Low Dimensional Text Representations for Sentiment Analysis NLP Tasks

Leonidas Akritidis[1*†] and Panayiotis Bozanis[1†]

[1]School of Science and Technology, International Hellenic University, 14th km Thessaloniki - Nea Moudania,Thermi, Thessaloniki, 570 01, Greece.

*Corresponding author(s). E-mail(s): lakritidis@ihu.gr;
Contributing authors: pbozanis@ihu.gr;
†These authors contributed equally to this work.

Abstract

Natural Language Processing (NLP) is presently among the hottest scientific fields with an enormous growth rate of the relevant research. Sentiment analysis is a popular NLP problem that aims at the automatic identification of the polarity in user reviews, tweets, blog posts, comments, forum discussions and so on. Unfortunately, the natural sparseness of text, along with its intimate high dimensionality renders the direct application of machine/deep learning models problematic. For this reason, the relevant literature contains a wealth of state-of-the-art dimensionality reduction methods that confront these issues. In this paper, we conduct an experimental study on the effects of dimensionality reduction in the area of sentiment classification. More specifically, we consider multiple feature selection and feature extraction techniques and we investigate their impact on the effectiveness and the efficiency of seven state-of-the-art classifiers. The experimental evaluation includes accuracy and execution time measurements on four benchmark datasets with various degrees of reduction aggressiveness. The results indicate that, in most cases, dimensionality reduction has indeed a beneficial impact on the running times, whereas the accuracy sacrifices are usually small. However, we also indicate several exceptions where this observation is not valid. These exceptions are appropriately highlighted and discussed.

Keywords: sentiment analysis, opinion mining, sentiment classification, text classification, dimensionality reduction, feature selection, feature extraction

1 Introduction

The growing requirement for accurately identifying the polarity of an opinion in an excerpt of text led to the establishment of the sentiment analysis research field. Powered by effective Natural Language Processing (NLP) algorithms, the sentiment analysis models are now playing a crucial role in a vast number of knowledge discovery tasks. Examples include blogs [1, 2], microblogs [3, 4], social networks [5, 6], e-Commerce reviews [7, 8], and so on.

In the vast majority of cases, sentiment analysis is treated as a text classification problem [9, 10]. If the involved text polarity is binary (i.e., positive or negative, good or bad), then binary text classification models are trained by utilizing two class labels. On the other hand, in case the polarity falls into a closed score range (e.g., 1–5, 1–10, etc.), then each individual score is treated as a separate class label and multi-class classification approaches are applied.

Text classification is one of the most well-studied machine learning problems, and numerous state-of-the-art models exist for almost every NLP-related application. Examples of such models are the Convolutional Neural Networks (CNNs) [11], the Recurrent Neural Networks/Long Short-Term Memory (RNNs/LSTMs) [12, 13], the attention-based Transformers [14], and many others. Due to their design, these deep learning models can identify the semantics of text either at word level [15, 16], or at sentence level [14, 17]. Apart from them, the classical machine learning methods are still of great usefulness because they are fast to train and they combine simplicity with decent accuracy [18].

When large document collections are involved, the underlying text is usually composed of a vast number of terms (words, n-grams, word combinations, dates, model numbers etc.), rendering its modelling hard. The traditional Vector Space Model (VSM) and the Bag-of-Words (BOW) technique have been quite popular solutions for some time. However, their difficulty in capturing the semantics of text and the usage of high dimensional vector representations introduced the requirement for more robust approaches.

Nowadays, there is a huge amount of research towards the introduction of semantically meaningful text representations, and low dimensionality is among the key requirements [16, 19]. Otherwise, feeding an algorithm (e.g., a classifier) with high-dimensional data is simply not feasible due to the extreme size of the input matrix and side the effects of the curse of dimensionality [20, 21].

Motivated by the importance of these problems, the aforementioned challenges and the limited number of similar studies in the relevant literature, this article presents an empirical study on the effect of dimensionality reduction in various sentiment analysis models. For this purpose, multiple feature selection (FS) and feature extraction (FE) techniques are examined.

The former select a subset of the most representative input variables from the original data so that the classification accuracy is not affected much [22]. The filter-based FS methods first compute the value of a statistical measure (e.g., mutual information, chi-squared, etc.) and then, they filter the features based on that value. On the other hand, the Wrapper-based FS methods

quantify the feature importance by considering the estimates produced by a classifier. In the sequel, they greedily append (or eliminate) the most (or the least) important of them to (from) the selected features set [23].

Feature extraction is another family of dimensionality reduction methods that generate informative and non-redundant features with the aim of producing better text representations [24, 25]. A portion of them construct a low-dimensional vector space, and then they project the original high-dimensional data onto that space. Indicatively, we refer Principal Component Analysis (PCA), Non-Negative Matrix Factorization (NMF) [26], and Singular Value Decomposition (SVD) [27], which are dominant approaches to this problem. The Autoencoders (AE) and the Variational Autoencoders (VAE) constitute an attractive alternative FE approach due to their ability to capture non-linear relationships between the input variables [15, 28].

This work studies the impact of the aforementioned FS and FE methods on the performance of various text classification models in sentiment analysis NLP tasks. Our contributions include:

- We conduct a quantitative study on how dimensionality reduction affects the effectiveness and the efficiency of multiple well-established text classifiers. The study focuses particularly on sentiment analysis applications. Therefore, a significant number of relevant datasets is employed.
- In contrast to most similar studies, we include both feature selection and feature extraction methods in our analysis.
- The analysis itself has a twin goal: to evaluate the performance of the involved algorithms in terms of both accuracy and running times.
- We verify that, in most cases, dimensionality reduction enhances the training times by making acceptable sacrifices in accuracy. However, we also highlight particular occasions where either the training times are not improved, or the effectiveness is degraded by a large margin.

The rest of this paper is organized as follows: In Section 2, we refer to the most significant works from the research fields of text classification, sentiment analysis, and dimensionality reduction. In the sequel, we present some preliminary elements and the basic notation in Section 3. Section 4 provides some brief descriptions of the dimensionality reduction algorithms that participate in this article. The experimental results of the study are presented and discussed in Sections 5 and 6, respectively. Finally, the conclusions of this work are summarized in Section 7.

2 Related Work

This paper involves two different research areas: text classification (in the context of sentiment analysis) and dimensionality reduction. Both areas have been explored extensively by the research community in the previous years, and numerous state-of-the-art algorithms have been published in the relevant literature. This section briefly presents the most remarkable among them.

2.1 Text Classification and Sentiment Analysis

As a scientific field, sentiment analysis was born and evolved together with social networks. The relevant literature includes a wide variety of opinion mining methods that apply to blogs, instant messengers, microblogs, product reviews, forum discussions, and so on. The first works were simple applications of traditional machine learning classifiers on small datasets. Indicatively, the study of [8] covered several linear, probabilistic, and rule-based classifiers, as long as a family of lexicon-based approaches.

The recent explosion of research on deep learning led to significantly more accurate models. One of the first works in the area applied the Word2Vec embeddings [15] to derive a semantically meaningful vector representation for each word. In the sequel, the word embeddings were used to train a deep Convolutional Neural Network (CNN) with the aim of identifying the polarity of an opinion [11]. The experimental evaluation of this model demonstrated its superior performance against Recursive Neural Networks (RNNs). Similarly, in [29] the authors applied a simple CNN model to predict user satisfaction from a collection of product reviews.

Long Short-Term Memory (LSTM) networks constitute another remarkable text classification technique. Originally introduced to model non-linearities in sequential data (e.g., time series), this model has been proved quite effective in sentiment analysis applications. For example, the authors of [13] employed an LSTM-based architecture to discover topics and conduct sentiment analysis in forum discussions about COVID-19. In addition, several researchers integrated powerful attention-based mechanisms into the LSTM models to further improve classification performance [12, 30].

Another portion of works injected stacks of convolutional layers into LSTM architectures to produce effective CNN/LSTM hybrid models. Specifically, Co-LSTM (Convolutional LSTM) is a recent hybrid approach that achieves domain-independent sentiment analysis of consumer reviews [31]. In a similar spirit, [32] introduced another combination of CNNs with Bidirectional LSTM (BiLSTM). The proposed model works with Doc2Vec embeddings and exhibited good performance in the task of opinion mining in long documents.

Some recent papers on sentiment classification highlighted the usefulness of Transformers in effectively identifying the polarity of an opinion. These models are applied to sequential data like RNNs, but they also include an attention module that is able to capture the text semantics in any position [14]. In [33], the authors introduced BMT-Net, a broad multitask Transformer network that exploits both feature-based and fine-tuning methods for applying pretrained language models to sentiment recognition tasks. Furthermore, the experimental study of [34] evaluated several pre-trained Transformer-based models and found them superior to five popular sentiment analysis tools.

Apparently, the scientific literature on the subject is very rich. For more information, the interested reader may also refer to the surveys of Zhang et al. [35] and Hussein [36]. The former reviewed multiple deep text classification models for document-level, sentence-level, and aspect-level sentiment analysis

tasks. On the other hand, the latter investigated the most important challenges in opinion mining and sentiment analysis.

2.2 Dimensionality Reduction for Text Data

Although the classification models of the previous subsection exhibit significant differences in their design and logic, they still require low-dimensional representations to operate efficiently. Some of them build these representations internally, whereas others require the input to be vectorized in advance. In this section, we briefly refer to some inspiring works in dimensionality reduction. However, more details are provided in Section 4.

The dimensionality reduction approaches are mainly divided into two categories: feature selection (FS) and feature extraction (FE). The former construct and preserve a subset of the most important features and discard the rest of them [22]. The removal of multiple components from the input vectors eventually leads to representations of lower dimensionality. A recent review on the most important FS algorithms was conducted in [37].

Regarding feature extraction, the area is governed by well-established matrix factorization and decomposition techniques, such as PCA, NMF [26], SVD [27], and others. In the context of sentiment analysis, we also encounter the works of [38] and [39]. The first constructs a Laplacian eigenmap (SS-LE) that quantifies the feature importance from the errors in sentiment classification. On the other hand, the second considers the structural information of text by applying a semi-supervised technique for assigning weights to the features of the input vectors.

3 Preliminary Elements

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ be a collection of n documents in raw text format. As mentioned earlier, in most cases, sentiment analysis is treated as a classification task. In order to feed $\mathcal{D}$ into a classifier, each document d_i must be transformed (vectorized) into a numerical vector of the form $\mathbf{x}^{(i)}$. A common technique for vectorizing a document is the Bag-of-Words (BOW) model, that replaces each term in the document by a numerical value. Notice that we use the word *term* instead of *word* to refer to additional text constructs such as bi-grams, tri-grams, word combinations, dates, or mixed strings that include both digits and characters.

One of the most important problems of BOW is that it is dictionary-based, and, as such, it generates very long and sparse vector representations. More specifically, a dictionary L that includes all the distinct terms in the collection $\mathcal{D}$ is initially constructed. If m is the size of L, then BOW dictates that a document $d_i \in \mathcal{D}$ is transformed into an m-dimensional vector $\mathbf{x}^{(i)}$ as follows: Initially, each element $x_j^{(i)} \in \mathbf{x}^{(i)}$ is mapped to the j-th entry of L. If d_i includes the term l_j, then $x_j^{(i)} > 0$, otherwise, we set $x_j^{(i)} = 0$. Obviously, since the number m of terms in the dictionary is very large and each document includes

only a small fraction of them, the produced vectors will contain a large number of zero elements.

One of the most popular BOW methods for determining the elements of $\mathbf{x}^{(i)}$ is the tf-idf model that applies the following formula:

$$x_j^{(i)} = \begin{cases} 0 & l_j \notin D_i \\ \text{tf}_j^{(i)} \cdot \text{idf}_j & l_j \in D_i \end{cases} \quad (1)$$

where $\text{tf}_j^{(i)}$ is the frequency of l_j in the document D_i, and $\text{idf}_j = \log(n/f_j)$ is the inverse document frequency of l_j, where f_j is the total number of documents that contain l_j.

The relevant literature includes a significant number of variants to tf-idf. For example, tf is a simplified variant that does not take into consideration the inverse document frequency, whereas tp-idf incorporates the position of a term in a document leading to semantically better representations. In this paper, we examine only the standard tf-idf. Nevertheless, the conclusions of this study apply to these variants too.

The embeddings are an alternative strategy for achieving text vectorization. One of the pioneering works in the area was Word2Vec [40], a group of unsupervised learning techniques that convert a word into a low-dimensional vector representation. Apart from their low dimensionality, the Word2Vec vectors are semantically meaningful, in a sense that similar words are represented by similar vectors. Other popular models for generating text vectors are Glove [16], BERT [19], and so forth.

Word2Vec comprises two model architectures: continuous bag-of-words (CBOW) and skip-gram. In both cases, the algorithm applies a fix-sized window that slides over the individual words of the documents. The difference between these two architectures is that CBOW predicts the current word from the window of surrounding context words, whereas skip-gram uses the current word to predict the surrounding window of context words. Word2Vec has been used heavily in many NLP applications. Current experience indicates that CBOW is faster, whereas skip-gram is more effective when treating rare words.

4 Dimensionality Reduction

The analysis begins with an $n \times m$ matrix $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}]^T$ that stores n m-dimensional input vectors. We consider that n and m are quite large, so $\mathbf{X}$ cannot be processed efficiently by a machine learning algorithm. In dimensionality reduction, the goal is to generate an $n \times d$ matrix $\mathbf{Z}$ so that: i) $d \ll m$, and ii) $\mathbf{Z}$ represents the properties of $\mathbf{X}$ as accurately as possible.

As mentioned earlier, there are mainly two kinds of methods that achieve this twin goal: the *feature selection (FS)* and *feature extraction (FE)*. In the first category, only the d most representative columns of $\mathbf{X}$ are preserved with regards to some criteria and the rest $m - d$ columns are discarded. In contrast, the feature extraction methods derive $\mathbf{Z}$ by applying a function, or a mathematical process to $\mathbf{X}$.

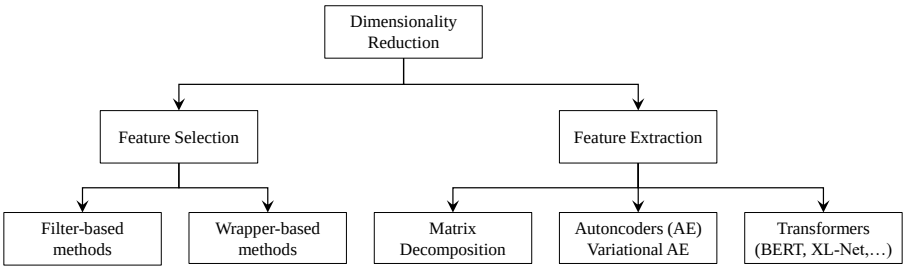

Fig. 1 Hierarchical categorization of dimensionality reduction algorithms.

In the following subsections, we describe the current state-of-the-art algorithms from both categories. An indicative hierarchy is illustrated in Figure 1.

4.1 Feature Selection

In feature selection, also known as variable selection, we are interested in preserving only the d most representative columns of $\mathbf{X}$.

4.1.1 Filter-based Feature Selection

These methods are applicable to supervised learning problems like the sentiment analysis task that is studied here. They select the d best components from the vectors of $\mathbf{X}$, based on the results of univariate statistical tests. The purpose of these tests is to quantify the correlation of each feature with the class labels and, eventually, determine how a single feature affects the class label. In this context, multiple types of tests can be performed, including the Analysis of Variance (ANOVA) F-value, information gain, mutual information, chi-squared (for non-negative features), Pearson correlation coefficient, Kendall's τ, and so on.

Compared to the Wrapper-based FS approaches, the Filter-based techniques do not require training a classifier. For this reason, they are significantly faster, whereas they are also considered less prone to overfitting.

4.1.2 Wrapper-based Feature Selection

The methods of this family are based on model-generated estimates of the importance of each input variable. More specifically, a classifier is trained by using different subsets of features, while it simultaneously computes the value of an importance metric [22]. For example, if the classifier is a linear model, such as Logistic Regression or Support Vector Machines (SVM), the importance can be represented by the value of its respective co-efficient in the model's equation. In the tree-based classifiers, like Decision Trees (DT) and Random Forests (RF), other measures can be applied, such as the Gini impurity, the Mean Decrease Accuracy, the Entropy, etc [41].

The recursive feature elimination algorithms initially train a classifier on the original set of features $\mathbf{X}$, and then acquire the importance of each feature

by considering one of the aforementioned model-generated estimates [42]. Next, the least important features are recursively removed from the feature set, until a d-dimensional set $\mathbf{Z}$ is eventually formed.

Contrary to recursive elimination, the sequential selection methods adopt a greedy approach [43]. The Forward-Sequential methods begin with an empty set of selected features ($\mathbf{Z} = \emptyset$), and greedily add new features to $\mathbf{Z}$ [44]. Initially, a classifier is trained by using all features and the most important among them, say x, is added to $\mathbf{Z}$. The classifier is trained again on the remaining set of features (i.e., after removing x), indicating a new feature to be added to $\mathbf{Z}$. The process terminates after d iterations, namely, when the set $\mathbf{Z}$ accommodates d elements.

On the other hand, the Backward-Sequential techniques operate on the reverse: they begin by filling $\mathbf{Z}$ with all m features of $\mathbf{X}$ (i.e., $\mathbf{Z} = \mathbf{X}$), and they greedily remove the least important among them [45]. The process stops when $\mathbf{Z}$ is left with d elements, and that happens after $m - d$ iterations.

Notice that Forward- and Backward-Sequential selection do not produce equivalent results. Since the requirement $d \ll m$ holds in text mining applications, then $d \ll (m - d)$ also holds. Therefore, the Forward-Sequential methods are considerably faster. Nonetheless, the Wrapper-based methods are in general much more computationally expensive than the Filter-based ones. Moreover, since the Wrapper-based methods train a machine learning model by using different sets of features, the probability of overfitting is high.

4.2 Feature Extraction

4.2.1 Matrix Factorization/Decomposition

The algorithms of this category achieve reduction by projecting the m-dimensional input vectors onto a d-dimensional space. Simultaneously, the computation of the d basis vectors of the latent space is performed, in such a way, so that one or more criteria is optimized.

In this spirit, Principal Component Analysis (PCA) considers that the best features are the ones that exhibit the highest degrees of variance. To prevent the input variables with large values —therefore, with large absolute variances— from dominating over the ones with small values, the original features are initially standardized (i.e., centered around 0).

In the sequel, the covariance between each pair of (standardized) features is computed, forming a $m \times m$ square covariance matrix $\mathbf{C}$. $\mathbf{C}$ is subsequently decomposed, and the d eigenvectors with the highest eigenvalues are used to fill the columns of a $m \times d$ projection matrix $\mathbf{W}$. The multiplication of the initial $n \times m$ matrix $\mathbf{X}$ with the $m \times d$ projection matrix $\mathbf{W}$ yields the $n \times d$ target matrix $\mathbf{Z}$; namely, $\mathbf{Z} = \mathbf{XW}$.

The application of the original PCA is problematic in applications that involve highly dimensional sparse vectors due to the standardization process. Hence, subtracting the mean value from the zero-valued vector components turns them into non-zero, rendering method computationally expensive. The

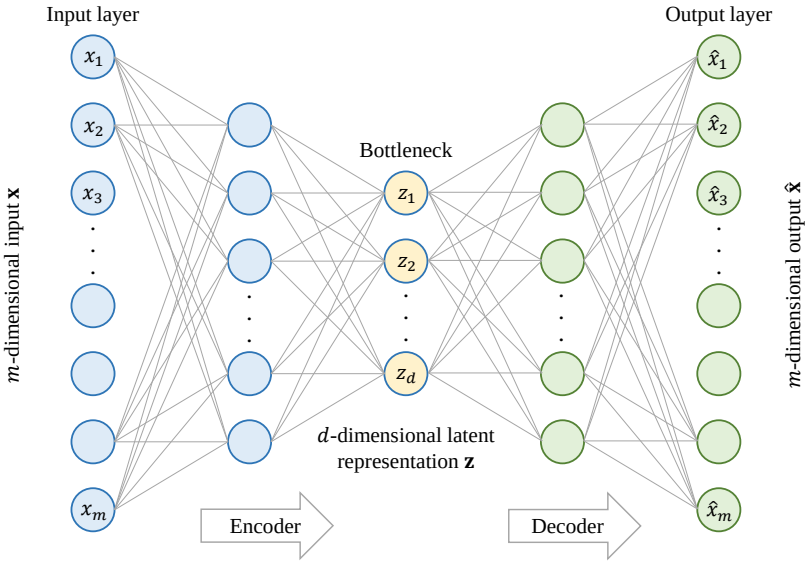

Fig. 2 A typical Autoencoder with 3 hidden layers.

approach that is considered more appropriate in this case is Singular Value Decomposition [27], and, particularly, a variant called Truncated SVD [46].

SVD is an algebraic process that decomposes an $n \times m$ matrix $\mathbf{X}$ into a square $n \times n$ matrix $\mathbf{U}$, a diagonal $n \times m$ matrix $\mathbf{\Sigma}$, and a $m \times m$ square matrix $\mathbf{V}$, so that $\mathbf{X} = \mathbf{U\Sigma V}^*$. Although it adopts a different logic, it yields results that are equivalent to those of PCA without standardization.

Non-Negative Matrix Factorization (NMF) is another technique that incorporates the additional constraint of non-negativity [26]. The goal is to decompose $\mathbf{X}$ into a $n \times d$ basis matrix $\mathbf{Z}$ and a non-negative $d \times m$ matrix $\mathbf{V}$, so that $\mathbf{X} \approx \mathbf{ZV}$. NMF computes $\mathbf{Z}$ and $\mathbf{V}$ by minimizing the error function, using the Frobenius norm $\|\mathbf{X} - \mathbf{ZV}\|_F$, subject to $\mathbf{Z} \geq 0$ and $\mathbf{V} \geq 0$. Apart from dimensionality reduction, NMF has native clustering capabilities, as it groups together the columns of the input matrix $\mathbf{X}$. Therefore, it is occasionally utilized in text clustering applications.

4.2.2 Autoencoders

The Autoencoder (AE) is topologically identical to a standard, fully-connected feed-forward neural network [47, 48]. Conceptionally, it is an unsupervised learning model, trained to reproduce its input $\mathbf{x}$ as accurately as possible. Hence, if $\hat{\mathbf{x}}$ is the output of the network, then the autoencoder attempts to minimize the reconstruction error $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$ that quantifies the distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$.

Figure 2 illustrates an Autoencoder with one input layer, one output layer and 3 hidden layers. The left part of this architecture comprises the Encoder part that transforms the input m-dimensional vectors $\mathbf{x} \in \mathbf{X}$ into

d-dimensional latent representations $\mathbf{z}$. Then, the bottleneck layer transfers these representations to the Decoder part, that, in turn, outputs a reconstruction $\hat{\mathbf{x}}$ of the initial vectors $\mathbf{x}$, so that $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$ is minimized. In most cases, the architecture of the Decoder is a mirror of that of the Encoder.

In practice, the loss functions optimized by an Autoencoder are usually linear combinations of the reconstruction error $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$ and a regularization parameter ϵ. This parameter is used to prevent the model, also known as Contractive Autoencoder, from simply memorizing the input, thus limiting the risk of overfitting. After the model has been trained, the Decoder part is discarded, and the Encoder is used to transform a m-dimensional input into a d-dimensional latent representation.

Although both the traditional matrix factorization techniques, like PCA and SVD, and the Autoencoders generate low-dimensional latent representations, the latter is capable of discovering non-linear relationships among the input variables. From this perspective, the Autoencoders learn non-linear manifolds, in contrast to PCA that projects the original data onto low-dimensional hyperplanes.

4.2.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a recent family of innovative language models that generate powerful vectorial text representations in an unsupervised manner [19]. One of the most remarkable features of BERT is that large models, that have been pre-trained with a variety of Web-scale text collections (e.g., Wikipedia), can be fine-tuned with respect to a particular application (e.g., question and answer datasets). This is achieved by simply appending an additional output layer after the pre-trained model, allowing the introduction of highly effective models.

In accordance to its name, BERT trains a Transformer on a bidirectional fashion [14, 49]. Transformers are deep learning Encoder-Decoder models that employ multi-head attention mechanisms with the aim of mapping a query and a set of key-value pairs to an output. In other words, the attention mechanism captures the semantic correlations between the words of a text. The experimental results on the relevant literature demonstrated that the bidirectional training of a Transformer can better capture the language context, compared to the single-directional models that adopt a left-to-right or right-to-left strategy.

The training process of BERT involves two key strategies: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). During training, the loss functions of MLM and NSP are optimized together in order to minimize the combined loss of the two strategies. The following list briefly outlines the basic elements of MLM and NSP.

- Masked Language Modeling (MLM): Given a sequence of words to be fed to BERT, this strategy replaces a percentage, say 15%, of the words with a mask token. In the sequel, the model tries to predict the masked words by analyzing the content of the adjacent, non-masked words. This is achieved

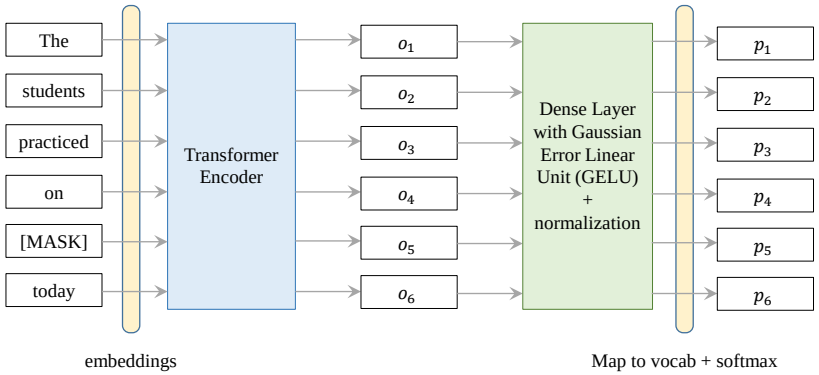

Fig. 3 Masked Language Modeling (MLM) with BERT.

by passing the output of the Transformer encoder to a classification layer (Figure 3). The output vectors are then multiplied by the embedding matrix and the probability of each word in the dictionary is computed by using the softmax function.

- Next Sentence Prediction (NSP): The input of this task is a pair of sentences. Given the first sentence in the pair, the goal of NSP is to predict whether the second sentence is subsequent to the first one inside a document. During training, only half of the input pairs contain sentences that are really subsequent in a document. The other half contain random sentences from the corpus.
  Before it is fed to the model, special tokens are injected into the text to indicate its beginning and the separation of the sentences. Then, each token is assigned an embedding that consists of 3 parts: the word, sentence, and positional embeddings. NSP predicts whether the second sentence is subsequent of the first one by firstly feeding the input to the Transformer. Then, a probability that reflects the connection of the two sentences is computed by applying softmax to the output of a simple classification layer.

As of May 2023, there are at least 24 pre-trained BERT models of different sizes, according to the number of their stacked hidden layers and the dimensionality of the vector representations[1] [50]. In addition, a significant number of alternatives have been proposed. RoBERTa is a popular BERT alternative that embodies several straightforward changes and carefully optimizes the training hyperparameters [51]. Its performance is considered superior to that of BERT. On the other hand, ALBERT [52] and DistilBERT [53] have been created to improve the training durations of BERT, while they yield only small performance sacrifices.

[1] https://github.com/google-research/bert

Table 1 Dataset characteristics

Dataset	Instances	d	Classes
IMDb Movie Reviews	50000	77026	2
Twitter US Airline Sentiment (TUSA)	14640	9849	3
Financial Tweets Sentiment (FTS)	28437	12138	3
Amazon Reviews	76295	26185	5

5 Experiments

This section describes the experimental part of our study. It is organized as follows: Subsections 5.1, 5.2, and 5.3 contain brief descriptions of the utilized datasets, the sentiment classification algorithms, and the dimensionality reduction methods that took part in the evaluation process. Next, the effectiveness and efficiency measurements are presented in Subsections 5.5 and 5.6, whereas a summarized analysis of the results is subsequently provided in the discussion of Section 6.

All the experiments were conducted on a commodity workstation with 32GB of RAM and an Intel CoreI7 12700K CPU. The system was also equipped with an NVIDIA GTX 3060 GPU that was used to accelerate the training of several deep learning models.

The code that we developed has been uploaded to a public GitHub repository[2] and it is free to modify, reproduce, and redistribute.

5.1 Datasets

The effects of dimensionality reduction in sentiment analysis applications were examined by using four popular, publicly available datasets. All of them were processed according to the methodology described in Section 3 to derive the tf-idf vectors and the CBOW embeddings. Additional filters were also applied including case-folding, punctuation removal, and stop word elimination. The number of documents in each dataset, the dimensionality of the input tf-idf vectors, and the number of the involved classes (i.e., opinion polarities) are shown in columns 2, 3, and 4 of Table 1, respectively.

The IMDb dataset[3] is a collection of 50000 movie reviews that serves as a benchmark dataset in numerous NLP tasks. It comprises two classes (positive or negative opinion); therefore, it is suitable for binary text classification problems. The tf-idf input vectors included approximately 77 thousand components, causing the curse-of-dimensionality to appear immediately: the input matrix contains about 3.85 billion values.

Since matrices of such sizes are overwhelming for a typical workstation, we applied another filter by limiting the vocabulary size to 15000 words. The filter initially performed stopword removal and, then, the 15000 most frequent words across the corpus were preserved. This approach rendered the executions of the involved algorithms feasible, while it yielded infinitesimal accuracy losses.

[2] https://github.com/lakritidis/SentimentAnalysis
[3] https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

The Amazon Reviews (AR) dataset[4] is another case of high-dimensional input. It comprises roughly 76 thousand product reviews, accompanied by numerical user opinions in the range $[1, 5]$. Each value in this range represents a different class. The produced tf-idf vectors contained about 26 thousand elements.

Twitter is a valuable data source for studying opinion mining problems. Hence, Twitter US Airline (TUSA)[5], includes more than 14 thousand tweets with ternary user opinions (positive/negative/neutral) on five major US airlines. Moreover, Financial Tweets Sentiment (FTS) is a crawl of about 28 thousand tweets[6] with ternary opinions on publicly traded companies. Each of these opinions (positive/negative/neutral) represent the three classes of the dataset.

5.2 Sentiment Classification Models

Although the deep learning text classifiers have been proved quite accurate in sentiment analysis tasks, for completeness reasons, we also included a number of classical machine learning algorithms in our tests. Specifically, the models that we employed are:

- Logistic Regression (LR): We used the LBFGS (Limited-memory BFGS) optimization algorithm with L2 regularization. LBFGS is memory-efficient approximation of the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) [54]. The maximum number of iterations was set to 300.
- Support Vector Machines (SVM): The Radial basis Function (RBF) kernel with L2 regularization was used for fitting. The regularization parameter and the kernel coefficient were set equal to $C = 1.0$ and $\gamma = 1/(d \cdot \sigma(X))$, where $\sigma(X)$ is the variance of the training example.
- Decision Trees (DT): The model was trained without setting a restriction to the maximum depth of the structure. So, we continued splitting the tree nodes until all leaves became pure. The Gini impurity measure was used to quantify the quality of each split. We also set the minimum number of samples required to split an internal node equal to 2.
- Random Forests (RF): The number of trees in the forest was set equal to 100. Each estimator was trained on the entire dataset by applying the same hyper-parameters as for DTs. The training process was performed in 8 parallel threads.
- Feed-Forward Neural Network (FFNN): This is a typical Multilayer Perceptron model with two fully-connected hidden layers including 50 and 300 neurons, respectively. We selected the Rectified Linear Unit (ReLU) function to compute neuron activations [55].
- Long Short-Term Memory RNN (LSTM): This model comprises 2 LSTM layers each one followed by a Dropout layer to prevent overfitting [56]. The Dropout rate was set equal to 0.2. The number of units in each LSTM layer

[4] https://jmcauley.ucsd.edu/data/amazon/
[5] https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[6] https://www.kaggle.com/vivekrathi055/sentiment-analysis-on-financial-tweets

was set equal to the dimensionality of the latent output space $\mathbf{Z}$, whereas their activation function was ReLU. We experimented with three such latent spaces with $d = 200$, $d = 500$, $d = 1000$ dimensions, so the number of units in each LSTM layer was 200, 500, and 1000, respectively. The output of the two LSTM layers was fed into two fully-connected layers with a Dropout layer (Dropout rate=0.2) between them.
The logistic function was selected as the activation of the output layer for binary classification, whereas softmax was used for multi-class classification. Regarding the initializers, we used Glorot [57], Orthogonal, and Zeroes initialization for the kernel weight matrix, the recurrent weight matrix and the biases, respectively. The training phase was conducted in batches of 256 items, through 20 epochs, that were accelerated by the underlying GPU.

- Recurrent Neural Network with bidirectional LSTM units (BiLSTM): The architecture consists of a single Bidirectional LSTM layer with identical hyper-parameters as those of the previous LSTM model. The output of BiLSTM was passed through a Dropout layer with a rate equal to 0.3, that, in turn, was connected to a single fully-connected layer. The activation functions were selected in the same spirit; namely, logistic and softmax for binary and multi-class tasks, respectively. Similarly to the LSTM model, BiLSTM was trained in 20 GPU-accelerated epochs and in batches of 256 samples.

The aforementioned hyperparameters have been applied uniformly in both binary (the IMDb dataset) and multi-class classification problems (the rest 3 datasets). All classes were considered of equal importance, so they were assigned a weight equal to 1. We used the binary cross-entropy loss function for the binary LR, FFNN, LSTM, and BiLSTM and the categorical cross-entropy loss function for the respective multi-class models. Regarding the SVM classifier, we applied the standard squared hinge loss function.

5.3 Dimensionality Reduction Methods

Each classifier of the previous subsection was trained on all four benchmark datasets. For each dataset, the following dimensional spaces were formed:

- Original: the feature space where the initial tf-idf vectors reside into.
- $d = 200$, $d = 500$, $d = 1000$: Three spaces with different numbers of basis vectors were formed by applying i) Truncated SVD, ii) NMF, and iii) an Autoencoder model to the initial tf-idf vectors. Regarding the Autoencoder model, we implemented the simple architecture of Figure 2.
- $d = 200$, $d = 500$, $d = 1000$: Three additional spaces with different numbers of basis vectors were created by applying the Continuous Bag of Words (CBOW) architecture of Word2Vec.
- $d = 200$, $d = 500$, $d = 1000$: Three spaces with different numbers of basis vectors were formed by applying the d-best feature selection (FS) method on the original data. The best features were identified by computing the mutual information (MI) between each input variable and the target variable. Then,

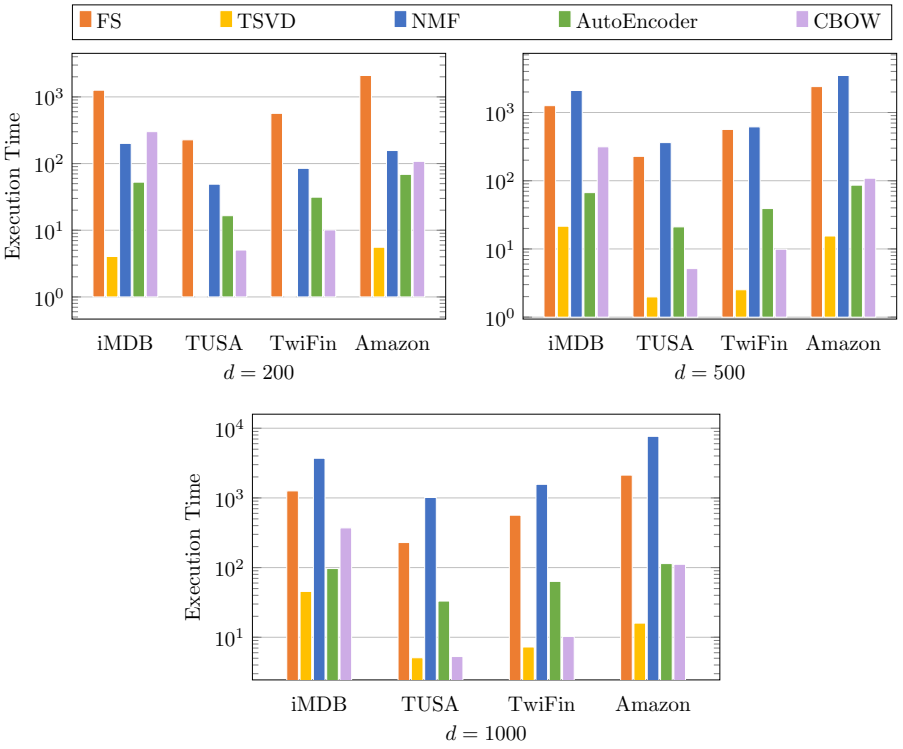

Fig. 4 Comparative bar plots of the execution times of various dimensionality reduction methods on the four datasets of Table 1.

the d variables with the highest MI scores were selected, whereas the rest of them were simply discarded from the dataset.

Figure 4 illustrates the running times of the aforementioned methods on the four datasets of Table 1. Each diagram concerns target spaces of different dimensionalities, namely $d = 200$, $d = 500$, and $d = 1000$. The vertical axes are plotted in logarithmic scale.

The results are unambiguous: in all cases, TSVD was, by far, the most efficient method. It outperformed all the adversary methods by one or more orders of magnitude, and this performance renders it an attractive approach for reducing data dimensionality. The second place was shared between the Autoencoder model and CBOW. In particular, on the IMDb and Amazon datasets, the former was faster than the latter, whereas the situation was reversed on the other two datasets.

On the other hand, FS was the slowest method on the 200-dimensional space. This is a rather expected observation, as the pairwise computations of mutual information are a computationally expensive process, and the cost increases linearly to the number of features. On the positive side, the running times of FS are unaffected by the size of the target space. Therefore, it takes

about the same amount of time to identify the 200, 500, or 1000 best features from a pool of tens of thousands of features.

In contrast, the running times of NMF are largely affected by the size of the target space due to the increased size of the output matrix. For this reason, NMF becomes the slowest method in the other two dimensional spaces ($d = 500$ and $d = 1000$).

5.4 Model Training

Before applying dimensionality reduction, each input dataset was split into a training and a test set with stratified sampling. Stratification guarantees approximately equal distributions of the target variable across the different splits. 70% of the input dataset was used for training the classifiers, whereas the rest 30% was reserved for testing their performance. The 5-fold cross validation technique was employed for validation.

To ensure reliable measurements, and to avoid the effect of "leakage" of the training set into the test set, each subset was preprocessed individually. In this context, the aforementioned preprocessing techniques (punctuation removal, case-folding, vectorization, etc.) were performed separately in the training and the test set. The same applies to the selected dimensionality reduction method.

5.5 Accuracy Measurements

Next, we proceed to the presentation of how dimensionality reduction affects the performance of text classification in sentiment analysis tasks. In this subsection, we focus on the accuracy measurements, while the running times of the classifiers are presented in the next subsection.

Figures 5, 6, 7, and 8 illustrate the accuracy achieved by each classifier in the IMDb, TUSA, Twitter Financial, and Amazon Reviews datasets, respectively. Each figure includes three diagrams that depict the algorithm performances on dimensional spaces of $d = 200$, $d = 500$, and $d = 1000$ features, top-to-bottom.

There are definitely a lot of numbers in these diagrams. Instead of describing them all, we will attempt to systematically derive the most important conclusions by grouping the measurements into three categories:

- *Conclusions (C)*: include the phenomena that are consistently observed in all 4 datasets, and in at least 2 out of 3 dimensional spaces.
- *Indications (I)*: are majority observations that are repeated in at least 3 out of 4 datasets, and in at least 2 out of 3 dimensional spaces.
- *Noticeables (N)*: include the observations that are repeated in at least 2 out of 4 datasets, and in at least 2 out of 3 dimensional spaces.

Non-Negative Matrix Factorization does not work well with Logistic Regression, SVMs, and the deep learning models LSTM and BiLSTM. More specifically, the accuracy degradation is particularly large in the linear models LR and SVM (C) and in the deep learning classifiers of the FTS dataset.

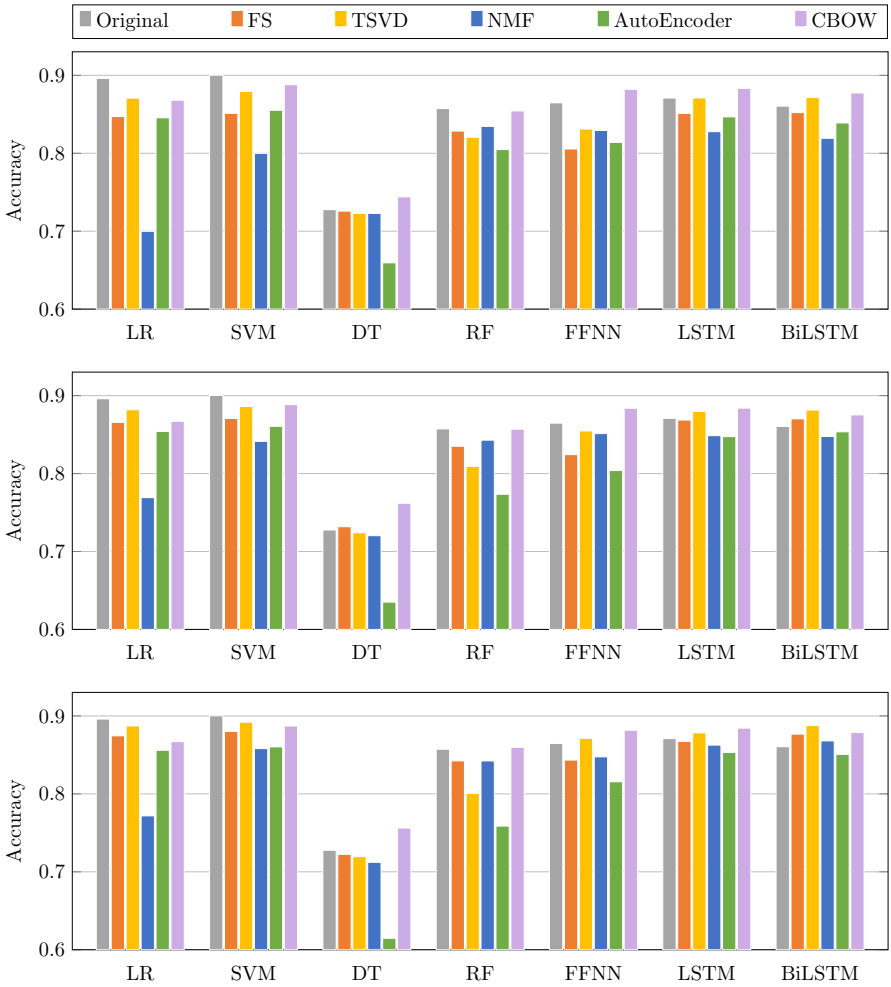

Fig. 5 Comparative bar plots of the accuracy of various classifiers on three low-dimensional feature spaces of the IMDb dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend.

In contrast, NMF is of some usefulness when tree-based learners are employed (C).

In contrast, Truncated Singular Value Decomposition co-operates effectively with the linear and the deep learning models (C). This is especially true in the 1000-dimensional spaces: when linear models are used, the classification accuracy is very close to the accuracy that is achieved in the original dimensional spaces (C). Remarkably, the deep learning classifiers are greatly benefited by TSVD, since the overall accuracy is improved in the reduced dimensional spaces over the original spaces (I).

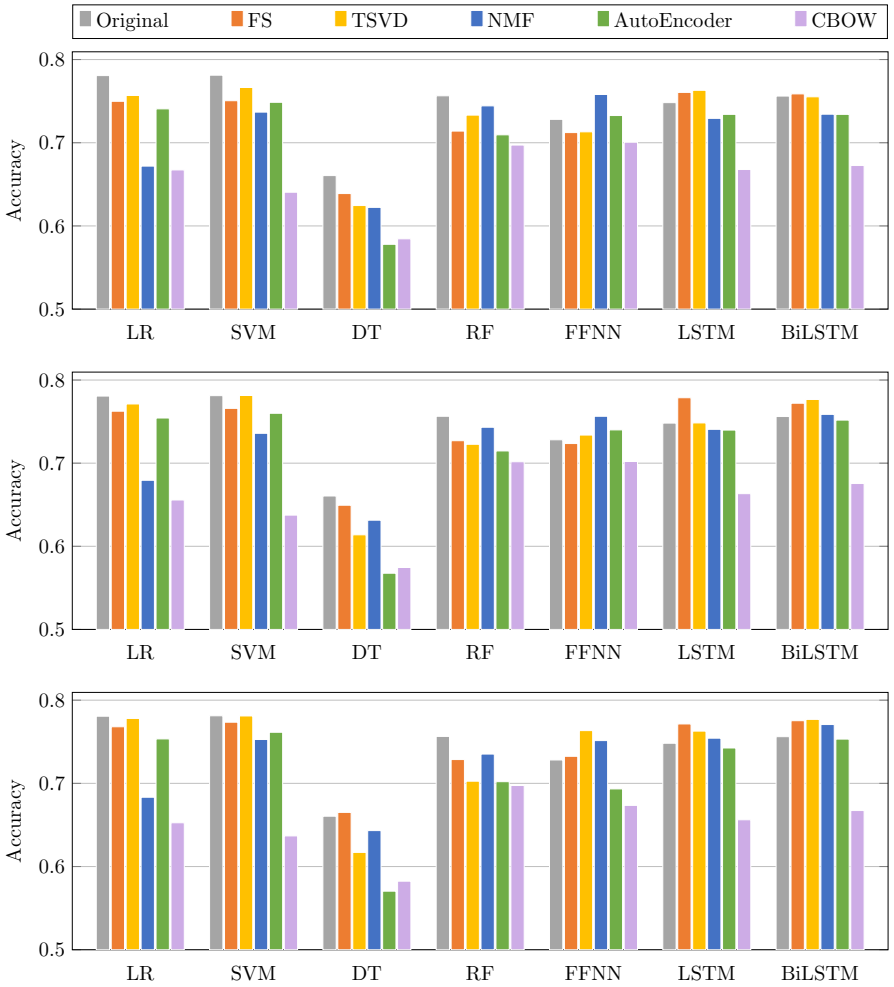

Fig. 6 Comparative bar plots of the accuracy of various classifiers on three low-dimensional feature spaces of the Twitter US Airline dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend.

Despite its simplicity, d-best feature selection (FS) was highly beneficial when deep learning models were employed. The results demonstrated that the performance of LSTM and BiLSTM was actually improved on the lower dimensional spaces, compared to the original input spaces (C). This means that, when the noisy features are removed from the data, these architectures can learn more effective models. With the exception of the IMDb dataset, FS was the most effective dimensionality reduction technique when working with LSTMs and BiLSTMs (I). Regarding the linear classifiers, FS was again among the most accurate methods (N). In particular, it outperformed all the other

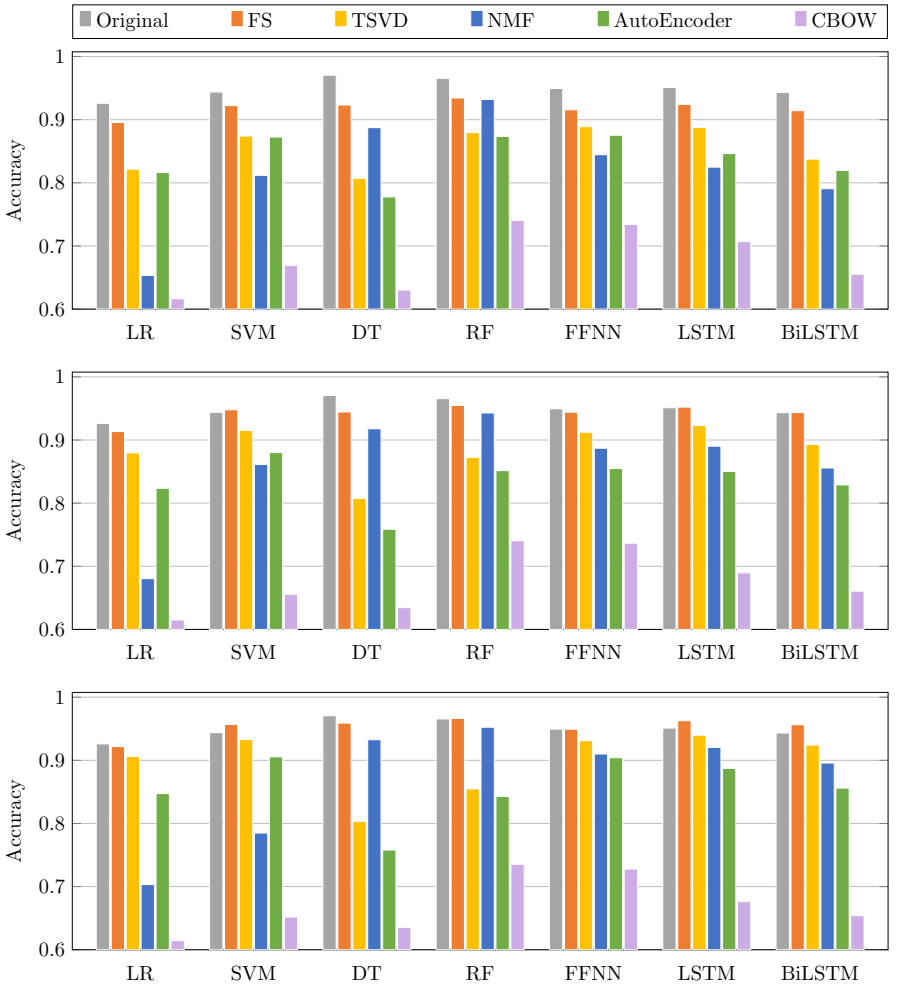

Fig. 7 Comparative bar plots of the accuracy of various classifiers on three low-dimensional feature spaces of the Financial Tweets Sentiment dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend.

methods on FTS and the Amazon Reviews dataset. As mentioned earlier, the most robust method on the IMDb and FTS datasets was TSVD. In most cases, FS was also effective when used in combination with tree-based learners (I).

Among all the tests that we conducted, there was not a single case where the Autoencoder-generated vectors led to remarkable accuracy results. Regarding LR, SVM, DT, and RF, the accuracy losses were significant in the majority of cases (I). These losses were somehow limited when neural nets were employed to conduct sentiment analysis. However, other techniques were more

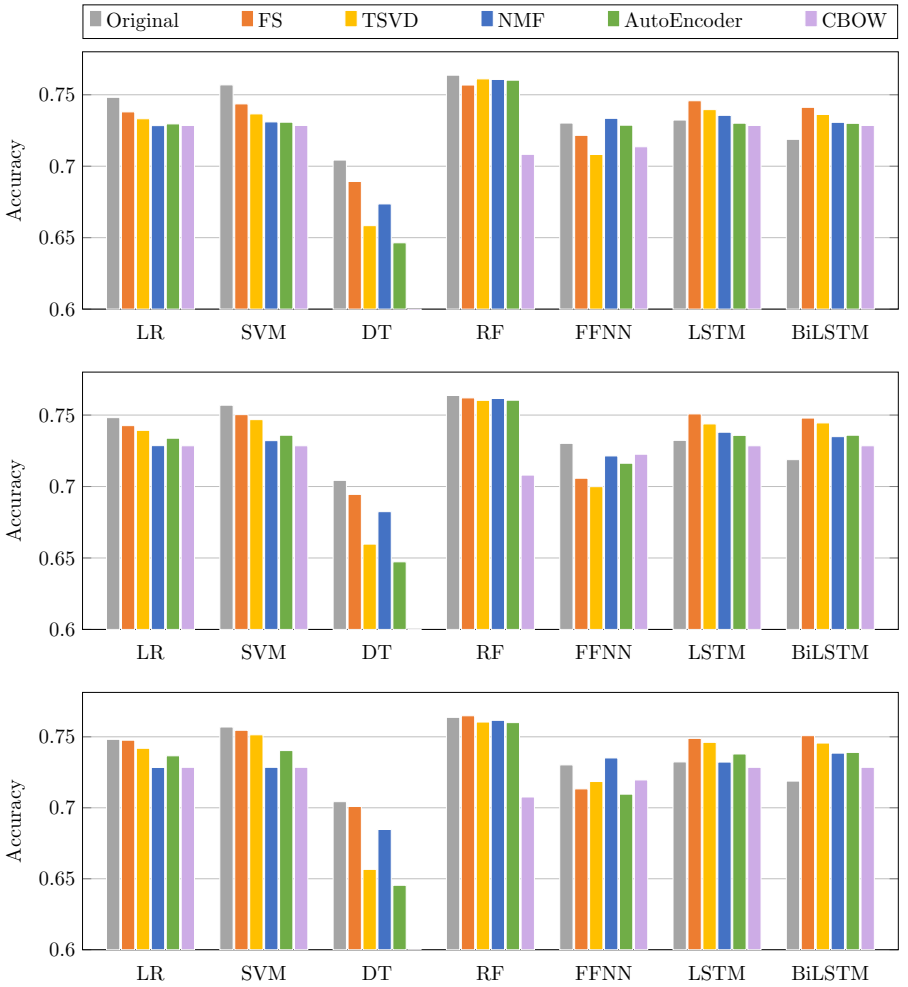

Fig. 8 Comparative bar plots of the accuracy of various classifiers on three low-dimensional feature spaces of the Amazon Reviews dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions respectively, generated by the 5 algorithms of the legend.

effective. In theory, the Autoencoders are able to capture non-linear relationships between the features. Nevertheless, this property was not verified experimentally by the simple architecture of Figure 2.

Finally, the results for CBOW were rather mixed. In the IMDb dataset, these vector representations performed quite well, especially when used in combination with deep learning models and tree-based classifiers. However, on the other datasets, the usage of Word2Vec embeddings resulted in accuracy degradations that in some cases were dramatic. Consequently, this method should

be used carefully, and not before some preliminary validation experiments indicate its usefulness (C).

Now, let us make some additional comments from the perspective of the classification algorithms: For Logistic Regression, Support Vector Machines, Long Short-Term Memory (LSTM) and Bidirectional LSTM, the best two dimensionality reduction algorithms were d-best Feature Selection and Truncated Singular Value Decomposition. Regarding the other three classifiers, the results were not so clear. More specifically, as to the IMDb dataset, CBOW was the most effective method for Decision Trees (DT), Random Forests (RF) and the Multilayer Perceptron (FFNN). On the other hand, concerning the Twitter US Airline and the Financial Sentiment datasets, the methods of choice for RF and FFNN were Non-Negative Matrix Factorization, and Feature Selection, respectively.

Finally, although a head-to-head comparison of the classification algorithms is out of the scope of this study, let us notice the effectiveness of the classical machine learning algorithms in the task of opinion mining. In the original feature spaces, these classifiers outperformed the deep learning models FFNN, LSTM, and BiLSTM. It was only after dimensionality reduction took place in some cases, where the deep learning architectures achieved superior performance. This indicates the competitiveness of the classical classifiers, and highlights that deep learning models require a considerable amount of fine tuning and large training data volumes before they achieve top performance.

5.6 Time Measurements

The improvement of model training times is another crucial reason for applying dimensionality reduction algorithms at the underlying data. Figures 9, 10, 11, and 12 illustrate these times for the IMDb, TUSA, Twitter Financial, and Amazon Reviews datasets, respectively. Similarly to the analysis of the previous subsection, each figure is divided into three diagrams that represent the training times at spaces including 200, 500, and 1000 dimensions, from top to bottom. In all diagrams, the vertical axes are in logarithmic scale.

At first, let us explain several observations that initially seem counterintuitive. In most cases, the training times of the Random Forest classifier were lower than those of the Decision Tree. The question "How an ensemble model with 100 estimators can be trained faster than a single estimator?" has two answers. The first one is that the single Decision Tree was trained by utilizing all d features of the reduced dataset, whereas Random Forest was trained by using $\sqrt{d}$ features. The second explanation was mentioned earlier: Random Forest was trained on 8 parallel threads, partially exploiting the 12 processing cores and the 20 available threads of our workstation's CPU.

Another observation that requires clarification concerns the small training duration of our deep learning models. As mentioned earlier, the learning procedure of these architectures was massively accelerated by the 3584 CUDA cores of the installed GPU.

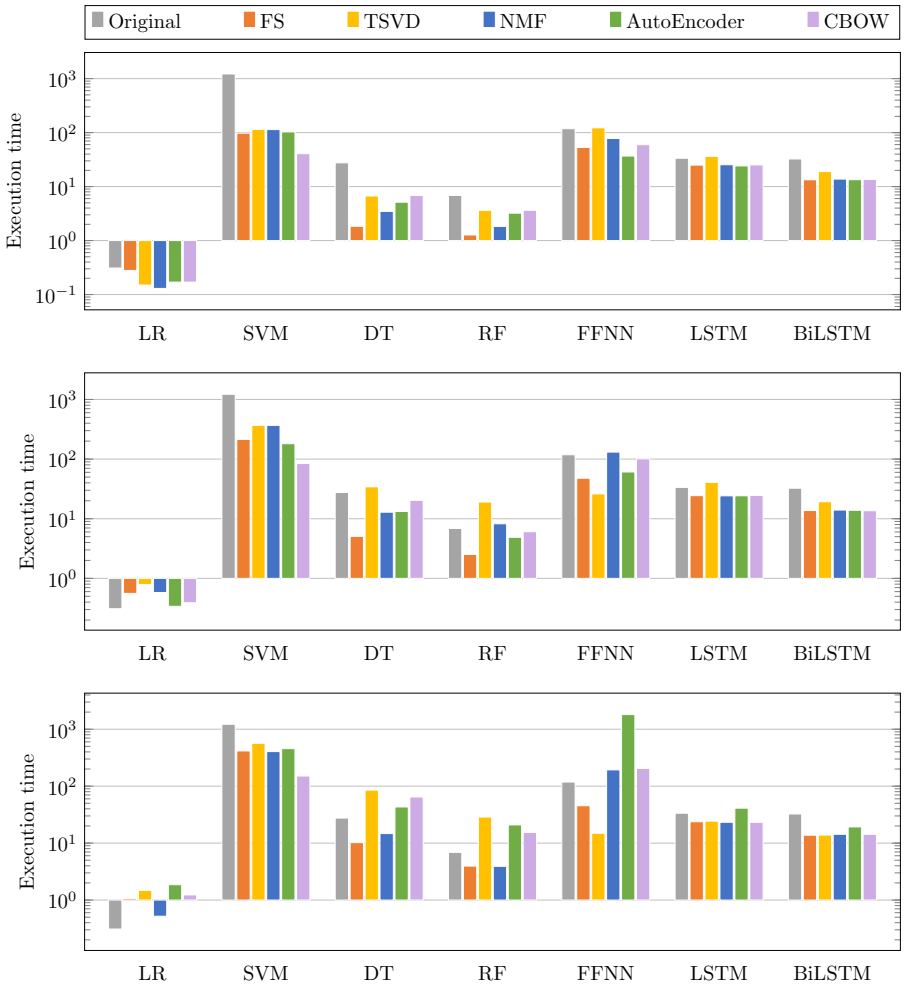

Fig. 9 Comparative bar plots of the execution times of various classifiers on three low-dimensional feature spaces of the IMDb dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend. The vertical axis is in logarithmic scale.

The diagrams indicate that, in the vast majority of cases, the application of a dimensionality reduction technique leads to a significant acceleration of the training procedure of the attested classifiers. However, we recorded several exceptions where the training times of some models were counter-intuitively increased. This is partially explained by the nature of the input data itself. Recall that the input documents are mostly reviews, comprised of a few tens of words. Therefore, although the respective tf-idf vector representations are very long, they include only a few tens of non-zero elements. In contrast, after dimensionality reduction, each document is represented by a dense vector that

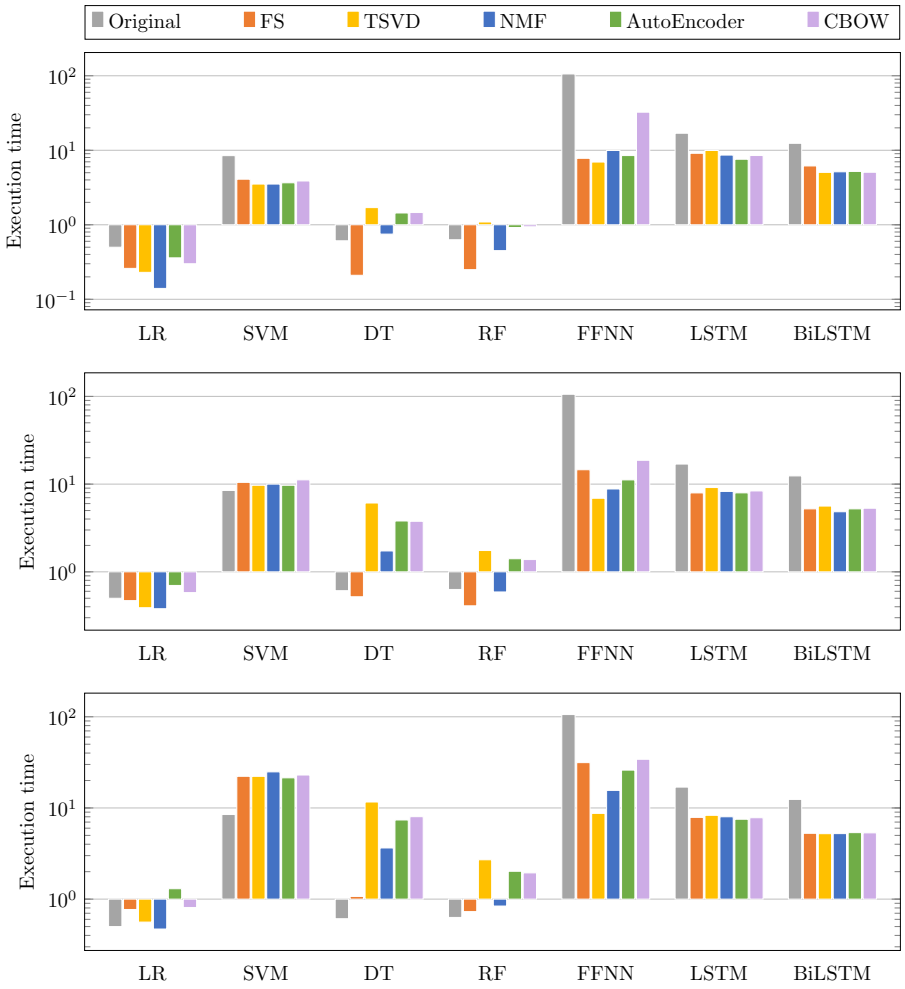

Fig. 10 Comparative bar plots of the execution times of various classifiers on three low-dimensional feature spaces of the Twitter US Airline dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend. The vertical axis is in logarithmic scale.

consists of a few hundreds non-zero elements. Consequently, in several cases, it may be computationally cheaper to train a model on large sparse vectors than short dense vectors.

In the discussion that follows, we adopt again the notions of *noticeable*, *indication* and *conclusion* that were introduced on the previous subsection. The first conclusion is that the application of dimensionality reduction was always beneficial for the training times of the BiLSTM model. Regarding the other two deep learning architectures, only FS, TSVD and CBOW consistently reduced the training durations of FFNN and LSTM in all four datasets. On

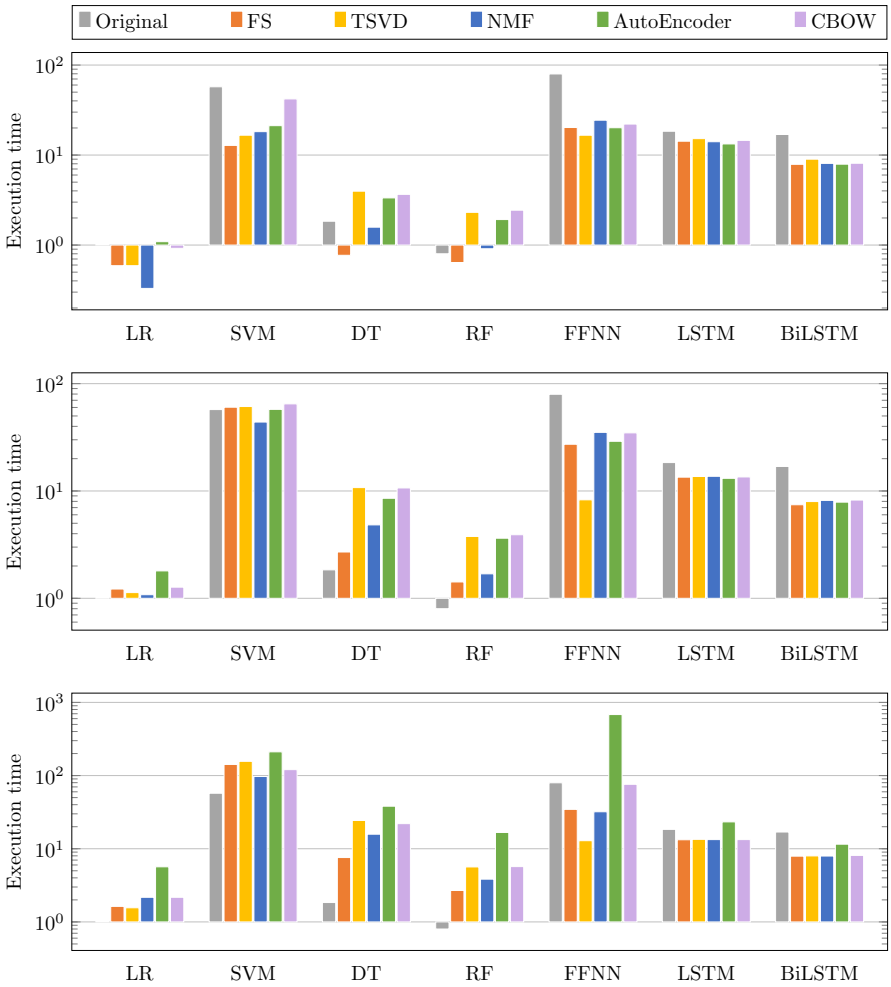

Fig. 11 Comparative bar plots of the execution times of various classifiers on three low-dimensional feature spaces of the Financial Tweets Sentiment dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend. The vertical axis is in logarithmic scale.

the other hand, the Autoencoder-generated vectors led to significant delays in FFNN and LSTM training, on the 1000-dimensional spaces of IMDb and FTS. Similarly, NMF had a negative impact on the training time of FFNN in the Amazon Reviews dataset.

The training process of the Logistic Regression classifiers was in all cases very fast, as it almost always consumed less than 10 seconds to complete. On the other hand, SVM with the RBF kernel was considerably slower. Dimensionality Reduction was beneficiary for this classifier only on the smallest dimensional space (C). Nevertheless, the situation was reversed on the 500

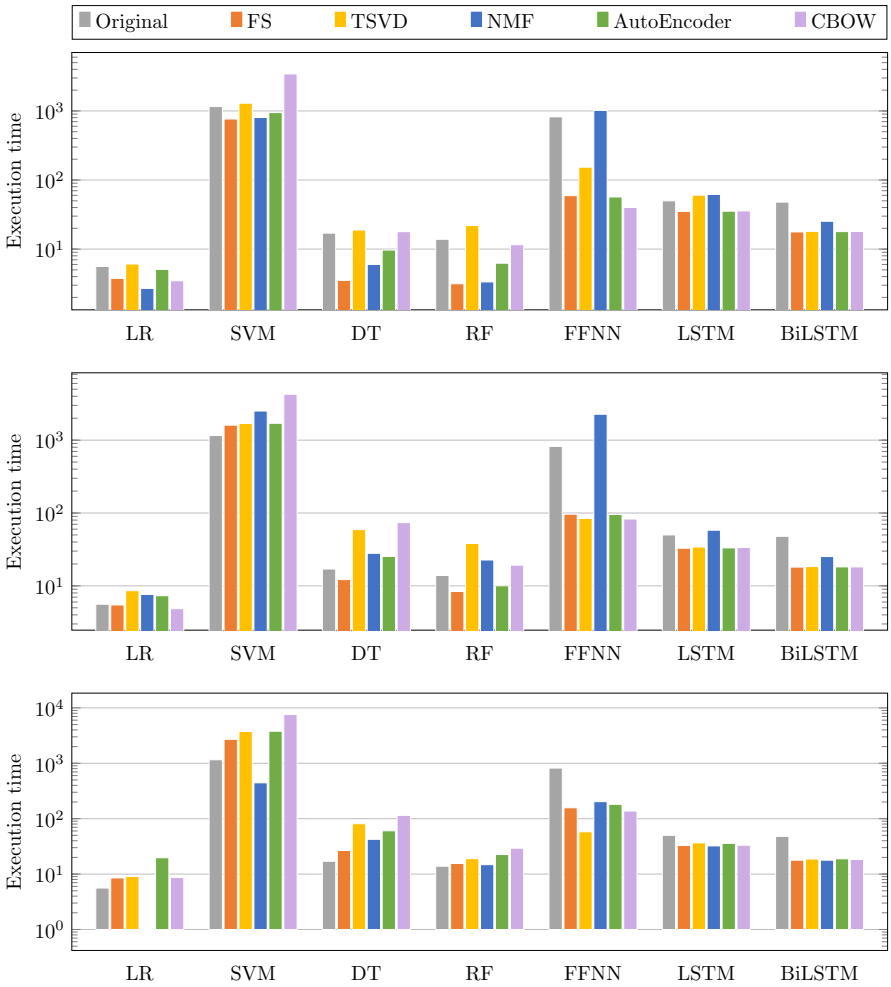

Fig. 12 Comparative bar plots of the execution times of various classifiers on three low-dimensional feature spaces of the Amazon Reviews dataset. The top, middle and bottom diagrams concern spaces of 200, 500, and 1000 dimensions, respectively, generated by the 5 algorithms of the legend. The vertical axis is in logarithmic scale.

and 1000-dimensional spaces, where on three of our four datasets, all methods decelerated the classifier significantly. The indication here is that, when SVM is going to be applied, then the target space must include 200, or fewer dimensions.

Regarding the tree based learners, Feature Selection was the method that yielded the greatest performance benefits (I). With the exception of the FTS dataset ($d = 500$ and $d = 1000$), this approach decreased the training times of both Decision Trees and Random Forests. It is remarkable that all the other reduction techniques had a negative impact in the training durations of both

models (C). A limited number of exceptions can only be found in small feature spaces that consist of fewer than 200 dimensions.

If we combine this observation with the accuracy measurements of the previous subsection, a more concrete conclusion derives: dimensionality reduction is not beneficial for Decision Trees and Random Forests, as, in the majority of cases, it decreases the prediction accuracy and increases the training times of these models.

6 Summary

In this section, we organize the key observations of the experimental evaluation that was described previously. In accordance to the analysis of Subsections 5.5 and 5.6, we group our findings in noticeable observations, indications, and strong conclusions. This information is summarized by the contents of Table 2.

In this table, the term "effective" reflects the ability of a dimensionality reduction technique to identify good features. Therefore, a reduction method is considered effective, when it causes no significant accuracy losses, or, even better, when it improves the accuracy of text classification.

Moreover, notice that the term "efficient" does not concern the running times of the reduction technique itself; we already examined this parameter in Subsection 5.3. In contrast, in Table 2, the "efficiency" of a dimensionality

Table 2 Summary of the findings of the experimental evaluation of Section 5. The terms "indication (I)", and "conclusion (C)" concern remarkable behaviors that have been repeatedly observed in 3 and 4 (out of 4) datasets, respectively.

Observation	Strength
FS improves the accuracy of LSTM and BiLSTM on the lower dimensional spaces, even when compared to the original input spaces.	C
FS is the most effective technique when deep learning LSTMs and BiLSTMs are applied.	I
FS and TSVD are the most effective methods when linear models (LR and SVM) are applied.	C
The simple Autoencoder-generated embeddings lead to significant accuracy losses.	I
The CBOW embeddings lead to mixed (small to large) accuracy losses. They should be used after validation.	C
Dimensionality reduction is always beneficial for the training times of BiLSTM.	C
FS, TSVD and CBOW are always beneficial for the training times of FFNN and LSTM.	C
AE and NMF are beneficial for the training times of FFNN and LSTM.	I
Dimensionality Reduction benefits the training times of SVM models when the target space is small (i.e., fewer than 200 dimensions).	C
With the exception of Feature Selection, dimensionality reduction has a negative impact on the training times of tree-based learners.	C

reduction method represents its ability to accelerate the training process of a sentiment classifier.

7 Conclusion

Sentiment analysis is among the most important applications of Natural Language Processing (NLP). It has been studied extensively as a text classification problem, and numerous researchers introduced various state-of-the-art models that improved the performance of the existing solutions. One of the greatest challenges in the area is the natural text sparsity and the high dimensionality that renders the input data barely manageable. Surprisingly, very few research works cover the issue of how the sentiment classification models perform with respect to the dimensionality of the input feature space.

In this paper, we conducted an experimental survey on the accuracy and the training times of multiple text classifiers in combination with various dimensionality reduction techniques. This work covers both classical and deep learning classifiers with respect to feature selection and feature extraction approaches. The results of our experiments demonstrated that, in the majority of cases, dimensionality reduction is indeed beneficiary for the training durations, while hurting accuracy by only a small margin. We also identified interesting cases where the reduced vector spaces render the underlying classifiers more effective. On the other hand, the tree-based learners, like Decision Trees and Random Forests, are not benefited, as the reduced spaces not only degraded their accuracy, but also rendered their training more computationally expensive.

Conflict of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

[1] Boldrini, E., Balahur, A., Martínez-Barco, P., Montoyo, A.: Using EmotiBlog to annotate and analyse subjectivity in the new textual genres. Data Mining and Knowledge Discovery **25**(3), 603–634 (2012)

[2] Akritidis, L., Bozanis, P.: Improving opinionated blog retrieval effectiveness with quality measures and temporal features. World Wide Web **17**(4), 777–798 (2014)

[3] Thelwall, M., Buckley, K., Paltoglou, G.: Sentiment in Twitter events. Journal of the American Society for Information Science and Technology **62**(2), 406–418 (2011)

[4] Stieglitz, S., Dang-Xuan, L.: Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. Journal of Management Information Systems **29**(4), 217–248 (2013)

[5] Ortigosa, A., Martín, J.M., Carro, R.M.: Sentiment analysis in Facebook and its application to e-learning. Computers in Human Behavior **31**, 527–541 (2014)

[6] Kaya, T., Bicen, H.: The effects of social media on students' behaviors; Facebook as a case study. Computers in Human Behavior **59**, 374–379 (2016)

[7] Mukherjee, S., Bhattacharyya, P.: Feature specific sentiment analysis for product reviews. In: Proceedings of the 13th International Conference on Intelligent Text Processing and Computational Linguistics, pp. 475–487 (2012)

[8] Medhat, W., Hassan, A., Korashy, H.: Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal **5**(4), 1093–1113 (2014)

[9] Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent Convolutional Neural Networks for text classification. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, pp. 2267–2273 (2015)

[10] Soong, H.-C., Jalil, N.B.A., Ayyasamy, R.K., Akbar, R.: The essential of sentiment analysis and opinion mining in social media: Introduction and survey of the recent approaches and techniques. In: Proceedings of the 9th IEEE Symposium on Computer Applications & Industrial Electronics, pp. 272–277 (2019)

[11] Ouyang, X., Zhou, P., Li, C.H., Liu, L.: Sentiment Analysis using Convolutional Neural Network. In: Proceedings of the 18th IEEE International Conference on Computer and Information Technology, pp. 2359–2364 (2015)

[12] Ma, Y., Peng, H., Khan, T., Cambria, E., Hussain, A.: Sentic LSTM: a hybrid network for targeted aspect-based sentiment analysis. Cognitive Computation **10**(4), 639–650 (2018)

[13] Jelodar, H., Wang, Y., Orji, R., Huang, S.: Deep sentiment classification and topic discovery on novel coronavirus or COVID-19 online discussions: NLP using LSTM Recurrent Neural Network approach. IEEE Journal of Biomedical and Health Informatics **24**(10), 2733–2742 (2020)

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in

Neural Information Processing Systems **30** (2017)

[15] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[16] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)

[17] Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: Proceedings of the 32nd International Conference on Machine Learning, pp. 957–966 (2015)

[18] Akritidis, L., Bozanis, P.: How dimensionality reduction affects sentiment analysis NLP tasks: An experimental study. In: Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 301–312 (2022)

[19] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep BiDirectional Transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[20] Abualigah, L.M., Khader, A.T., Al-Betar, M.A., Alomari, O.A.: Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Systems with Applications **84**, 24–36 (2017)

[21] Akritidis, L., Alamaniotis, M., Fevgas, A., Bozanis, P.: Confronting sparseness and high dimensionality in short text clustering via feature vector projections. In: Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence, pp. 813–820 (2020)

[22] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H.: Feature selection: A data perspective. ACM Computing Surveys **50**(6), 1–45 (2017)

[23] Jović, A., Brkić, K., Bogunović, N.: A review of feature selection methods with applications. In: Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, pp. 1200–1205 (2015)

[24] Humeau-Heurtier, A.: Texture Feature Extraction Methods: A survey. IEEE Access **7**, 8975–9000 (2019)

[25] Mutlag, W.K., Ali, S.K., Aydam, Z.M., Taher, B.H.: Feature extraction methods: A review. In: Journal of Physics: Conference Series, vol. 1591, p. 012028 (2020)

[26] Wang, Y.-X., Zhang, Y.-J.: Non-Negative Matrix Factorization: A comprehensive review. IEEE Transactions on Knowledge and Data Engineering **25**(6), 1336–1353 (2012)

[27] Shyamasundar, L., Rani, P.J.: Twitter sentiment analysis with different feature extractors and dimensionality reduction using supervised learning algorithms. In: Proceedings of the 2016 IEEE Annual India Conference, pp. 1–6 (2016)

[28] Kingma, D.P., Welling, M., *et al.*: An introduction to variational autoencoders. Foundations and Trends in Machine Learning **12**(4), 307–392 (2019)

[29] Liao, S., Wang, J., Yu, R., Sato, K., Cheng, Z.: CNN for situations understanding based on sentiment analysis of Twitter data. Procedia Computer Science **111**, 376–381 (2017)

[30] Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 606–615 (2016)

[31] Behera, R.K., Jena, M., Rath, S.K., Misra, S.: Co-LSTM: Convolutional LSTM model for sentiment analysis in social Big Data. Information Processing & Management **58**(1), 102435 (2021)

[32] Rhanoui, M., Mikram, M., Yousfi, S., Barzali, S.: A CNN-BiLSTM model for document-level sentiment analysis. Machine Learning and Knowledge Extraction **1**(3), 832–847 (2019)

[33] Zhang, T., Gong, X., Chen, C.P.: BMT-Net: Broad multitask transformer network for sentiment analysis. IEEE Transactions on Cybernetics **2**(57), 6232–6243 (2021)

[34] Zhang, T., Xu, B., Thung, F., Haryono, S.A., Lo, D., Jiang, L.: Sentiment analysis for software engineering: How far can pre-trained transformer models go? In: Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution, pp. 70–80 (2020)

[35] Zhang, L., Wang, S., Liu, B.: Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(4), 1253 (2018)

[36] Hussein, D.M.E.-D.M.: A survey on sentiment analysis challenges. Journal of King Saud University-Engineering Sciences **30**(4), 330–338 (2018)

[37] Venkatesh, B., Anuradha, J.: A review of feature selection and its

methods. Cybernetics and Information Technologies **19**(1), 3–26 (2019)

[38] Kim, K., Lee, J.: Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction. Pattern Recognition **47**(2), 758–768 (2014)

[39] Kim, K.: An improved semi-supervised dimensionality reduction using feature weighting: Application to sentiment analysis. Expert Systems with Applications **109**, 49–65 (2018)

[40] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[41] Grabczewski, K., Jankowski, N.: Feature selection with decision tree criterion. In: Proceedings of the 5th International Conference on Hybrid Intelligent Systems, pp. 1–6 (2005)

[42] Chen, X.-w., Jeong, J.C.: Enhanced recursive feature elimination. In: Proceedings of the 6th International Conference on Machine Learning and Applications, pp. 429–435 (2007)

[43] El Aboudi, N., Benhlima, L.: Review on wrapper feature selection approaches. In: Proceedings of the 2016 International Conference on Engineering & MIS, pp. 1–5 (2016)

[44] Ververidis, D., Kotropoulos, C.: Sequential forward feature selection with low computational cost. In: Proceedings of the 13th European Signal Processing Conference, pp. 1–4 (2005)

[45] Nguyen, H.B., Xue, B., Liu, I., Zhang, M.: Filter based backward elimination in wrapper based pso for feature selection in classification. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, pp. 3111–3118 (2014)

[46] Hansen, P.C.: The TruncatedSVD as a method for regularization. BIT Numerical Mathematics **27**(4), 534–553 (1987)

[47] Tschannen, M., Bachem, O., Lucic, M.: Recent advances in autoencoder-based representation learning. arXiv preprint arXiv:1812.05069 (2018)

[48] Meng, Q., Catchpoole, D., Skillicom, D., Kennedy, P.J.: Relational autoencoder for feature extraction. In: Proceedings of the 2017 International Joint Conference on Neural Networks, pp. 364–371 (2017)

[49] Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. Advances in Neural Information Processing Systems **34**, 15908–15919 (2021)

[50] Turc, I., Chang, M.-W., Lee, K., Toutanova, K.: Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. arXiv preprint arXiv:1908.08962v2 (2019)

[51] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692 (2019)

[52] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv preprint arXiv:1909.11942 (2019)

[53] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)

[54] Liu, D., Nocedal, J.: On the limited memory method for large scale optimization: Mathematical programming b. Mathematical Programming B **45**(3), 503–528 (1989)

[55] Nair, V., Hinton, G.E.: Rectified Linear Units improve restricted Boltzmann Machines. In: Proceedings of the 27th International Conference on Machine Learning, pp. 807–814 (2010)

[56] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent Neural Networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)

[57] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)