IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers

KOSTADIN MISHEV[1], ANA GJORGJEVIKJ[1], IRENA VODENSKA[2], LUBOMIR T. CHITKUSHEV[2], AND DIMITAR TRAJANOV.[1]

[1]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, Republic of North Macedonia (e-mail: {kostadin.mishev,dimitar.trajanov}@finki.ukim.mk, gjorgjevikj.ana@students.finki.ukim.mk)

[2]Financial Informatics Lab, Metropolitan College, Boston University, USA (e-mail: {vodenska, LTC}@bu.edu)

Corresponding author: Kostadin Mishev (e-mail: kostadin.mishev@finki.ukim.mk).

The work in this paper was partially financed by the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje.

ABSTRACT

Financial and economic news is continuously monitored by financial market participants. According to the efficient market hypothesis, all past information is reflected in stock prices and new information is instantaneously absorbed in determining future stock prices. Hence, prompt extraction of positive or negative sentiments from news is very important for investment decision-making by traders, portfolio managers and investors. Sentiment analysis models can provide an efficient method for extracting actionable signals from the news. However, financial sentiment analysis is challenging due to domain-specific language and unavailability of large labeled datasets. General sentiment analysis models are ineffective when applied to specific domains such as finance. To overcome these challenges, we design an evaluation platform which we use to assess the effectiveness and performance of various sentiment analysis approaches, based on combinations of text representation methods and machine-learning classifiers. We perform more than one hundred experiments using publicly available datasets, labeled by financial experts. We start the evaluation with specific lexicons for sentiment analysis in finance and gradually build the study to include word and sentence encoders, up to the latest available NLP transformers. The results show improved efficiency of contextual embeddings in sentiment analysis compared to lexicons and fixed word and sentence encoders, even when large datasets are not available. Furthermore, distilled versions of NLP transformers produce comparable results to their larger teacher models, which makes them suitable for use in production environments.

INDEX TERMS Sentiment Analysis, Finance, Natural Language Processing, Text Representations, Deep-learning, Encoders, Word Embedding, Sentence Embedding, Transfer-learning, Transformers, Survey

I. INTRODUCTION

The latest advances in Natural Language Processing (NLP) have received significant attention due to their efficiency in language modeling. These language models are finding applications in various industries as they provide powerful mechanisms for real-time, reliable, and semantic-oriented text analysis. Sentiment analysis is one of the NLP tasks that leverages language modeling advancements and is achieving improved results. According to the Oxford University Press dictionary [1], sentiment analysis is defined as the process of computationally identifying and categorizing opinions expressed in a text, primarily to determine whether the writer's attitude towards a particular topic or product is positive, negative, or neutral. Sentiment analysis is becoming an essential tool for transforming emotions and attitudes into actionable information.

Designing and building deep-learning-based sentiment analysis models require substantial datasets for training and testing. While there are several large, publicly available sentiment-annotated datasets, they are mostly related to products and movies. Many sentiment analysis models [1]–[4] use these datasets and achieve good performance in related domains. However, the application of these models in different domains is challenging because each domain has

[1]https://lexico.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

a unique set of words for emotion expression.

The financial domain is characterized by a unique vocabulary, which calls for domain-specific sentiment analysis. Prices observed in financial markets reflect all available information related to traded assets [5], hence new information allows stakeholders to make well-informed and timely decisions. The sentiments expressed in news and tweets influence stock prices and brand reputation, hence, constant measurement and tracking of these sentiments is becoming one of the most important activities for investors. Studies have used sentiment analysis based on financial news to forecast stock prices [6]–[8], foreign exchange and global financial market trends [9], [10] as well as to predict corporate earnings [11].

Given that the financial sector uses its own jargon, it is not suitable to apply generic sentiment analysis in finance because many of the words differ from their general meaning. For example, "liability" is generally a negative word, but in the financial domain it has a neutral meaning. The term "share" usually has a positive meaning, but in the financial domain, share represents a financial asset or a stock, which is a neutral word. Furthermore, "bull" is neutral in general, but in finance, it is strictly positive, while "bear" is neutral in general, but negative in finance. These examples emphasize the need for development of dedicated models, which will extract sentiments from financial texts.

Sentiment analysis in finance has become an important research topic, connecting quantitative and qualitative measures of financial performance. A seminal study by Loughran and McDonald [12] shows that word lists developed for other disciplines misclassify common words in financial texts. Hence, Loughran and McDonald created an expert annotated lexicon of positive, negative, and neutral words in finance, which better reflect sentiments in financial texts. In [13], the authors introduce a Twitter-specific lexicon, which, in combination with the DAN2 machine learning approach, produces more accurate sentiment classification results than support vector machine (SVM) approach while using the same Twitter-specific lexicon.

Machine learning methods for sentiment extraction have been applied on datasets of tweets or news [14]–[18]. In [15], the authors use various machine-learning binary classifiers to obtain StockTwits tweets sentiments. They show that the SVM classifier is more accurate compared to Decision Trees and Naïve Bayes classifier. In [16], Atzeni et al. test the performance of various regression models in combination with statistical and semantic methods for feature extraction to predict a real-valued sentiment score in micro-blogs and news headlines, and show that semantic methods improve classification accuracy.

Researchers have used lexicon-based approaches in combination with machine-learning models. The authors in [18] show that such combinations are more efficient for sentiment extraction than using single models. However, regular machine-learning methods are unable to extract complex features and to keep the order of words in a sentence. These tasks require the use of deep-learning approaches, which allow for complex feature extraction, location identification, and order information [19].

Deep-learning methods [20] use a cascade of multiple layers of non-linear processing units for complex feature extraction and transformation. Each successive layer uses the output from the previous layer as input, thus extracting complex features which in many cases can be useful for generating learning patterns and relationships beyond immediate neighbors in the sequence. Many studies confirm the efficiency of deep-learning models, including recurrent neural network (RNN) [21], [22], convolutional neural networks [23]–[25] and attention mechanism [19], [26] in sentiment extraction in finance. The great success of deep-learning approaches in NLP is mainly due to the introduction and improvement of text representation methods, such as word [27]–[29] and sentence encoders. [30]–[33] These convert words/sentences into vector representation, making them suitable as input for neural networks. These representations keep the semantic information coded into words and sentences, which is crucial for sentiment extraction.

Recent developments in NLP, deep-learning, and transfer-learning have significantly improved the sentiment extraction from financial news and texts [17], [34]–[37]. In [35], Yang et al. incorporate inductive transfer-learning methods such as ULMFiT [38] for sentiment analysis in finance, and the results show improvements in sentiment classification compared to traditional transfer-learning approaches. The superior performance of recent NLP transformers, BERT and RoBERTA, in sentiment analysis is evaluated in [37], where the effectiveness of using the RoBERTa model is compared to dictionary-based models.

Studies have used sentiment analysis based on financial news to forecast stock prices [6]–[8], foreign exchange and global financial market trends [9], [10] as well as to predict corporate earnings [11].

This paper aims to survey approaches to sentiment analysis, including combinations of machine-learning and deep-learning models with lexicon-based feature extraction methods and word and sentence encoders, up to the most recent NLP transformers. The goal is to apply these approaches to finance. We evaluate and compare model effectiveness when trained under same conditions and on the same dataset. The main contribution of this paper is the development of an evaluation platform, which we use to assess the performance of NLP methodologies for text feature extraction in finance.

We show that recent advances in deep-learning and transfer-learning methods in NLP increase the accuracy of sentiment analysis based on financial headlines. Moreover, our results indicate that lexicon-based approaches can be efficiently replaced by modern NLP transformers.

The rest of the paper is organized as follows. Section II provides an overview of NLP methods for text representation: lexicon-based and statistical, as well as word and sentence encoders. Section III presents NLP transformers, their architectures and objectives, as a separate group of deep-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

learning models for text classification, which we evaluate in extraction of finance text sentiments. Section IV describes the dataset that we created to evaluate text representation methods. Section V presents the evaluation platform that we build for measuring model performances. Section VI reports the results, and section VII concludes the paper and considers future applications.

II. TEXT REPRESENTATION METHODS

A. LEXICON-BASED KNOWLEDGE EXTRACTION

Lexicon-based sentiment analysis methods rely on domain-specific knowledge represented as a lexicon or dictionary. The process of sentiment calculation is based on identifying and keeping words that hold useful information while removing words that are not related to sentiments in finance.

Commonly used lexicons and dictionaries in finance are General Inquirer (GI), Harvard IV-4 (HIV4) [39], Diction [40] [41], and Loughran and McDonald's (LM) [12] word lists.

To infer the sentiment, we evaluate the Loughran-McDonald lexicon (a financial lexical rule-based tool) and the general-purpose Harvard IV-4 dictionary (general sentiment dictionary). We calculate the sentiment polarity using the Lydia system [42]. Each of the words in the sentences is categorized into either a positive or a negative group based on its sentiment in the lexicon (Eq. 1). If polarity>0, then the sentence is classified as positive, and if polarity<0, then the sentence is classified as negative.

$$Polarity = \frac{Pos - Neg}{Pos + Neg} \tag{1}$$

When using machine-learning (ML) and deep-learning (DL) classifiers, we extract the headline features by replacing the words in the sentence with the sentiment value, specified in the dictionary. Next, we input the newly generated sequence into the neural network to classify the text. The DL's output soft-max layer calculates the probability that the sequence belongs to either the positive or negative sentiment labels.

B. STATISTICAL METHODS

1) Count vectors

Count Vectorizer (CV) is a simple statistical approach to text representation which converts a collection of text documents into a matrix of token counts, thus reducing the entire sentence into a single vector. The positions in the vector represent the number of appearances of each word in the sentence. The CV algorithm performs feature extraction by using a vocabulary of words (tokens) which can be built from the same text corpus, or input manually (a-priori) from an external resource. The vocabulary limits the number of features which can be extracted from the text.

The CV approach for text representation has some drawbacks. First, the ordering information gets lost due to the methodology for term "squeezing." Second, the contextual information of the sentence is hidden, although it is crucial for sentiment extraction. These issues can be partially solved by using n-gram vectorizers where two, three or more consecutive words are put together in order to form tokens. Another issue with CV is that it shadows the important words that hold decision-making features for classifiers, because it pays more attention to general, frequent words such as "like," "but," and," "or," which do not add meaningful information. As a result, important text features may vanish, which calls for more sophisticated algorithms.

2) TF-IDF term weighting

TF-IDF (Term frequency - inverse document frequency) is an algorithm for statistical measurement, which evaluates the relevance of a word in a document within a corpus of documents. It addresses the feature-vanishing issue of CV algorithms by re-weighting the count frequencies of the words (tokens) in the sentence according to the number of appearances of each token. The algorithm works by multiplying two metrics: term frequency (TF), which calculates the number of occurrences of a term in the sequence (Eq.3), and inverse document frequency (IDF), which penalizes the feature count of the term if it appears in more sentences within the corpus (Eq.4),

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{2}$$

$$tf(t, d) = log(1 + freq(t, d)) \tag{3}$$

$$idf(t, D) = log(\frac{N}{count(d \in D : t \in d)}) \tag{4}$$

where t denotes the term, d denotes the document, D denotes the corpus of documents and N is the total number of documents.

In this study, we assess the feature extraction performance of the uni-gram and 2-gram count vectorizers as well as the TF-IDF term weighting in combination with machine-learning classifiers and deep-neural networks.

C. WORD ENCODERS

Statistical features do not provide semantics of the contextually close words, which means that words with similar meaning will not have similar codes. Many NLP tasks such as sentiment analysis, question-answering and text generation require detailed semantic knowledge that is not provided by CV and TF-IDF. To overcome these challenges, researchers have introduced word encoders [43] to convert discrete words into high-dimensional vectors composed of real numbers, using a procedure called word embedding. Word encoders help with understanding the context of the sentences, which improves the extracted features. These models are based on the principle of distributional hypothesis [44], in which the meaning of words is evidenced by the context. This approach establishes a new area of research in NLP called distributional semantics, which is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

core of many contemporary NLP techniques, including word encoders. These methods are called distributional semantic models (DSM), also known in the literature as vector space or semantic space models of meaning [45]–[47].

The word encoders classify the words that appear in the same context as semantically similar to one another, hence assigning similar vectors to them. This retained semantic information is very useful for classifiers or neural networks.

In this section, we provide an overview of the most popular word encoders: Word2Vec [43], GloVe [28] and FastText [48] [29], which exemplify different approaches in modeling word embeddings.

1) Word2Vec

In 2013, a team of researchers at Google, led by Tomas Mikolov, introduced the breakthrough model for word representation called Word2Vec [43] [27], which marked the beginning of a spectacular evolution in NLP. Mikolov and his collaborators proposed two model architectures for computing continuous vector representations of words by using the unsupervised approach: Continuous Bag-of-Words (CBOW) and Continuous Skip-gram Model (Fig. 1). The CBOW architecture predicts the current word based on the context, while in the Skip-Gram architecture, the distributed representation of the input word is used to predict the context [43]. The authors show the effectiveness of the proposed methodology experimentally, using several NLP applications, including sentiment analysis. Additionally, they demonstrate that the Skip-gram architecture gives more accurate results for large datasets because it generates more general contexts.

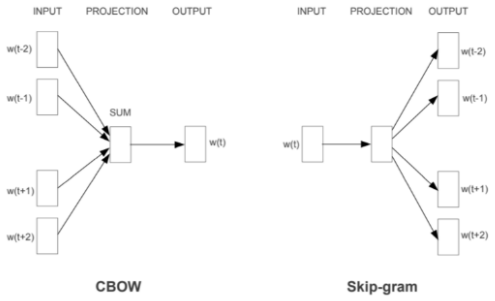

FIGURE 1. Word2Vec CBOW and Skip-Grams architectures [43]

The main drawback of Word2Vec is its inability to handle unknown or out-of-vocabulary (OOV) words. If the model has not encountered a word before, it will be unable to interpret it or build a vector for it. Additionally, Word2Vec does not support shared representations at sub-word level, which means that it will create two completely different vector representations for words which are morphologically similar, like agree/agreement or worth/worthwhile [29].

In our analysis, we use a pre-trained version of Word2Vec on the Google News corpus, which contains almost 3 million English words represented by 300-dimensional vectors.

2) GloVe

In 2014, a team of researchers at Stanford University proposed GloVe, an improved methodology for word encoding, based on a solid mathematical approach [28]. GloVe overcomes the drawbacks of Word2Vec in the training phase, improving the generated embeddings. It emphasizes the importance of considering the co-occurrence probabilities between the words rather than single word occurrence probabilities themselves. The model combines two classes of methods for distributed word representations: global matrix factorization and Skip-grams are used to extract better features by examining the relationships between words. The global matrix factorization method can capture the overall statistics and relationships between words. On the other hand, Word2Vec Skip-gram's method is efficient in extracting the local context and capturing the word analogy. Both methods are successfully incorporated into the GloVe encoder, thus outperforming Word2Vec in many NLP tasks. GloVe is widely used as a word encoder for NLP-based sentiment analysis [49]–[51].

3) FastText

In 2016, the Facebook research laboratory introduced a novel method for word encoding called FastText, which tackles the generalization problem of unknown words [48] [29]. FastText differs from previous models in its ability to build word embeddings at a deeper level by harnessing sub-words and characters. In this method, words become a context and word embedding is calculated based on combinations of lower-level embeddings. Each word is represented as a bag of character n-grams. For example, the word "finance," given n=3, will be represented by the following character n-grams: $< fi, fin, ina, nan, anc, nce, ce >$. The main algorithm behind FastText is Word2Vec. Learning the sub-word information enables training of embeddings on smaller datasets and generalization to unknown words. FastText shows improved results in text classification [52], even in structurally rich languages such as Turkish [53] and Arabic [54], which require morphological analysis instead of assigning a distinct vector to each word.

We evaluate pre-trained FastText vectors in order to assess their performance on financial texts. We use the wiki-news-300d-1M pre-trained model, which wraps 1 million word vectors trained on Wikipedia's 2017 corpus and the statmt.org[2] news dataset, where each embedding consists of 300 dimensions.

4) ELMo

In 2018, a team of researchers at Allen Institute for Artificial Intelligence developed an advanced word encoder called ELMo (Embeddings from Language Models) [55], whose word embeddings are learned from a deep bidirectional

[2]http://statmt.org/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

language model (biLM), pre-trained on large corpora of textual data. The essential feature, which makes ELMo different from previous word encoders, is that it produces contextual word embeddings considering the whole context in which the word is used. Hence, we can obtain different embedding for the same word in a different context, a major improvement from previous encoders, which always produce a static embedding. To tackle out-of-vocabulary (OOV) tokens, ELMo uses character-derived embedding, leveraging the morphological clues of words, thus improving the quality of word representations.

D. SENTENCE ENCODERS

In 2014, the idea of encoding entire sentences surpassed word encoding. The primary purpose of sentence encoders is to learn fixed-length feature vectors that encode the syntax and semantic properties of variable-length sentences. While a simple sentence embedding model can be built by averaging the individual word embeddings for every word of the sentence, this approach loses the inherent context and sequence of words as valuable information that should be retained in many tasks.

The main weakness of using sentence encoders to handle variable-length text input is related to the fixed size of the produced vectors. Long and short sentences are treated equally, producing the same number of extracted features, thus diluting the embeddings.

In this section, we outline recent and most prevalent sentence encoders [2], [30]–[33], to assess their ability to extract important features in sentence representation of financial headlines.

1) Doc2Vec

In 2014, the first successful sentence encoder, Doc2Vec [30] introduced an approach for representing variable-length fragments of texts (sentences, paragraphs, and documents) as fixed-size dense vectors, a.k.a. paragraph vectors. These vectors are trained to predict words in documents. Their primary goal is to make an appropriate distributed representation of large texts, overcoming the weaknesses of bag-of-words methods. Paragraph vectors combine word vectors to build phrase-level or sentence-level representations. They epitomize a distributed memory model, holding the context of the paragraph and contributing to the prediction task of the next word in combination with word vectors. Additionally, paragraph vectors can be used as features for the paragraph, which can be fed as input to a classifier or to a neural network, making them appropriate for evaluation of sentiment analysis in financial headlines. To obtain sentence embeddings, we use a Doc2Vec approach, which is pre-trained on English Wikipedia texts.

2) Skip-Thought Vectors

Skip-Thought Vectors [31] are models that use encoder-decoder architecture for sequence modeling based on unsupervised learning. These models use continuity of texts, extracted from books, to train an encoder-decoder method. The model tries to reconstruct the surrounding sentences of an encoded passage in order to remap their syntactic and semantic meaning into similar vector representations. The encoder generates a sentence vector, and the decoder is used to generate the surrounding sentences. The model uses a Recurrent Neural Networks (RNN) encoder with Gated Recurrent Unit (GRU) [56] activations, and an RNN decoder uses a conditional GRU. The use of the attention layer provides for a dynamic change of the source sentence representation. Depending on the encoder type, two separate models are trained: uni-skip and bi-skip. Uni-skip passes sentences in the correct order and extracts 2400 features. The Bi-skip model uses two encoders. One of them passes the sentence in the correct order and the other passes the sentence in reverse order, extracting a total of 2400 features. Due to their generative nature, Skip-Thought vectors are appropriate and effective for neural machine translation and classification tasks. The main shortcoming of this approach is the arduous task assigned to the decoder [57], as the next sentence prediction requires modeling aspects that are, in most cases, irrelevant to the meaning of the sentence.

3) InferSent

InferSent [32] is a supervised approach to learning sentence embeddings using natural language inference (NLI) data. NLI captures universally useful features, thus learning universal sentence embeddings in a supervised manner. The training dataset used by this model is the Stanford Natural Language Inference (SNLI) dataset that contains 570k human-generated English sentence pairs, manually annotated with one of the three labels: entailment, contradiction, or neutral.

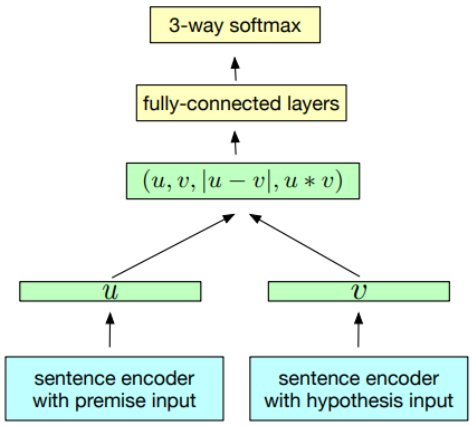

FIGURE 2. InferSent training scheme [32]

Fig. 2 shows a shared encoder used for encoding the premise u and the hypothesis v. In order to extract relations between u and v, three matching methods are applied: concatenation (u, v), element-wise product $u * v$ and absolute element-wise difference $|u - v|$. Next, the resulting

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

feature vector is applied as input to the 3-class classifier to evaluate the relationship between u and v based on the extracted features. Experimentally, the best architecture for the encoder is shown to be the BiLSTM network with max pooling. This approach outperforms Skip-Thought vectors in many NLP tasks.

In our study, we assess the performances of two publicly available versions of InferSent. The first version is trained with Stanford's GloVe as word encoder and the second is trained with Facebook's FastText.

4) Universal Sentence Encoder

In March 2018, Google researchers published their first version of a model which converts variable-length sentences into 512-dimensional vectors, called Universal Sentence Encoder (USE) [2]. The model is able to embed not only sentences, but also words and entire paragraphs. USE uses the concept of transfer-learning to leverage the knowledge extracted from large datasets to improve the results when limited training data is available.

We evaluate the USE encoder, which is based on Deep Averaging Network (DAN) architecture as shown in Fig.3. Input embeddings for words and bi-grams are first averaged and then passed through a feed-forward deep neural network (DNN) to produce sentence embeddings. The computational time is linear in the length of the input sentence.

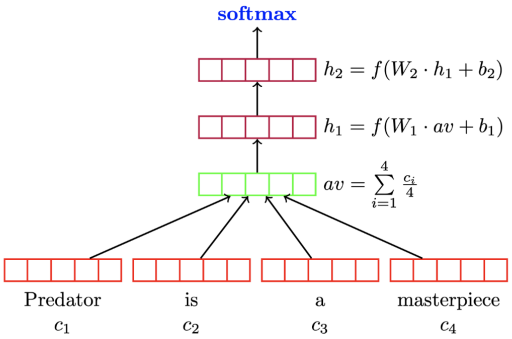

FIGURE 3. USE based on DAN architecture [2]

USE models are trained on a variety of data sources: Wikipedia, news, question-answer pages, and discussion forums. These models are based on transfer-learning experiments with several datasets to evaluate the efficiency of the encoder. The results show that sentence encoders outperform transfer-learning methodologies that use word-level embeddings alone.

The main issues with USE (DAN model) are related to the use of averaging techniques that cannot recognize negation phrases like "not good." This refers to using contextualized embeddings, which considers the influence of other words in producing sentence embedding.

In our analysis, we assess the two latest versions of USE (4 and 5) that can be found at the TensorFlow Hub repository[3].

5) Language-Agnostic Sentence Representations (LASER)

In 2019, Facebook researchers [33] introduced an architecture for universal language-agnostic multilingual sentence representations (LASER) for 93 languages by using a single BiLSTM encoder with a shared Byte Pair Encoding (BPE) vocabulary for different languages. The main contribution of the LASER methodology is that it provides a framework for zero-shot transfer-learning. LASER leverages one model, trained on one language, to be used in another language without the need for pre-training. This is accomplished by LASER's ability to bring semantically similar sentences, written in different languages, close to each other in the embedding space.

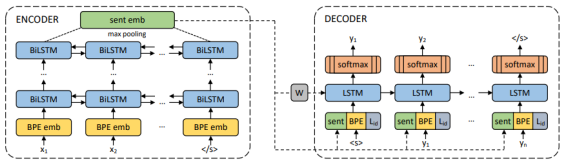

FIGURE 4. LASER architecture [33]

Sentence embeddings are obtained by applying a max-pooling operation to the output of the BiLSTM encoder. The same encoder is used for all 93 languages. The byte-pair encoding (BPE) vocabulary is learned based on the concatenation of all training corpora, hence, it does not require specific information about the input language. LASER's encoder architecture, illustrated in Fig. 4, is shown to be efficient even for low-resource languages.

In this study, we evaluate LASER on English texts, though the same model that we build here can be used for sentiment analysis in texts written in the other 92 languages supported by LASER.

III. NLP TRANSFORMERS

The pre-trained word and sentence embeddings show good performance for NLP tasks due to their ability to retain the semantics and the syntax of the words in the sentence. The transfer-learning task, in this case, allows for the information that has been learned from unlabeled data to be used in tasks with relatively small labeled data to achieve higher accuracy. Although such embeddings have proven to be powerful, they lack context-based mutability. Word2Vec, GloVe, and FastText use fixed embeddings for each of the words, thus producing one-to-one mapping, which in many cases is not appropriate and requires additional attention. Recent research studies have proposed methods that produce different embeddings for the same word, taking into consideration specific contexts [3], [55], [58]. As an illustration of context importance, we analyze the following two sentences that contain the word "Apple": *"Apple Inc*

[3]https://tfhub.dev/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

performed well this year." and *"Apple fruits are exported to various countries."* In the first sentence, Apple refers to the technology company Apple, headquartered in the US, while in the second sentence, apple refers to the fruit, with a completely different meaning. The encoders, however, will produce the same encoding for both words regardless of the contexts. This problem highlights the need for contextualized embeddings for the word *"Apple."*

A. NLP TRANSFORMER ARCHITECTURE

A transformer represents an architecture that transforms one sequence into another by using two models: encoder and decoder. Unlike previously described standard sequence-to-sequence models, which are based on LSTM/GRU units, the paper "Attention is All You Need" [59] introduces a novel, breakthrough transformer architecture based solely on multi-headed self-attention mechanisms. There are three reasons for choosing self-attention instead of recurrent layer: computational complexity, parallelization, and learning long-range dependencies between words in the sequence, all of which are crucial for building contextualized embeddings. By using this approach, transformers have shown improved results in machine translation and other related tasks.

This method uses positional embedding to remember the order of words in the sequence. The main building blocks in the encoder/decoder modules are Multi-Head Attention and Feed Forward layers, as shown in the Attention-based transformer architecture (Fig.5).

The scaled dot-product attention mechanism is described by equations 5 and 6.

$$a = softmax(\frac{QK^T}{\sqrt{n}}) \tag{5}$$

In Eq.5, the attention weights a represent the influence of each word in the sequence (Q) by all the other words (K) in the same sequence. Q is a matrix that contains the query (vector representation of one word in the sequence), K are all keys (all vector representations of all the words in the sequence) and n is dimensionality of the query/key vectors. The softmax function is used to ensure that weights a have a distribution between 0 and 1. Considering a, a self-attention is calculated by using Eq.6, which represents a weighted sum of values (V), where V is the vector obtained from the encoder.

$$Attention(Q, K, V) = aV \tag{6}$$

A multi-head attention mechanism calculates the scaled dot-product attention multiple times in parallel. The independent outputs are concatenated and linearly transformed into expected dimensions. Multi-head attention is obtained by using Eq. 7:

$$MultiHead(Q, K, V) = [head_1, head_2, ...head_h]W^O \tag{7}$$

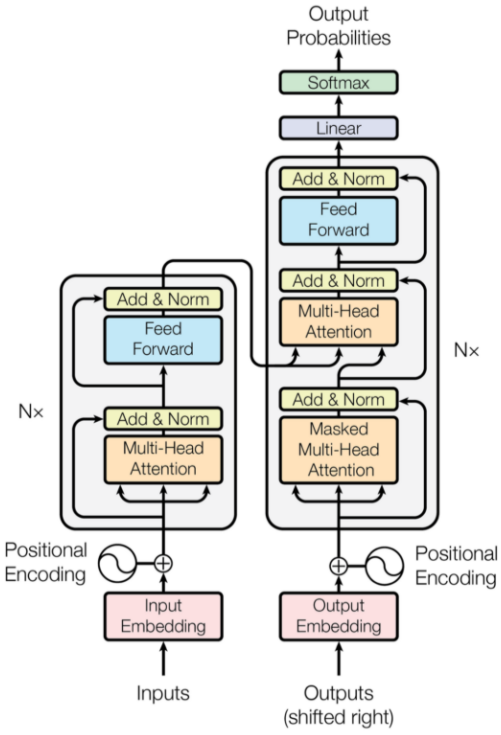

FIGURE 5. The Transformer architecture [59]

Each of the $head_i$ can be calculated by Eq. 8:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_V^i) \tag{8}$$

where W_i^Q, W_i^K, W_V^i and W^O are parameter matrices, which the model needs to learn. Multi-head attentions have an important role in obtaining the contextual embeddings when using NLP transformers.

A pre-training phase is an unsupervised learning approach where an unlabeled text corpus is introduced into the transformer architecture to produce text representations based on an objective function used by the transformer. This is a relatively expensive task, but the learned token or generic sentence representations can be used in many other tasks using transfer-learning. Later, the representation can be fine-tuned in order to recognize the specifics of the task and to achieve better results. Fine-tuning is performed by adding an additional dense layer after the last hidden state, recommended for using transformers in classification and regression tasks [3]. The transformer performs supervised learning (fine-tuning) on the labeled sentiment dataset, which is relatively inexpensive compared to pre-training.

NLP transformers are applicable to many different text classification problems, such as binary sentiment classification, which we use in our analysis.

1) BERT

In 2018, Devlin et al. [3] leveraged the transformer architecture to introduce a revolutionary language representation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

model, called BERT (Bidirectional Encoder Representations from Transformers). This model started the new era in NLP, with state-of-the-art performance achieved on most NLP tasks. BERT leverages the unsupervised learning approach to pre-train deep bidirectional representations from large unlabeled text corpora by using two new pre-training objectives — masked language model (MLM) and next sentence prediction (NSP). BERT overcomes the limitation of previous language models, which incorporate only unidirectional representations of words in sentences. It builds a bidirectional masked language model, which predicts randomly masked words in the sentence, enriching the contextual information of the words.

BERT is based on conventional, auto-regressive (AR) language modeling. The process of pre-training is performed by maximizing the likelihood between the tokens x in a text sequence $\mathrm{x} = [x_1, ..., x_T]$. Let $\hat{\mathrm{x}}$ denote the same text sentence with masked tokens and $\overline{\mathrm{x}}$ be an array of masked tokens. The training objective for BERT is to reconstruct $\overline{x}$ from $\hat{x}$ by Eq.9:

$$\max_{\theta} \quad \log p_\theta(\overline{\mathrm{x}}|\hat{\mathrm{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(\mathrm{x_t}|\hat{x}) =$$

$$\sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{x})_t^T e(x_t)\right)}{\sum_{x'} \exp(H_\theta(\hat{x})_t^T e(x'))} \quad (9)$$

where,

- $e(x')$ denotes the embedding of the token x;
- $m_t = 1$, if x_t token of the text sequence x is masked;
- H_θ is a Transformer which transforms each token of text sequence into a hidden vector.

BERT assumes that all masked tokens $\overline{x}$ are mutually independent, which is the main rationale behind the approximation of the joint conditional probability $p(\overline{x}, \hat{x})$ in Eq.9. Another advantage that differentiates BERT from previous AR methods is the ability to increase the context information $H_\theta(x)_t$ by accessing the tokens placed on the left and the right side of token t.

BERT has two versions: BERT-base, with 12 encoder layers, hidden size of 768, 12 multi-head attention heads and 110M parameters in total; and BERT-large, with 24 encoder layers, hidden size of 1024, 16 multi-head attention heads and 340M parameters. Both of these models have been trained on English Wikipedia and BookCorpus [60].

2) FinBERT

FinBERT [61] is a version of BERT intended for the finance domain. It is pre-trained on a financial text corpus which consists of 1.8M news articles from Reuters TRC2 dataset, published between 2008 and 2010. Compared to other pre-trained versions of BERT, FinBERT model has achieved a 15% improvement in accuracy in text classification tasks specifically applied to financial texts.

3) XLNet

The XLNet model, developed by Google Brain and Carnegie Mellon University, addresses the disadvantages of BERT, improves its architectural design for pre-training, and produces results that outperform BERT in 20 different tasks. It utilizes a generalized AR model where the next token is dependent on all previous tokens, thus avoiding corrupted input caused by masking of the words, performed by BERT. The limitations of BERT include neglecting the dependency between masked tokens as it assumes that they are mutually independent variables. On the other hand, XLNet considers these tokens in the process of context building and assumes that masked words are mutually dependent.

Additionally, XLNet uses Permutation Language Modeling (PLM) to capture bidirectional context by maximizing the expected log-likelihood of a sequence given all possible permutations of words in a sentence. This means that XLNet enriches the contextual information of each position by leveraging the tokens from all the other positions found on the left and on the right sides of the token. Specifically, for a sequence x of length T, there are $T!$ different orders on which the algorithm performs auto-regressive factorizations.

Let $\mathcal{Z}_T$ be the set of permutations of the words in a sentence of length T. $\mathrm{x}_{\mathrm{z}_{<\mathrm{t}}}$ denotes the first $t-1$ elements of the permutation $\mathrm{z} \in \mathcal{Z}_T$. The PLM objective is given in Eq. 10.

$$\max_{\theta} \quad \mathbf{E}_{\mathrm{z}\sim\mathcal{Z}} \sum_{t=c+1}^{|z|} \log p_\theta(x_{z_t}|\mathrm{x}_{\mathrm{z}_{<\mathrm{t}}}) \quad (10)$$

The hyperparameter c can be derived from the hyperparameter K, where $c = |z|(K-1)/K$, and it represents the cutting-point of the division of vector z into non-target $\mathrm{z}_{\leq \mathrm{c}}$ and target $\mathrm{z}_{>\mathrm{c}}$ subsequences.

As shown in Eq.9 and Eq.10, both BERT and XLNet perform partial prediction, due to optimization. The main difference lies in the choice of tokens used for context modeling. BERT predicts the masked tokens, assuming that targets are mutually independent, while XLNet predicts the last token in a factorization order $\mathrm{z}_{>c}$.

The following example $[Wells, Fargo, is, a, bank, in, USA]$ explains the difference. Assume that our goal is to predict "Wells Fargo." In order to use $[Wells, Fargo]$ as prediction targets, BERT masks them, and XLNet samples the factorization order [is,a,bank,in,USA,Wells,Fargo]. Using Eq. 9, BERT will compute:

$$\mathcal{J}_{\mathrm{BERT}} = \log p(\mathrm{Wells} \mid \mathrm{is, a, bank, in, USA}) + \log p(\mathrm{FARGO} \mid \mathrm{is, a, bank, in, USA}) \quad (11)$$

Using Eq. 10, XLNet will compute:

$$\mathcal{J}_{\mathrm{XLNet}} = \log p(\mathrm{Wells} \mid \mathrm{is, a, bank, in, USA}) +$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

$$\log p(\text{FARGO} \mid \text{Wells}, \text{is}, \text{a}, \text{bank}, \text{in}, \text{USA}) \quad (12)$$

These examples show that both BERT and XLNet compute the objective differently. XLNet captures important dependencies between prediction targets, such as (Wells, Fargo), which BERT omits. Hence, XLNet combines the advantages of AR and auto-encoding methods by using a generalized AR pre-training approach with a permutation language modeling objective, in order to improve the results in NLP.

4) XLM

The Cross-lingual Language Model (XLM) [62] has a transformer architecture that is mainly used for modeling cross-lingual features. XLM is pre-trained using several objectives:

- Causal Language Modeling (CLM) - next token prediction.
- Masked Language Modeling (MLM) - approach similar to BERT's objective for masking random tokens in the sentence.
- Translation Language Modeling (TLM) - supervised approach, which harnesses parallel streams of textual data written in different languages in order to improve cross-lingual pre-training support.

In our analysis, we use XLM for text classification tasks to perform sentiment analysis of texts in English. We explore bi-directional context of the tokens in sentences to perform Masked Language Modeling (MLM), which is the best approach for our evaluation task.

5) ALBERT

To overcome the shortcomings of using large pre-training natural language representations such as GPU/TPU, memory limitations, and longer training times, in 2019 Google Research and Toyota Technological Institute jointly released a new model that introduces BERT's smaller and more scalable successor, called ALBERT [63]. ALBERT is based on two-parameter reduction methods: cross-layer parameter sharing and sentence ordering objectives, in order to lower memory consumption and increase the training speed of BERT. ALBERT outperforms BERT in several tasks, including text classification [64]. ALBERT uses a significantly reduced number of parameters in sentiment analysis, compared to BERT and XLNet.

6) RoBERTa

The RoBERTa model, introduced by the Facebook research team in 2019 [4], offers an alternative optimized version of BERT. Retrained on a dataset ten times larger, with improved training methodology and different hyper-parameters, RoBERTa removes the Next Sentence Prediction (NSP) objective and adds dynamic masking of words during the training epochs. These changes and features show better performances compared to BERT in many NLP tasks, including text classification.

7) DistilBERT

DistilBERT, introduced in October 2019 [65], is based on a methodology that reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. The technique that produces a compression of the original model is known as knowledge distillation. The compact (student) model is trained to reproduce the full output distribution of the larger (teacher) model or ensemble of models. Rather than training with a cross-entropy over the hard-targets (one-hot encoding of the classes), the student obtains the knowledge based on a distillation loss over the soft-target probabilities of the teacher. The distillation loss L_{ce} is calculated by using the Eq. 13.

$$L_{ce} = \sum_i t_i * log(s_i) \quad (13)$$

where t_i and s_i are the estimated probabilities of the teacher and student respectively. This objective results in a richer training signal, since soft-target probabilities enforce stricter constraints compared to a single hard-target.

We assess the performances of three distilled versions (students) of the following transformers (teachers): BERT-base-cased, BERT-base-uncased, and RoBERTa-base.

8) XLM-RoBERTa

The XLM-RoBERTa (XLM-R) [66] model is a multilingual model trained on one hundred different languages by using 2.5TB of filtered CommonCrawl data and it is based on Facebook's RoBERTa model. XLM-R achieves solid performance gains for a wide range of cross-lingual transfer tasks, including text classification. Additionally, XLM-RoBERTa offers a possibility of multilingual modeling without decreasing per-language performance, which makes it more attractive for evaluation compared to other transformers.

XLM-R follows the XLM approach [62], trained with a Masked Language Modeling (MLM) objective with minor changes to the hyper-parameters of the original XLM model.

In our analysis, we evaluate the performance of two different pre-trained XLM-R models: $\text{XLM} - \text{R}_{base}$ and $\text{XLM} - \text{R}_{Large}$, which differ in the size of their parameters.

9) BART

In October 2019, the Facebook research team published a novel transformer called BART [67] with an architecture similar to both BERT [3] and GPT2 (Generative Pre-Training 2) [68]. BART outperforms other transformers in generation tasks such as text summarizing and question answering. BART leverages the advantages of the bidirectional encoder from BERT and the GPT AR decoder. The auto-regressive approach means that GPT considers left to right dependence of the words in a sentence, which makes it more appropriate for text-generation compared to BERT. BART's encoder and decoder are connected by cross-attention. Each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

decoder layer performs attention over the final hidden state of the encoder output. This mechanism enables the model to generate output that is closely connected to the original input.

The fine-tuned model concatenates the input sentence with the end of sequence (EOS) token and passes these components as input to the BART encoder and decoder. The representation of the EOS token is used to classify the sentiment expressed in the sentence. In this study, we fine-tune BART and adapt it to sentiment analysis in finance.

IV. DATASETS

We use publicly available datasets that have been labeled by financial experts to perform a reliable evaluation of the ML models in predicting sentiments of financial headlines. We perform binary classifications to designate each of the sentences as bullish (positive) or bearish (negative), as described in the following subsections.

A. FINANCIAL PHRASE BANK

The Financial Phrase-Bank dataset [69] consists of 4845 English sentences selected randomly from financial news found on the LexisNexis database. These sentences have been annotated by 16 experts with a background in finance and business. The annotators were asked to give labels according to how they think the information in the sentence might influence the mentioned company's stock price. The dataset also includes information regarding the agreement levels on sentences among annotators. All sentences are annotated with three labels: Positive, Negative, and Neutral. The distribution of sentiment labels is presented in Table 1.

B. SEMEVAL 2017 TASK 5

The second dataset used in this paper is provided by the SemEval-2017 task "Fine-Grained Sentiment Analysis on Financial Microblogs and News" [70]. The Financial News Statements and Headlines dataset consists of 2510 news headlines, gathered from different publicly available sources such as Yahoo Finance. Each headline (instance) is annotated by three independent financial experts, and a sentiment score, in the range between -1 and 1, is assigned to each statement. A score of -1 means that the statement (message) is bearish or very negative, and a score of 1 means that the statement is bullish or very positive. We convert these sentiment scores into sentiment labels (bullish/bearish). The conversion process is performed by using Eq. 14.

$$L = \begin{cases} Bullish, & \text{if } score > 0 \\ Bearish, & \text{if } score < 0 \\ Neutral, & \text{if } score = 0 \end{cases} \tag{14}$$

After the conversion, the number of sentences per label is presented in Table 1.

The dataset used for evaluation is a combination of both datasets. To address the imbalance between positive and negative sentences, we perform a balancing by extracting

TABLE 1. Datasets statistics

Dataset/Category	Neutral	Positive	Negative
Financial Phrases	2879	1363	604
SemEval2017-Task5	38	653	451
Total	2917	2016	1055

1093 positive and another 1093 negative sentences, which we merge into one dataset. Additionally, we shuffle the datasets and we set aside stratified 80% of all sentences as a training and stratified 20% of the remaining sentences as a validation set. At the end, our balanced training set includes 1748 samples, and a balanced validation set consisting of 438 samples.

C. DATA PRE-PROCESSING

Financial headlines, similar to other real world text data, are likely to be inconsistent, incomplete and contain errors. Hence, to prepare the data, we perform initial pre-processing that includes tokenization, stop-word removal, and stemming. Additionally, we extract the named entities (organizations and people) from the headlines and replace them with their general nouns. For example, Microsoft is replaced with <CMPY>, or London with <CITY>.

We impose a min-max length of sentences to 3-64 words. After this initial filtering, we obtain the distributions of the number of words per sentence for the training set (Fig. 6) and for the validation set (Fig. 7).

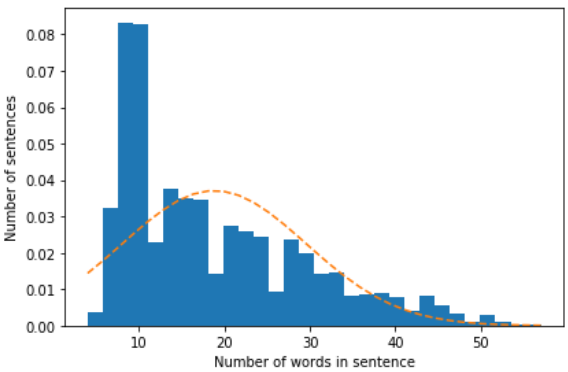

FIGURE 6. Distribution of number of words in training set

When evaluating lexicon-based and word encoders, we perform left padding to sentences in order to fix their size, due to their variable length. Considering the maximum size of the sentences given in Figs. 6 and 7, we pad them to 64 word length. When using sentence encoders, we do not pad the sequences due to the ability of the sentence encoders to encode sentences to fixed-size vectors.

V. SENTIMENT ANALYSIS PLATFORM

We evaluate the sentiment analysis methods by using the general platform, consisting of five phases shown in Fig. 8,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

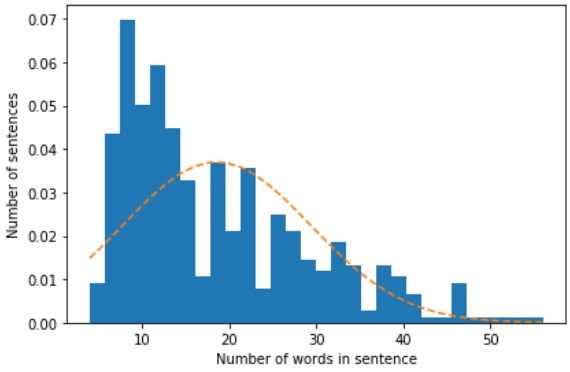

FIGURE 7. Distribution of number of words in validation set

as follows:

- In the first phase, we create our working dataset based on the Financial Phrase Bank and the SemEval 2017 dataset.
- In the second phase we apply data pre-processing functions as described in subsection IV-C.
- The third phase performs text encoding by using various text representation methods in order to extract features from the pre-processed texts. We evaluate the following text representation methods: domain lexicons, statistical models for feature extraction, word encoders, sentence encoders and NLP transformers.
- In the fourth phase, these embeddings are fed as input to various machine-learning or deep-learning classifiers, thus enabling us to evaluate many encoding-classifier combinations.
- In the fifth phase, we compare the real and predicted labels using several binary classification performance metrics.

The sentiment analysis platform is implemented in Python 3.6. The shallow models are developed using Tensorflow Keras [71] while the pre-trained versions of NLP transformers are retrieved from the Hugging Face repository [72]. The sentiment analysis modules are published at the GitHub repository[4].

In the following subsections, we present the details of machine-learning and deep-learning classifiers, fine-tuning of NLP transformers and evaluation metrics.

A. MACHINE-LEARNING CLASSIFIERS

In our evaluation analysis, we use two machine-learning classifiers: Support Vector Classifier (SVC), as a representative of Support Vector Machines (SVM), and an Extreme Gradient Boosting (XGB) [73], [74], as a representative of gradient-boosted decision trees. We chose the XGB model because it has achieved impressive results in many Kaggle competitions, in the structured data category. When using the ML classifiers, we perform a GridSearch approach for retrieving the best hyper-parameters.

[4]https://github.com/f-data/finSENT

B. DEEP-NEURAL NETWORKS (DNN)

Deep-learning methods [75] are achieving outstanding results in many fields, including: signal processing [76], computer vision [77], speech processing [78]–[80] and text classification [81].

The text representations and the features extracted from the evaluation methods are fed as input into Convolutional Neural Networks (CNN) [23] and Recurrent Neural Networks (RNN) [82] in order to proceed with the classification. While RNN networks work well in sequence modeling and capturing long-term dependencies, CNN networks are more efficient in capturing spatial or temporal correlations and in reducing data dimensionality.

In order to improve the architecture of previous DNN networks, novel mechanisms have been introduced. One of them is the Attention mechanism [83], which helps RNN networks focus on specific parts of the input sequence, facilitating the learning and improving the prediction. The Attention mechanism is widely used in encoder-decoder architectures due to its ability to highlight important parts of the contextual information.

Bidirectional RNN networks are often used to collect features from both directions. A forward RNN $\overrightarrow{h}$ gathers token features from the start (x_1) to the end (x_n), while the backward RNN $\overleftarrow{h}$ processes the tokens in reverse direction, from (x_n) to (x_1). The resulting hidden state h uses both sets of features concatenating $\overrightarrow{h}$ and $\overleftarrow{h}$ as shown in Eq.15:

$$h_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i} \tag{15}$$

where $\oplus$ denotes the concatenation function.

In our analysis, we used shallow RNN and CNN networks in order to evaluate the features from text representations. These shallow neural networks consist of three main layers: the input (embedding) layer, the hidden layer, and the output layer. The input layer uses text representation methods (lexicons/word or sentence encoders) to extract the feature vectors from the headlines. It then gives the vector as an input to the recurrent or convolutional hidden layer to extract complex features from the text representation methods. The output layer uses a softmax activation function to make the final classification. We then add an attention layer after the hidden layer to evaluate its effectiveness. Furthermore, we build an additional group of GRU and LSTM networks, which support bidirectional feature extraction, to assess their performance in finance-based sentiment analysis as described in [84]. and we use binary cross-entropy loss function when training the models. The ADAM (Adaptive Learning Rate) optimization algorithm [85] is used to find optimal weights in the networks. We use a maximum of one hundred training epochs for all DL models. We impose early stopping when the validation loss does not diminish after ten

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

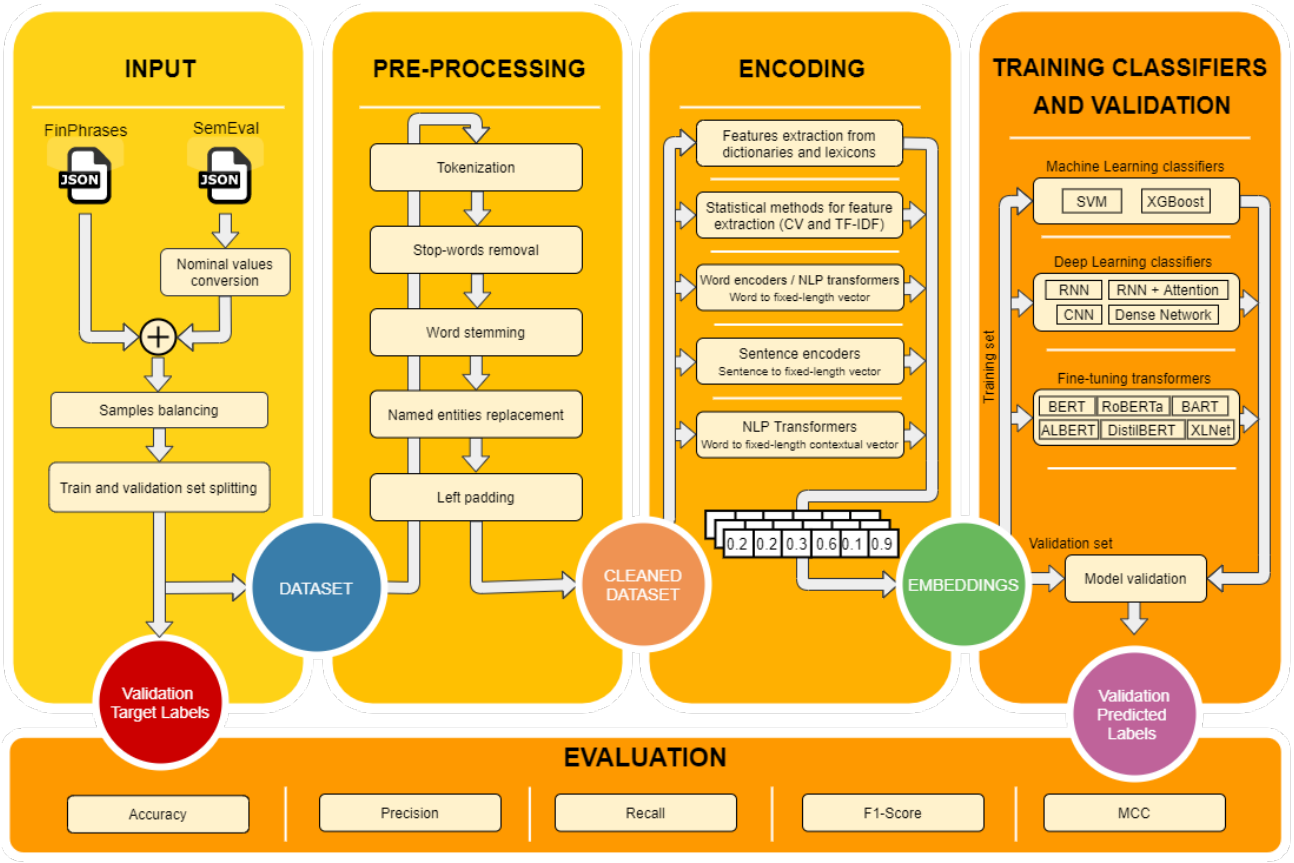

FIGURE 8. Sentiment analysis platform architecture

epochs to prevent over-fitting. Finally, we use dropout layer as regularization in the CNN network [86].

C. MODEL FINE-TUNING

To evaluate NLP transformers, we use pre-trained models from the Hugging Face's repository [72]. For finBERT, we use the language model trained on TRC2 dataset, published on the GitHub repository[5]. We fine-tune the transformers with the training dataset by adding only one dense layer after the last hidden state. The dense layer outputs the probabilities of sentence classification. Transformer's hyper-parameter settings during the fine-tuning phase are not model agnostic and they are directly related to the quality of the model.

D. EVALUATION METRICS

We evaluate the models for sentiment analysis of financial headlines, and present the results chronologically, based on the models' publication date. We first evaluate lexicon-based methods, using Harvard IV-4 and Loughran-McDonald dictionaries. Next, we evaluate word encoders as pioneers in modern NLP feature engineering approaches. Here, we use word encoders with shallow RNN architectures, described in Section V. Subsequently, we examine the performance of sentence encoders with a shallow dense layer and CNN architectures. Finally, we measure the efficiency of the latest NLP transformers, described in Section III.

As a main evaluation metric, we chose Matthews Correlation Coefficient (MCC) (16), where TP and TN are True Positive and True Negative samples accordingly, and FP and FN are the False Positive and False Negative number of samples which are misclassified.

$$MCC = \frac{tp * tn - fp * fn}{\sqrt{(tp + fp)(fn + tn)(fp + tn)(tp + fn)}} \quad (16)$$

MCC is widely used in assessing binary classification performance with a range between -1 (completely wrong binary classifier) and 1 (completely accurate binary classifier). It takes into consideration true and false positives and negatives, thus providing a balanced measure, which can be used even if the classes have different sample sizes.

VI. RESULTS AND DISCUSSION

In this section, we present the model evaluation results.

In Table 3, we report on the performance of the lexicon-based models by using hand-crafted feature engineering, based on the Loughran-McDonald (LM) financial and general Harvard IV-4 dictionaries. We perform the evaluations

[5]https://github.com/ProsusAI/finBERT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 2. Average performances of models grouped by text representation method

Model	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
HIV4	0.616	0.614	0.642	0.591	0.625	0.24
LM	0.636	0.605	0.781	0.491	0.68	0.288
CV	0.813	0.83	0.793	0.833	0.811	0.626
TF-IDF	0.815	0.82	0.807	0.823	0.813	0.631
Word2Vec	0.792	0.793	0.792	0.793	0.792	0.585
FastText	0.778	0.777	0.777	0.78	0.776	0.557
GloVe	0.796	0.785	0.816	0.775	0.8	0.593
ELMO	0.871	0.875	0.865	0.877	0.87	0.742
Doc2Vec	0.756	0.785	0.708	0.805	0.744	0.517
STV	0.755	0.77	0.725	0.785	0.746	0.513
InferSent	0.813	0.825	0.795	0.831	0.81	0.628
USE	0.825	0.841	0.801	0.848	0.82	0.65
LASER	0.826	0.846	0.796	0.855	0.82	0.652
BERT-base	0.898	0.913	0.881	0.916	0.897	0.797
BERT-large	0.928	0.928	0.929	0.927	0.928	0.854
xlnet-base-cased	0.913	0.91	0.918	0.909	0.914	0.827
xlnet-large-cased	0.922	0.922	0.922	0.922	0.922	0.845
xlm-mlm	0.932	0.932	0.932	0.932	0.932	0.863
finBERT	0.89	0.87	0.918	0.863	0.893	0.782
DistilBert-base	0.892	0.887	0.897	0.886	0.892	0.783
RoBERTa-BASE	0.941	0.949	0.932	0.95	0.94	0.881
RoBERTa-LARGE	0.941	0.941	0.941	0.941	0.941	0.882
distilroberta-base	0.927	0.931	0.922	0.932	0.927	0.854
ALBERT-base	0.887	0.893	0.879	0.895	0.886	0.774
ALBERT-large	0.913	0.913	0.913	0.913	0.913	0.845
ALBERT-xlarge	0.924	0.93	0.916	0.932	0.923	0.847
ALBERT-xxlarge	0.936	0.94	0.932	0.941	0.936	0.872
xlm-roberta-large	0.922	0.934	0.909	0.936	0.921	0.845
BART-large	0.947	0.95	0.945	0.95	0.947	0.895

by using the Lydia system polarity detection, machine-learning classifiers, and deep-learning models, as described in previous sections. As expected, the Loughran-McDonald features outperform the Harvard IV-4 general-purpose sentiment analysis dictionary. Hence, feature extraction with a domain-specific dictionary is a better approach for sentiment analysis tasks. The best performing model is the XGB classifier using LM features, achieving MCC=0.327. Additionally, we find that RNN networks outperform CNN and fully-connected dense networks. The improved results are due to the RNN networks' ability to remember sequential data, which is crucial for classification of sentences. Furthermore, the bidirectional context and attention layer improve the results when used in combination with RNN networks.

In Table 4, we present the results of the experiments performed on features extracted from statistical methods. We use ML classifiers and a deep neural network classifier based on fully connected dense layers. These methods show good results, achieving an MCC score of 0.667, almost twice as good as the lexicon-based methods.

In Table 5, we present the evaluation results of the word encoders. Generally, the best score is achieved when using Stanford's GloVe with Bidirectional GRU and attention layer (MCC=0.704). Here, the attention layer increases the MCC score by 0.04 compared to the BiGRU method without the attention layer (MCC=0.666). Additionally, the evaluated word encoders achieve better results when used with RNN networks, which further learn the context from the attention layer. In all tests, the GRU units outperform the LSTM units.

The features extracted from word encoders are significantly better compared to the features extracted by using lexicons and dictionaries. Furthermore, the word encoders perform better than statistical methods for feature extraction, which implies that incorporating semantic meaning into the word representation is useful for classification.

The results obtained from the evaluation of sentence encoders are presented in Table 6. InferSent, developed by Facebook, is the best performing sentence-based encoder. Its version 2 uses a simple architecture composed of fully connected dense layers which averages FastText word embeddings, thus outperforming Doc2Vec, Universal Sentence Encoder (USE), Skip-Thought-Vectors, and LASER. Additionally, InferSent outperforms the word encoder FastText, which implies that the InferSent's algorithm for averaging the word embeddings has superior efficiency for sentence context representation. Furthermore, we find that the FastText version of InferSent outperforms the GloVe version of InterSent. When using sentence vector representation, ML classifiers are more effective than CNN and a fully connected dense network.

In Table 7, we present the results of the first contextual word encoder, ELMo, which we evaluate in combination with ML classifiers (SVC, XGB) and DL classifier models (Dense, CNN and RNN). ELMo embeddings outperform the evaluated word encoders with fixed embeddings. This

confirms the hypothesis that contextual word vectors extract better features than the fixed ones. Additionally, concatenated vectors of words embeddings in combination with BiGRU network and an attention layer outperform the other ML and DL networks.

We also evaluate the popular NLP transformers which support text classification. We fine-tune them with training data in order to bias the embeddings towards financial sentiment analysis. All transformer architectures outperform word and sentence encoders, as shown in Table 8. Hence, contextualized embeddings perform semantic tasks better than their non-contextualized counterparts. Among the family of BERT transformers, BERT-Large-uncased achieves the best score in classification, with MCC=0.859. Although FinBERT was pre-trained on Reuters financial texts, it does not perform as well as the other pre-trained versions of BERT, which use Wikipedia and BookCorpus as text corpora for pre-training.

RoBERTa's dynamic masking increases the efficiency of the BERT algorithm by 0.023. DistilBERT retains more than 95% of the accuracy while having 40% fewer parameters. A distilled version of RoBERTa achieves as good results as BERT-large while using half the parameters of the teacher RoBERTa-base model. Among the ALBERT family of transformers, ALBERT-xxlarge pre-trained model outperforms the other ALBERT versions, obtaining MCC=0.881. Additionally, ALBERT outperforms the BERT model. The cross-language model (XLM) also outperforms BERT and XLNet. XLM-MLM-en-2048 achieves the best result, with MCC=0.863, among all XLM versions. Finally, the latest NLP transformer, Facebook's BART, outperforms all the other NLP transformers when applied to finance data, achieving the best MCC score of 0.895.

We show the performances of text representation approaches in Table 2, while the performance of each method chronologically is shown in Fig. 9.

VII. CONCLUSION

This paper presents a comprehensive chronological study of NLP-based methods for sentiment analysis in finance. The study begins with the lexicon-based approach, includes word and sentence encoders and concludes with recent NLP transformers. The NLP transformers show superior performances compared to the other evaluated approaches. The main progress in sentiment analysis accuracy is driven by the text representation methods, which feed the semantic meaning of the words and sentences into the models. The results achieved by the best models are comparable to expert's opinion. The evaluations were performed on a relatively small dataset of approximately 2000 sentences. Even though the dataset is not large, we obtained good results, suggesting that this approach is appropriate for domains where large annotated data is not available.

Distilled versions (Distilled-BERT and Distilled-RoBERTa) of NLP transformers achieve text classification performances comparable to their large, uncompressed teacher models. Hence, they can be effectively used in text classification production environments, where the need for light-weight, responsive, energy-efficient and cost-saving models is essential.

The results of this study can be applied in areas such as finance, where decision-making is based on sentiment extraction from massive textual datasets. The findings imply that selected models can be successfully used for forecasting stock market trends and corporate earnings, decision-making in securities trading and portfolio management, brand reputation management as well as fraud detection and regulation [87]–[89].

Although this approach was constructed for sentiment analysis in the finance domain, it can be extended to other areas such as healthcare, legal and business analytics.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

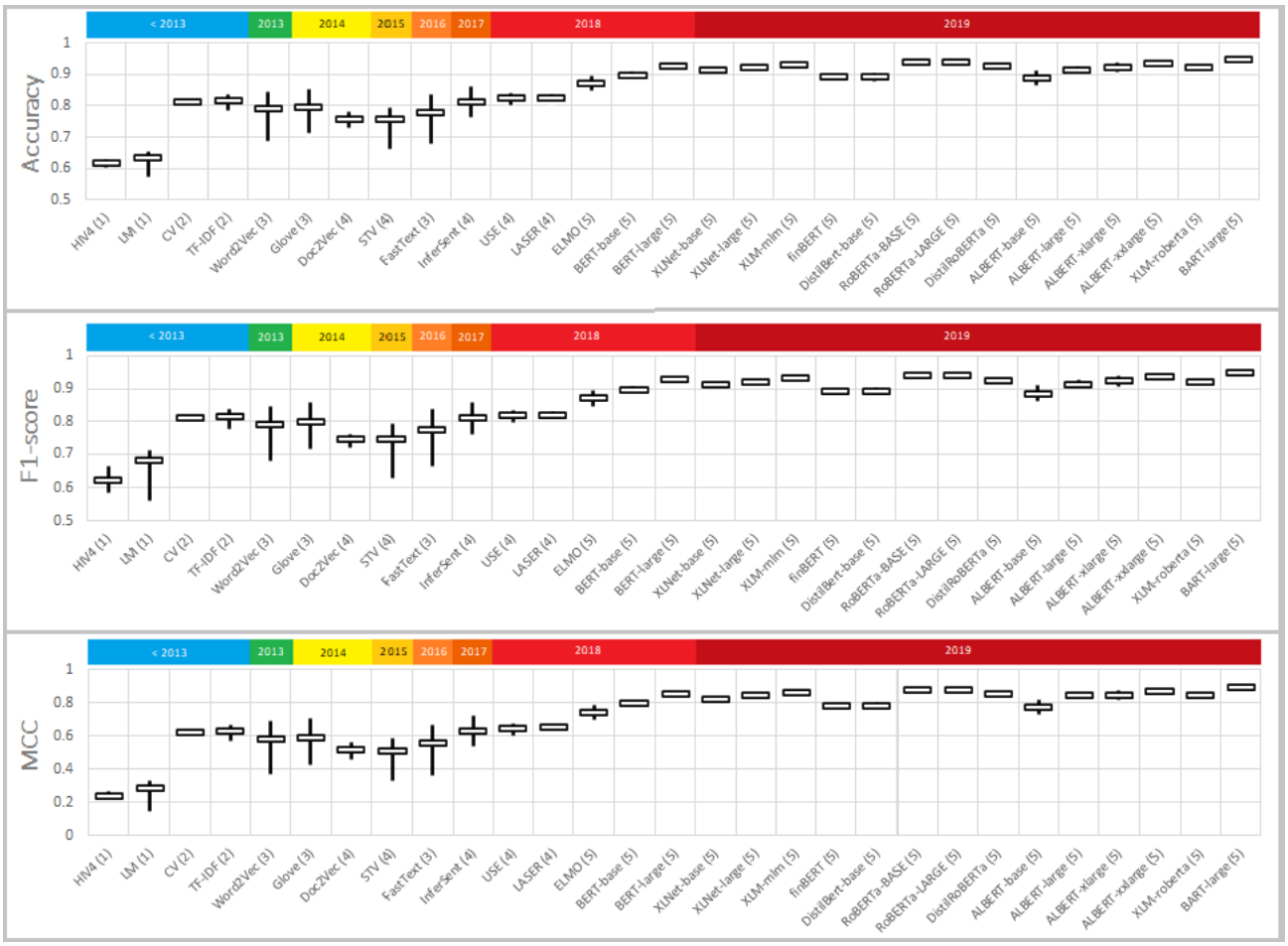

FIGURE 9. Sentiment analysis models' performances grouped by text representation method. On the chart, the top of the line for each model is the maximum, the bottom is the minimum and the white rectangle is the average performance per group. The numbers in the brackets represent the group of the text representation method, where (1) - lexicon-based methods, (2) - statistical methods, (3) - word encoders, (4) - sentence encoder, (5) NLP Transformer

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

.

APPENDIX A RESULTS

TABLE 3. Lexical Rule-based approach results

Model	Model Description	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
HIV4 + Lydia System	Harvard IV-4 features + Sentiment polarity	0.602	0.576	0.776	0.429	0.661	0.219
HIV4 + SVC	Harvard IV-4 features + SVC (kernel="linear", C=10)	0.618	0.632	0.566	0.671	0.597	0.238
HIV4 + XGB	Harvard IV-4 features + XGB classifier	0.614	0.589	0.753	0.474	0.661	0.237
HIV4 + Dense	Harvard IV-4 features + Dense network 32 units, 2,146 params	0.607	0.618	0.562	0.653	0.589	0.218
HIV4 + CNN	Harvard IV-4 features + CNN network 64 filters, 16 kernel size, 9,282 params	0.611	0.613	0.607	0.616	0.610	0.229
HIV4 + LSTM	Harvard IV-4 features + LSTM network 64 LSTM units, 25,090 params	0.625	0.626	0.621	0.630	0.624	0.258
HIV4 + LSTM + Attention	Harvard IV-4 features + LSTM network + Attention Layer 64 LSTM units, 34,242 params,	0.613	0.611	0.625	0.601	0.619	0.228
HIV4 + BiLSTM	Harvard IV-4 features + BiLSTM network 64 LSTM units, 50,178 params	0.627	0.628	0.625	0.628	0.627	0.256
HIV4 + BiLSTM + Attention	Harvard IV-4 features + BiLSTM network + Attention Layer 64 LSTM units, 34,242 params	0.613	0.616	0.607	0.619	0.612	0.228
HIV4 + BiGRU	Harvard IV-4 features + BiGRU network 64 GRU units, 41,730 params	0.622	0.612	0.676	0.569	0.642	0.256
HIV4 + BiGRU +Attention	Harvard IV-4 features + BiGRU network + Attention 64 GRU units, 25,794 params	0.623	0.629	0.637	0.607	0.633	**0.267**
LM + Lydia System	Loughran-McDonald features + Sentiment Polarity	0.573	0.576	0.547	0.598	0.562	0.146
LM + SVC	Loughran-McDonald features + SVC(kernel="linear",C=10)	0.648	0.606	0.845	0.452	0.706	0.322
LM + XGB	Loughran-McDonald features + XGB classifier	0.648	0.604	0.858	0.438	0.709	**0.327**
LM + Dense	Loughran-McDonald features + Dense network 32 units, 2,146 params	0.644	0.613	0.781	0.507	0.687	0.291
LM + CNN	Loughran-McDonald features + CNN network 64 filters, 16 kernel size, 9,282 params	0.640	0.610	0.785	0.495	0.687	0.292
LM + GRU	Loughran-McDonald features + GRU network 64 GRU units, 20,866 params	0.631	0.605	0.762	0.500	0.675	0.280
LM + GRU + Attention	Loughran-McDonald features + GRU network + Attention Layer 64 GRU units, 13,874 params	0.649	0.617	0.794	0.504	0.694	0.313
LM + BiGRU	Loughran-McDonald features + BiGRU network 64 GRU units, 41,730 params	0.643	0.611	0.794	0.493	0.690	0.302
LM + BiGRU + Attention	Loughran-McDonald features + BiGRU network + Attention Layer 64 GRU units, 25,794 params	0.652	0.618	0.799	0.505	0.697	0.316

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 4. Statistical methods results

Model	Model Description	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
CV + SVC	Count Vectorizer + SVC (kernel="rbf", C=100)	0.811	0.837	0.772	0.849	0.803	0.623
CV + XGB	Count Vectorizer + XGB classifier	0.811	0.831	0.798	0.824	0.814	0.622
CV + Dense	Count Vectorizer + Dense layer 64 units, 226 178 params	0.817	0.823	0.808	0.826	0.816	**0.635**
TF-IDF + SVC	TF-IDF Vectorizer + SVC (kernel="rbf", C=1)	0.833	0.823	0.849	0.817	0.836	**0.667**
TF-IDF + XGB	TF-IDF Vectorizer + XGB classifier	0.785	0.802	0.758	0.813	0.779	0.572
TF-IDF + Dense	TF-IDF Vectorizer + Dense network 64 units, 226,178 params	0.826	0.836	0.813	0.84	0.824	0.653

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 5. Fixed word embedding encoders results

Model	Model Description	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
Word2Vec (concat) + SVC	Word2Vec (concat) + SVC(kernel="rbf",C=50)	0.728	0.717	0.753	0.703	0.735	0.457
Word2Vec (concat) + XGB	Word2Vec (concat) + XGB classifier	0.687	0.693	0.671	0.703	0.682	0.375
Word2Vec (concat) + BiLSTM	Word2Vec(concat) + Bidirectional LSTM 100 LSTM units, 372,770 parameters	0.808	0.814	0.799	0.817	0.806	0.617
Word2Vec (concat) + BiGRU	Word2Vec (concat) + Bidirectional GRU + 100 GRU units, 292,570 parameters	0.821	0.816	0.831	0.812	0.823	0.643
Word2Vec (concat) + BiLSTM + Attention	Word2Vec (concat) + Bidirectional LSTM + Attention Layer 100 LSTM units, 373,034 parameters	0.824	0.829	0.817	0.831	0.822	0.648
Word2Vec (concat) + BiGRU + Attention	Word2Vec (concat) + Bidirectional GRU + Attention Layer 100 GRU units, 292,834 parameters	0.842	0.831	0.858	0.826	0.845	**0.685**
Word2Vec (concat) + CNN	Word2Vec (concat) + CNN network filters=100, kernel=4, 267,498 parameters	0.833	0.848	0.813	0.854	0.830	0.667
FastText (concat) + SVC	FastText (concat) + SVC (kernel="rbf",C=10)	0.772	0.774	0.767	0.776	0.771	0.543
FastText (concat) + XGB	FastText (concat) + XGB	0.772	0.774	0.767	0.776	0.771	0.543
FastText (mean) + SVC	FastText (vector averaging) + SVC (kernel="linear",C=50000)	0.795	0.797	0.79	0.799	0.794	0.589
FastText (mean) + XGB	FastText (vector averaging) + XGB	0.772	0.774	0.767	0.776	0.771	0.543
FastText (concat) + BiLSTM	FastText (concat) + Bidirectional LSTM 100 units (372,770 parameters)	0.772	0.774	0.767	0.776	0.771	0.543
FastText (concat) + BiGRU	FastText (concat) + Bidirectional GRU + 100 units, 292,570 parameters	0.829	0.805	0.867	0.790	0.835	0.660
FastText (concat) + BiLSTM + Attention	FastText (concat) + Bidirectional LSTM + Attention Layer 100 units, 373,034 parameters	0.820	0.802	0.849	0.789	0.825	0.640
FastText (concat) + BiGRU + Attention	FastText (concat) + Bidirectional GRU + 100 units, 292,834 parameters	0.833	0.817	0.858	0.808	0.837	**0.667**
FastText (concat) + CNN	FastText (concat) + CNN network filters=100, kernel=4, 267,498 parameters	0.787	0.794	0.776	0.799	0.785	0.575
GloVe (concat) + SVC	GloVe (concat) + SVC (kernel="rbf",C=200)	0.735	0.737	0.731	0.740	0.734	0.470
GloVe (concat) + XGB	GloVe (concat) + XGB	0.715	0.712	0.721	0.708	0.717	0.429
GloVe (mean) + SVC	GloVe (vector averaging) + SVC (kernel="linear",C=50)	0.813	0.816	0.808	0.817	0.812	0.626
GloVe (mean) + XGB	GloVe (vector averaging) + XGB	0.742	0.77	0.689	0.795	0.728	0.487
GloVe (concat) + LSTM	GloVe (concat) + LSTM 100 units (186,934 parameters)	0.801	0.764	0.872	0.730	0.814	0.608
GloVe (concat) + GRU	GloVe (concat) + GRU + 100 units, 146,834 parameters	0.811	0.830	0.780	0.840	0.804	0.622
GloVe (concat) + BiLSTM	GloVe (concat) + Bidirectional LSTM + Attention Layer 100 units,373,034 parameters	0.820	0.810	0.836	0.803	0.822	0.639
GloVe (concat) + BiGRU	GloVe (concat) + Bidirectional GRU + 100 units, 292,570 parameters	0.833	0.828	0.840	0.826	0.834	0.666
GloVe (concat) + BiLSTM + Attention	GloVe (concat) + Bidirectional LSTM + Attention Layer 100 units, 373,034 parameters	0.826	0.812	0.849	0.803	0.830	0.654
GloVe (concat) + BiGRU + Attention	GloVe (concat) + Bidirectional GRU + 100 units, 292,834 parameters	0.851	0.837	0.872	0.831	0.854	**0.704**
GloVe (concat) + CNN	GloVe (concat) + CNN network filters=100, kernel=4, 267,498 parameters	0.769	0.734	0.844	0.694	0.785	0.545

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 6. Sentence encoders

Model	Model Description	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
Doc2Vec + SVC	Doc2Vec features (enwiki_dbow) + SVC (kernel="rbf",C=1000)	0.776	0.827	0.698	0.853	0.757	**0.559**
Doc2Vec + XGB	Doc2Vec features (enwiki_dbow) + XGB classifier	0.730	0.748	0.694	0.767	0.720	0.462
Doc2Vec + Dense	Doc2Vec features (enwiki_dbow) + Dense network 128 units, 528 610 parameters	0.763	0.780	0.730	0.794	0.755	0.529
Skip-Thought-Vectors (Uni-Grams) + Dense	Skip-Thought-Vectors (Uni-Grams) + Dense network 128 units, 528 610 parameters	0.769	0.772	0.761	0.776	0.767	0.538
Skip-Thought-Vectors (Bi-Grams) + SVC	Skip-Thought-Vectors features with SVC classifier (kernel="linear",C=10)	0.764	0.771	0.753	0.776	0.762	0.529
Skip-Thought-Vectors (Bi-Grams) + XGB	Skip-Thought-Vectors features with XGB classifier	0.765	0.771	0.753	0.776	0.762	0.530
Skip-Thought-Vectors (Bi-Grams) + Dense	Skip-Thought-Vectors + Dense network 128 units, 528 610 parameters	0.788	0.818	0.740	0.836	0.777	**0.578**
InferSent (GloVe) + SVC	InferSent features + SVC (kernel="rbf",C=0.025)	0.820	0.837	0.795	0.845	0.815	**0.640**
InferSent (GloVe) + XGB classifier	InferSent features + XGB	0.820	0.827	0.808	0.831	0.818	0.639
InferSent (GloVe) + CNN	InferSent (GloVe) features CNN network 1024 filters, 20 kernel size, 54 370 parameters	0.767	0.783	0.740	0.794	0.761	0.537
InferSent (GloVe) + Dense	InferSent GloVe features + Dense network 128 units, 528 610 parameters	0.817	0.831	0.793	0.840	0.811	0.631
InferSent (FastText) + SVC	InferSent FastText features + SVC (kernel="rbf",C=200)	0.858	0.879	0.831	0.886	0.854	**0.718**
InferSent (FastText) + XGB	InferSent (FastText) features + XGB classifier	0.847	0.862	0.826	0.868	0.844	0.695
InferSent (FastText) + CNN	InferSent (FastText) features 1024 filters, 20 kernel size, 54 370 parameters	0.769	0.773	0.763	0.776	0.768	0.543
InferSent (FastText) + Dense	InferSent (FastText) features + Dense network 128 units, 528 610 parameters	0.829	0.837	0.818	0.84	0.828	0.665
USE 4 (DAN) + SVC	USE 4 (DAN) + SVC (kernel="rbf",C=1000)	0.815	0.845	0.772	0.858	0.807	0.633
USE 4 (DAN) + XGB	USE 4 (DAN) + XGB	0.801	0.814	0.781	0.822	0.797	0.603
USE 4 (DAN) + Dense	USE 4 (DAN) + Dense network, 397 538 parameters	0.829	0.839	0.812	0.844	0.825	0.659
USE 5 (DAN) + SVC	USE 5 (DAN) + SVC (kernel="linear",C=0.25)	0.836	0.855	0.808	0.863	0.831	**0.672**
USE 5 (DAN) + XGB	USE 5 (DAN) + XGB classifier	0.833	0.854	0.804	0.863	0.828	**0.668**
USE 5 (DAN) + Dense	USE 5 (DAN) + Dense network (128 units, 528 610 parameters)	0.833	0.834	0.831	0.835	0.833	0.665
LASER + SVC	LASER + SVC (kernel="linear",C=10)	0.822	0.851	0.781	0.863	0.814	0.646
LASER + XGB	LASER + XGB classifier	0.822	0.837	0.799	0.845	0.818	0.645
LASER + Dense	LASER + Dense network 128 units, 528 610 parameters	0.833	0.851	0.808	0.858	0.829	**0.667**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 7. Contextual word embedding encoders results

Model	Model Description	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
ELMo (mean) + SVC	ELMo (vector averaging) + SVC (kernel="rbf", C=50)	0.872	0.869	0.877	0.868	0.873	0.744
ELMo (mean) + XGB	ELMo (vector averaging) + XGB (lr=0.2, estimators=400)	0.861	0.873	0.845	0.877	0.858	0.722
ELMo (mean) + Dense Network	ELMo (vector averaging) + Dense Network 128 units, 528,610 parameters	0.863	0.86	0.868	0.858	0.864	0.727
ELMo (concat) + CNN	ELMo (vector concatenation) + CNN Network 100 filters, 4 kernel size, 267,498 parameters	0.852	0.87	0.826	0.877	0.848	0.702
ELMo (concat) + LSTM	ELMo (vector concatenation) + LSTM Network 128 LSTM units, 1,647,106 parameters	0.886	0.900	0.868	0.904	0.884	0.772
ELMo (concat) + GRU	ELMo (vector concatenation) + GRU Network 128 GRU units, 1,237,762 parameters	0.884	0.882	0.886	0.881	0.884	0.767
ELMo (concat) + LSTM + Attention	ELMo (vector concatenation) + LSTM Network + Attention 128 LSTM units, 1,639,330 parameters	0.879	0.881	0.877	0.881	0.879	0.758
ELMo (concat) + GRU + Attention	ELMo (vector concatenation) + GRU Network + Attention 128 GRU units, 1,229,986 parameters	0.888	0.886	0.89	0.885	0.888	0.778
ELMo (concat) + BiLSTM + Attention	ELMo (vector concatenation) + Bidirectional LSTM Network + Attention, 3,278,626 parameters	0.888	0.89	0.886	0.89	0.888	0.776
ELMo (concat) + BiGRU + Attention	ELMo (vector concatenation) + Bidirectional GRU Network + Attention, 2,459,938 parameters	0.89	0.89	0.89	0.89	0.89	**0.781**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3009626, IEEE Access

TABLE 8. Transformers results

Model	Model Details	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
BERT-Base-cased	12-layers, 768-hidden, 12-heads, 110M parameters. Trained on cased English text.	0.892	0.913	0.868	0.918	0.890	0.786
BERT-Base-uncased	12-layers, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.	0.904	0.912	0.895	0.913	0.903	0.808
BERT-Large-cased	24-layers, 1024-hidden, 16-heads, 340M parameters. Trained on cased English text.	0.927	0.912	0.945	0.909	0.928	0.856
BERT-Large-uncased	24-layers, 1024-hidden, 16-heads, 340M parameters. Trained on lower-cased English text.	0.929	0.943	0.913	0.945	0.928	0.859
FinBERT	12-layers, 768-hidden, 12-heads, 110M parameters. Trained on Reuters TRC2 dataset	0.890	0.870	0.918	0.863	0.893	0.782
XLNet-Base-cased	12-layers, 768-hidden, 12-heads, 110M parameters. XLNet English model	0.913	0.910	0.918	0.909	0.914	0.827
XLNet-Large-cased	24-layers, 1024-hidden, 16-heads, 340M parameters. XLNet Large English model	0.922	0.922	0.922	0.922	0.922	0.844
RoBERTa-Base	12-layers, 768-hidden, 12-heads, 125M parameters RoBERTa using the BERT-base architecture	0.941	0.949	0.932	0.950	0.940	0.881
RoBERTa-Large	24-layers, 1024-hidden, 16-heads, 355M parameters RoBERTa using the BERT-large architecture	0.941	0.941	0.941	0.941	0.941	0.882
distilbert-base-uncased	6-layers, 768-hidden, 12-heads, 66M parameters The DistilBERT model distilled from the BERT model bert-base-uncased checkpoint	0.902	0.904	0.899	0.904	0.902	0.804
distilbert-base-cased	6-layers, 768-hidden, 12-heads, 65M parameters The DistilBERT model distilled from the BERT model bert-base-cased checkpoint	0.881	0.871	0.895	0.868	0.883	0.763
distilroberta-base	6-layers, 768-hidden, 12-heads, 82M parameters The DistilRoBERTa model distilled from the RoBERTa model roberta-base checkpoint.	0.927	0.931	0.922	0.932	0.927	0.854
ALBERT-base-v1	12 repeating layers, 128 embedding, 768-hidden, 12-heads 11M parameters	0.865	0.881	0.845	0.886	0.862	0.731
ALBERT-large-v1	24 repeating layers, 128 embedding, 1024-hidden, 16-heads, 17M parameters	0.904	0.912	0.895	0.913	0.903	0.808
ALBERT-xlarge-v1	24 repeating layers, 128 embedding, 2048-hidden, 16-heads, 58M parameters	0.911	0.917	0.904	0.918	0.910	0.822
ALBERT-xxlarge-v1	12 repeating layers, 128 embedding, 4096-hidden, 64-heads, 223M parameters	0.941	0.945	0.936	0.945	0.940	0.881
ALBERT-base-v2	12 repeating layers, 128 embedding, 4096-hidden, 64-heads, 223M parameters	0.909	0.905	0.913	0.904	0.909	0.817
ALBERT-large-v2	24 repeating layers, 128 embedding, 1024-hidden, 16-heads, 17M parameters	0.922	0.915	0.932	0.913	0.923	0.845
ALBERT-xlarge-v2	24 repeating layers, 128 embedding, 2048-hidden, 16-heads, 58M parameters	0.936	0.944	0.927	0.945	0.935	0.872
ALBERT-xxlarge-v2	12 repeating layers, 128 embedding, 4096-hidden, 64-heads, 223M parameters	0.932	0.935	0.927	0.936	0.931	0.863
XLM-MLM-en-2048	12-layers, 2048-hidden, 16-heads XLM English model	0.931	0.932	0.932	0.931	0.931	0.863
XLM-MLM-ende-1024	6-layers, 1024-hidden, 8-heads XLM English-German model trained on the concatenation of English and German wikipedia	0.867	0.861	0.877	0.858	0.869	0.735
XLM-RoBERTa-Base	12-layers, 125M parameters, 768-hidden, 8-heads	0.913	0.924	0.899	0.926	0.912	0.827
XLM-RoBERTa-Large	24-layers, 355M parameters 1027-hidden, 16-heads	0.922	0.934	0.909	0.936	0.921	0.845
BART-Large	12-layers, 1024-hidden, 16-heads, 406M parameters	0.947	0.950	0.945	0.950	0.947	**0.895**

REFERENCES

[1] A. Yenter and A. Verma, "Deep cnn-lstm with combined kernels from multiple branches for imdb review sentiment analysis," in 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pp. 540–546, IEEE, 2017.

[2] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., "Universal sentence encoder," arXiv preprint arXiv:1803.11175, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[5] B. G. Malkiel, "The efficient market hypothesis and its critics," Journal of economic perspectives, vol. 17, no. 1, pp. 59–82, 2003.

[6] M.-Y. Day and C.-C. Lee, "Deep learning for financial sentiment analysis on finance news providers," in 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1127–1134, IEEE, 2016.

[7] L. Dodevska, V. Petreski, K. Mishev, A. Gjorgjevikj, I. Vodenska, L. Chitkushev, and D. Trajanov, "Predicting companies stock price direction by using sentiment analysis of news articles," in The 15th Annual International Conference on Computer Science and Education in Computer Science, June 28 – July 1, 2019, Fulda/Bamberg, Germany.

[8] W. Souma, I. Vodenska, and H. Aoyama, "Enhanced news sentiment analysis using deep learning methods," Journal of Computational Social Science, vol. 2, no. 1, pp. 33–46, 2019.

[9] S. F. Crone and C. Koeppel, "Predicting exchange rates with sentiment indicators: An empirical evaluation using text mining and multilayer perceptrons," in 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pp. 114–121, IEEE, 2014.

[10] C. Curme, H. E. Stanley, and I. Vodenska, "Coupled network approach to predictability of financial market returns and news sentiments," International Journal of Theoretical and Applied Finance, vol. 18, no. 07, p. 1550043, 2015.

[11] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. Chitkushev, W. Souma, and D. Trajanov, "Forecasting corporate revenue by using deep-learning methodologies," in 2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO), pp. 115–120, IEEE, 2019.

[12] T. Loughran and B. McDonald, "When is a liability not a liability? textual analysis, dictionaries, and 10-ks," The Journal of Finance, vol. 66, no. 1, pp. 35–65, 2011.

[13] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network," Expert Systems with applications, vol. 40, no. 16, pp. 6266–6282, 2013.

[14] N. Li, X. Liang, X. Li, C. Wang, and D. D. Wu, "Network environment and financial risk using machine learning and sentiment analysis," Human and Ecological Risk Assessment, vol. 15, no. 2, pp. 227–252, 2009.

[15] G. Wang, T. Wang, B. Wang, D. Sambasivan, Z. Zhang, H. Zheng, and B. Y. Zhao, "Crowds on Wall Street: Extracting value from collaborative investing platforms," in Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, pp. 17–30, 2015.

[16] M. Atzeni, A. Dridi, and D. R. Recupero, "Fine-grained sentiment analysis on financial microblogs and news headlines," in Semantic Web Evaluation Challenge, pp. 124–128, Springer, 2017.

[17] S. Agaian and P. Kolm, "Financial sentiment analysis using machine learning techniques," International Journal of Investment Management and Financial Innovations, vol. 3, pp. 1–9, 2017.

[18] S. Sohangir, N. Petty, and D. Wang, "Financial sentiment lexicon analysis," in 2018 IEEE 12th International Conference on Semantic Computing (ICSC), pp. 286–289, IEEE, 2018.

[19] S. Sohangir, D. Wang, A. Pomeranets, and T. M. Khoshgoftaar, "Big data: Deep learning for financial sentiment analysis," Journal of Big Data, vol. 5, no. 1, p. 3, 2018.

[20] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, p. e1253, 2018.

[21] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in Proceedings of the 2015 conference on empirical methods in natural language processing, pp. 1422–1432, 2015.

[22] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," arXiv preprint arXiv:1503.00075, 2015.

[23] Y. Kim, "Convolutional neural networks for sentence classification," arXiv preprint arXiv:1408.5882, 2014.

[24] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in Advances in neural information processing systems, pp. 649–657, 2015.

[25] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 562–570, 2017.

[26] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 1480–1489, 2016.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, pp. 3111–3119, 2013.

[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, 2014.

[29] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[30] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in International conference on machine learning, pp. 1188–1196, 2014.

[31] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in Advances in neural information processing systems, pp. 3294–3302, 2015.

[32] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," arXiv preprint arXiv:1705.02364, 2017.

[33] M. Artetxe and H. Schwenk, "Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond," Transactions of the Association for Computational Linguistics, vol. 7, pp. 597–610, 2019.

[34] X. Man, T. Luo, and J. Lin, "Financial sentiment analysis (fsa): A survey," in 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), pp. 617–622, IEEE, 2019.

[35] S. Yang, J. Rosenfeld, and J. Makutonin, "Financial aspect-based sentiment analysis using deep representations," arXiv preprint arXiv:1808.07931, 2018.

[36] C.-H. Du, M.-F. Tsai, and C.-J. Wang, "Beyond word-level to sentence-level sentiment analysis for financial reports," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1562–1566, IEEE, 2019.

[37] L. Zhao, L. Li, and X. Zheng, "A bert based sentiment analysis and key entity detection approach for online financial texts," arXiv preprint arXiv:2001.05326, 2020.

[38] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," arXiv preprint arXiv:1801.06146, 2018.

[39] P. J. Stone, D. C. Dunphy, and M. S. Smith, "The general inquirer: A computer approach to content analysis.," 1966.

[40] J. L. Rogers, A. Van Buskirk, and S. L. Zechman, "Disclosure tone and shareholder litigation," The Accounting Review, vol. 86, no. 6, pp. 2155–2183, 2011.

[41] A. K. Davis, W. Ge, D. Matsumoto, and J. L. Zhang, "The effect of manager-specific optimism on the tone of earnings conference calls," Review of Accounting Studies, vol. 20, no. 2, pp. 639–673, 2015.

[42] W. Zhang and S. Skiena, "Trading strategies to exploit blog and news sentiment," in Fourth international aAAI conference on weblogs and social media, 2010.

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

[44] Z. S. Harris, "Distributional structure," Word, vol. 10, no. 2-3, pp. 146–162, 1954.

[45] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," Psychological review, vol. 104, no. 2, p. 211, 1997.

[46] M. Sahlgren, The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. PhD thesis, 2006.

[47] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," Journal of artificial intelligence research, vol. 37, pp. 141–188, 2010.

[48] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 427–431, Association for Computational Linguistics, April 2017.

[49] Y. Sharma, G. Agrawal, P. Jain, and T. Kumar, "Vector representation of words for sentiment analysis using glove," in 2017 international conference on intelligent communication and computational techniques (icct), pp. 279–284, IEEE, 2017.

[50] P. Lauren, G. Qu, J. Yang, P. Watta, G.-B. Huang, and A. Lendasse, "Generating word embeddings from an extreme learning machine for sentiment analysis and sequence labeling tasks," Cognitive Computation, vol. 10, no. 4, pp. 625–638, 2018.

[51] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," Expert Systems with Applications, vol. 117, pp. 139–147, 2019.

[52] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," arXiv preprint arXiv:1712.09405, 2017.

[53] R. Velioğlu, T. Yıldız, and S. Yıldırım, "Sentiment analysis using learning approaches over emojis for turkish tweets," in 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 303–307, IEEE, 2018.

[54] A. A. Altowayan and A. Elnagar, "Improving arabic sentiment analysis with sentiment-specific embeddings," in 2017 IEEE International Conference on Big Data (Big Data), pp. 4314–4320, IEEE, 2017.

[55] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," arXiv preprint arXiv:1802.05365, 2018.

[56] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.

[57] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," arXiv preprint arXiv:1803.02893, 2018.

[58] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in Proceedings of the 27th International Conference on Computational Linguistics, pp. 1638–1649, 2018.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008, 2017.

[60] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in Proceedings of the IEEE international conference on computer vision, pp. 19–27, 2015.

[61] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," arXiv preprint arXiv:1908.10063, 2019.

[62] G. Lample and A. Conneau, "Cross-lingual language model pretraining," arXiv preprint arXiv:1901.07291, 2019.

[63] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," arXiv preprint arXiv:1909.11942, 2019.

[64] Y. Yang, H. Cai, Y. Ruan, K. O'Connor, G. Gonzalez-Hernandez, J. Perrone, A. Sarker, et al., "Text classification models for the automatic detection of nonmedical prescription medication use from social media," 2020.

[65] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.

[66] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," arXiv preprint arXiv:1911.02116, 2019.

[67] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," arXiv preprint arXiv:1910.13461, 2019.

[68] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI Blog, vol. 1, no. 8, p. 9, 2019.

[69] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts," Journal of the Association for Information Science and Technology, vol. 65, no. 4, pp. 782–796, 2014.

[70] K. Cortis, A. Freitas, T. Daudert, M. Huerlimann, M. Zarrouk, S. Handschuh, and B. Davis, "Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news," in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 519–535, 2017.

[71] F. Chollet et al., "Keras." https://keras.io, 2015.

[72] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," ArXiv, vol. abs/1910.03771, 2019.

[73] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of statistics, pp. 1189–1232, 2001.

[74] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: extreme gradient boosting," R package version 0.4-2, pp. 1–4, 2015.

[75] L. Deng, D. Yu, et al., "Deep learning: methods and applications," Foundations and Trends® in Signal Processing, vol. 7, no. 3–4, pp. 197–387, 2014.

[76] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," IEEE Signal Processing Magazine, vol. 28, no. 1, pp. 145–154, 2010.

[77] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," Computational intelligence and neuroscience, vol. 2018, 2018.

[78] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8599–8603, IEEE, 2013.

[79] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, et al., "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4779–4783, IEEE, 2018.

[80] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," arXiv preprint arXiv:1710.07654, 2017.

[81] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," Neurocomputing, vol. 337, pp. 325–338, 2019.

[82] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," arXiv preprint arXiv:1605.05101, 2016.

[83] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[84] K. Mishev, A. Gjorgjevikj, R. Stojanov, I. Mishkovski, I. Vodenska, L. Chitkushev, and D. Trajanov, "Performance evaluation of word and sentence embeddings for finance headlines sentiment analysis," in International Conference on ICT Innovations, pp. 161–172, Springer, 2019.

[85] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[87] T. Rao, S. Srivastava, et al., "Analyzing stock market movements using twitter sentiment analysis," 2012.

[88] J. Smailović, M. Grčar, N. Lavrač, and M. Žnidaršič, "Predictive sentiment analysis of tweets: A stock market application," in International Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data, pp. 77–88, Springer, 2013.

[89] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng, "News impact on stock price return via sentiment analysis," Knowledge-Based Systems, vol. 69, pp. 14–23, 2014.