IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 June 2023, accepted 3 July 2023, date of publication 10 July 2023, date of current version 14 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3293733

APPLIED RESEARCH

Domain-Specific Sentiment Analysis: An Optimized Deep Learning Approach for the Financial Markets

MEHDI YEKRANGI AND NIKOLA S. NIKOLOV
Department of Computer Science and Information Systems, University of Limerick, Limerick, V94 T9PX Ireland
Corresponding author: Mehdi Yekrangi (mehdi.yekrangi@ul.ie)

This work was supported in part by the Science Foundation Ireland under Grant 18/CRT/6223.

ABSTRACT Although different studies are caried out by deep learning models for financial markets sentiment analysis, there is a lack of specific embedding method that regards the domain. Therefore, the goal of this study is to discover what type of embedding techniques along with different classification algorithms work better for the financial markets' sentiment analysis to present an optimized embedding method in the domain. In this paper we present a broad comparative study of multiple classification models trained for sentiment analysis and will improve their performance with an optimized embedding layer. We use a heterogeneous corpus of both formal (news headlines) and informal (tweets) text to increase the robustness and build the models with CBOW, GloVe, and BERT pre-trained embeddings as well as developing an optimized embedding layer to improve the results. The best results reported here are by our LSTM model with the fine-tuned embedding layer, which has an accuracy of 0.84 and a macro-average F1-score of 0.8. Our results give evidence that the fine-tuned embedding is superior to utilising pretrained CBOW, GloVe, and BERT embeddings for financial markets sentiment analysis. We train SVM, MLP, CNN, generic RNN and LSTM models by a comprehensive approach in input data and algorithms. As a result, a sentiment analysis model is presented with a robust performance for different datasets in the domain.

INDEX TERMS Sentiment analysis, natural language processing, deep learning, financial markets, text mining, neural networks.

I. INTRODUCTION

The development of Web 2.0 technologies in the last decade, specifically social media and micro-blogging, have enabled access to unprecedented amount of information in a timely manner. The insights this information provides can be leveraged in many domains including financial markets and investing. The content created on various social media and blog platforms has changed the way that investors get access to information about stocks, brands, and currencies and the way they react to it [1]. This information can be collected and evaluated by text mining technologies, one of them being *sentiment analysis* (SA). SA, also called *opinion mining*, is a classification task for the purpose of discovering views expressed by people in the textual data and extracting emotional polarities [2]. In the financial markets' domain, sentiment is used mostly for predicting future market movements. To that end, most SA research has been conducted on news and micro-blogging sites with either traditional machine learning algorithms including support vector machine (SVM), random forest (RF), Naïve Bayes (NB) or artificial neural networks (ANN) [3].

One of the challenges in SA is the high dimensionality of the input data. The training of traditional text classification models typically requires the employment of a dimensionality reduction step, while the state-of-the art deep learning algorithms natively deal with high-dimensional data. Despite high dimensional input, deep learning algorithms are able to discover dependencies between words in a sentence as part of the training process and select the best features accordingly [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Justin Zhang.

Another challenge in SA is the *context dependency issue*, i.e., when a single word has different meanings or connotations in different contexts. For instance, according to the WordNet dictionary [4], the word ''strong'' has a positive subjectivity. However, in each of the following three sentences expressing opinions about the market, it has a different connotation.

- The sell trend is *strong*. (negative)
- The buy trend is *strong*. (positive)
- Both buy and sell side pressures are *strong*. (neutral)

This issue arises when using the more traditional text representations such as *term frequency-inverse document frequency* (TF-IDF) as well as when employing the relatively newer *embedding* methods. In pre-trained embedding methods, such as Word2Vec [5] and GloVe. ([6], the words in a document are represented regarding their context. Although these methods are much more advanced than TF-IDF, and do take the context into account, but they still may be unable to discriminate between different sentiments or connotations of the same word, such as in the examples above with the word ''strong''. This embedding issue is due to the fact that most pre-trained embedding methods regard the general application of a word in a context, while disregarding the topic information of the specific document, in which the word is used [7].

Non-linear relationships between the elements of a sentence make it difficult to measure the sentiment of a word, given other words in the same sentence. To that end, some deep learning algorithms along with fine-tuned embeddings are able to factor in such contextual relationships as topical information and grammatical structures [8] by employing hidden layers and a high-dimensional feature space which allow the extraction of features that reflect both the context and relationship dependencies in the text. Because of their structure and ability to extract best features, deep learning models have achieved promising results [9]. The predominant deep learning algorithms which have been used for financial markets' SA include convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory (LSTM) networks [2].

In this study, SA in the financial markets' domain is carried out by both deep learning and traditional machine learning algorithms. Each type of the models is examined with respect to different parameters which can impact on the models' performance in the particular domain of financial markets. In particular, this paper contributes to the body of knowledge by presenting the results from a comparative study of various embedding methods as well as various machine learning algorithms for training an effective and robust model for SA in the financial markets domain. In the next section, the most common terms and techniques in this field are introduced briefly. Following that, the existing gaps in the domain are addressed by a review of recent related work. Next, the methodology and experiments sections present the various methods and algorithms towards filling the gaps and how they have been applied in the experimental work. The results of applying different methods are discussed and reviewed in detail in the results and discussion section. Finally, we draw conclusions from this study and outline possible future directions of work.

Before proceeding with the review of related work, some terms and techniques, commonly used in natural language processing (NLP) are briefly introduced here. Two types of algorithmic techniques are employed and examined in this work, namely embedding and classification algorithms for text pre-processing and classification, respectively. Embedding is a term used for text representation in a vector space, i.e., words/sentences are represented in the form of real-valued vectors to be fed as input into a classifier. Embedding and vectorization are used interchangeably in this paper. The embedding models and methods, used in this work, are the following ones:

- *CBOW embeddings* [5]: A pre-trained embedding algorithm based on neural networks. In a Word2Vec algorithm, the words are vectorized by 300 dimensions based on the context of the corpus. There are two types of Word2Vec algorithms: *continuous bag of words* (CBOW) and *Skip-Gram*. While in CBOW the target word is vectorized based on its surrounding (context) words, the skip-gram embedding vectorizes the surrounding words based on the target word.
- *GloVe embeddings* [6]:GloVe stands for global vectors for word representation. It is another pre-trained embedding algorithm, which is based on words co-occurrence in the corpus representing the words in 100-dimesion vectors.
- *Sequence embeddings*: A simple embedding method in which each word in a document is represented by a unique number. Therefore, a text document (e.g., a sentence) is represented by a sequence of numbers potentially padded with zeros for having all documents represented by vectors of equal length.
- *BERT embeddings* [10]: BERT stands for bidirectional encoder representation from transformers, which is a new pre-trained language representation model. By considering both left and right context for all layers, BERT pretrains deep bidirectional representations from unlabelled text. The text is then represented by 768-dimesion vectors as output. BERT is available in different variations. The variation used in this work is ''bert-base-uncased''.
- *Keras embeddings*: In contrast to pre-trained embeddings, the Keras library allows to train an embedding by adding an embedding layer to a neural network as its first layer. The input to this embedding layer can be, for example, sequence embedding vectors (see above), which gets fine-tuned by it. In our experiments we refer to this embedding as *Keras embedding*.[1]

[1]More about the embedding method can be found at https://keras.io/api/layers/core_layers/embedding/

This study focuses on deep learning methods for classification while also comparing their results to those of traditional classifiers. The classification algorithms we experimented with are the following ones:

- *Rule-based scoring*: In this classification algorithm, a sentence is scored based on the sentiment score of each word in it. These sentiment scores are pre-defined in a lexicon and can be 1 for positive, -1 for negative, and 0 for neutral words.
- *Support vector machines (SVM)*: SVM is a supervised machine learning algorithm that can be used for classification and regression purposes. In SVM, each data point is projected into an n-dimensional space, where n is the number of features. The algorithm searches for the best hyperplane in that can segregate the classes with the highest accuracy.
- *Multi-layer perceptron (MLP)*: MLP is a class of feedforward artificial neural network (ANN) which is comprised of a series of fully connected layers. An MLP model consists of the input, hidden, and output layers. The hidden and output layers utilize non-linear activation functions at each node and the neural network gets trained based on a supervised learning technique called backpropagation.
- *Convolutional neural network (CNN)*: CNNs are also a type of feedforward neural networks, which work based on convolutional kernels and pooling layers that slide over input nodes to reduce dimensionality while keeping the information.
- *Recurrent neural network (RNN)*: RNNs are a class of neural networks that are adopted to work with sequential data in which one data point is dependent upon the previous data point. By using a memory state, RNNs store the information of previous inputs to be used for generating the next output.
- *Long short-term memory (LSTM)*: LSTM is a special kind of RNN, which is capable of capturing long-term dependencies in the input sequential data.

II. RELATED WORK

In this section, various deep learning approaches for SA are reviewed to address the gaps and possible improvements. In a recent study by Dang et al. [11], the most popular deep learning algorithms including MLP, CNN and RNN are investigated and compared to each other on multiple datasets. For each of the deep learning algorithms, two embeddings, a TF-IDF and a Word2Vec are fed into the neural network. The goal is to compare the performance of different neural networks and two different embedding techniques for sentiment polarity detection. In conclusion, the RNN applied on Word2Vec embeddings, outperforms MLP and CNN. However, the simultaneous use of pre-trained word embeddings and a fine-tuned embedding layer in the same neural network is arguable.

In a study by Jangid et al. [12], deep learning models are applied with the aim of polarity prediction as well as aspect extraction. Two neural network architectures, namely LSTM and CNN, are applied for aspect recognition and polarity score prediction, respectively. However, the dataset size is small, and the reported findings are limited to just one type of neural network for each task, which are LSTM for aspect recognition and CNN for sentiment prediction. Some other studies tie sentiments in a corpus to the corresponding market movements. Specifically in the domain of financial markets, Souma et al. [13] carry out SA by deep learning with the aim of predicting market movements. Market sentiment is measured with regard to news headlines and their impact on the stock market return after a one-minute interval. The approach for labelling the news is based on the market returns after the news is published, which is quite different than other labelling approaches. In that study, GloVe vectors are used as embeddings to be fed into an LSTM neural network. Therefore, the study is based on just one type of neural networks and one type of embedding. The deep learning model classifies news headlines as either positive or negative without considering potential neutral headlines with an average accuracy of 75.5%. In a similar work by Vicari and Gaspari [8], LSTM is applied to the embedding layer's vectors. Their method, again, is based on only one type of embedding and neural network. The reported accuracy is at most 58%. Another study by Sohangir et al. [9] with the same goal of predicting future stock price, examines both LSTM and CNN on tweets extracted from StockTwits[2] to find any potential relationship between market price and the related tweets. They achieve better results by CNN than LSTM. However, there is no discussion about the embedding method used for the input data, which can have a significant impact on the results. In another recent work by Yıldırım et al. [3], both traditional machine learning and deep learning algorithms are applied on another dataset of tweets collected from StockTwits and the results are compared. Same as in the study of Vicari and Gaspari [8], the input vectors are trained in the embedding layer by feeding into different neural networks the performances of which are compared to the performance of traditional machine learning algorithms. The results indicate the superior performance of LSTM at 80.8% accuracy, which is higher than the accuracy of other deep learning and traditional machine learning algorithms attempted. It should be noted that this work considers only positive and negative labels. In another study [14], a derivative BERT embedding called Fin-BERT is pre-trained on the financial corpus. Having fed Fin-BERT embedding to the classifiers, the reported F1-score is 0.84. However, the training dataset in that research only includes formal language text, such as news, and the size of it is one third of the training dataset used in our research.

The SA models proposed in the recent literature differ in a few aspects including the neural networks architecture, the embedding, the number of classes, and the datasets being used, which are noteworthy addressing towards developing

[2]https://stocktwits.com/

TABLE 1. A summary of most recent related SA studies with deep learning approach.

Study	Goal	Neural network/Embedding	Limitation
[11]	Comparing different neural networks for SA	MLP, CNN, RNN/TF-IDF, Word2Vec, Keras embedding layer	No financial corpus amongst the datasets, using pre-trained W2V and Keras embedding simultaneously
[12]	Aspect recognition and polarity detection in financial tweets	LSTM, CNN/Fast Text, Godin, GloVe, Word2Vec, Keras embedding layer	Small dataset size, just one type of neural networks for each task
[13]	Predicting future stock price by SA	LSTM/GloVe	Just one type of corpus (news) as dataset, just one type of neural networks and embeddings examined, no neutral label
[8]	Predicting stock price by SA	LSTM/Keras embedding layer	Just one type of corpus (news) as dataset, just one type of neural network and embedding examined, no neutral label
[9]	Predicting stock price by SA	LSTM, CNN/-	Embedding method not clear, no neutral label
[3]	Comparing different deep learning and traditional machine learning algorithms for financial tweets SA	RNN, LSTM, GRU/Embedding layer	Just one type of embedding examined, no neutral label
[14]	Comparing classification results by different transfer learning embeddings	LSTM/ELMo, ULMFit, BERT	Small dataset size, just one type of corpus (news) as dataset

an improved SA model. With respect to these aspects related to the input data as well as various algorithms affecting the results, there is a lack of comparative studies on the deep learning approaches used for SA [11]. Therefore, in this study we compare three well-known neural network architectures for SA specifically in the financial markets' domain. The embedding technique is another predominant parameter on the performance of the models. There are various embedding techniques, which can be either pre-trained models such as CBOW or an embedding layer added to the neural network. This study aims to answer which embedding method is effective for SA in the financial markets' domain and how to optimize it.

The number of classes is another factor, which is treated differently by different studies. While typically SA involves classifying sentiments in three categories (positive, negative, neutral), the neutral label has been disregarded in some research works [3], [13], which in turn limits the study [15]. Another issue with SA models in the financial markets' domain is the insufficient robustness assessment.

Many studies [3], [8], [13] in the domain focus just on one source of data, mostly either news headlines or tweets for training the deep learning models. Nevertheless, the terms and phrases that are used in news media are different than what is used in many tweets as colloquial language. In other words, there are many terms and phrases, which can be found either only in colloquial language or in formal language. Models trained on only one of these sources may not perform as well when applied to a dataset from the other source. In an attempt towards solving this issue, data sources of both types are included in our work to develop a robust model, capable of performing well on different financial corpora.

In summary, what distinguishes our study from other similar studies are:

1) Optimized embedding, which is focused on the domain.
2) Comparative approach of different neural networks as well as different embedding techniques.
3) Consideration of the robustness of the models by employing different sources of data.
4) Consideration of the neutral label in SA.

Aspects of recent related SA studies are summarized in Table 1.

III. DATASETS AND METHODS

In this section, we introduce the datasets employed in our work and briefly list the methods leveraged at the various steps of the machine learning workflow.

A. DATASETS

We employ datasets for training both embeddings and classification models. The process of pre-training embeddings is unsupervised, i.e., it does not need a labelled dataset and it can be done by different algorithms such as CBOW and GloVe, and BERT. The goal is for each word to get represented by a unique vector that is then fed into a classifier. For pre-training CBOW and BERT embeddings, we collected a dataset of 328,326 unique tweets about crypto currencies from Twitter (in the view of cryptocurrencies being a hot trend on the financial markets) and 140,000 unique tweets on the currency, stock and gold markets. This dataset is collected as a source of various words that can be used to express

opinions about the financial markets. The crypto currency and all other financial markets tweets are from the periods 9/6/2021-13/6/2021 and 1/9/2022-1/11/2022, respectively. As a result, 27,951 unique meaningful (non-gibberish) words are extracted to pre-train CBOW and BERT embeddings model. There are two reasons that we add crypto currency to our dataset for pre-training the embeddings. First, it increases the diversity in the dataset in terms of financial markets being represented, and second, since the cryptocurrencies were a hot topic in the data collection period, it allowed us to collect a staggering number of relevant tweets in a short period of time. We refer to this dataset as DS1 in the remainder of the paper. It should be noted that DS1 is just used for pre-training CBOW and BERT embedding models, while the classifiers are trained on a dataset that comprises different topics including stock markets, commodities, corporations, currencies, crypto currencies, etc.

We train classifiers on a dataset combining three different sources. The first one [16] consists of 10,631 tweets about stock markets. The second one [17] consists of 4,821 various news headlines about financial markets. And the third is a dataset consisting of 1,000 tweets about cryptocurrencies collected and labelled in this study. The tweets in the third dataset are labelled by three academic annotators, two of whom were asked to label each tweet as ''positive'', ''negative'', or ''neutral'', and the third annotator resolved conflicts between the other two. The most unanimous label is assigned to each tweet. In total, we work with a dataset of 16,452 samples. In order to examine the performance of the models, we also experimented with two subsets of the entire dataset, one of 6,687 samples, and another of 11,336 samples. In all datasets, the ratio of the samples originating from the three initial datasets is the same. In the remainder of the paper, the entire labelled dataset is referred as DS2.

Before training an SA model, we first apply text pre-processing consisting in tokenization, removing stop words, punctuation, non-letter characters and links, lemmatization, and lowercasing. Having cleaned and prepared the data, we train multiple SA models by varying the vectorization and the dimensionality reduction methods applied as well as the classification algorithms. In the remainder of this section, we give more details about the methods considered.

B. FEATURE EMBEDDING

The way text documents are represented for the purpose of SA can be based on simple methods such as frequency, scoring, sequence embedding, or more advanced methods such as embeddings by Word2Vec, GloVe, or BERT. In NLP tasks, features may be selected by traditional methods such as lexicon-based algorithms, part of speech (POS) tagging or by advanced deep learning algorithms [11]. In lexical methods, lexicons and pre-defined words such as adjectives or words with high frequency in some documents are selected as features to perform SA. On the other hand, in deep learning algorithms, feature selection is performed while training the neural network. These two approaches are examined by different experiments in this work to measure their impact on the models' performances.

C. CLASSIFICATION ALGORITHM

The choice of a classification algorithm is yet another variable that affects the model being trained. The algorithms can be either traditional machine learning algorithms such as SVM and Random Forest, to mention a few, or deep learning algorithms based on training a neural network. In this work we experimented with SVM, MLP, CNN, RNN, and LSTM as well as a variety of embedding methods, which are discussed further in the following sections.

We employ two baseline models: a lexicon-based model in which each sentence is scored and classified based on the cumulative polarity of the words in it, and an SVM model. Thereafter, several other models are experimented. We first introduce the baseline models and discuss the various aspects that influence them. Then, more complex models are trained and compared against the baseline models as well as against each other.

IV. EXPERIMENTAL SETUP AND RESULTS

Given the specific domain of application as financial markets and the requirements for sentence-level SA such as word dependencies, the performance of different models can vary significantly. In this section, various experiments and the acquired results are presented. Therefore, in the following section, we can have a thorough discussion and examination of the models.

A. BASELINE MODELS

First, we trained two baseline SA models, a lexicon-based and a ML-based one [18]. While in lexicon-based models the sentiment of a text document depends on the labels of words in a lexicon [19], in ML-based models, the sentiment is classified by a ML model trained on a labelled dataset [20].

1) LEXICON-BASED MODEL

The lexicon-based models consist of two steps. First, a lexicon of words is built, then the lexicon is applied for determining the sentiment of entire text documents. In order to build the pre-defined lexicon, we used the financial market's sentiment lexicon [21], developed in our previous work as the seed words resource to be expanded. The lexicon consists of 210 words such as *bullish*, *decline*, *loss*, etc., which are commonly used to express opinions about financial markets. The words in the lexicon are then vectorized by a CBOW model, trained on the dataset of 468k tweets about cryptocurrencies (see Section III-A). The reason of choosing CBOW for embedding rather than more traditional algorithms such as TF-IDF is the context dependency of the word vectors that are produced based on each words' neighbours in the corpus, represented in a high dimensional vector space. The CBOW embedding is a better fit than

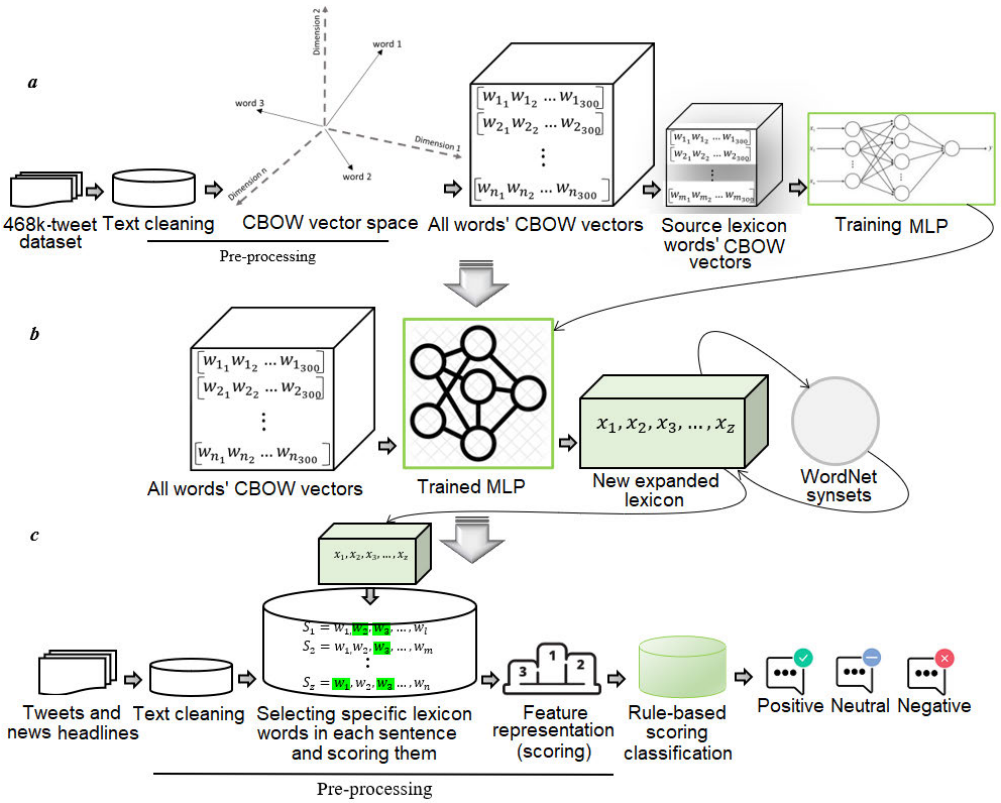

FIGURE 1. a) Training a MLP neural network by the source lexicon words b) building a new expanded lexicon c) classifying tweets by the new expanded lexicon.

skip-gram for our application, as we work with a big dataset while looking for the vector representation of more frequent (as opposed to less frequent) words in the financial markets' domain [5].

An MLP model is then trained on the lexicon word vectors. In this model, there are 300 nodes for CBOW vectors in the first layer, followed by 30 nodes in the hidden layer and 3 nodes in the output layer. Having trained the neural network on the lexicon words, all the words in DS1 are fed into the trained model in order to be classified as either positive, negative, or neutral. Those of them that are labelled either positive or negative are then added to the lexicon to expand it. We also expand the lexicon with the synonyms of both positive and negative labelled words by an iterative approach based on WordNet synsets [22].

As one of the baseline SA models, we employ the expanded lexicon for labelling the sentences in DS2. In this process, each sentence gets a score based on the words in it and their corresponding label in the expanded lexicon. For example, in the sentence *I'm still incredibly bullish on bitcoin*, there is the positive-labelled word *bullish* which gives one positive score to the whole sentence, while all other words' scores are counted 0 as neutral because they are not present in the expanded lexicon. The accuracy acquired by this method is 48%, i.e., 15% better than the accuracy of random classification (33% for each of three labels), but clearly it has room for improvement. The process of training the MLP neural network for creating the lexicon, and subsequently classifying the text documents is illustrated in Figure 1.

2) SVM MODEL

As another baseline model, we experimented with SVM with polynomial, Gaussian and radial basis function (RBF) kernels, respectively. In this set of experiments, each sentence was vectorized by CBOW, GloVe, and BERT embeddings. As a result, RBF kernel performed better than other kernels. On the test dataset, as presented in Table 2, the SVM algorithm had similar accuracy for both CBOW and GloVe vectors as the input. The best accuracy in this set of experiments was 69%, achieved by applying SVM on BERT embeddings.

B. DEEP LEARNING MODELS

In the remainder of this section, we introduce the deep learning algorithms for training SA models in the financial markets' domain. In all models, four types of text representations, namely CBOW, GloVe, BERT, and Keras embeddings are utilized and compared. The embedding for CBOW consists of 300 dimensions; GloVe has vectors of 100 dimensions, BERT embeddings are vectors with 768 dimensions, and the Keras embeddings results are vectors of 1024 dimensions. While CBOW, GloVe, and BERT are pre-trained embeddings, the Keras embeddings get fine-tuned during training the models. Regarding pre-trained

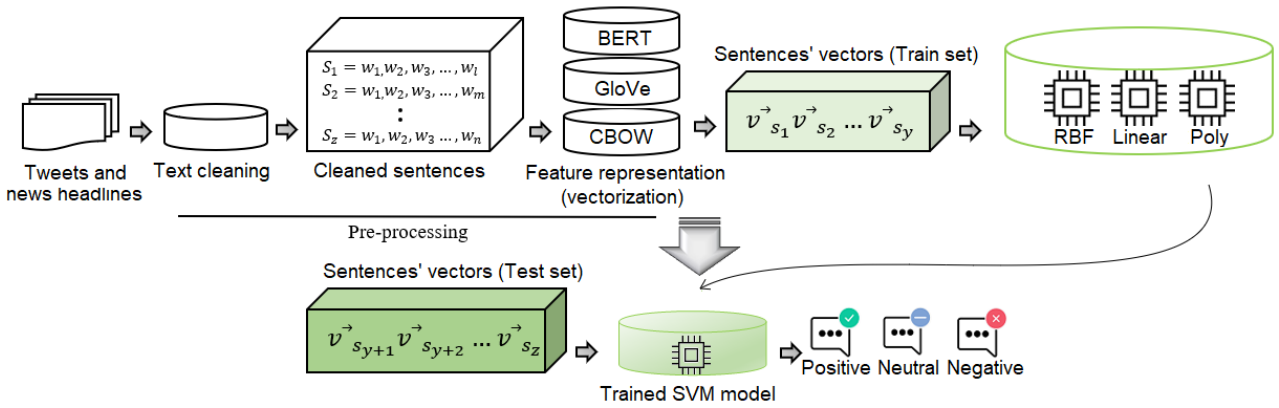

FIGURE 2. Classifying tweets and news headlines by the SVM algorithm.

TABLE 2. Traditional SA algorithms, configurations and results.

Dataset size	Feature extraction/selection	Feature representation	Classification	Acc
6,687 samples	Bag of words/Lexicon	Count scoring	Lexicon Scoring	0.49
11,336 samples	Bag of words/Lexicon	Count scoring	Lexicon Scoring	0.44
6,687 samples	Bag of words	CBOW	SVM	0.48
11,336 samples	Bag of words	CBOW	SVM	0.57
16,452 samples	Bag of words	CBOW	SVM	0.63
6,687 samples	Bag of words	Glove	SVM	0.33
11,336 samples	Bag of words	Glove	SVM	0.59
16,452 samples	Bag of words	Glove	SVM	0.63
6,687 samples	Bag of words	BERT	SVM	0.61
11,336 samples	Bag of words	BERT	SVM	0.67
16,452 samples	Bag of words	BERT	SVM	**0.69**

embeddings, there are two issues that need to be addressed as below.

1) SIMILAR VECTORS FOR OPPOSITE WORDS

Both CBOW and GloVe embeddings are based on words' statistical co-occurrences and each word's representation is determined by accompanying words. Also, BERT embeddings are based on semantic co-occurrences. However, all embeddings are from unlabelled text, which in turn can result in to some deficiencies for SA in specific domains such as financial markets. There are some opposite words in the corpus which can occur in similar contexts, and thus being represented by similar CBOW/GloVe vectors, while being labelled differently. Same for BERT, as the embedding is based on masked language modelling (MLM) [10], which relies on contextual patterns, two different words (sentimentally) may have similar contextual pattern, hence, leading to similar sentence vectors. For example, there are two very common words *buy* and *sell* in the financial corpus with the opposite meaning. In the corpus, they have many accompanying words in common which leads to similar vector representations. There are multiple sentences like the two below in which, opposing words such as *buy* and *sell*, *bullish* and *bearish*, etc. have the same context.

- *The broker has given a* ***buy*** *recommendation on the asset.*
- *The broker has given a* ***sell*** *recommendation on the asset.*

The pre-trained vector representations of such words cannot be used reliably to compare their similarity and classify them sentimentally [23].

2) SSAME WORD, DIFFERENT CONNOTATIONS

Some very common words in the corpora can carry both positive and negative sentiments, which is interpreted as polysemy. In other words, same word with a unique embedding has different sentiment polarities depending on the context [7]. For instance, the words *pressure* and *high* would convey different connotations in the opinions of the following sentences.

- ***Buying pressure*** *is still* ***high***.
- ***Selling pressure*** *is still* ***high***.

While in the first sentence the opinion is about market price appreciation expressed by *buying pressure*, the second opinion conveys the exact opposite sentiment. However, they both use the words *pressure* and *high* to describe two opposite market conditions. As shown in the samples, the connotation of some words such as *pressure* and *high*

TABLE 3. MLP SA algorithms, configurations and results.

Dataset size	Feature representation	Acc
6,687 samples	CBOW	0.52
11,336 samples	CBOW	0.6
16,452 samples	CBOW	0.69
6,687 samples	GloVe	0.56
11,336 samples	GloVe	0.65
16,452 samples	GloVe	**0.73**
6,687 samples	BERT	0.59
11,336 samples	BERT	0.61
16,452 samples	BERT	0.66
6,687 samples	Keras embeddings	0.52
11,336 samples	Keras embeddings	**0.70**
16,452 samples	Keras embeddings	**0.75**

TABLE 4. CNN SA algorithms, configurations and results.

Dataset size	Feature representation	Acc
6,687 samples	CBOW	0.33
11336 samples	CBOW	0.55
16,452 samples	CBOW	0.57
6,687 samples	GloVe	0.49
11,336 samples	GloVe	0.56
16,452 samples	GloVe	0.62
6,687 samples	BERT	0.63
11336 samples	BERT	0.63
16,452 samples	BERT	0.63
6,687 samples	Keras embeddings	0.52
11,336 samples	Keras embeddings	0.68
16,452 samples	Keras embeddings	**0.72**

can be both positive and negative, while there can only be one CBOW/GloVe/BERT representation for each unique word in the corpus. However, depending on the context, when the vector of an entire sentence is fed into a neural network, it should be interpreted as positive in some cases and negative in others. Thus, a successful model could handle SA challenges at sentence level, which are primarily context sensitivities and the dependencies that words have on each other in a sequence as a sentence [24]. Such dependencies are addressed in the deep learning experiments while pre-trained and fine-tuned embeddings are compared as well in different neural networks.

3) MLP EXPERIMENTS

In the first set of experiments as MLP models, the neural networks are fed with CBOW, GloVe, BERT, and Keras embeddings. Then the input is passed through three hidden layers of 150 nodes each. Finally, the output layer consists of 3 nodes.

Given the results, the MLP model performed better than the baseline SVM by CBOW and GloVe embeddings. We also gathered evidence that the models' accuracy is improvable by expanding the dataset. As presented in Table 3, MLP has the best accuracy with the Keras embeddings. However, the pre-trained GloVe embedding too, leads to a good accuracy when used as the embedding method in MLP neural networks.

4) CNN EXPERIMENTS

In the next set of experiments, a CNN architecture is used for training a SA model. The convolutional layer of the CNN consists of 32 nodes in either a two-dimensional 9×9 kernel or one-dimensional kernel of length 9. It is followed by a max pooling layer with either a two-dimensional window of size 2×2 or one-dimensional window of size 4. Having flattened the max pooling layer output, it is passed through a dense layer of 100 nodes to get the 3 output nodes activated by the softmax function. Overall, the CNN models, achieved lower accuracy scores than the MLP and SVM models, for any type of embeddings. Therefore, the CNN trained on CBOW, Glove, and BERT embeddings could not beat the baseline SVM performance on these embeddings. Amongst all CNN models, the model with the Keras embedding obtained the best accuracy. Nevertheless, its accuracy was lower than what was achieved with the same embedding by MLP (see Section 4.3).

5) RNN EXPERIMENTS

RNNs are well-known for their success in text mining and sentiment analysis. Therefore, in the next set of experiments, we train both a generic RNN as well as one of its variations, LSTM. Similar to the previous experiments, we train RNN and LSTM models on CBOW, GloVe, BERT, and Keras embeddings. In the RNN architecture, a simple RNN layer of 100 neurons is used, followed by a dense layer of 50 nodes, which leads to an output layer of 3 nodes. Again, the Keras embeddings led to the best accuracy. However, while RNN performed better than our MLP and CNN models when using a Keras embedding layer, it did not perform as well with pre-trained CBOW, GloVe, and BERT embeddings. Therefore, same as CNN, it could not perform better than the baseline SVM model on pre-trained embeddings.

6) LSTM EXPERIMENTS

Finally, we also experiment with training LSTM models. In LSTM, the position of a word in a document is considered as a feature (sequential feature). Based on it, long-term dependencies in the text are taken into account. LSTM is explicitly designed to avoid the long-term dependency problem that arises while working at either sentence or document level. In a sentence/document, there can be distant words, which despite the distance between them, are related. For example, there are some words in the sentence below, which have an impact on each other, while there is a significant distance between them.

"The seller market is very strong and their power is dominant."

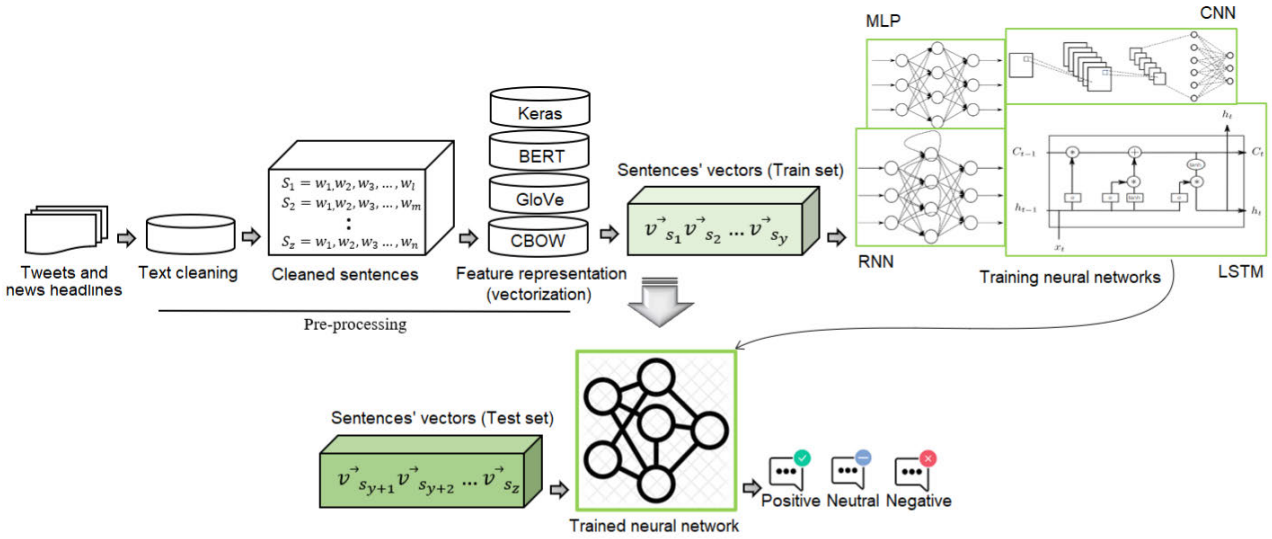

FIGURE 3. Training several neural networks by different embedding methods for sentiment classification.

TABLE 5. RNN SA algorithms, configurations and results.

Dataset size	Feature representation	Acc
6,687 samples	CBOW	0.33
11,336 samples	CBOW	0.44
16,452 samples	CBOW	0.57
6,687 tweets	GloVe	0.33
11,336 samples	GloVe	0.44
16,452 samples	GloVe	0.57
6,687 tweets	BERT	0.40
11,336 samples	BERT	0.50
16,452 samples	BERT	0.56
6687 samples	Keras embeddings	0.57
11,336 samples	Keras embeddings	**0.73**
16,452 samples	Keras embeddings	**0.80**

There are three words in the sentence with their connotations depending on each other. If instead of the word *seller*, there was *buyer*, the sentence would be about a *buyer market*, and together with the words *strong* and *dominant*, it would give the sentence a positive sentiment. Though, in this sentence, the words *strong* and *dominant* are used to describe a *seller market* and the sentiment is negative. While classifying the text, if we do not regard the dependencies of the adjectives (*strong*, *dominant*) and the noun (*seller*) that they are describing, the sentiment of the sentence may be misclassified. Such dependencies can be detected in an LSTM neural network.

Again, we experiment with the pre-trained embeddings as well as Keras embeddings. The embeddings are fed into an LSTM with 64 neurons, followed by a dropout layer and a 3-node dense output layer with softmax as the activation function. To prevent overfitting, the training is done with a 0.2 dropout rate, by which a random number of inputs are set to 0 at each step during the training time. Having trained the LSTM neural network with all embedding methods, the Keras embeddings again led to the best result, which is also the best result among all models that we have experimented with. Moreover, the accuracy on CBOW and GloVe embeddings was higher than what was achieved by the baseline SVM model. The results achieved by our LSTM models are summarised in Table 6.

In summary, when training the models presented here, we employed various feature extraction, selection and representation approaches leveraging multiple embedding methods and input vector dimensionalities. In terms of machine learning, we used both traditional machine learning and deep learning algorithms, experimenting with various hyper parameter values and dataset sizes. This type of experimenting enables us to see which problems (discussed in subsection B) can be addressed more precisely by which approach.

An illustration of the different stages of training and testing the deep learning models is presented in Figure 3.

V. RESULTS AND DISCUSSION

In this section, we first present and discuss the results of the models with multiple embedding techniques and classification algorithms. Then we take a detailed look at the best model while comparing it with a similar model in the domain and evaluate its robustness.

The focus of our study is on the effect of multiple factors, including feature embedding, the classification algorithm, and the training set on the accuracy of the SA model. We also compare the training times.

In almost all deep learning models as well as in the SVM model we observe better accuracy as we increase the size of the training set. It can be noted that this is not the case with the lexicon-based method, utilised for our first baseline model, as it is does not get trained.

TABLE 6. LSTM SA algorithms, configurations and results.

Dataset size	Feature representation	Acc
6,687 samples	CBOW	0.34
11,336 samples	CBOW	0.68
16,452 samples	CBOW	**0.73**
6,687 samples	GloVe	0.58
11,336 samples	GloVe	**0.71**
16,452 samples	GloVe	**0.77**
6,687 samples	BERT	0.38
11,336 samples	BERT	0.47
16,452 samples	BERT	0.59
6,687 samples	Keras embeddings	0.57
11,336 samples	Keras embeddings	**0.77**
16,452 samples	Keras embeddings	**0.84**

A. EMBEDDING TECHNIQUES

In the deep learning experiments, the embedding method can have a big impact on the accuracy of the model [25]. We used pre-trained CBOW, GloVe, and BERT embeddings as well as fine-tuned Keras embedding by adding an embedding layer to the neural networks. The pre-trained CBOW and BERT embeddings are trained on a financial corpus, which makes them suitable for SA in the financial markets' domain. The advantage of pre-trained embeddings is their training on large datasets, allowing them to recognize the contextual patterns and dependencies. Accordingly, embeddings such as CBOW and GloVe consider statistical co-occurrences while BERT embeddings focus on semantic co-occurrences. However, pre-trained embeddings are not based on labelled text, which in turn may raise challenges for specific domains SA. By experimenting with multiple models and different embeddings, we observed that the pre-trained embeddings do not perform better than the fine-tuned embeddings for financial markets SA. There are instances in domain-specific SA, in which some non-sentiment words (such as buy, plunge, return etc.) in general applications may have positive/negative connotations in the domain. Therefore, as an embedding method, which is based on the labelled financial corpus, the fine-tuned embedding led to better results in all deep learning experiments. As stated earlier in section IV-B, one of the biggest challenges for SA in the financial markets' corpus relates to different connotations for the same words. Depending on the context, these connotations can inversely change. In order to gain a better view of how different embeddings perform in such instances, we measured the output results of the best-performing classifier (LSTM) on the embeddings. Although for the most instances in the dataset there was a unanimous class labelled by the classifier, there were some instances like the ones in Table 7 that were classified with different labels in different embeddings, which in turn lead to the gap in the results by different embeddings. Owing to its domain-specific training and the labelled text, the fine-tuned embedding led to accurate classification in such tricky situations. As a result, it constantly leads to high accuracy in the models. Hence, it is demonstrated to be the preferred embedding method used in deep learning approaches for SA of financial markets corpus with both formal (news and articles) and informal (tweets and comments) text.

TABLE 7. Different outputs by embeddings in tricky instances.

Instance	Embedding	Class
Buying pressure** is still **high	CBOW	negative
	GloVe	positive
	BERT	neutral
	fine-tuned Keras embedding	positive
Selling pressure** is still **high	CBOW	negative
	GloVe	positive
	BERT	neutral
	fine-tuned Keras embedding	negative

Still, some good results were achieved by GloVe and CBOW with the GloVe embedding performing better than CBOW. The global frequency of words' cooccurrence in the corpus is information leveraged for training GloVe embedding, while for CBOW, it is regarded locally (neighbouring words). The better performance by GloVe embedding in our case indicates the improving role of global statistics when vectorizing the words. Same as CBOW and GloVe, the BERT embedding could not beat the fine-tuned embedding performance.

B. CLASSIFICATION MODELS

We experimented with three types of neural networks with the aim of finding the best neural network architecture for SA of text documents containing information about financial markets. Our LSTM model obtained the highest accuracy followed by the RNN model, both with the added embedding layer. Owing to its architecture, by preserving the sequential information of the input documents, LSTM proved to outperform other SA models. Apart from LSTM, it is worth noting that MLP could obtain higher accuracy than other algorithms on the pre-trained CBOW and GloVe embeddings. That is, MLP could be a better choice than CNN and RNN when working with these pre-trained embeddings. Training time is another factor when choosing a ML algorithm. As shown in Table 8, apart from the lexicon-based model, which does not have any learning stage, the shortest training time is the one of CNN, thanks to its convolutional layer. Therefore, although LSTM, RNN, and MLP achieve higher accuracies, we can say that CNN presents a better trade-off between training time and accuracy. A summary of all the results is presented in Table 8.

C. ROBUSTNESS OF THE LSTM MODEL

Finally, we did further evaluation of the LSTM models with the fine-tuned embeddings, the results of which are presented in Table 9. We evaluated three different models. The model

TABLE 8. Various embedding-classifier configurations, experimented for financial markets SA.

No	Dataset size	Feature extraction/selection	Feature representation		Classification	Acc	Training time (sec)
1	6,687 samples	Bag of words/Lexicon	Count scoring		Scoring	0.49	t_{max}<1
2	11,336 samples	Bag of words/Lexicon	Count scoring		Scoring	0.44	
3	6,687 samples	Bag of words	Pre-trained	CBOW	SVM	0.48	t_{max}=6531
4	11,336 samples	Bag of words	Pre-trained	CBOW	SVM	0.57	
5	16,452 samples	Bag of words	Pre-trained	CBOW	SVM	0.63	
6	6,687 samples	Bag of words	Pre-trained	Glove	SVM	0.33	
7	11,336 samples	Bag of words	Pre-trained	Glove	SVM	0.59	
8	16,452 samples	Bag of words	Pre-trained	Glove	SVM	0.63	
9	6,687 samples	Bag of words	Pre-trained	BERT	SVM	0.61	
10	11,336 samples	Bag of words	Pre-trained	BERT	SVM	0.67	
11	16,452 samples	Bag of words	Pre-trained	BERT	SVM	0.69	
12	6,687 samples	Deep learning	Pre-trained	CBOW	MLP	0.52	t_{max}=2000
13	11,336 samples	Deep learning	Pre-trained	CBOW	MLP	0.60	
14	16,452 samples	Deep learning	Pre-trained	CBOW	MLP	0.67	
15	6,687 samples	Deep learning	Pre-trained	GloVe	MLP	0.56	
16	11,336 samples	Deep learning	Pre-trained	GloVe	MLP	0.65	
17	16,452 samples	Deep learning	Pre-trained	GloVe	MLP	**0.73**	
18	6,687 samples	Deep learning	Pre-trained	BERT	MLP	0.59	
19	11,336 samples	Deep learning	Pre-trained	BERT	MLP	0.61	
20	16,452 samples	Deep learning	Pre-trained	BERT	MLP	0.66	
21	6,687 samples	Deep learning	Fine-tuned	Keras embeddings	MLP	0.52	
22	11,336 samples	Deep learning	Fine-tuned	Keras embeddings	MLP	**0.70**	
23	16,452 samples	Deep learning	Fine-tuned	Keras embeddings	MLP	**0.75**	
24	6,687 samples	Deep learning	Pre-trained	CBOW	CNN	0.33	t_{max}=215
25	11,336 samples	Deep learning	Pre-trained	CBOW	CNN	0.55	
26	16,452 samples	Deep learning	Pre-trained	CBOW	CNN	0.57	
27	6,687 samples	Deep learning	Pre-trained	GloVe	CNN	0.49	
28	11,336 samples	Deep learning	Pre-trained	GloVe	CNN	0.56	
29	16,452 samples	Deep learning	Pre-trained	GloVe	CNN	0.62	
30	6,687 samples	Deep learning	Pre-trained	BERT	CNN	0.63	
31	11,336 samples	Deep learning	Pre-trained	BERT	CNN	0.63	
32	16,452 samples	Deep learning	Pre-trained	BERT	CNN	0.63	
33	6,687 samples	Deep learning	Fine-tuned	Keras embeddings	CNN	0.52	
34	11,336 samples	Deep learning	Fine-tuned	Keras embeddings	CNN	0.68	
35	16,452 samples	Deep learning	Fine-tuned	Keras embeddings	CNN	**0.72**	
36	6,687 tweets	Deep learning	Pre-trained	CBOW	RNN	0.33	t_{max}=1847
37	11,336 samples	Deep learning	Pre-trained	CBOW	RNN	0.44	
38	16,452 samples	Deep learning	Pre-trained	CBOW	RNN	0.57	
39	6,687 tweets	Deep learning	Pre-trained	GloVe	RNN	0.33	
40	11,336 samples	Deep learning	Pre-trained	GloVe	RNN	0.44	
41	16,452 samples	Deep learning	Pre-trained	GloVe	RNN	0.57	
42	6,687 samples	Deep learning	Pre-trained	BERT	RNN	0.40	
43	11,336 samples	Deep learning	Pre-trained	BERT	RNN	0.50	
44	16,452 samples	Deep learning	Pre-trained	BERT	RNN	0.56	
45	6,687 samples	Deep learning	Fine-tuned	Keras embeddings	RNN	0.57	
46	11,336 samples	Deep learning	Fine-tuned	Keras embeddings	RNN	**0.73**	
47	16,452 samples	Deep learning	Fine-tuned	Keras embeddings	RNN	**0.80**	
48	6,687 samples	Deep learning	Pre-trained	CBOW	LSTM	0.34	t_{max}=2363
49	11,336 samples	Deep learning	Pre-trained	CBOW	LSTM	0.68	
50	16,452 samples	Deep learning	Pre-trained	CBOW	LSTM	**0.73**	
51	6,687 samples	Deep learning	Pre-trained	GloVe	LSTM	0.58	
52	11,336 samples	Deep learning	Pre-trained	GloVe	LSTM	**0.71**	
53	16,452 samples	Deep learning	Pre-trained	GloVe	LSTM	**0.77**	
54	6,687 samples	Deep learning	Pre-trained	BERT	LSTM	0.38	
55	11,336 samples	Deep learning	Pre-trained	BERT	LSTM	0.47	
56	16,452 samples	Deep learning	Pre-trained	BERT	LSTM	0.59	
57	6,687 samples	Deep learning	Fine-tuned	Keras embeddings	LSTM	0.57	
58	11,336 samples	Deep learning	Fine-tuned	Keras embeddings	LSTM	**0.77**	
59	16,452 samples	Deep learning	Fine-tuned	Keras embeddings	LSTM	**0.84**	

TABLE 9. Performance metrics of LSTM-Keras embedding models on datasets with different number of samples.

				Results									
					Recall			Precision			F1-score		
LSTM	Samples	Train	Test	Acc	pos	neg	neu	pos	neg	neu	pos	neg	neu
1	11,336	9,219	2,117	0.77	0.74	0.76	0.80	0.79	0.77	0.76	0.76	0.76	0.78
2	16,452	13,161	3,291	0.84	0.78	0.72	0.89	0.82	0.75	0.86	0.80	0.73	0.87
3	16,452	13,161	1,668	0.80	0.74	0.80	0.86	0.87	0.87	0.70	0.80	0.83	0.77

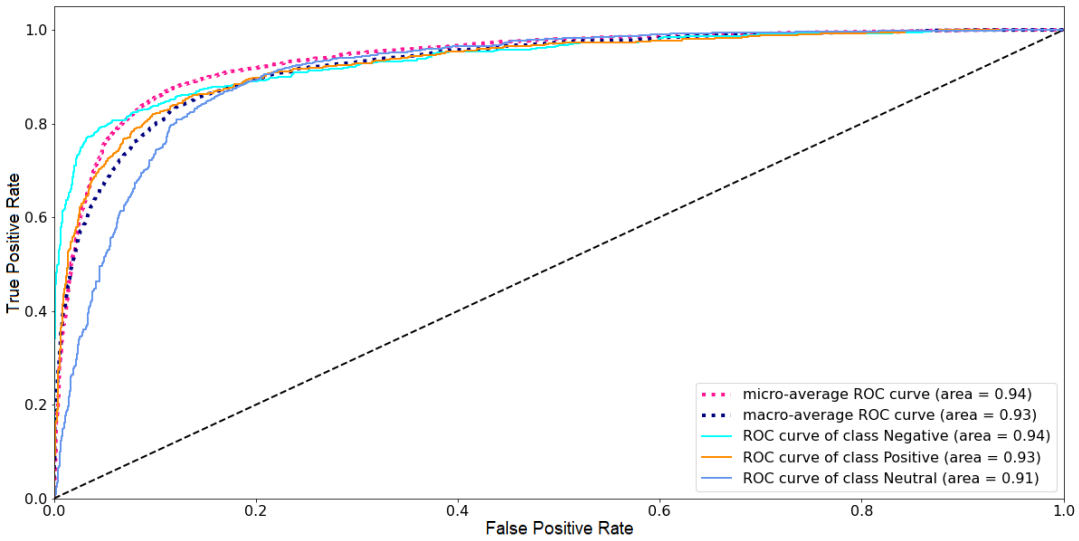

FIGURE 4. ROC curve for different classes by LSTM2.

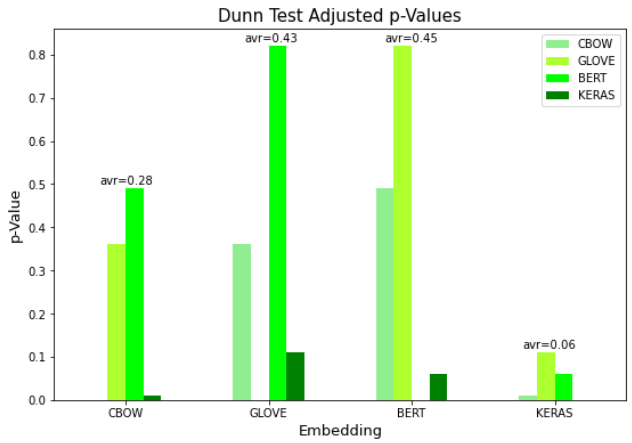

FIGURE 5. Obtained p-values for significance of differences by Dunn test.

trained on 9,219 documents (LSTM1) has accuracy of 77% and an F1-score in the interval [0.76, 0.78].

As we increased the size of the training set to 13,161 documents (LSTM2) the accuracy raised to 83.9% and the F1-score remained high in the interval [0.73, 0.87]. However, by increasing the size of the training dataset and keeping it balanced, the remaining set of samples used for test became unbalanced. Hence, in order to balance the test set, we removed some samples from it bringing it down to 1,668 text documents (LSTM3). Having tested the model on the test dataset with evenly distributed labels, still a high accuracy of 80% was achieved, which is indicative of the model's unbiased performance. The F1-score also remained high.

Furthermore, a proper metric for classifiers on imbalanced data is receiver operating characteristic or ROC curve which is not biased towards majority or minority classes [26]. In LSTM2, the number of "neutral" labels is twice larger than the "positive" and "negative" labels. Hence, ROC analysis was performed on the results to evaluate the model's performance on each specific label.

The ROC of a classifier with no discriminative power is on or below the diagonal line, while the ROC of a classifier with a good discriminative power bends towards the top left corner of the plot and has a big area under the curve (AUC). Therefore, the higher the AUC, the better the model can distinguish different classes. As shown in Figure 4, LSTM2 has AUC well above 0.9 for all labels. The micro/macro average ROC curves represent the cumulative performance of the model over all three classes. Since the test dataset is imbalanced, their properties are indicative for the quality of the model. As shown in Figure 4, the model performs robustly and is unbiased in classifying all labels. It is also worth noting that the LSTM model on the fine-tuned embeddings achieved above average accuracy compared to other embeddings and models with just two thirds of the dataset, i.e., the minimum

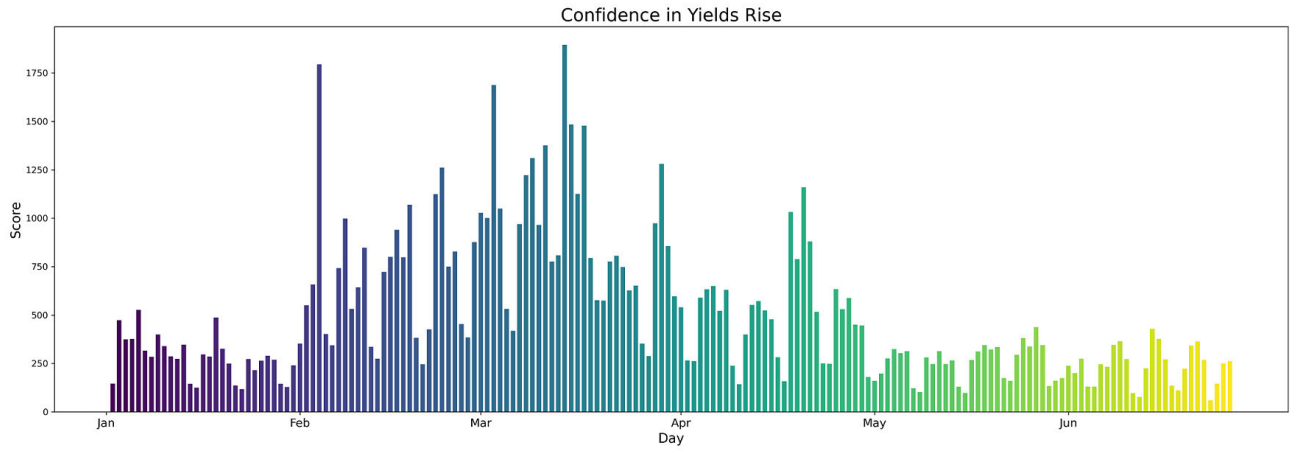

FIGURE 6. Sentiment score trends towards US10Y yields.

number required to achieve above average (77%) accuracy in the model is around 11,000 data rows.

D. STATISTICAL ANALYSIS

In order to further evaluate the obtained results, we measured the statistical difference in the set of accuracies of the various models that we experimented with. Our sample set consists of 16 observations, which are the average accuracies resulted from applying four types of embeddings on four types of algorithms as SVM, MLP, CNN, and RNN (LSTM included) presented in Table 8. Two types of tests as pre-hoc and post-hoc were employed to measure the significance of difference among the results. One common pre-hoc test is Friedman test [27], which is a non-parametric test to assess significant differences between the results. By doing Friedman test, the associated p-value with the test statistics can be determined. The p-value represents the probability of the observed (or more extreme) differences occurring among the groups if the null hypothesis (no difference among the groups) is true. In other words, a small p-value suggests the evidence that the results are significantly different, while a high p-value rejects that significance.

After a significant difference is obtained by Friedman test, a post-hoc test was used for pairwise comparison between the groups. There are different post-hoc tests based on rank sums such as Dunn [28], Nemenyi [29], and Games-Howell [30] that can be used. However, selecting the test depends on the specific context and requirements. In our study, as the differences between the rankings do not follow a normal distribution, the suitable test is Dunn test. Unlike other tests, Dunn test does not rely on normal distribution assumption and is robust against violation of normal distribution. Therefore, by doing Dunn test, the significance of pairwise differences will be determined based on the adjusted p-values.

It should be noted that the appropriate p-value depends on the sample size. In other words, while a larger sample size needs a lower p-value, a higher p-value can be chosen for a smaller sample size [31]. Our sample set consists of 16 observations. Thus, as the sample size is relatively low, we chose p-value of 0.1 for measuring significance of difference.

By applying Friedman [27] test on the observed results, we obtained a p-value of 0.07. The obtained p-value is less than 0.1 and suggests that the probability of observing such (or more extreme) differences in the results, if the null hypothesis is true, will be 7%, which is relatively significant and suggestive of evidence against the null hypothesis. This p-value leads us to the post-hoc test to measure pairwise differences between the observations. Based on the results of Dunn test, illustrated in Figures 5, it can be seen that the average p-value for the fine-tuned Keras embedding group is less than for other groups, reflecting the statistical difference of the results that were obtained by this embedding compared to other embeddings' results. Therefore, we can conclude that the obtained results by applying fine-tuned Keras embedding are significantly different than other embeddings' results.

VI. APPLICATION AND INSIGHTS

By applying the model on myriads of texts published as tweets, news headlines, reports, etc., every day, the output will be the overall sentiment towards various entities in the markets including brands, currencies, stock indexes, etc. Therefore, the model's output on historical data will be collective sentiments, represented as sentiment trends in the markets. As an illustration in Figure 6, the sentiments extracted from more than 120,000 tweets over the period of January-June 2023 are represented. The tweets have been specifically towards United States 10-year Treasury bills' yields (US10Y) and how individuals/experts were thinking about US10Y during this period. Consequently, the overall sentiment in each day with the underlying trend (ascending/descending) is presented to the investors as an added feature to technical and fundamental analysis. Put simply, the sentiment trends will reflect how confident market participants are towards yield increase in the future, which

will significantly affect investment options. Therefore, the result will be an added value while decision making about future market movements in light of sentiments towards a specific entity (here US10Y).

By studying the model's output, the investors get educated about past, present, and emerging interests about particular assets in the financial markets. The results can provide insights into the mood and emotions of market participants, which can affect the behavior of financial instruments. The application of the model in the financial markets can be described as follows:

- Predicting stock prices: the output can be used to analyze the tone of news articles, social media posts, and other sources of market sentiment. By tracking the sentiment of these sources, analysts are more informed while predicting future stock prices.
- Identifying market trends: By analyzing the sentiment of social media posts and other sources, traders can identify emerging market trends and take positions accordingly. This can help them to stay ahead of the curve and make profitable trades.
- Monitoring brand reputation: Companies can use sentiment analysis to monitor their brand reputation and track the sentiment of customer feedback. This can help them to identify areas for improvement and respond quickly to negative feedbacks.
- Evaluating investment opportunities: Investors can use the model to evaluate the sentiment surrounding potential investment opportunities. This can help them to make more informed decisions and avoid investing in companies with a negative sentiment.

Overall, the SA model can help investors, industries, and traders to make more informed decisions with an added value as sentiment to their framework.

VII. CONCLUSION AND FUTURE WORKS

Although there have been studies comparing different deep learning algorithms for sentiment analysis, the comparison has been mostly limited to neural network architectures, while the embedding part has been overlooked. Also, many studies have been done either only on news headlines or only on micro-blogging data such as tweets. This paper presents a broad comparative study of multiple deep-learning algorithms for SA in the financial markets' domain with respect to different parameters including embedding and classification algorithms as well as corpus. By a diversified corpus as dataset, we included both formal (news headlines) and informal (tweets) text documents in order to evaluate the robustness of the SA models. Aiming for a robust performance in the models, we developed an optimized embedding, which leads to the best performance in all models. Same as other pre-trained embeddings, BERT could not beat the fine-tuned embedding in the domain as it is trained on general corpus without labelled text. Therefore, the best model is the LSTM with the fine-tuned embedding layer added as the first layer to it. As a result, the pair of LSTM and fine-tuned embedding outperformed all other models in the study with an accuracy of 83.9% and an F1-score in the interval [0.73-0.87]. The findings of the research imply the need for specific embeddings in the financial domain. Apart from the financial domain, the proposed method can be applied to other domains as well, provided that the model is trained on the relevant dataset. The main difference of the proposed method compared to general models is its emphasis on the embedding layer. By applying this method on a specific domain, the words are embedded more precisely with regard to the specific meanings they may have in the context. Therefore, by saving the words in a high-dimensional embedding vector and leveraging the dimensions by LSTM neural networks, the words' meanings and dependencies in the target domain will be more accurately preserved, which in turn leads to more domain-specific (rather than general) classification. In order to apply the proposed method on other domains, the appropriate pre-processing parameters such as embedding dimensions and hyper parameters such as LSTM layers, need to be considered with regard to the domain's specifics.

Although there have been good results reported for pre-trained embeddings, such embeddings are not developed for specific domains but the general applications. In other words, the performance in general domains cannot be indicative of the same results for domain-specific applications. The dependency between the words is another determinant while measuring sentiments in the financial corpus. LSTM neural networks proved to retain such dependencies while being applied on the fine-tuned embeddings. As a result, context and dependencies were both regarded by the embedding and classification algorithm leading to the best performance.

Given the scarcity of large labelled datasets in the financial markets' domain, in the next stage we intend to apply our best LSTM model with a human-in-the-loop approach on a large dataset of text documents related to financial markets to achieve a trustworthy large labelled dataset that is essential for sentiment analysis tasks in the financial markets. Also, with the rise of most recent large language models (LLMs) such as ChatGPT and the noticeable performance of our proposed model, we intend to compare it as a domain-specific language model with other LLMs while using far less resources as data and hardware requirements.

The presented model proves valuable specifically for SA in the context of financial markets. Therefore, the studied application is on this domain's corpus. However, we believe that the proposed methodology is not limited to this domain and can be successfully applied in training models for related problems, such as election poll prediction.

REFERENCES

[1] B. M. Barber and T. Odean, "The internet and the investor," *J. Econ. Perspect.*, vol. 15, no. 1, pp. 41–54, Feb. 2001, doi: 10.1257/jep.15.1.41.

[2] X. Man, T. Luo, and J. Lin, "Financial sentiment Analysis(FSA): A survey," in *Proc. IEEE Int. Conf. Ind. Cyber Phys. Syst. (ICPS)*, May 2019, pp. 617–622, doi: 10.1109/ICPHYS.2019.8780312.

[3] S. Yildirim, D. Jothimani, C. Kavaklioglu, and A. Basar, "Deep learning approaches for sentiment analysis on financial microblog dataset," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5581–5584, doi: 10.1109/BigData47090.2019.9006056.
[4] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An on-line lexical database," *Int. J. Lexicography*, vol. 3, no. 4, pp. 235–244, 1990, doi: 10.1093/ijl/3.4.235.
[5] T. Mikolov. K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781
[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.
[7] Z. Y. Y. Ren, M. Zhang, and D. Ji, "Improving Twitter sentiment classification using topic-enriched multi-prototype word embeddings," in *Proc. 13th AAAI Conf. Artif. Intell.* Phoenix, AZ, USA: AAAI Press, 2016, pp. 3038–3044.
[8] M. Vicari and M. Gaspari, "Analysis of news sentiments using natural language processing and deep learning," *AI Soc.*, vol. 36, no. 3, pp. 931–937, Sep. 2021, doi: 10.1007/s00146-020-01111-x.
[9] S. Sohangir, D. Wang, A. Pomeranets, and T. M. Khoshgoftaar, "Big data: Deep learning for financial sentiment analysis," *J. Big Data*, vol. 5, no. 1, pp. 1–25, Dec. 2018, doi: 10.1186/s40537-017-0111-6.
[10] C. M. J. Devlin, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language Understanding," 2019, *arXiv:1810.04805*.
[11] N. C. Dang, M. N. Moreno-García, and F. De la Prieta, "Sentiment analysis based on deep learning: A comparative study," *Electronics*, vol. 9, no. 3, p. 483, Mar. 2020, doi: 10.3390/electronics9030483.
[12] H. Jangid, S. Singhal, R. R. Shah, and R. Zimmermann, "Aspect-based financial sentiment analysis using deep learning," in *Proc. Companion Web Conf. Web Conf. (WWW)*, 2018, pp. 1961–1966, doi: 10.1145/3184558.3191827.
[13] W. Souma, I. Vodenska, and H. Aoyama, "Enhanced news sentiment analysis using deep learning methods," *J. Comput. Social Sci.*, vol. 2, no. 1, pp. 33–46, Jan. 2019, doi: 10.1007/s42001-019-00035-x.
[14] D. Araci, "FinBERT: Financial sentiment analysis with pre-trained language models," 2019, *arXiv:1908.10063*.
[15] I. Chaturvedi, E. Cambria, R. E. Welsch, and F. Herrera, "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges," *Inf. Fusion*, vol. 44, pp. 65–77, Nov. 2018, doi: 10.1016/j.inffus.2017.12.006.
[16] S. A. N. Tabari, T. Peddi, M. Hadzikadic, and W. Zadrozny, "A comparison of neural network methods for accurate sentiment analysis of stock market tweets," in *Proc. Workshop Mining Data Financial Appl.*, vol. 11054. Cham, Switzerland: Springer, 2019, pp. 51–65.
[17] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 4, pp. 782–796, Apr. 2014, doi: 10.1002/asi.23062.
[18] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguistics*, vol. 37, no. 2, pp. 267–307, Jun. 2011, doi: 10.1162/COLI_a_00049.
[19] P. D. Turney, "Thumbs up or thumbs down: Semantic orientation applied to unsupervised classification of reviews," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 417–424, doi: 10.3115/1073083.1073153.
[20] L. L. P. Bo and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning," in *Empirical Methods NLP*. Philadelphia, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86, doi: 10.3115/1118693.1118704.
[21] M. Yekrangi and N. Abdolvand, "Financial markets sentiment analysis: Developing a specialized lexicon," *J. Intell. Inf. Syst.*, vol. 57, no. 1, pp. 127–146, Aug. 2021, doi: 10.1007/s10844-020-00630-9.
[22] S.-M. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proc. 20th Int. Conf. Comput. Linguistics (COLING)*, 2004, p. 1367, doi: 10.3115/1220355.1220555.
[23] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 496–509, Feb. 2016, doi: 10.1109/TKDE.2015.2489653.
[24] Y. Choi, Y. Kim, and S.-H. Myaeng, "Domain-specific sentiment analysis using contextual feature generation," in *Proc. 1st Int. CIKM Workshop Topic-Sentiment Anal. Mass Opinion*, Nov. 2009, pp. 37–44, doi: 10.1145/1651461.1651469.
[25] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 23, Dec. 2021, Art. no. e5909, doi: 10.1002/cpe.5909.
[26] N. Japkowicz, "Assessment metrics for imbalanced learning," in *Imbalanced Learning: Foundations, Algorithms, and Applications*, Y. M. H. He, Ed. Hoboken, NJ, USA: Wiley, 2013, pp. 187–206.
[27] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, Dec. 1937, doi: 10.1080/01621459.1937.10503522.
[28] O. J. Dunn, "Multiple comparisons using rank sums," *Technometrics*, vol. 6, no. 3, pp. 241–252, 1964, doi: 10.2307/1266041.
[29] P. B. Nemenyi, *Distribution-Free Multiple Comparisons*. Princeton, NJ, USA: Princeton Univ., 1963.
[30] P. A. Games and J. F. Howell, "Pairwise multiple comparison procedures with unequal N's and/or variances: A Monte Carlo study," *J. Educ. Statist.*, vol. 1, no. 2, pp. 113–125, 1976.
[31] E. Gómez-de-Mariscal, V. Guerrero, A. Sneider, H. Jayatilaka, J. M. Phillip, D. Wirtz, and A. Muñoz-Barrutia, "Use of the *p*-values as a size-dependent function to address practical differences when analyzing large datasets," *Sci. Rep.*, vol. 11, Oct. 2021, Art. no. 20942, doi: 10.1038/s41598-021-00199-5.

MEHDI YEKRANGI received the B.S. degree in information technology from the Iran University of Science and Technology, in 2015, and the M.S. degree in e-commerce from Islamic Azad University, in 2019. He is currently pursuing the Ph.D. degree with the Big Data and Analytics Research Group (BDARG), University of Limerick. His research interests include natural language processing, sentiment analysis, quantitative analysis, and financial markets modeling. He is also a member of the Big Data and Analytics Research Group (BDARG), University of Limerick.

NIKOLA S. NIKOLOV received the B.Sc. degree from Sofia University in 1995, and the Ph.D. degree from the University of Limerick, in 2002. He is currently a Lecturer with the Department of Computer Science and Information Systems and the Co-Head of the Big Data and Analytics Research Group, University of Limerick. He was a principle investigator on research projects funded by the Science Foundation Ireland and the Irish Research Council. He is the author of more than 60 research articles in network visualization and collaborative filtering. He is a member of the IEEE Technical Committee on Visual Analytics and Communication.