Computational Economics
https://doi.org/10.1007/s10614-025-10901-8

Enhancing Sentiment Analysis in Stock Market Tweets Through BERT-Based Knowledge Transfer

Emre Cicekyurt[1] · **Gokhan Bakal**[1]

Accepted: 9 February 2025

Abstract

One of the widely studied text classification efforts is sentiment analysis. It is a specific examination involving natural language processing and machine learning methods to understand semantic orientation from textual data. Working social media posts, such as tweets, for sentiment analysis, is quite common among researchers due to the speed of information dissemination. In this regard, forecasting stock market tweets is a widely studied research topic. Some studies have revealed a strong connection between sentiment and stock market performance, while others have not found any notable associations. The proposed work shows two distinct approaches to sentiment analysis over the stock market tweets. The first approach employs traditional machine learning algorithms, including logistic regression, random forest, and XGBoost. The second approach constructs deep learning (as a subfield of machine learning) models using LSTM and CNN algorithms to classify the test instances into positive, negative, or neutral classes through ten randomly shuffled data splits. In this study, the labeled data size is gradually increased utilizing a pre-trained model, FinBERT. It is exclusively employed to label unlabeled data instances to integrate them into the experiments. The goal is to monitor the effect of the additional newly-labeled examples on the sentiment analysis performance. The experiments showed that the average F1-score improved by **20%** for the deep learning models and **17%** for the machine learning models. In the end, the paper reveals a strong positive correlation between training data size and the classification performance of the experimental approaches.

Keywords Financial tweets · Text mining · Machine & deep learning

✉ Gokhan Bakal
gokhan.bakal@agu.edu.tr

Emre Cicekyurt
emre.cicekyurt@agu.edu.tr

[1] Department of Computer Engineering, Abdullah Gul University, Erkilet Bulvd., 38080 Kayseri, Turkey

Published online: 26 February 2025

1 Introduction

Text classification is an essential task since it enables the extraction of relevant information from vast amounts of text and can be applied to various domains for distinct purposes, including sentiment analysis, spam detection, and topic modeling. Sentiment analysis (a.k.a. opinion mining) is the identification and extraction process of personal information from textual data input. The analysis includes natural language processing and text mining techniques to determine the attitudes, opinions, and emotions of the text's author(s) intensively (Meng et al., 2023; Sungur and Bakal, 2025). Recently, there has been a growing interest in using sentiment analysis to predict stock market movements. The rapidly emerging use of social media platforms, such as Twitter, has provided a vast amount of unstructured data that can be analyzed to gain insights into the public opinion of a particular stock or market item. Twitter is one of the most widely operated social media platforms globally, with over 330 million monthly active users. Twitter users generate an enormous volume of textual data called tweets, which can contain valuable information about users' opinions and sentiments. It has become a popular data source for researchers interested in examining public opinion on various topics (Sohangir et al, 2018; Sun et al, 2023; Erkantarci and Bakal, 2023; Bakal and Abar, 2021), including stock market movements.

Sentiment analysis on stock market tweets is critically significant because it can provide valuable insights into market trends and investor sentiment. The stock market is a complex and dynamic system in which conventional research studies, such as rule-based and fundamental analysis, may not always provide a complete comprehensive market appraisal. In this sense, some researchers examined the influence of operating distinct financial resources to improve forecasting accuracy through deep learning approaches (Day and Lee, 2016). However, sentiment analysis of tweets can provide additional information about the market directions that can help investors make more concrete decisions (Tumasjan et al., 2010). Furthermore, the major obstacle to working on social media datasets in sentiment analysis is the need for available labeled instances to build supervised learning models (Aroyehun and Gelbukh, 2018).

In this experimental research, **our principal hypothesis is that a cumulatively increasing number of newly labeled examples (annotated by a pre-trained BERT-based model) incorporated into our experimental models enhanced classification performances regardless of the models built by distinct algorithms**. Thus, even though there is no human annotator, employing a pre-trained model as an annotator can drastically boost classification performance. Additionally, we demonstrated that we could rely on the power of the pre-trained models for at least labeling a gigantic volume of unlabeled textual examples for further research directions since labeling large volumes of unlabeled instances by human annotators is almost impractical.

The fundamental contributions of this study are listed below:

- The dataset employed in constructing the model underwent scrutiny to substantiate the advantages of increasing information flow.
- The classification performances of distinct traditional machine learning and deep learning models were reported for comparative experiments.
- The effect of the state-of-the-art pre-trained Fin-BERT (Financial-Bidirectional Encoder Representations from Transformers) model's prediction performance was gradually associated with the overall performance of the proposed approach.

The rest of the paper continues as follows. Section 2 briefly mentions relevant literature, while Sect. 3 describes the dataset used in the study. Section 4 presents the methodologies used during the model composition with the experimental details. Section 5 shows the results and evaluates the model performances. Eventually, Sect. 6 concludes the proposed work and provides a general summary, while Sect. 7 mentions some limitations and potential future works.

2 Related Efforts

Numerous examinations have been conducted to investigate the associations between the sentiment analysis of tweets and stock market performances. The study performed by Bollen et al. (2011); Khafaga et al. (2023) disclosed that Twitter posts could be used to understand the encoded sentiment and to predict stock market changes with an accuracy of 86%, where the dataset used consisted of over 2 million tweets and a subset of Dow Jones Industrial Average (DJIA) stocks. Technically, the authors employed a lexicon-based approach, where they manually assigned a sentiment score to each word in the tweets and computed an overall sentiment score to identify the final sentiment category. Another study conducted by Tumasjan et al. (2010) utilized a similar approach to analyze the sentiment of tweets about political parties during the German federal election. The authors found that the sentiment of tweets about a particular party was positively correlated with the election results for that party, while the sentiment of tweets about the overall election was positively correlated with the election results (Tun and Khaing, 2023). In contrast, a study by Liu (2012) found that sentiment analysis on tweets about stocks was not a reliable predictor of stock market performance. The authors used a tweet dataset and stock prices from 2012 to 2013 and found no significant correlation between the sentiment of tweets and stock prices. They addressed this lack of correlation and entailed that many tweets about stocks were not informative but were either promotional or irrelevant to the topic. As a critical study, Loughran and McDonald (2011) have explored the role of textual information in corporate disclosures by highlighting the limitations of relying solely on negative word counts to assess the tone of a financial text. They found that commonly used negative word lists developed for other disciplines, such as the Harvard Dictionary, often misclassify common words in financial contexts. A more recent study by Gupta and Chen (2020) utilized a sentiment analysis experiment combination of machine learning techniques and Term Frequency-Inverse Document Frequency (TF-IDF) to predict stock market price differences happening periodically. The authors used a dataset containing nine-month stock tweets and daily stock data and identified that their model could predict stock price movements with an accuracy range between 75% and 85%. Plus, they revealed that the sentiment of tweets was a significant predictor of stock price movements (Bacco et al., 2024; Gandhudi et al., 2024). Yet another relevant effort employed by Antweiler and Frank (2004) examined the influence of Internet stock message boards on market movements, explicitly focusing on the 45 companies in the Dow Jones Industrial Average and the Dow Jones Internet Index. They analyzed over 1.5 million messages posted on Yahoo! Finance and Raging Bull, using computational linguistics methods to measure the bullishness level. Their study compared this information to news coverage in the Wall Street Journal, which served as a control group. The key finding of their research was that stock messages, particularly those expressing bullish sentiment, can help predict market volatility. While the effect on stock returns was statistically significant, it was economically small. This situation suggests that the impact of message board sentiment on stock prices may be limited, but it does play a role in market dynamics. Interestingly, their analysis also revealed a correlation between disagreement among the posted messages and increased trading volume. This finding aligns with the notion that information asymmetry and disagreement can contribute to higher volatility in financial markets. As a fact, financial texts often exhibit unique linguistic patterns that complicate sentiment analysis; for instance, financial news frequently avoids negative language, preferring to negate favorable terms. This tendency, known as 'hedging,' can make it difficult for traditional sentiment analysis models to capture the true sentiment expressed in financial texts accurately. Studies (Antweiler and Frank, 2004; Akbiyik et al., 2023) have explored this phenomenon and highlighted the need for specialized approaches to sentiment analysis in financial contexts.

Generally, the previous studies indicate that sentiment analysis on tweets can be a practical approach for predicting stock market movements despite a few contradictive outcomes. Besides, some investigations have found a strong correlation between sentiment and stock market performance, while others have not encountered any significant correlation. Among these studies, the critical factors affecting the consequences, such as the dataset dealt with, the method utilized, and the specific period selected, come into prominence.

3 Dataset & Data Preprocessing

Between April 9th and July 16th, 2020, 943,672 tweets were obtained using hashtags such as #SPX500, references to the top 25 companies in the S&P 500 index, and #stocks by Taborda et al. (2021). Among these tweets, 1300 were annotated by manual efforts and labeled as positive (529), neutral (349), or negative (422). The data distribution is illustrated in Fig. 1.

The data instances were obtained using the Twitter REST API search (Kumar et al., 2014), specifically with the "`Tweepy`" Python package (Roesslein, 2020), which simplifies the process of interacting with the API for developers. The API only allows access to tweets from the past seven days and has the option to filter tweets by language. The tweets used in this study were filtered to include only those

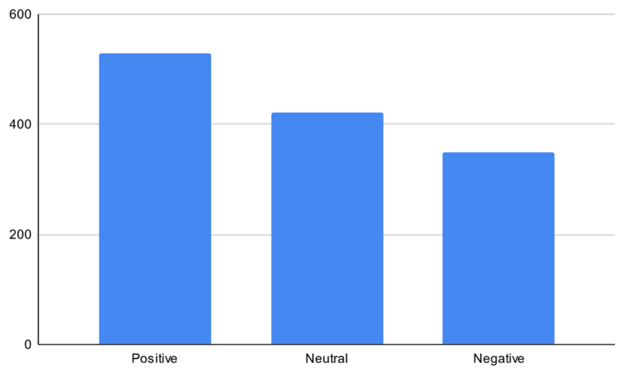

Fig. 1 Number of instances per class type

written in English. The data collection was generated by searching Twitter with the hashtags as listed in Table 1. Due to the large amount of data obtained, only the tweets' content and the date they were created were retained for the analysis (Taborda et al., 2021).

It is important to note that the analysis period includes the COVID-19 pandemic, a period of significant global uncertainty and market volatility. During this time, individuals and investors may have relied more heavily on information from media channels like Twitter for insights and guidance. This increased reliance on social media information during uncertain times could have influenced the results of our study. Therefore, it is crucial to interpret our findings in the context of this specific period, acknowledging the potential impact of the pandemic on investor sentiment and market dynamics

4 Methods

Sentiment analysis (Medhat et al., 2014) is a technique intensively utilizing natural language processing, text mining, and computational linguistics to determine and extract personal information/orientation from source data. This approach has become increasingly significant in various fields, such as movies, news, articles,

Table 1 Keywords used in twitter data collection

SPX500	#SP500	SPX500	SP500
$SPX	#stocks	$MSFT	$AAPL
$AMZN	$FB	$BBRK.B	$GOOG
$JNJ	$JPM	$V	$PG
$MA	$INTC	$UNH	$BAC
$T	$HD	$XOM	$DIS
$VZ	$KO	$MRK	$CMCSA
$CVX	$PEP	$PFE	

and social media posts. Considering the finance and economy disciplines, sentiment analysis is applied to stock market tweets to analyze whether they are positive, negative, or neutral. The process of sentiment analysis applications can be evaluated by two approaches; traditional machine learning and deep learning models. N-gram representations of unigram, bigram, and trigram and their combinations are widely employed while applying these methodologies (Cavnar et al., 1994). The overall execution flow of the study is shown in Fig. 2.

4.1 Data Preprocessing

In this study, we consider financial tweets to be unstructured text; this is because they are not organized in a predefined format or schema. These tweets are essentially free-form text, with varying lengths, structures, and levels of detail. This lack of a predefined structure makes it challenging to analyze these tweets using traditional methods, such as relational databases. Instead, we employ natural language processing (NLP) techniques, including text mining and machine learning, to extract meaningful features from this raw text.

Data preprocessing in natural language processing is crucial because it helps to improve the model's accuracy by cleaning the data before the analysis. This condition is because the raw textual data often contains noisy items, such as special

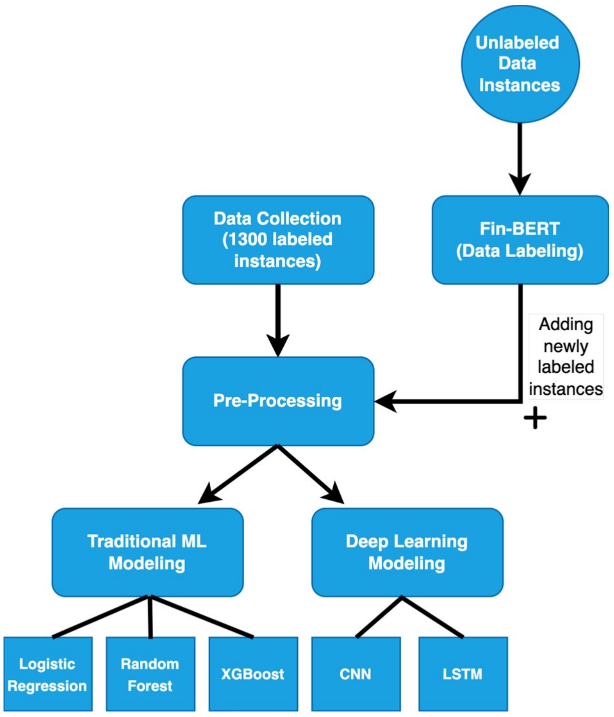

Fig. 2 Overall graphical representation of the experiments

characters, numbers, and stop-words, which can negatively affect the performance. Thus, preprocessing helps remove these noises by reforming the text such that the input is more suitable for analysis. The preprocessing steps consist of tokenization, lemmatization, and stop-word removal. These operations drastically improve the accuracy and efficiency of NLP models (Silva and Ribeiro, 2003; Al-Shargabi et al., 2011).

The labeled dataset used in this study comprises 1,300 tweets and requires additional cleaning operations to remove any unnecessary information, including acronyms, emoticons, pictures, and URLs. Due to the dataset's emoji ambiguity and the context-dependent meanings, we removed the emojis before conducting the analysis to mitigate modeling challenges. Additionally, instances of three or more consecutive letters are replaced with two letters. Then, the tweets are lemmatized using the WordNetLemmatizer function of the Natural Language Toolkit (NLTK) library (Loper and Bird, 2002), which transforms the words into their base form. After the cleaning phase, the dataset is divided into three subsets: training, testing, and validation, as in the proportions of 70%, 20%, and 10%, respectively. Additionally, ten distinct shuffled versions of the training, testing and validation datasets are composed to ensure the robustness of the study.

4.2 Traditional Machine Learning Modeling

Data scientists use different machine learning (ML) algorithms for classification and regression purposes. In this study, logistic regression, random forest, and xgboost algorithms are employed to build traditional machine learning models that predict tweet instances' sentiment labels. An ML pipeline that includes a TF-IDF vectorizer is built to create a model via the vectorizer converting textual data into a numerical matrix representation for training the models.

4.2.1 N-gram Representation

An N-gram is a sequence of *N* words from a textual instance, where *N* is a positive integer number. N-grams are broadly used in natural language processing and computational linguistics to analyze the statistical properties of the text and understand the conveyed meaning (Cavnar et al., 1994). For example, *unigrams* represent single words, and *bigrams* denote two consecutive words, while *trigrams* point to three consecutive terms. Technically, N-gram representations are exploited by using the frequency and co-occurrence of words in a text which is valuable for the NLP tasks, such as language modeling, text classification, and information retrieval.

4.2.2 TF-IDF Vectorizer

TF-IDF Vectorizer is a process that converts a collection of textual documents into a matrix of TF-IDF (term frequency-inverse document frequency) feature values (Joachims, 1996). It is openly available in the scikit-learn library in Python language and is exploited in text mining and information retrieval tasks (Pedregosa et al., 2011). The vectorizer begins by counting the number of times each word appears in each document, known as the term frequency (TF). Then it scales this count by the inverse of the number of documents in which the word appears, known as the inverse document frequency (IDF). The resulting TF-IDF value for each term in each document is then used as a feature cell in the matrix. The vectorizer also provides options for the text preprocessing steps, such as lowercasing, removing stop words, and lemmatizing, as well as other options for adjusting the parameters of the TF-IDF calculation. For instance, the `max_features` parameter is set to 1000 to include a vocabulary that only considers the top 1000 features ordered according to frequency across the corpus. Also, the `ngram_range` parameter is determined to cover the combinations of unigram, bigram, and trigram representations, and there are six possibilities in total.

4.2.3 Logistic Regression Classifier

Logistic Regression is a type of supervised learning algorithm that is used for classification problems (Shah et al., 2020). It models the relationship between the dependent variable (the target variable) and one or multiple independent variables (the features) by utilizing a logistic function, which is the sigmoid function. The logistic function transforms the input data into a probability that ranges between 0 and 1. The threshold of 0.5 is typically selected to determine the class of the target variable, with values above 0.5 indicating one class and values below 0.5 indicating another class in binary classification. The logistic regression model is trained by utilizing `maximum likelihood estimation` to learn the weights that are associated with each feature and the intercept. During the prediction phase, the input data is multiplied by the weights and added to the intercept, and the result is fed into the logistic function to obtain the predicted probability. The predicted class is then determined based on the threshold value. Typically, logistic regression models are widely used in various applications, including image & text classification, spam filtering, and medical diagnosis (Bakal and Kavuluru., 2015; Josephine et al., 2021).

4.2.4 Random Forest Classifier

Random Forest is an ensemble learning algorithm for regression and classification tasks. It is an extension of the decision tree algorithm, in which multiple decision trees are learned, and their outputs are combined to make a final prediction (Breiman, 2001). The primary idea behind random forests is to decorate the variance of the individual trees and reduce the overfitting issue, a usual learning problem with decision trees. In the training phase, multiple decision trees are generated by randomly selecting a subset of features for each split and using bootstrapped training data samples. Then, the prediction for a new instance is made by taking the majority vote from all the decision trees in the forest. The randomness in the feature selection and data samples ensures that the trees are diverse, reducing the overfitting risk. Random Forest classifiers have proven to be robust and accurate (Rodriguez-Galiano et al., 2012), and they are relatively easy to implement compared to other models. They also can handle high-dimensional data and non-linear relationships between features and target variables. They are broadly operated in various applications, including image classification, credit risk analysis, and customer churn prediction.

4.2.5 XGBoost Classifier

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm, which is an ensemble learning technique used for regression and classification problems. XGBoost is a highly optimized and scalable implementation designed to maintain large and high-dimensional datasets. The XGBoost algorithm combines multiple decision trees in a sequential form, where each tree aims to correct the mistakes made by the previous trees. The training process involves iteratively adding trees to the model and updating the weights associated with each instance in the training data based on the prediction error of the previous trees. The final prediction is generated by taking a weighted average of the predictions from all trees (Chen and Guestrin, 2016). It is also known for its fast training speed, high accuracy, and ability to handle missing values and large amounts of data. Furthermore, it provides several hyperparameters that can be tuned to improve the model performance, such as the learning rate, the number of trees, and the tree size. These hyperparameters can be optimized using cross-validation or grid search. Grid search is a hyperparameter optimization technique that systematically evaluates a predefined set of hyperparameter values for a machine learning model. It exhaustively tests various combinations of these parameters and selects the combination that yields the best performance based on a specific metric. Alternatives to grid search, such as random search or Bayesian optimization, are also often used in model tuning, especially when the hyperparameter space is large.

4.3 Deep Learning Modeling

Deep learning (DL) is a subfield of machine learning that is inspired by the structure and function of the human brain structure, known as artificial neural networks. These networks consist of multiple layers of interconnected nodes or artificial neurons, which process and transmit information. The connections between the neurons have weights that are learned during training, allowing the network to make predictions or decisions based on input data (LeCun et al., 2015). In DL models, multiple hidden layers are used to extract and learn higher-level features from raw input data. The depth of these networks allows them to model complex non-linear relationships and capture abstract patterns in the data. The depth of deep learning networks, while enabling the capture of intricate relationships, is critically enhanced by the use of non-linear activation functions. These functions, such as ReLU (Rectified Linear Unit), introduce non-linearity into the model's decision boundaries. This non-linearity is essential for modeling complex, non-linear relationships within the data. Without non-linear activation functions, the network would effectively be a linear model, severely limiting its ability to represent intricate patterns or map complex interactions present in the financial data. The inclusion of non-linear activation functions in our deep learning models (e.g., CNN and LSTM) allows these models to learn hierarchical representations of the data. This is a crucial aspect of deep learning's capacity to model intricate patterns and enhance the prediction accuracy in our analysis. There are various deep learning architectures, including feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other contemporary models, such as LSTM (long short-term memory) and BERT architectures. These architectures have been adapted and developed to handle specific problems and have led to essential breakthroughs in various fields, such as computer vision, natural language processing, and intelligent gaming systems.

4.3.1 Convolutional Neural Networks Model Classifier

The convolutional Neural Networks (CNN) model is a specific deep learning architecture designed originally for image and video recognition tasks. The model is built using convolutional layers, which apply filters to local regions of the input data to extract features (Gu et al., 2018). These features are then passed through multiple non-linear activation functions, and the resulting feature maps are processed by pooling layers, which reduce the spatial dimensions while retaining important information. Considering text classification, CNNs are highly effective in capturing local patterns and features in the text, allowing them to make accurate predictions even when the input text is of variable length. A typical CNN structure is demonstrated in Fig. 3.

In text classification tasks, the input to the CNN is generally a matrix representation of the text, where each row corresponds to a word in the text, and each column represents a feature of the word, such as its word embedding or one-hot encoding. Convolutional filters are then applied to local regions of the input matrix to extract local features, and then max-pooling is used to aggregate the information from distinct textual sections. In the CNN architecture, multiple convolutional and pooling layers can be stacked to form a deep network to capture more complex and abstract features in the text. The final layer of the network is typically a fully connected layer, which outputs the predicted labels.

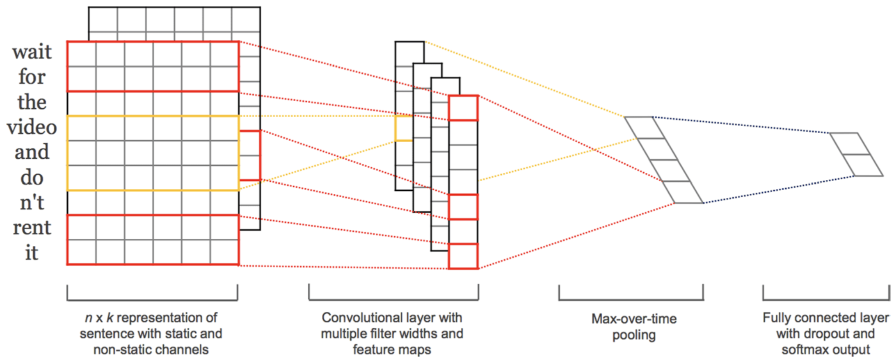
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 3 General CNN architecture (Cezanne, 2020)

4.3.2 Long Short Term Memory Model Classifier

Long Short-Term Memory (LSTM) is a popular RNN architecture that is commonly used in natural language processing and text classification tasks. LSTMs are particularly designed to overcome the vanishing and exploding gradient problems in traditional RNNs when processing long data sequences (Sherstinsky, 2020). This point is achieved by introducing memory cells, gates, and hidden states that control the flow of information in the network (Yu et al., 2019). In text classification, LSTMs are typically trained on large text corpora and then used to predict the class label for a given input text. The LSTM processes the input sequence word by word, updating its hidden state and memory cells based on the current term and previous hidden state. Then, the LSTM's final hidden state is passed through a fully connected layer to produce a prediction of the class label. This architecture allows the LSTM model to capture long-term dependencies in the textual data and generate more robust and accurate predictions.

A typical cell structure is represented in Fig. 4. The LSTM network architecture consists of a memory cell, three gates (input, output, and forget), and three fully connected layers. The memory cell is responsible for storing the information, while the gates control the information flow into and out of the cell. The input gate determines which part of information should be added to the cell state, and the forget gate determines which information should be discarded from the cell state, while the output gate decides which information should be outputted as the final output. Combining the memory cell and specific gates allows LSTM cells to retain long-term dependencies, which is critical for tasks such as language modeling, speech recognition, machine translation, and particularly sequence-based models.

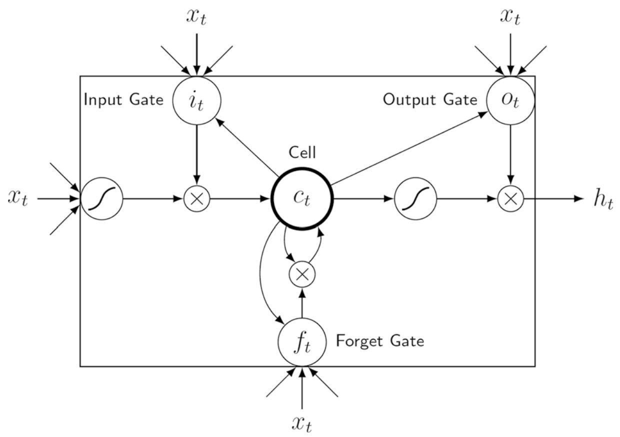

Fig. 4 A typical LSTM cell structure (Melli, 2024)

4.3.3 Pretrained Models and The Fin-BERT Model

Transformers, a type of neural network architecture, have gained significant attention in natural language processing due to their ability to capture contextual relationships between words in a sentence. Unlike RNNs, which process sequences sequentially, transformers utilize attention mechanisms to consider all words simultaneously, which enables them to capture long-range dependencies more effectively (Vaswani, 2017). This parallel processing capability is particularly beneficial for tasks involving complex language, such as sentiment analysis, where understanding the context surrounding a word or phrase is crucial. The use of attention allows the model to weigh the importance of different words in relation to the others, providing a more holistic understanding of the sentiment expressed. While promising, the direct application of transformer models to financial tweet analysis, as highlighted in our literature review, is still a relatively unexplored area with potential for improvement.

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing model developed by Google. It belongs to the Transformer architecture family and is designed for unsupervised learning tasks in NLP (Bozdag et al., 2024). What sets BERT apart is its bidirectional context awareness, enabling it to consider both left and right contextual words when predicting a target word and enhancing its understanding of word relationships. BERT achieves this by pre-training on large corpora and learning contextualized embeddings for words. During the training phase, it predicts masked words in sentences, capturing intricate contextual information.

The FinBERT is a pre-trained natural language processing model which is built to analyze the sentiment of financial textual elements. It is developed by fine-tuning the BERT language model on a large financial corpus, specifically, the Financial PhraseBank by Malo et al. (2014), to improve its performance on financial sentiment classification tasks (ProsusAI, 2020). The primary rationale for utilizing the FinBERT pre-trained model stems from its construction using a sufficiently extensive dataset. Furthermore, a deliberate choice was made to employ the FinBERT model, distinct from other BERT-based models, due to its alignment with the specific domain under consideration in this experimental work.

4.4 Experimental Details & Configurations

The principal motivation is to monitor the performance improvements of the models when further knowledge, which the pre-trained BERT model obtains, is provided to the original model. Hence, we created gradually increasing additional labeled data sample sets of 5000, 10,000, 15,000, 30,000, and 40,000. These datasets are incorporated into the original dataset to enhance the classification performances over different models. Ultimately, the performance evaluations are obtained through the average performance metrics of *the ten distinctly shuffled data splits* scores.

4.4.1 Traditional ML Model Configurations

During the traditional models' constructions, the logistic regression model's parameters are chosen as default values (i.e., `l2` *penalty*, `C`-*Inverse of regularization as* 1.0, `multi_class` *as multinomial*) since there was no noticeable improvement with the parameter optimization. Similarly, the random forest model's parameters are selected as the `criterion` (*split quality function*) as *gini*, `min_samples_split` (*minimum number of instances needed to split*) as 2, and `min_samples_leaf` as 1. Plus, the employed voting mechanism of the random forest model we used is usual majority voting as default. Regarding the xgboost model, the hyperparameters parameters are as follows: `booster` type of *gbtree*, `learning rate- eta` of 0.3, and `max_depth` of 6.

4.4.2 Deep Learning Model Configurations

In the CNN model construction, the network architecture began with an input layer as a text vectorizer to convert the textual instances into the numerical matrix representation. Then, it was followed by the `embedding layer` transforming high-dimensional vectors into low-dimensional space. Afterward, a `1DConvolutional` layer with `ReLu` (Rectified Linear Unit) activation function was integrated into the network to extract the contextual features and followed by a `GlobalMaxPooling1D` layer to downsample the input representation by taking the maximum value along the second/time axis. Lastly, the network structure ended with the output layer functioning with the `softmax activation` function. The embedding layer dimension was determined by the maximum vocabulary size of 10,000 and the maximum instance sequence length of 13. The `convolution filter` size of 32 and the kernel size of five were used in the convolution layer, while the output layer contains three units indicating each sentiment category. The whole architecture was compiled with the `categorical-cross-entropy` loss function and optimized by the `Adam optimizer` function.

Similarly, the LSTM model was initialized with an embedding layer and two `LSTM` layers containing 256 and 128 neural units, respectively. Then, the LSTM network was constructed by four dense layers in which the unit sizes were 128, 128, 64, 32, and 32. Here, the idea of decreasing the number of neurons from 128 to 32 is to have a more relevant and dense representation of the previous layer information. Finally, the model was finalized by the output layer with the `softmax activation` function containing three categorical units (positive, neutral, and negative). The LSTM model was regularized by `drop-out` layers with a rate of 0.2 after each layer to prevent over-fitting issues. The overall LSTM network was compiled with the `categorical-cross-entropy` loss function and optimized by the `Adam optimization` algorithm.

5 Results and Evaluations

Consequently, this work aims to classify the financial tweet instances according to their contextual sentiment information and monitor the performance effects of the gradually-increased number of labeled examples. Hence, we present the weighted average performance scores of the traditional ML (logistic regression, random forest, and xgboost) models over the ten randomly shuffled data splits in Table 2. As the critical evaluation metric, the average F1 score was computed over the average precision and recall scores.

When we inspect the table, the first noticeable outcome is that the best F1 score was obtained by the logistic regression model built with the unigram feature space. Unlike the logistic regression models, the best F1 score was achieved by the set of {unigram ∪ bigram} features combination in random forest models, even though the scores were slightly different. Among the xgboost classifier experiments, again, the model using unigram feature space yielded the highest F1 score as opposed to the other feature space configurations. The reason for the lowest performing models utilizing bigram feature space is that bigrams are less frequent compared to other features. Overall, the model constructed with the logistic regression classifier using unigram features gained the highest classification performance. Considering the deep learning models, we demonstrate the performance scores over the ten shuffled data splits obtained from the CNN and LSTM models in Table 3. Although the CNN model outperformed the LSTM model, the CNN model did not achieve any better performances than the best-performing logistic regression and random forest models.

Surprisingly, the LSTM model performed the worst among the best model configurations, even if there were a few cases where it outperformed. The potential reason for this situation is that the average length of tweet instances is much smaller

Table 2 Performance metrics of traditional ML models on the initial dataset

Model	N-gram type	Avg. precision	Avg. recall	Avg. F1-score
Logistic regression	Unigram	0.579	**0.511**	**0.542**
	Bigram	0.532	0.423	0.471
	Trigram	0.602	0.403	0.482
	Unigram ∪ Bigram	0.594	0.490	0.537
	Bigram ∪ Trigram	0.563	0.422	0.482
	Unigram ∪ Bigram ∪ Trigram	**0.608**	0.483	0.538
Random forest	Unigram	0.547	0.5	0.522
	Bigram	0.417	0.396	0.406
	Trigram	0.447	0.392	0.417
	Unigram ∪ Bigram	**0.548**	**0.501**	**0.523**
	Bigram ∪ Trigram	0.420	0.395	0.407
	Unigram ∪ Bigram ∪ Trigram	0.533	0.492	0.511
XGBoost	Unigram	**0.526**	**0.487**	**0.505**
	Bigram	0.425	0.372	0.396
	Trigram	0.445	0.361	0.398
	Unigram ∪ Bigram	0.520	0.482	0.5
	Bigram ∪ Trigram	0.428	0.372	0.398
	Unigram ∪ Bigram ∪ Trigram	0.512	0.476	0.493

Highest scores within the corresponding setup for each table are shown in bold

Table 3 Performance metrics of deep learning models on the initial dataset

Model	Avg. precision	Avg. recall	Avg. F1-score
CNN	**0.516**	**0.515**	**0.515**
LSTM	0.456	0.447	0.451

Highest scores within the corresponding setup for each table are shown in bold

than typical textual examples to maintain the learning operation in sequential models. As mentioned in the introduction section, we observed the effect of the gradually-increased number of additional examples, labeled by the pre-trained Fin-BERT model, on the classification performance and showed their performance scores gathered by traditional ML models in Tables 4 and 5. The most significant outcome from the experimental results table is that increasing the number of new examples helps to boost the models' classification power. To better illustrate the performance differentiation, we have generated a bar chart representing the best sub-models *which are Random Forest classifiers with unigram features* configurations, as shown in Fig. 5.

When the most successful models are considered regardless of the classifier type in each configuration where additional instances are cumulatively included, we achieved at least a 1% F1 score improvement. Another striking point is that the models using a random forest classifier outperformed other models in all configurations. One possible explanation for this outcome is that the Random Forest algorithm exploits an ensemble classification approach using multiple decision trees. Among all distinct model configurations, the model employing the random forest algorithm

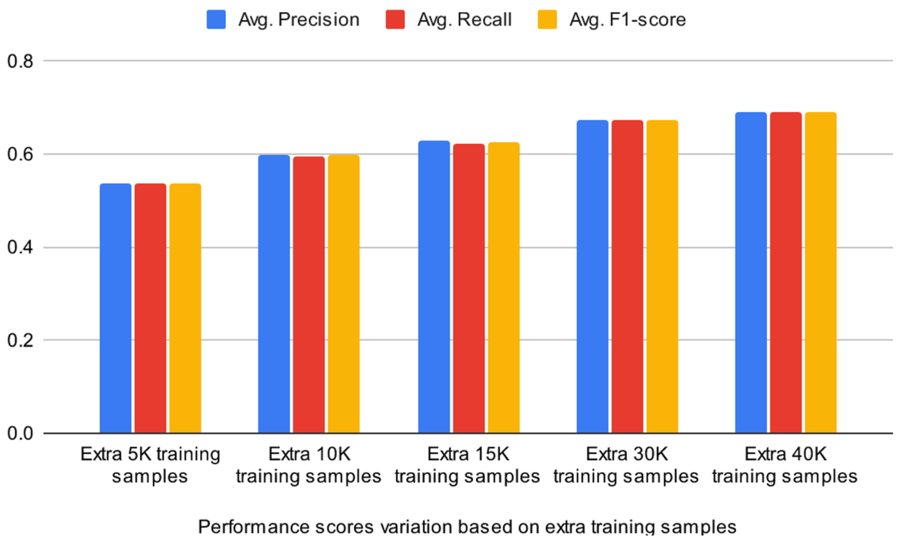

Fig. 5 Performance differentiation by the number of incorporated extra examples in random forest models

Table 4 Results of traditional ML models on the additional data sizes: 5K and 10K instances

Model	N-gram type	Average precision	Average recall	Average F1-score
Logistic regression on new 5K samples	Unigram	**0.531**	**0.531**	**0.531**
	Bigram	0.468	0.457	0.462
	Trigram	0.501	0.437	0.466
	Unigram ∪ Bigram	0.527	0.528	0.527
	Bigram ∪ Trigram	0.469	0.453	0.460
	Unigram ∪ Bigram ∪ Trigram	0.526	0.527	0.526
Random forest on new 5K samples	Unigram	**0.538**	**0.538**	**0.538**
	Bigram	0.462	0.449	0.455
	Trigram	0.498	0.437	0.465
	Unigram ∪ Bigram	0.532	0.53	0.530
	Bigram ∪ Trigram	0.465	0.447	0.455
	Unigram ∪ Bigram ∪ Trigram	0.526	0.525	0.525
XGBoost on new 5K samples	Unigram	**0.53**	**0.529**	**0.529**
	Bigram	0.464	0.444	0.453
	Trigram	0.509	0.426	0.463
	Unigram ∪ Bigram	0.528	0.527	0.527
	Bigram ∪ Trigram	0.462	0.441	0.451
	Unigram ∪ Bigram ∪ Trigram	0.528	0.527	0.527
Logistic regression on new 10K samples	Unigram	**0.594**	**0.595**	**0.594**
	Bigram	0.515	0.496	0.505
	Trigram	0.53	0.464	0.494
	Unigram ∪ Bigram	0.59	0.592	0.590
	Bigram ∪ Trigram	0.508	0.486	0.496
	Unigram ∪ Bigram ∪ Trigram	0.588	0.589	0.588
Random forest on new 10K samples	Unigram	**0.599**	**0.596**	**0.597**
	Bigram	0.511	0.489	0.499
	Trigram	0.527	0.464	0.493
	Unigram ∪ Bigram	0.595	0.591	0.592
	Bigram ∪ Trigram	0.511	0.487	0.498
	Unigram ∪ Bigram ∪ Trigram	0.595	0.592	0.593
XGBoost on new 10K samples	Unigram	0.593	0.591	0.591
	Bigram	0.521	0.49	0.505
	Trigram	0.549	0.455	0.497
	Unigram ∪ Bigram	**0.595**	**0.593**	**0.593**
	Bigram ∪ Trigram	0.513	0.487	0.499
	Unigram ∪ Bigram ∪ Trigram	0.593	0.59	0.591

Highest scores within the corresponding setup for each table are shown in bold

with the unigram feature space on 40K newly added auto-annotated examples yielded the best precision (69%), recall (68%), and F1 score (69%). Contrary to the best model, the lowest model achieved an F1 score of 45%. Yet another considerable

Table 5 Results of traditional ML models on the additional data sizes: 15K, 30K, and 40K instances

Model	N-gram type	Average precision	Average pecall	Average F1-score
Logistic regression on new 15K samples	Unigram	**0.617**	**0.617**	**0.617**
	Bigram	0.533	0.512	0.522
	Trigram	0.545	0.476	0.508
	Unigram ∪ Bigram	0.613	0.614	0.613
	Bigram ∪ Trigram	0.525	0.501	0.512
	Unigram ∪ Bigram ∪ Trigram	0.609	0.609	0.609
Random forest on new 15K samples	Unigram	**0.63**	**0.623**	**0.626**
	Bigram	0.538	0.515	0.526
	Trigram	0.54	0.474	0.504
	Unigram ∪ Bigram	0.628	0.62	0.623
	Bigram ∪ Trigram	0.528	0.501	0.514
	Unigram ∪ Bigram ∪ Trigram	0.629	0.621	0.624
XGBoost on new 15K samples	Unigram	0.622	0.618	0.619
	Bigram	0.542	0.512	0.526
	Trigram	0.569	0.471	0.515
	Unigram ∪ Bigram	0.625	0.62	0.622
	Bigram ∪ Trigram	0.531	0.503	0.516
	Unigram ∪ Bigram ∪ Trigram	**0.626**	**0.621**	**0.623**
Logistic regression on new 30K samples	Unigram	**0.648**	**0.648**	**0.648**
	Bigram	0.529	0.514	0.521
	Trigram	0.538	0.47	0.501
	Unigram ∪ Bigram	**0.648**	**0.648**	**0.648**
	Bigram ∪ Trigram	0.519	0.501	0.509
	Unigram ∪ Bigram ∪ Trigram	0.647	0.647	0.647
Random forest on new 30K samples	Unigram	**0.675**	**0.672**	**0.673**
	Bigram	0.542	0.521	0.531
	Trigram	0.539	0.471	0.502
	Unigram ∪ Bigram	0.671	0.668	0.669
	Bigram ∪ Trigram	0.532	0.508	0.519
	Unigram ∪ Bigram ∪ Trigram	0.668	0.665	0.666
XGBoost on new 30K samples	Unigram	**0.667**	**0.66**	**0.663**
	Bigram	0.548	0.505	0.525
	Trigram	0.555	0.461	0.503
	Unigram ∪ Bigram	0.663	0.657	0.659
	Bigram ∪ Trigram	0.53	0.499	0.514
	Unigram ∪ Bigram ∪ Trigram	0.664	0.658	0.660
Logistic regression on new 40K samples	Unigram	**0.66**	**0.659**	**0.659**
	Bigram	0.535	0.519	0.526
	Trigram	0.544	0.47	0.504
	Unigram ∪ Bigram	0.658	0.658	0.658
	Bigram ∪ Trigram	0.526	0.506	0.515
	Unigram ∪ Bigram ∪ Trigram	0.654	0.654	0.654

Table 5 (continued)

Model	N-gram type	Average precision	Average pecall	Average F1-score
Random forest on new 40K samples	Unigram	**0.691**	**0.689**	**0.690**
	Bigram	0.552	0.532	0.541
	Trigram	0.547	0.473	0.507
	Unigram ∪ Bigram	0.686	0.683	0.684
	Bigram ∪ Trigram	0.542	0.518	0.529
	Unigram ∪ Bigram ∪ Trigram	0.684	0.682	0.682
XGBoost on new 40K samples	Unigram	**0.679**	**0.671**	**0.674**
	Bigram	0.553	0.513	0.532
	Trigram	0.572	0.465	0.512
	Unigram ∪ Bigram	0.677	0.67	0.673
	Bigram ∪ Trigram	0.539	0.504	0.520
	Unigram ∪ Bigram ∪ Trigram	0.674	0.668	0.670

Highest scores within the corresponding setup for each table are shown in bold

point is that 13 out of 15 unique models achieved higher performances using only unigram features.

As the results indicate in Table 6, the DL models also yielded a similar rising trend as in the traditional ML models when the number of new examples was gradually increased in the experimental configurations. When we interpret the impact of the DL experiments utilizing cumulatively more annotated instances, the LSTM model captured more discriminative contextual information than the CNN model when we introduced new samples. Here, the strange case we noticed is that the CNN model had around a loss of 0.5% precision, recall, and F1 score when we included 5K additional examples.

Table 6 Performance metrics of DL models on the additional dataset

	CNN			LSTM		
Additional Set	Avg. precision	Avg. recall	Avg. F1-score	Avg. precision	Avg. recall	Avg. F1-score
New 5K samples	0.501	0.501	0.501	0.47	0.461	0.465
New 10K samples	0.575	0.574	0.574	0.549	0.544	0.546
New 15K samples	0.6	0.6	0.6	0.577	0.575	0.575
New 30K samples	0.65	0.648	0.648	0.648	0.648	0.648
New 40K samples	0.668	0.668	0.668	0.671	0.667	0.668

To visually keep track of the rising trend in the classification performances of the DL models, we drew a bar chart portraying the differentiation trends of precision, recall, and F1 score metrics in Fig. 6. From the resulting graphic, the remarkable point is that both the LSTM and CNN models achieved the same classification performances when the models were fed by 30K and 40K new instances. Another considerable outcome is that the LSTM architecture never outperformed the CNN model in any configurations.

As can be noticed from the results in Tables 4, 5 and 6, the traditional ML models (including logistic regression, random forest, and xgboost) performed better than DL models in each distinct configuration. The reason for this situation is that the textual length of the tweet examples was not long enough to capture the contextual dependencies in the dataset. Unlike the DL models, the traditional ML models could learn the contextual associations reasonably better among the words by using short sizes of n-gram features.

As shown in Table 7, the models demonstrate a slightly different performance profile across the three sentiment classes. This table offers a more granular view of the classification quality by providing a more robust assessment of our deep learning models and confirming our overall observation of significant model performance gains.

6 Conclusion

Sentiment analysis stands as a pivotal research endeavor within the machine learning and data science fields, where it finds widespread applications across various domains, encompassing individual opinions on products or services to collective expressions in social media posts. The conclusive findings presented in Sect. 5 unequivocally validate our hypothesis that the incremental inclusion of new samples

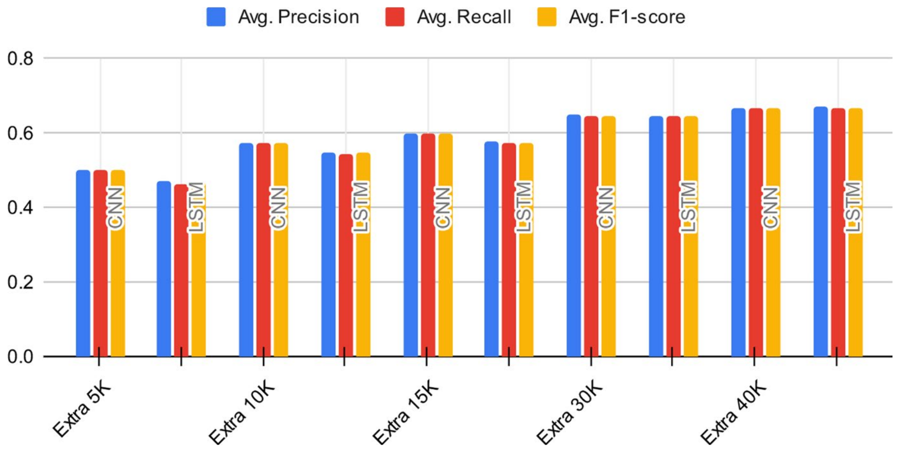

Fig. 6 Performance differentiation by the number of incorporated extra examples in CNN and LSTM models

Table 7 Performances of each class for the deep learning models

Model	Class tone	Avg. precision	Avg. recall	Avg. F1-score
LSTM	Positive	0.647	0.642	0.644
	Negative	0.721	0.709	0.714
	Neutral	0.640	0.652	0.645
CNN	Positive	0.646	0.645	0.643
	Negative	0.701	0.722	0.713
	Neutral	0.657	0.640	0.646

significantly enhances the performance of the models. Specifically, the optimal machine learning model, constructed using the random forest algorithm with unigram features and 40,000 new instances, achieved an F1 score of 69%, which outperforms the initial model's F1 score of 54% in its best configuration (*a logistic regression model employing unigram features*). This noteworthy improvement of 15% in F1 score reaffirms the effectiveness of our approach, a result consistently observed across both traditional machine learning and deep learning model experiments.

7 Limitations & Future Works

While the study contributes valuable insights to sentiment analysis in the context of stock market tweets, there are certain limitations and opportunities for future research. One limitation lies in the reliance on labeled data, and the study's focus on sentiment analysis may not capture the full spectrum of market dynamics. Additionally, the effectiveness of the proposed approaches may vary across different market conditions, and external factors influencing sentiment expression could impact the generalizability of the models. Yet another limitation is that the original data collection method described in this study may not be functional due to the re-branding of Twitter to X and the associated changes to the platform's APIs.

Although we briefly expose the advantage of transferring knowledge provided by the pre-trained Bert model, a control subset of the labeled examples (*by randomly selecting*) might be subjected to human annotators to validate the labeling results. By performing this additional operation in the future, we can confirm the success of the proposed Bert-based approach. Another potential future direction could be an exploration of ensemble models or hybrid approaches that combine traditional machine learning algorithms with deep learning techniques. This fusion might offer synergistic advantages by warranting further investigation. Overall, addressing these considerations would contribute to advancing the understanding and application of sentiment analysis in forecasting stock market trends.

Acknowledgements Not applicable.

Author Contributions E.C. carried out the experiments and wrote the initial manuscript draft. G.B. supervised the project and conceived the original idea. Also, G.B. wrote the final manuscript.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). No funds, grants, or other types of support were received.

Data Availability The dataset we used is available via the link: https://ieee-dataport.org/open-access/stock-market-tweets-data.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical Approval Not applicable.

References

Akbiyik, M. E., Erkul, M., Kämpf, K., Vasiliauskaite, V., & Antulov-Fantulin, N. (2023). Ask" who", not" what": Bitcoin volatility forecasting with twitter data. In: *Proceedings of the sixteenth ACM international conference on web search and data mining,* pp. 688–696.

Al-Shargabi, B., Al-Romimah, W., & Olayah, F. (2011). A comparative study for arabic text classification algorithms based on stop words elimination. In: *Proceedings of the 2011 international conference on intelligent semantic web-services and applications*, pp. 1–5.

Antweiler, W., & Frank, M. Z. (2004). Is all that talk just noise? The information content of internet stock message boards. *The Journal of finance, 59*(3), 1259–1294.

Aroyehun, S. T., Gelbukh, A. (2018). Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In: *Proceedings of the first workshop on trolling, aggression and cyberbullying (TRAC-2018)*, pp. 90–97.

Bacco, L., Petrosino, L., Arganese, D., Vollero, L., Papi, M., & Merone, M. (2024). Investigating stock prediction using lstm networks and sentiment analysis of tweets under high uncertainty: A case study of north american and european banks. *IEEE Access.*

Bakal, G., & Abar, O. (2021). On comparative classification of relevant covid-19 tweets. In: *2021 6th international conference on computer science and engineering (UBMK)*, IEEE, pp. 287–291.

Bakal, G., & Kavuluru, R. (2015). Predicting treatment relations with semantic patterns over biomedical knowledge graphs. In: *International conference on mining intelligence and knowledge exploration*, Springer, pp. 586–596.

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science, 2*(1), 1–8.

Bozdag, M., Sevim, N., & Koç, A. (2024). Measuring and mitigating gender bias in legal contextualized language models. *ACM Transactions on Knowledge Discovery from Data*. https://doi.org/10.1145/3628602

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Cavnar, W. B., Trenkle, J. M., et al. (1994). N-gram-based text categorization. In: *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, Las Vegas, NV, pp. 161–175.

Cezanne, C. (2020). Cnn for text classification. https://cezannec.github.io/CNN_Text_Classification, accessed: 2024-09-12.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Day, M. Y., & Lee, C. C. (2016). 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)*, IEEE, pp. 1127–1134.

Erkantarci, B., & Bakal, G. (2023). An empirical study of sentiment analysis utilizing machine learning and deep learning algorithms. *Journal of Computational Social Science,* pp. 1–17.

Gandhudi, M., Alphonse, P. J. A., Fiore, U., & Gangadharan, G. R. (2024). Explainable hybrid quantum neural networks for analyzing the influence of tweets on stock price prediction. *Computers and Electrical Engineering, 118*(109), 302.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77*, 354–377.

Gupta, R., & Chen, M. (2020). Sentiment analysis for stock price prediction. In: *2020 IEEE conference on multimedia information processing and retrieval (MIPR)*, IEEE, pp. 213–218.

Joachims, T. (1996). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science.

Josephine, J., Ulaganathan, M., Shenbagavalli, A., Venkatraman, B., & Menaka, M. (2021). Statistical analysis on breast thermograms using logistic regression for image classification. In: *2021 IEEE Bombay section signature conference (IBSSC)*, IEEE, pp. 1–6.

Khafaga, D. S., Auvdaiappan, M., Deepa, K., Abouhawwash, M., & Karim, F. K. (2023). Deep learning for depression detection using twitter data. *Intelligent Automation & Soft Computing, 36*(2), 1301–1313.

Kumar, S., Morstatter, F., & Liu, H. (2014). *Twitter data analytics*. Springer.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies, 5*(1), 1–167.

Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. arXiv preprint cs/0205028.

Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-ks. *The Journal of Finance, 66*(1), 35–65.

Malo, P., Sinha, A., Korhonen, P., Wallenius, J., & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology, 65*(4), 782–796.

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

Melli, G. (n.d.) (2024). Long short-term memory (lstm) unit. https://www.gabormelli.com/RKB/Long_Short-Term_Memory_%28LSTM%29_Unit accessed: 2024-09-12.

Meng, M., Zhang, Y., Ma, Y., Gao, Y., & Kong, W. (2023). Eeg-based emotion recognition with cascaded convolutional recurrent neural networks. *Pattern Analysis and Applications, 26*(2), 783–795.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & Vanderplas, J. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research, 12*, 2825–2830.

ProsusAI (2020). finbert: Financial sentiment analysis with bert. https://github.com/ProsusAI/finBERT.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing, 67*, 93–104.

Roesslein, J. (2020). Tweepy: Twitter for python!.

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research, 5*, 1–16.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena, 404*(132), 306.

Silva, C., & Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. In: *Proceedings of the international joint conference on neural networks*, IEEE, pp. 1661–1666.

Sohangir, S., Wang, D., Pomeranets, A., & Khoshgoftaar, T. M. (2018). Big data: Deep learning for financial sentiment analysis. *Journal of Big Data, 5*(1), 1–25.

Sun, L., Li, Q., Liu, L., & Su, Y. (2023). Unsupervised multimodal learning for image-text relation classification in tweets. *Pattern Analysis and Applications, 26*(4), 1793–1804.

Sungur, K. S., & Bakal, G. (2025). Beyond visual cues: Emotion recognition in images with text-aware fusion. *Displays,* p. 102958.

Taborda, B., de Almeida, A., Dias, J. C., Batista, F., & Ribeiro, R. (2021). *Stock market tweets data. IEEE Dataport.*

Tumasjan, A., Sprenger, T., Sandner, P., et al. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In: *Proceedings of the international AAAI conference on web and social media,* pp. 178–185.

Tun, Y. M., & Khaing, M. (2023). A large-scale sentiment analysis using political tweets. *International Journal of Electrical & Computer Engineering (2088-8708)* 13(6).

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Yu, Y., Si, X., Hu, C., et al. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation, 31*(7), 1235–1270.