models like sentence embeddings.

## 3 Setting the Stage

In order to keep our paper self-contained, we report in this section a brief overview of concepts necessary to comprehend our proposal. We define a text corpus and each of its components, i.e., documents and sentences, as follows:

**Definition 1.** A *corpus* is a structured set of textual documents, represented as a collection  $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$  where  $\mathcal{T}$  is the corpus and  $T_i$  an individual document within the corpus. A *document*  $T$  is a sequence of sentences, where each sentence is denoted by  $S_j$ , and the entire text is represented as  $T = \{S_1, S_2, \dots, S_m\}$ ,

where  $m$  is the total number of sentences in the text. Finally, a *sentence*  $S$  is an ordered sequence of words.

For example, a corpus might be a set of film or theater scripts, a document might be a scene or an opera, and a sentence might be an actor's line. Also, we can consider a set of songs as a corpus, a song as a document, and a verse of the song as a sentence. We can now define the Text Classification problem as follows:

**Definition 2.** Given a corpus  $\mathcal{T}$  with a vector of finite integer labels (or classes) assigned to each document  $\mathbf{y} \in \mathbb{N}^n$ , the *Text Classification Problem* is the task of training a function  $f$  from the space of possible inputs  $\mathcal{T}$  to a probability distribution over the class values in  $\mathbf{y}$ .

In the following, we establish a connection between text and time series, defining them in similar and coherent ways.

**Definition 3.** A *time series dataset*  $\mathcal{X} = \{X_1, \dots, X_n\} \in \mathbb{R}^{n \times d \times m}$  is a collection of  $n$  time series. A *time series*  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_d\} \in \mathbb{R}^{d \times m}$  is a set of  $d$  signals. A *signal*, or dimension,  $\mathbf{x} = \{x_1, \dots, x_m\} \in \mathbb{R}^m$  is a sequence of  $m$  real-valued observations sampled at equal time intervals. When  $d = 1$ , a time series is *univariate*, while if  $d > 1$ , the time series is *multivariate*.

Consequently, we define the Time Series Classification problem as follows:

**Definition 4.** Given a time series dataset  $\mathcal{X}$  with a vector of finite integer labels  $\mathbf{y} \in \mathbb{N}^n$ , *Time Series Classification* is the task of training a function  $f$  from the space of possible inputs  $\mathcal{X}$  to a probability distribution over the class values in  $\mathbf{y}$ .

Given the formulations above, a parallel can be drawn between a text corpus  $\mathcal{T}$  and a time series dataset  $\mathcal{X}$ , and a document  $T$  and a time series  $X$ . Consequently, the only difference between the two problems is in the type of dataset used, i.e.,  $\mathcal{T}$  vs  $\mathcal{X}$ . By exploiting the parallelism between time series and text, our intuition is that we can solve the Text Classification Problem through TSA approaches. Our idea is to exploit interpretable machine learning methods on time series [25] to build algorithms able to identify the most discriminative subsequences of a time series and project them back into the original text. This would allow us to perform text classification in a human-understandable way.

We focus on interpretable classification through shapelets, i.e., time series subsequences representing a particular class within a dataset [30]. A subsequence is an ordered and contiguous subpart of a signal, formally:

**Definition 5.** Given a signal  $\mathbf{x}$ , a *subsequence*  $\mathbf{s} = \{x_j, \dots, x_{j+l-1}\}$  of length  $l$  is an ordered sequence of values such that  $1 \leq j \leq m - l + 1$ .

To extract shapelets from a dataset, candidate shapelets are generated, and their distances to the time series in the dataset are calculated. Then, their quality is assessed based on how well they separate different classes, and the best shapelets are selected based on their quality scores. After that, each time series is represented as a feature vector, where each feature corresponds to the distance between the time series and one of the shapelets [7]. Formally:

**Definition 6.** Given a time series dataset  $\mathcal{X} \in \mathbb{R}^{n \times d \times m}$ , a shapelet discovery function, *shp\_discovery*, extracts a set  $Q$  of  $q$  discriminative shapelets, i.e.,  $\text{shp\_discovery}(\mathcal{X}) = Q \in \mathbb{R}^{q \times l}$ . Then, a transform function, *shp\_transform*, converts  $\mathcal{X}$  into a real-valued tabular dataset,  $D$ , obtained by taking the minimum Euclidean distance between each time series in  $\mathcal{X}$ , and each shapelet in  $Q$ , via a sliding-window, i.e.,  $\text{shp\_transform}(\mathcal{X}, Q) = D \in \mathbb{R}^{n \times q}$ .

Once the time series dataset is converted through the shapelet transform, an interpretable classifier such as a Decision Tree [3] can be used, having the advantage of an interpretable feature representation. Given these notions, we can now easily link the concept of a time series shapelet to that of a *subdocument* by defining it as an ordered and contiguous subpart of a document, formally:

**Definition 7.** Given a document  $T$ , a *subdocument*  $P = \{S_j, \dots, S_{j+l-1}\}$  is an ordered sequence of  $l$  sentences, such that  $1 \leq j \leq m - l + 1$ .

Therefore, by finding important subsequences in a time series, i.e., shapelets, we can find the most discriminative parts in a corresponding text. Consequently, the real challenge we face in this paper consists in converting a text into a time series. Our proposal to accomplish this task, and to allow solving interpretable text classification through time series classification, is illustrated in the next section.

## 4 Text to Time Series Conversion

In this section, we describe TOTS, a framework to turn Text to Time series. The TOTS framework is a text to time series conversion workflow formed by three core steps: tokenization, feature extraction, and aggregation. We regard TOTS as a framework because every step can be implemented differently. In this work, we defined its main steps and realized only some possible variants. However, the TOTS structure leaves space to integrate various alternatives easily.

A summary of TOTS is illustrated in Algorithm 1. Given a text corpus  $\mathcal{T}$ , TOTS returns a time series dataset  $\mathcal{X}$  where the  $i^{\text{th}}$  time series  $X \in \mathcal{X}$  is the time series representation of the corresponding  $i^{\text{th}}$  document  $T \in \mathcal{T}$ . First, TOTS initializes the empty time series dataset  $\mathcal{X}$  (line 1). Then, for each document  $T \in \mathcal{T}$ , it runs the conversion of  $T$  into  $X$  and adds it to  $\mathcal{X}$  (lines 2–10). The first step of TOTS on  $T$  is tokenization, in which the document is split into sentences and readied for further analysis (line 4). After that, for each sentence  $S$  in the tokenized document  $T'$  (lines 5–7), TOTS extracts characteristic features describing  $S$  through the features extraction function *feat\_extr* and places them into a vector  $\mathbf{v} \in \mathbb{R}^d$  where  $d$  is the number of features. Supposing a document  $T'$  formed by  $m$  sentences, the sequence of  $m$  vectors  $\mathbf{v}$  is concatenated into the matrix  $X \in \mathbb{R}^{m \times d}$ . This matrix  $X \in \mathbb{R}^{m \times d}$  can be viewed as a multivariate time series. Thus, we can transpose  $X$  in order to have a proper multivariate time series  $X' \in \mathbb{R}^{d \times m}$  where different rows model different signals, i.e., features in this case, and different columns capture different timesteps (line 8). The various signals of the multivariate time series are aggregated into a univariate one

**Algorithm 1:** TOTS( $\mathcal{T}$ , *tokenize*, *feat\_extr*, *aggregate*)

---

```

Input :  $\mathcal{T}$  - text corpus, tokenize - splitting function,
        feat_extr - feature extraction function, aggregate - aggregation function
Output:  $\mathcal{X}$  - time series dataset

1  $\mathcal{X} \leftarrow \emptyset;$  // init. time series dataset
2 for  $T \in \mathcal{T}$  do // for each document
3    $X \leftarrow \emptyset;$  // init. time series
4    $T' \leftarrow \textit{tokenize}(T);$  // tokenize document
5   for  $S \in T'$  do // for each sentence
6      $\mathbf{v} \leftarrow \textit{feat\_extr}(S);$  // extract feature vector
7      $X \leftarrow X \cup \{\mathbf{v}\}$  // store feature vector
8    $X' \leftarrow X^\top;$  // transpose feature vector matrix
9    $X' \leftarrow \textit{aggregate}(X');$  // aggregate multivariate time series
10   $\mathcal{X} \leftarrow \mathcal{X} \cup \{X'\};$  // store time series
11 return  $\mathcal{X}$ 

```

---

through the aggregation function *aggregate* (line 9). The function *aggregate* has no effect on  $X'$  when the time series is already univariate, i.e.,  $d = 1$ .

Once a given text corpus  $\mathcal{T}$  is converted into a time series dataset  $\mathcal{X}$  through TOTS, we can run any TSA approach exploiting the advance of a clear correspondence between texts and time series. In particular, we can use an interpretable shapelet-based time series classifier. In the remainder of this section, we illustrate some alternatives to implement the three functions used by TOTS.

**Tokenization.** The first step in our approach consists in defining the granularity of the final time series by splitting the original text. Tokenization involves breaking up a given text into units, called *tokens*, that can be individual words, phrases, or whole sentences [13]. In TOTS, we tokenize at the sentence level.

**Definition 8 (Sentence Tokenization).** Given a text document,  $T$ , a tokenization function, *tokenize*, splits the document into tokens, creating a set of  $m$  sentences  $T' = \{S_1, S_2, \dots, S_m\} = \textit{tokenize}(T)$ .

Here, we use the term “sentence” loosely, i.e., not as a sequence of words ending with a punctuation mark but as a grammatically complete sequence expressing a full thought. Text splitting is a crucial step that may vary depending on the nature of the text and the specific problem. For example, in a dialogue, a timestep may correspond to a speaker’s turn, while in a book, it may correspond to a whole paragraph. If the focus is on song lyrics, line splitting is instead the most sensible option. We use a real 66-line-long rap lyric from the Song Lyrics dataset as a running example to illustrate the various step of the proposed framework (see Sect. 5 for further details). The newline character ( $/$ ) is adopted as the splitting criterion for this example. The following are the first six lines, i.e.,  $T_{0:5}$ :

$T_{0:5}$     Say brah / In this game called life / It’s charces , decisions, and consequences / I decided to change my  
           life, for the better / So anybody that’s out there seeking conviction / because of profanity in my music /

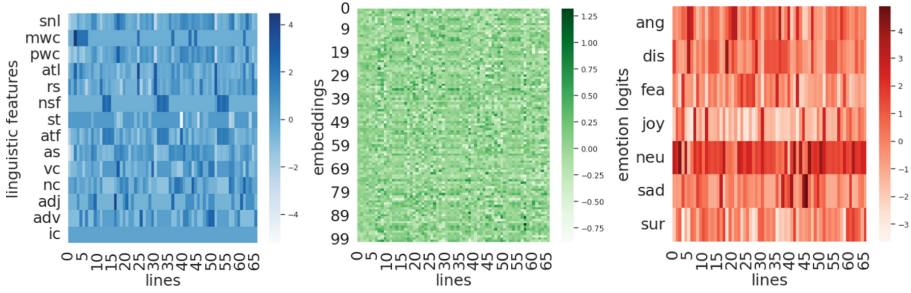
**Feature Extraction.** The second step consists in extracting features from each token, i.e., each sentence in our setting. We present here alternatives to extract features from a document and to implement the *feat\_extr* function defined as:

**Definition 9 (Feature Extraction).** The feature extraction function  $feat\_extr$  takes as input a sentence  $S$ , and returns a vector  $\mathbf{v}$  containing  $d$  characteristics of  $S$ , i.e.  $feat\_extr(S) = \{v_1, \dots, v_d\} = \mathbf{v} \in \mathbb{R}^d$ .

There exist many different feature extraction approaches in NLP. We design and implement three alternatives: one based on linguistic features, one based on sentence embeddings, and an approach relying on sentiment/emotions.

**Linguistic Features.** Computational linguistics offers several methods for extracting meaningful *linguistic features* within a text [5, 13, 19]. Features such as type-token ratio can be extracted through tokenization [13], measuring the lexical diversity of a text by calculating the ratio of unique words (types) to the total number of words (tokens) in a text. Readability scores, such as the Flesch-Kincaid and Dale-Chall formulas [5], can also be viewed as features assessing the complexity of a text. Part-of-speech tagging, such as the Universal POS tagsets [19], can instead be used to identify the grammatical category of each word in a text. These linguistic features provide valuable information about the structure and complexity of a text and can be used in conjunction with other features to improve NLP tasks. In TOTS, we define the function  $feat\_extr$  to extract the following features  $\mathbf{v}$  from a given sentence  $S$ : sentence\_length (`snl`), monosyl\_words\_count (`mwc`), polysyl\_words\_count (`pwc`), avg\_token\_length (`at1`), readability\_score (`rs`), normalized\_sentence\_freq (`nsf`), sentence\_ttr (`st`), avg\_token\_freq (`atf`), alliteration\_score (`as`), verb\_count (`vc`), noun\_count (`nc`), adj\_count (`adj`), adv\_count (`adv`), intj\_count (`ic`). With linguistic feature extraction, a dynamic representation of text characteristics emerges as a multivariate time series. This provides insights into the changing grammatical and phonological qualities from sentence to sentence. Figure 1 (left) shows the linguistic feature-based conversion for the rap text above (all features are normalized). We notice that `nsf` and `atf` can recognize repeating patterns in the text, identifying the three changes between chorus and verse (lines 12:15, 34:36, 56:59). Further, monosyllabic words (`mwc`) have a generally low frequency in the text, except for the first few sentences. The only constant feature throughout the text is the number of interjections (`ic`).

**Sentence Embeddings.** Sentence embeddings are high-dimensional vectors that encode the semantic meaning of a sentence into a space where similar sentences are spatially closer [20]. Several NLP models have been developed to output embeddings. In TOTS, we implement the function  $feat\_extr$  through Sentence-BERT (SBERT) [20] and Doc2Vec [17], in order to extract the embedding  $\mathbf{v}$  from a given sentence  $S$ . Doc2Vec takes a document as input and outputs embeddings capturing context, while Sentence-BERT uses Siamese and Triplet networks to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. The sentence embeddings  $S$  of these models are “static” vectorial representations,  $\mathbf{v}$ , of sentences. However, considering the sequence of embeddings  $X'$ , we can capture the relationship between subsequent sentences. Figure 1 (center) shows an example of a 100-dimensional embedding vector of Doc2Vec for each input sentence. Sentence embeddings are not directly



**Fig. 1.** From left to right: multivariate time series obtained through feature extraction via (i) linguistic features, (ii) text embeddings, (iii) sentiment analysis.

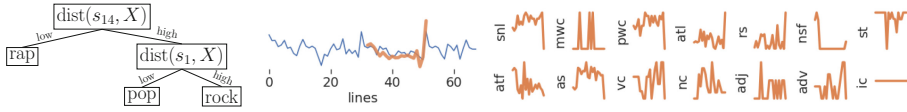
interpretable by humans, but they can capture complex semantic information, which is extremely useful for machine learning predictors.

**Sentiment/Emotion Features.** The logit layer of a sentiment/emotion analysis model produces a vector of scores or activations for each possible output class, indicating the model’s confidence or belief that the input sentence corresponds to each possible sentiment/emotion. Examples of such models are VADER (Valence Aware Dictionary and sEntiment Reasoner) [11], a lexicon and rule-based sentiment analysis tool attuned explicitly to sentiments expressed in social media, and RoBERTa (Robustly Optimized BERT Pretraining Approach) [15], a variant of BERT that has been shown to achieve state-of-the-art performance on several NLP tasks, including sentiment analysis. While originating from a transformer model, logits are more interpretable than embeddings as they provide a sort of expectation of an input sentence for a certain sentiment/emotion.

In TOTS, we implement the function *feat\_extr* to extract the list of sentiments/emotions  $\mathbf{v}$  from a given sentence  $S$  through both VADER and RoBERTa, which, in a time series context, can track the fluctuation of sentiment and emotions within the text, providing dynamic information instead of static analysis. VADER provides a single sentiment score, while RoBERTa outputs logits for the following emotions: Anger (**ang**), Disgust (**dis**), Fear (**fea**), Joy (**joy**), Neutral (**neu**), Sadness (**sad**), and Surprise (**sur**). Figure 1 (right), shows an example of a multivariate time series obtained with RoBERTa. This series depicts a mostly neutral document, with a high peak of sadness on line 42 (“*Rest in peace and then deceased but we still strugglin while you sleep*”).

**Aggregation.** In order to use a shapelet-based interpretable machine learning model, we need to reduce multivariate time series into univariate ones. We accomplish this task by defining an aggregation function, *aggregate*, that takes a multivariate time series as input and “compresses” it into a univariate time series without changing the number of observations  $m$ . Formally:





**Fig. 2.** Shapelet analysis approach on a linguistic time series aggregated with PCA.

**Definition 10 (Aggregation Function).** An aggregation function *aggregate* takes as input a multivariate time series  $X \in \mathbb{R}^{d \times m}$ , with  $d > 1$ , and compresses it into a univariate time series  $X' = \text{aggregate}(X)$ , where  $X' \in \mathbb{R}^{1 \times m}$ .

In this work, we experiment with two naive approaches such as *average* and *max* aggregation, and with a complex dimensionality reduction method such as Principal Component Analysis (PCA) [24]. Aggregation by taking the average may be sufficient when multivariate dimensions represent the same phenomenon detected by different models, such as the sentiment or the emotion computed by two different transformers, which is averaged for a more robust prediction. On the other hand, aggregation by taking the maximum could be enough when the different signals in a time series represent logits of different sentiments, highlighting the intensity of the prevalent emotion at a specific timestep. More sophisticated approaches like PCA may be required for more complex signals, like those resulting from embeddings. PCA dynamically detects the significant time series signals that include characteristic patterns of the original data because the significance of each signal is represented in each component of the transformation [24].

Figure 2 (left) displays in blue the univariate time series resultant from the PCA aggregation from the linguistic features. The signal is hardly interpretable at first look, but, as illustrated in the following, the contribution of each signal toward the final component can be retrieved, providing insights into the most relevant signals at specific timesteps, i.e., for specific sentences in our setting.

**Time Classification.** Once a given text corpus,  $\mathcal{T}$ , has been converted into the corresponding time series dataset,  $\mathcal{X}$ , by TOTS, i.e.,  $\mathcal{X} = \text{TOTS}(\mathcal{T})$ , we can extract a set of  $q$  shapelets,  $Q$ , from  $\mathcal{X}$  with Learning Shapelets (LS) [7]. LS learns shapelets through gradient descent optimization and is regarded as a state-of-the-art approach. In the example in Fig. 2, we use the extracted shapelets with a decision tree classifier to distinguish between rap and rock lyrics transparently. The resulting tree is extremely simple and, using only two of the extracted subsequences ( $s_{14}$  and  $s_1$ ), can discern between the two genres, by looking at the distance between the shapelets and the text conversion. Here, to aid interpretability, we present distances as “high” or “low” instead of specific values. Hence, there are only three rules to classify songs: *if*  $\text{dist}(s_{14}, X)$  is low *then* the class = rap, else, *if*  $\text{dist}(s_1, X)$  is low *then* class = pop, else class = rap.

Figure 2 (center) displays our running example for shapelet  $s_{14}$  (in orange). We notice that the best alignment of the shapelet with the time series begins at index 32 and ends at index 55 included. With this information, the shapelet can be mapped back to its multivariate components, i.e., the subsequences between 32 and 55 of each signal depicted in Fig. 2 (right). Furthermore, the same indexes can be mapped back to the original text by unveiling the lines between 32 and 55

**Table 1.** Dataset information.

| <i>dataset</i>      | <i>id</i> | <i>classes</i> | <i>records</i> | <i>labels</i>                           |
|---------------------|-----------|----------------|----------------|---|
| SongLyrics          | pprc      | 2              | 24000          | pop, rock                               |
| SongLyrics          | rcrp      | 2              | 24000          | rock, rap                               |
| SongLyrics          | lyr3      | 3              | 36000          | pop, rap, rock                          |
| SongLyrics          | rsub      | 5              | 25000          | pop-rock, metal, indie, hard-rock, punk |
| WikipediaMoviePlots | mplt      | 4              | 7512           | drama, comedy, thriller, horror         |
| 20Newsgroups        | 20ng      | 5              | 1775           | talk, religion, sci, rec, comp          |

in the original lyric, which the model uses to make a prediction. In the following, we show the document  $T$  from sentence 28 to sentence 59 to better appreciate the text highlighted by the shapelet.

$T_{28:59}$       Sometime we do bad, but we all in it / You gotta learn to dream, cause there's No Limit, ya heard me? / - singing / Y'all don't know what we goin through / Y'all don't know what they put us through / Y'all don't know what we goin through / Y'all don't know what they put us through / Don't treat me like a disease, cause my skin darker than yers / And my environment is hostile, nuttin like your suburbs / I'm from the ghetto, home of poverty - drugs and guns / Where hustlers night life for funds but, makin crumbs / in the slums in the street, in the cold in the heat / Rest in peace and then deceased but we still strugglin while you sleep / And the game never change it's still the same since you passed / We get beat and harassed, whenever them blue lights flash / To the little homies in the hood, claimin wards and wearin rags / Tryin to feel a part of a family he never had / And it's sad, I feel his pain, I feel his wants / To avoid bein locked up, there's do's and don'ts / Use your head little soldier, keep the coke out your system / that ? out your veins, that won't do away with the pain / Only prayers will get you through, ain't no use to bein foolish / Ain't got one life to live, so be careful how you use it / - singing / Y'all don't know what we goin through / Y'all don't know what they put us through / Y'all don't know what we goin through / Y'all don't know what they put us through /

From the comparison between the shapelet and the text, we can observe how the text evolves. For instance, at the beginning of the shapelet, the normalized sentence frequency drops (**nsf**), indicating the end of the chorus and the beginning of the verse. A slight increase at the end highlights the beginning of a new chorus. Further, the alliteration score seems to grow in the verse, with the more rhythmic repetition of sounds (“*To the little homies in the hood, claimin wards and wearin rags*”). **sn1** and **pwc** represent the higher length of sentences in the verse w.r.t. the chorus. Other subsequences are harder to interpret in this instance, such as the number of adjectives (**adj**) and verbs (**vc**).

## 5 Experiments

We experiment with TOTS<sup>1</sup> on three datasets to assess the correctness and effectiveness of the proposed transformation.

**Datasets.** The first dataset is **Song Lyrics**, containing lyrics associated with the artist’s genres. We created four different balanced subsets of this dataset, **pprc**, **rcrp**, **lyr3**, and **rsub**, containing different labels, as described in Table 1. We split Song Lyrics line-by-line with *tokenize*, removing duplicates and non-English text (e.g., *Chorus 2x*). For **20ng**, we used sentences as tokens, removing hyperlinks, HTML tags, email addresses, symbol repetitions, and expanding contractions. For **mplt**, we merged coherent genre labels, tokenizing at the sentence

<sup>1</sup> Code available at: <https://github.com/mattiapggioli/lyrics2ts>.

level. We discarded sentences with less than 20 lines for all datasets to avoid generating very short time series and performed an 80/20% train/test split.

**Experimental Setting.** We detail here the alternative implementations adopted to realize the function *feat\_extr* and *aggregate*. *Linguistic features* are derived using the `textstats` and `NLTK` packages. Regarding *sentence embedding* methods, for Sentence-BERT [20] (SBE) we used the *all-MiniLM-L12-v2* model provided by `SentenceTransformers`, while for Doc2Vec [17] (D2V) we used `Gensim` after using its tokenizer with lowercasing<sup>2</sup>. For *sentiment features* (SEN), we used VADER [11] through the NLTK library, which outputs a compound score, ranging from  $-1$  (extremely negative) to  $+1$  (extremely positive). Thus, the resulting time series are univariate and require no aggregation. Finally, for emotion features (EMO), we used *emotion-english-distilroberta-base*, extracting the emotion logits of the last layer. As *aggregate* functions, we tested naive *avg* and *max* by simply applying the respective `numpy` functions column-wise and PCA by adopting the `scikit-learn` implementation. We experimented with PCA by (i) fitting and transforming each time series separately (*pca*), and (ii), by fitting a global PCA model on the entire multivariate time series dataset and using it to transform each time series into a univariate one (*gpca*). In the latter, the idea is to consider timesteps as individual observations in a vector space that we want to reduce in one dimension and time series as movements within it.

**Assessing Conversions Correctness.** In this experiment, we assess the correctness of the different conversion workflows that can be realized through the TOTS framework. We measured the correctness by checking if similar texts are mapped to similar time series after the conversion in a controlled experiment on the `lyr3` dataset. Formally, given a document  $T$  from the corpus  $\mathcal{T}$ , a document  $T' \neq T$  that by construction is similar to  $T$ , i.e., is obtained by altering  $T$ , and a document  $T'' \neq T$  randomly selected from  $\mathcal{T}$ , our desiderata is that the distance between  $\text{TOTS}(T)$  and  $\text{TOTS}(T')$  is smaller than the distance between  $\text{TOTS}(T)$  and  $\text{TOTS}(T'')$ . Thus, similar documents should be converted in similar time series. Since we are comparing time series, we adopt the Dynamic Time Warping (DTW) distance [23]. Hence, given a corpus  $\mathcal{T}$ , a corpus of similar documents  $\mathcal{T}'$ , and a randomly shuffled corpus  $\mathcal{T}''$ , we define the correctness score  $CS$  as:

$$CS = \frac{1}{n} \sum_i^n \mathbb{1}[\text{dtw}(\text{TOTS}(T_i), \text{TOTS}(T'_i)) < \text{dtw}(\text{TOTS}(T_i), \text{TOTS}(T''_i))]$$

where  $CS$  is the percentage of times the desiderata holds. In practice, we sampled 50 song lyrics per genre from `lyr3`, i.e.,  $\mathcal{T}$  and, for each of them, we created a similar lyric by applying text augmentation line by line, i.e.,  $\mathcal{T}'$ . For this purpose, we used the *ContextualWordEmbsAugmenter* of the `nlpaug` library, which replaces words in a text with their contextually similar counterparts using a pre-

<sup>2</sup> We set the following parameters: `dm = 1`, `vector_size = 100`, `min_count = 2`, `epochs = 20`, `window = 5`.

trained contextual word embedding model. Then, we associated each original text in  $\mathcal{T}$  with a randomly selected one, i.e.,  $\mathcal{T}''$ . Finally, we computed  $CS$ .

**Table 2.** CS metric for `lyr3`. The best *aggregate* for each *feat\_extr* method are in bold.

|                  | <i>feat_extr</i> | D2V          | SBE          | LIN          | EMO          | SEN          |
|------------------|------------------|--------------|--------------|--------------|--------------|--------------|
| <i>aggregate</i> | <i>avg</i>       | <b>0.740</b> | 0.667        | 0.840        | <b>0.813</b> |              |
|                  | <i>max</i>       | <b>0.740</b> | 0.693        | 0.713        | 0.693        | <b>0.800</b> |
|                  | <i>pca</i>       | 0.593        | 0.540        | 0.633        | 0.713        |              |
|                  | <i>gpca</i>      | 0.720        | <b>0.767</b> | <b>0.997</b> | 0.786        |              |

Table 2 shows the results of this experiment w.r.t. different types of feature extraction and aggregation functions (the higher, the better). Excluding *gpca*, D2V performs better than SBE, with an average difference of about 0.05. However, with *gpca* applied, SBE demonstrates the highest performance among sentence embedding approaches, outperforming D2V. The *gpca* method demonstrates significant superiority among those based on linguistic features. Traditional *pca* demonstrates poor results not only against *gpca* but also to *max* and *avg*. In summary, the best aggregation approaches seem to be *avg* and *gpca*. However, the single sentiment signal SEN, without any aggregation, scores surprisingly high. As for runtime performance, the fastest method is SEN, with a runtime of 5.3 ms per sentence, followed by LIN and D2V with an average execution time of 44–48 ms. SBE takes longer, with an average execution time of 1.120s, and the slowest model is EMO taking on average 2.83s<sup>3</sup>.

Given these results, we chose one instance for each feature extraction method to experiment with the classification task. In particular, we selected SBE with *gpca*, which produced the best results among the embeddings, despite being less efficient than D2V. For LIN, we also picked the *gpca* method, which proved extremely accurate during validation. Finally, for the sentiment/emotion method, we selected SEN, given that it performed well, with extremely fast runtimes.

**Classification Benchmark.** This section evaluates the performance of interpretable ML models applied to solve the text classification problem. Regarding our proposal, after having selected the most promising functions *feat\_extr* and *aggregate* as described in the previous section, we applied TOTS on the text corpus obtaining the corresponding time series datasets, i.e.,  $\mathcal{X} = \text{TOTS}(\mathcal{T})$ . To achieve our goal of interpretable text classification with explanations based on the dynamical properties of text, we extracted the shapelets from  $\mathcal{X}$  through a *shp\_discovery* function, and we turned  $\mathcal{X}$  into  $D$  with a *shp\_transform* function. In particular, we obtained *shp\_discovery* and  $D$  with the *LearningShapelets* function of [tslearn](#)<sup>4</sup>. Then, we trained the following ML models selected for their

<sup>3</sup> Experiments were run on a ThinkPad E595. AMD Ryzen 5 3500U CPU, 8 gb RAM.

<sup>4</sup> We set the number of shapelets to extract  $q$  using the provided heuristic, and Adam as optimizer training for 2000 epochs per dataset.

**Table 3.** Classification accuracy (higher is better). The best results by column, i.e., by TOTS conversion, are bolded, best results by dataset are underlined.

|             | pprc |            |            | rcrp       |            |            | lyr3       |            |            | rsub       |            |            | mplt       |            |            | 20ng       |            |            |            |
|-------------|------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
|             | LIN  | SEN        | SBE        | LIN        | SEN        | SBE        | LIN        | SEN        | SBE        | LIN        | SEN        | SBE        | LIN        | SEN        | SBE        | LIN        | SEN        | SBE        |            |
| <i>shp</i>  | DT   | .53        | .54        | .57        | .79        | .73        | .74        | .52        | .50        | .48        | .22        | .23        | .24        | .28        | .33        | .31        | .27        | .31        | .32        |
|             | RF   | .59        | <b>.60</b> | <b>.64</b> | <b>.86</b> | .81        | <b>.82</b> | <b>.61</b> | .60        | .56        | .24        | .27        | <b>.28</b> | .33        | <b>.39</b> | <b>.39</b> | <b>.32</b> | .34        | <b>.41</b> |
|             | LG   | .60        | <b>.60</b> | <b>.64</b> | <b>.86</b> | .81        | <b>.82</b> | <b>.61</b> | .60        | <b>.57</b> | .24        | <b>.28</b> | <b>.28</b> | .31        | <b>.39</b> | .37        | .29        | .35        | .38        |
| <i>feat</i> | DT   | .55        | .54        | .55        | .80        | .76        | .71        | .53        | .52        | .47        | .21        | .24        | .23        | .29        | .33        | .31        | .25        | .29        | .34        |
|             | RF   | .60        | .59        | .62        | <b>.86</b> | <b>.83</b> | .78        | .60        | .60        | .54        | <b>.25</b> | .27        | .26        | <b>.34</b> | <b>.39</b> | .36        | .26        | <b>.39</b> | <b>.41</b> |
|             | LG   | <b>.61</b> | <b>.60</b> | <b>.64</b> | <b>.86</b> | <b>.83</b> | .79        | <b>.61</b> | <b>.61</b> | .56        | <b>.25</b> | .27        | <b>.28</b> | .33        | .38        | .35        | .27        | .34        | .40        |
| <i>knn</i>  | EUC  | .52        | .55        | .56        | .56        | .51        | .55        | .38        | .37        | .38        | .22        | .22        | .24        | .32        | .35        | .33        | .27        | .24        | .33        |
|             | DTW  | .54        | .54        | .60        | .75        | .68        | .67        | .51        | .47        | .44        | .22        | .23        | .26        | .29        | .33        | .34        | .28        | .30        | .32        |

interpretability properties on the shapelet-transformed dataset  $D$  (*shp*), i.e., a Decision Tree (DT), a Random Forest (RF), and LightGBM (LG). As a competitor, we extracted global time series statistics (*feat*) such as the minimum, maximum, mean, variance, skewness, and kurtosis on  $\mathcal{X}$ , and then we train the tree-based models DT, RF, and LG. In this setting, classifiers are only statically interpretable because all the temporal references given by the time series are completely lost. Finally, in line with instance-based explanation approaches [8], we experimented also with k-Nearest-Neighbors (*knn*) trained directly on  $\mathcal{X}$ . In particular, we experiment with *knn* with  $k = 5$  using the Euclidean distance (EUC) and DTW with a 3-window Sakoe Chiba band [23], adopting the `pyts` library<sup>5</sup>.

Table 3 presents the accuracy of the various classifiers. The column header represents the different dataset conversions of TOTS, i.e., we convert each of the six datasets using the three best approaches from the previous section for a total of 18 dataset representations. The rows represent different classifiers, i.e., based on shapelets (*shp*), static global features (*feat*), and distances (*knn*). The best results in each column are in bold, highlighting the best feature extraction and aggregation. The best approach overall for each dataset is underlined.

At first glance, the best-performing classifiers are RF and LG, with DT, EUC, and DTW always having subpar performance. In general, shapelets and global features perform similarly, with their respective best models tying in all of the six datasets. However, as shown in the example rap lyric, the advantage of using shapelets is to look at the importance of specific paragraphs in the text, which is impossible with global features. Regarding TOTS conversion alternatives, SBE wins in 20ng and pprc, while LIN is the overall best for rcrp. Classifiers trained on SEN have slightly lower performance, likely because they are based on a single

<sup>5</sup> Given the computational complexity of DTW on large datasets, we first used Piecewise Aggregate Approximation (PAA) to reduce the length of the time series by 80% and then kept one-third of the records for each class, selected using the ClusterCentroids method of `imblearn`.

sentiment, which may not be sufficient for the classification. The similar performance of embeddings and linguistic features is promising for explainability. It demonstrates that, for specific problems, using domain knowledge to extract interpretable features can achieve similar results to non-interpretable embeddings.

As a final note, we highlight that the purpose of TOTS at this stage is not to beat standard NLP approaches applied to the whole text but to define a way of using TSA approaches for text classification. While our approach may not perform as well as standard NLP classifiers, we offer a unique way to analyze text by taking into account local patterns rather than relying solely on the properties of the entire text. This allows for a more nuanced understanding of the text and its underlying dynamics. Overall, a sentence-based explanation can provide a more fine-grained and interpretable classification. For example, when analyzing a song, a sentence-based explanation can help identify the most relevant lines or sections to the classification result. Finally, in datasets such as 20ng, containing multiple topics, a sentence-based explanation can provide insights into how different parts of the text contribute to the classification result.

## 6 Conclusion

We have introduced TOTS, a method that represents text as a time series using TSA techniques and NLP approaches. Our formalization enables the conversion between text and time series, enhancing interpretability by capturing local textual patterns. Additionally, TOTS allows for easy transformation back to text, facilitating human interpretation. Through experiments, we showed that our text to time series conversion uncovers new insights and patterns not easily observable with traditional NLP approaches. A potential limitation of TOTS is its reliance on multiple independent steps, where the quality of underlying models can influence the overall performance. For example, we acknowledge that aggregating time series into univariate ones is a strong simplification, and directly analyzing multivariate text-time series could be more effective. Moreover, instead of exclusively relying on shapelets, alternative patterns could be tested for the classification task. Combining features and patterns offers a promising approach to extracting local characteristics and global dynamic trends, capturing the entire document’s semantic context. After further improvements, we plan to compare TOTS against state-of-the-art NLP models and study possible avenues of integration with Large Language Models. Finally, we plan on extending text shapelets’ interpretability to unsupervised analyses like clustering or topic modeling, where sequentiality can be incorporated by localizing the analysis on extracted sequential patterns.

**Acknowledgment.** This work is partially supported by the EU NextGenerationEU programme under the funding schemes PNRR-PE-AI FAIR (Future Artificial Intelligence Research), PNRR-SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics - Prot. IR0000013, H2020-INFRAIA-2019-1: Res. Infr. G.A.

871042 *SoBigData++*, G.A. 761758 *Humane AI*, G.A. 952215 *TAILOR*, ERC-2018-ADG G.A. 834756 *XAI*, and CHIST-ERA-19-XAI-010 SAI, and by the Green.Dat.AI Horizon Europe research and innovation programme, G.A. 101070416.

## References

1. Badea, I., Trausan-Matu, S.: Text analysis based on time series. In: ICSTCC 2013, pp. 37–41. IEEE (2013)
2. Bagnall, A., et al.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *DAMI* **31**, 606–660 (2017)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Routledge (2017)
4. Chalkiadakis, I., Zaremba, A., Peters, G.W., Chantler, M.J.: On-chain analytics for sentiment-driven statistical causality in cryptocurrencies. *Blockchain: Res. Appl.* **3**(2), 100063 (2022)
5. Dale, E., Chall, J.S.: A formula for predicting readability: instructions. *Educ. Res. Bull.* **27**, 37–54 (1948)
6. Fu, T.C.: A review on time series data mining. *Eng. Appl. Artif. Intell.* **24**(1), 164–181 (2011)
7. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: SIGKDD 2014, pp. 392–401. ACM (2014)
8. Guidotti, R., Monreale, A., Spinnato, F., Pedreschi, D., Giannotti, F.: Explaining any time series classifier. In: CogMI 2020, pp. 167–176. IEEE (2020)
9. Harris, Z.S.: Distributional structure. *Word* **10**(2–3), 146–162 (1954)
10. Hassani, H., Beneki, C., Unger, S., Mazinani, M.T., Yeganegi, M.R.: Text mining in big data analytics. *Big Data Cogn. Comput.* **4**(1), 1 (2020)
11. Hutto, C., Gilbert, E.: Vader: a parsimonious rule-based model for sentiment analysis of social media text. In: ICWSM 2014, vol. 8, pp. 216–225 (2014)
12. Jing, L.P., Huang, H.K., Shi, H.B.: Improved feature selection approach TFIDF in text mining. In: ICMLC 2002, vol. 2, pp. 944–946. IEEE (2002)
13. Kaplan, R.M.: A method for tokenizing text. *Inquiries into words, constraints and contexts* 55 (2005)
14. Khurana, D., Koli, A., Khatter, K., Singh, S.: Natural language processing: state of the art, current trends and challenges. *Multim. Tools Appl.* **82**(3), 3713 (2023)
15. Liu, Y., et al.: Roberta: a robustly optimized BERT pretraining approach. arXiv preprint [arXiv:1907.11692](https://arxiv.org/abs/1907.11692) (2019)
16. Makridakis, S., Wheelwright, S.C., Hyndman, R.J.: *Forecasting Methods and Applications*. Wiley, Hoboken (2008)
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. *ICML* **2013**, 26 (2013)
18. Muthukumar, P., Zhong, J.: A stochastic time series model for predicting financial trends using NLP. arXiv preprint [arXiv:2102.01290](https://arxiv.org/abs/2102.01290) (2021)
19. Petrov, S., Das, D., McDonald, R.: A universal part-of-speech tagset. In: LREC’12, pp. 2089–2096 (2012)
20. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint [arXiv:1908.10084](https://arxiv.org/abs/1908.10084) (2019)
21. Ruiz, E.J., Hristidis, V., Castillo, C., Gionis, A., Jaimes, A.: Correlating financial time series with micro-blogging activity. In: WSDM 2012, pp. 513–522 (2012)
22. Şenel, L.K., Utlu, I., Yücesoy, V., Koc, A., Cukur, T.: Semantic structure and interpretability of word embeddings. *IEEE/ACM TASLP* **26**(10), 1769–1779 (2018)

23. Senin, P.: Dynamic time warping algorithm review. Information and Computer Science Dept. University of Hawaii at Manoa Honolulu, USA 855(1–23), 40 (2008)
24. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time-series motif from multi-dimensional data based on mdl principle. *Mach. Learn.* **58**, 269–300 (2005)
25. Theissler, A., Spinnato, F., Schlegel, U., Guidotti, R.: Explainable AI for time series classification: a review, taxonomy and research directions. *IEEE Access* **10**, 100700–100724 (2022)
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł, Polosukhin, I.: Attention is all you need. *NIPS* **2017**, 30 (2017)
27. Wang, B., Huang, H., Wang, X.: A novel text mining approach to financial time series forecasting. *Neurocomputing* **83**, 136–145 (2012)
28. Wang, X., McCallum, A., Wei, X.: Topical n-grams: phrase and topic discovery, with an application to information retrieval. In: *ICDM*, pp. 697–702. IEEE (2007)
29. Yang, T., Lee, D.: T3: on mapping text to time series. In: *AMW* (2009)
30. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: *SIGKDD 2009*, pp. 947–956 (2009)
31. Zhang, W., Yoshida, T., Tang, X.: A comparative study of TF\* IDF, LSI and multi-words for text classification. *Expert Syst. Appl.* **38**(3), 2758–2765 (2011)