Stock Price Prediction with Online News Sentiment Analysis

Alkın Er, Mehmet Rıfat Özkurt, Yiğit Ali Selçuk  
*Bilkent University   
Department of Electrical and Electronics Engineering*Ankara, Turkey

*Abstract*—The prediction of stock market prices has always been an enthuse among financiers and researchers. Although the current leap of deep learning, RNN architectures, decision trees and regression algorithms based on historical stock price data improved the ability of making a valid future prediction; this approach is still insufficient regarding the high volatility of large companies’ stock prices, in which the diverse investor sentiment and political incidents are strong factors. In this paper, we propose an approach of utilizing the task of sentiment analysis together with the historical daily stock price data of Tesla. To do so, BERT architecture will be specified on financial news data for text classification and sentiment analysis, and the output tensors will be fed in an LSTM recurrent neural network together with financial historical data. The altogether system is expected to predict the movement of the stock price for the next day.

Keywords—NLP, natural language processing, stock price prediction, online news sentiment analysis, financial news, BERT, LSTM, RNN, Tesla, historical stock price data, stock market.

# Introductıon

Stock price trend prediction is a quite significant area of research in finance, attracting investors, traders and researchers. The current improvement in the regression algorithms, Neural Networks (NN) and Deep Neural Networks (DNN) resulted in valuable insights for making informed investment decisions and maximizing returns. Still, the improvements are insufficient because of the high volatility in stock prices, regarding the market is highly dependent on factors other than financial insights. Therefore, the developments in the Natural Language Processing field after 2010’s, utilizing textual information besides historical market data resulted in more robust architectures on stock market prediction and became a widely famous approach among researchers.

The aim of this paper will be to contribute the existing literature on making stock price movement predictions, with improvements of widely used architectures to obtain more accurate results. The significance of our work will be the new improvements of those architectures, namely “Bidirectional Encoder Representations from Transformers (BERT)” for sentiment analysis of online news corpora and “Long-Short Term Memory (LSTM)” for stock price prediction. Our paper will communicate with related work and represent how and why our results are different than those suggesting an almost perfectly accurate stock prediction. To do so, we will firstly

utilize BERT for sentiment analysis, specifically on financial corpora obtained from Financial Phrase Bank [1] and FIQA [2]. The procedure will follow the utilization of LSTM architecture, using the sentiment score and label outputs of online news analysis of Tesla in twelve months from SeekingAlpha [3] & Benzinga [4] and finance data from Yahoo Finance [5] in the same time period. It is expected to obtain a good approximation of how the trend of stock prices will respond the news and the previous trend, if not the prediction of exact stock prices with a negligible amount of error.

The following of this report will include a “Related Work” section where the literature in this field will be reviewed. Then in the “Methods” section, the procedure of how the data is collected, organized and formatted in a brand new dataset and additionally the motivation behind the selection of architectures, models, parameters and methods will be examined. Following that section, illustrations of our results will be represented and discussed in “Results”, and those results will be evaluated in “Discussion & Conclusion” in the light of previous work.

# Related Work

## Bidirectional Encoder Representations from Transformers (BERT)

BERT is a transformer-based, pre-trained language model that is widely used for NLP purposes. Due to its robust self-attention mechanism, large training corpus and availability, different forms of this model are prevalent in many NLP research areas, including financial analysis. Some of these researches are studied below.

One study that utilizes BERT model for financial sentiment analysis and stock price prediction is by Wang and Luo [6], which focused on the stock price of a very controversial company on social media, Gamestop (GME). This stock was subject to huge social media manipulations in 2021, making it the most shorted stock of the year, which is why it is relevant for such a sentiment analysis task on social media posts. The authors used a pretrained BERT model to analyze tokens of social media (Reddit) posts on the given stock. For the sentiment labeling model, VADER module of Python’s NLTK library was chosen. The semantic embedding outputs of word2vec and the BERT model were added to the outputs and compared. As for the BERT model, a pre-trained BERT which was trained on general-purpose corpora, namely Wikipedia and BookCorpus, and a sequence classification task were chosen to produce the [‘CLS’] token output of the input sequences. These [‘CLS’] tokens were then fed to different models such as SVM, SCM, Random Forest and MLP for sentiment classification purposes. The authors realize the unpredictable fluctuations of the chosen stock and the probable insufficiency of traditional prediction methods on this particular case. Overall, the authors were unable to describe a strong relationship between the sentiment extracted from text and stock price movement. Using a general BERT model trained on various types of corpora might not have been very efficient in analyzing financial statements. For this reason, finance-specific BERT models are also used in academic research.

Finance-specific BERT models are created by fine-tuning a pre-trained BERT model entirely on financial corpora. This provides BERT a more robust understanding of financial collocations, nuances and use of jargon. Araci (2019) [7] introduces one of these finance-specific models trained on a pre-trained BERT model, called FinBERT. The author further trains the “bert-base-uncased” twice to obtain application-specific models. To improve target domain adaptation, the pre-trained BERT model was initially trained with a large financial communication corpus called TRC2 by Reuters. After this training, a second application-specific corpus was used to train the financial model for different tasks such as sentence classification and regression. For the sentence classification task, a labelled sentiment dataset called “Financial Phrasebank” from Malo et al. [8] is used. In the second training, the classifier network is trained. This is mainly achieved by methods such as gradual unfreezing and discriminative fine-tuning to avoid training the lower layers of the original model, which would cause catastrophic forgetting [7]. Overall, Araci’s FinBERT model was compared to three traditional classifier methods: LSTM classifier with GLoVe embeddings, ELMo embeddings and another ULMFit classifier. The author found FinBERT to outperform these models in all three metrics: loss, accuracy and F1 score. While this somewhat underlines the importance of application-specific models, this study lacks the stock price prediction component, which is still a major concern.

Another construction of a financial BERT model that bears the same name FinBERT is done by Yang et al. (2020), [9] which employs a similar strategy in fine-tuning the pretrained language model. The general-purpose model is first trained on a financial corpus consisting of annual formal corporate reports mandated by the U.S. Security Exchange Commision. These 60490, reports belong to 3000 different companies and were later merged with other corpora such as earnings conference call transcripts and analyst reports. This dataset is used to train a general-purpose financial language model. Unlike Araci’s work, a linear layer was used as a classification layer instead of a deep architecture. Then, this general-purpose model was further fine-tuned for sentence classification tasks, for which multiple datasets were tested. Two of these datasets are FiQA [10] and Financial Phrasebank [8] datasets, similar to Araci’s work [7]. The performance of those was compared in the results. Overall, FinBERT was found to outperform the pre-trained model with increases in the accuracy score ranging from 4% to 15.4%. Among the datasets used, the descending order of accuracy is observed as: AnalystTone [9], Financial Phrasebank, FiQA. Although this study proves the success of a financial language model, similar to Araci’s study, it does not investigate stock price movement prediction. Therefore, the overall success of the model in out study’s task is very much questionable. Our paper aims to employ a similar fine-tuning process on a BERT model and merge this functionality with another stock prediction model to perform a news-trading/stock prediction task.

## Long-Short Term Memory (LSTM)

The early studies in this field was focused on using historical financial data. Interpreting financial information as a time series data, Song, Zhou and Han [11] suggested that the results of 5 neural network models including BP, RBF, GRNN, SVMR, LS-SVMR can extract meaningful information from historical price. Moreover, they also demonstrated that BP was the best model for this task, regarding accuracy and robustness in stock forecasting and also was quite robust. Still, this work also demonstrated that the Market Efficiency Hypothesis [12] is not likely to be valid, which claims that the market is not affected from the parameters other than economic/market conditions, as like a closed system.

As the denial of Market Efficiency Hypothesis suggests, researchers focused on the effect of parameters other than the market information. Considering the huge impact of political incidents and news on market, the researchers utilized the approaches that NLP suggests for processing the news data along with the historical financial data. Khedr, Salama, and Yaseen [13], by using PRNewswire archive, proposed a system of providing customers financial consultancy. After using a Naïve Bayes algorithm for sentiment analysis of the text, they join feature set by date and k-NN algorithms for prediction model. However, they only predict the future trend of the stock prices, which is, whether the stock will raise or fall. Elegamy et al. [14] used a new approach of using Random Forest besides text mining algorithms for extraction of critical indicators. In spite of its success of implementing Random Forest classifier for %98.89 accuracy, it still lacks the ability of predicting stock market values precisely.

In the light of previous studies, with the inclusion of NLP, the new studies produced more robust models. Alostad and Davulcu [14] proposed a novel approach for directional prediction of stocks using news data from Twitter, specifically on breaking news. The architecture contains two branches, one processes Feature Selection algorithm of hourly news, and the other labels that news with hourly stock chart. They utilize LogisticR classifier with 1-gram keyword features, leading high directional prediction accuracy at a narrow time stamp. With a similar approach, Mohan et al. [15] used different variations of RNN-LSTM architecture to build up a prediction model for Apple’s stock prices. The most accurate results yielded from RNN-pt and RNN-pp models, which stand for RNN-LSTM model with prices and text as input and RNN-LSTM model with prices and text polarity as input. They demonstrated that the most robust model was RNN-pp with a quite small deviation from the actual prices, with a higher prediction insight than any other architectures that are previously used for this task.

Based on all those previous work in this field, we decided to use BERT for sequence classification for practical purposes and its robustness in sentiment analysis tasks. Based on Mohan et al.’s [15] results, inputting LSTM model the sentiment scores obtained from headlines of Tesla analysis articles and historical financial data of Tesla is expected to yield more accurate results. Another remarkable reason for this decision is the success and the accuracy of LSTM architecture regarding time-series data.

# Methodology

This section details the implementation of the two designated models, where the first one is dedicated to perform sentiment analysis on historical news data and the second model is dedicated to use this sentiment output and historical finance data on the stock of TESLA Inc. to predict the next day’s price movement (as percent return) for the company. The second model makes predictions on a day-to day basis and does not predict closing prices for further dates, therefore it always requires the entire stock data of the previous day, only to be able to predict the next day’s return. Since it is always fed with historical actual data, it will closely follow the actual stock trend of the given company and update itself on error. Implementation details of the models are in the following sections, along with the details on the datasets used and preprocessing of those.

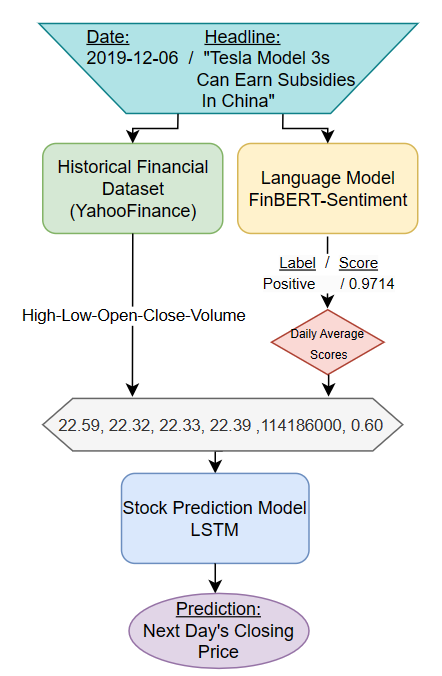


Fig. 1: The outline of the project with sample inputs and outpus.

## Data Collection & Processing

To fine tune project’s language model, labeled training datasets are obtained from Financial Phrase Bank [1] and FIQA [2]. Financial Phrase Bank dataset includes nearly 5000 sentences which are in human standards. When two datasets are merged, 5322 total unique values are obtained. These two datasets consist of financial statements and their corresponding sentiment labels which belong to following categories; *positive, neutral and negative*. FinBert is a model which is a Bert-uncased trained with financial corpus. Financial text classification is added to FinBert in fine tune training. To feed the training data set to language model, they have been concatenated. The sentiment labels were convertor to numeric labels 0, 1 and 2, for “neutral”, “positive” and “negative”*.* Preprocessing stages which are tokenizing the headlines, lemmatizing, stop word and punctuation removal from the headlines were done by BERT tokenizer. Test, train and evaluation sets are split with 80-10-10 method. After fine-tuning was completed, financial news data gathered from another source were fed into fine-tuned model to perform sentiment analysis.

Financial news data is obtained from two resources which are SeekingAlpha [3] & Benzinga [4]. These datasets include more than 8000 news headlines between 2010 and 2020. Obtaining financial news data is quite hard and expensive. Thus, in this study, available news is used. SeekingAlpha news data consists of entirely analysis about Tesla. The reason why this dataset is used that the headlines of analysis are unambiguous which means that it is more convenient to classify them regarding to their meaning. Most of the time, the general news uses a general language which means it is not biased and it is neutral. Thus, this does not affect sentiment label and score. Benzinga dataset is bigger and more ambiguous. This means that news headlines have more formal and general language. It must be said that the merged dataset consists of mostly positive and neutral. To use them, two datasets are merged. The data is processed such that data looks like headline-date-ticker. In merged dataset duplicates are removed to avoid having the same elements in training dataset and evaluation or test dataset. Then, to find news about Tesla filtering is done with the words in the headlines such as including "tesla", "elon", "musk", "electric car" or "tsla" and the tickers which contain "tsla", "tslaq" or "tesla". After filtering, total 3500 data points are gathered. Filtered data is given to the language model as input and outputs are sentiment label and sentiment score. Then, since some days have more than one financial news, news in the same day, scores are added for positive labels and subtracted for negative labels, and sentiment labels are changed regarding changes in the score. As a result, each day has unique score and label. In this part, total 800 data points are obtained. Then, the score is normalized to get meaningful data. In addition, sentiment labels of dates which have below 0.10 score are changed to “neutral” to achieve more coherent data.

For historical financial data, Yahoo Finance API has been used [5]. The API gives high-values, low-values, open-values, close-values, trading volume and adj close-values of the Tesla stock for a given date. High and low values give highest and lowest price of the stock at the given date. Open and close provide opening and closing price of the stock. Volume data value indicates number of trades has been done in the given period. In other words, it indicates activity of the stock. Since prediction model uses date, open, close and volume, from the API, a data set is created and it is adjusted accordingly. Dataset includes daily stock prices and with indicators which mentioned above. The main reason why daily data used is that in used news dataset, there is not enough news to cover intraday changes. Therefore, using intraday stock data to predict movement in the stock is meaningless with the available news dataset. Then, by looking the date, each day’s stock data is matched and merged with its sentiment score and label from the dataset which is obtained before. This dataset will be used to train an LSTM stock prediction model whose inputs are the values provided above and outputs are next day’s adjusted price. Therefore, following day’s adjusted closing prices were added as training data labels. Hereby, dataset is completed for prediction model training and optimizing.

## Language Model

As was introduced in the previous sections, a pre-trained BERT model is chosen for the sentiment analysis part. Since training a finance-specific language model from pre-trained “bert-base-uncased” requires a vast corpus, FinBERT model by Yang et al. [9] was chosen as the general-purpose financial language model to start with. As the tokenizer, BertTokenizer is used as it is readily available and successful in sequence classification tasks. To achieve sentence classification, BertForSequenceClassification object in Python was called with the FinBERT model. Our BertForSequenceClassification object uses the tradition approach of feeding the output [‘CLS’] token of the input sequence to the classifier head to be further sent to a softmax function as logits before reaching the final prediction state. This [‘CLS’] token has a default length of 768 and represents the sentence embeddings. A simplified diagram of this process can be seen in the figure below. Fine-tuning only changes the checkpoint values in the classifier head.

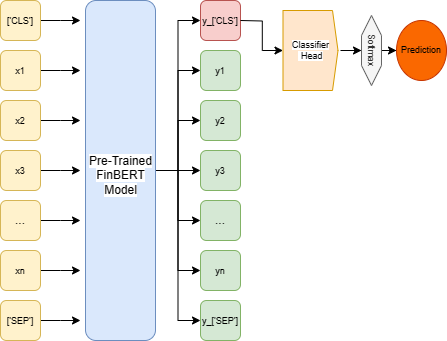


Fig 2. The simplified diagram of the FinBERT model, where the [‘CLS’] token output is directly fed into the classifier head as a sentence embedding.

Our custom dataset explained in Part A was called to be preprocessed. A total number of 6568 labelled sentences were split into train, test and validation sets, corresponding to 80%, 10% and 10% of the total data respectively. Sentence preprocessing was performed on the input data and PyTorch library was used to create formatted tensors consisting of the input ids, token type ids, attention masks and labels.

For the training, several parameters were set as training arguments, such as the evaluation strategy, batch size, learning rate, number of epochs, weight decay and evaluation metric. For the evaluation metric, accuracy was selected. It is the simplest metric to maximize and can be replaced with more complex metric on demand. Minimizing the loss is another option for optimization metrics. For our model, the loss is calculated by PyTorch as cross-entropy loss and the classifier is a Softmax function. Other arguments were tested in different trials by an optimizer and the corresponding plots are provided in the results section with brief explanations.

After setting the training arguments and hyperparameters, the model was trained on the given dataset. The first results for training accuracy, training loss, evaluation accuracy and evaluation loss were received. Then, Optuna was used to study and optimize these hyperparameters to maximize the accuracy. After the optimization process, the best parameters were recorded and the training was run once again with these parameters. The final model was saved.

To obtain the sentiment analysis results of news headlines, the news headline dataset described in Part A was fed as input to the model. The sentiment labels and the scores were recorded along with the headline dates. This data was saved to be used later in the stock prediction model. This concludes the fine-tuning steps followed to construct the sentiment analysis model. The model can be reached from the HuggingFace API as “rifatozkurt/finbert-sentiment-v1”. The performance of the model will be discussed in the “Results” section.

## Long-Short Term Memory (LSTM) based Stock Prediction Model

Because of its success in yielding highly accurate results for time-series future prediction, LSTM architecture is quite popular for financial forecasting tasks. LSTM architecture is based on Recurrent Neural Networks (RNN), which are able to capture nonlinear short-term time-series data. Addressing the vanishing gradient problem of RNN which prevents the networks’ capability to learn long-term dependencies [16], LSTM architecture is proposed by Hochreiter and Schmidhuber [17] in 1997. Their modification on traditional RNN architecture is the addition of a cell state, which allows LSTMs to remember or forget information over long sequences. The LSTM architecture as a network topology is as follows:

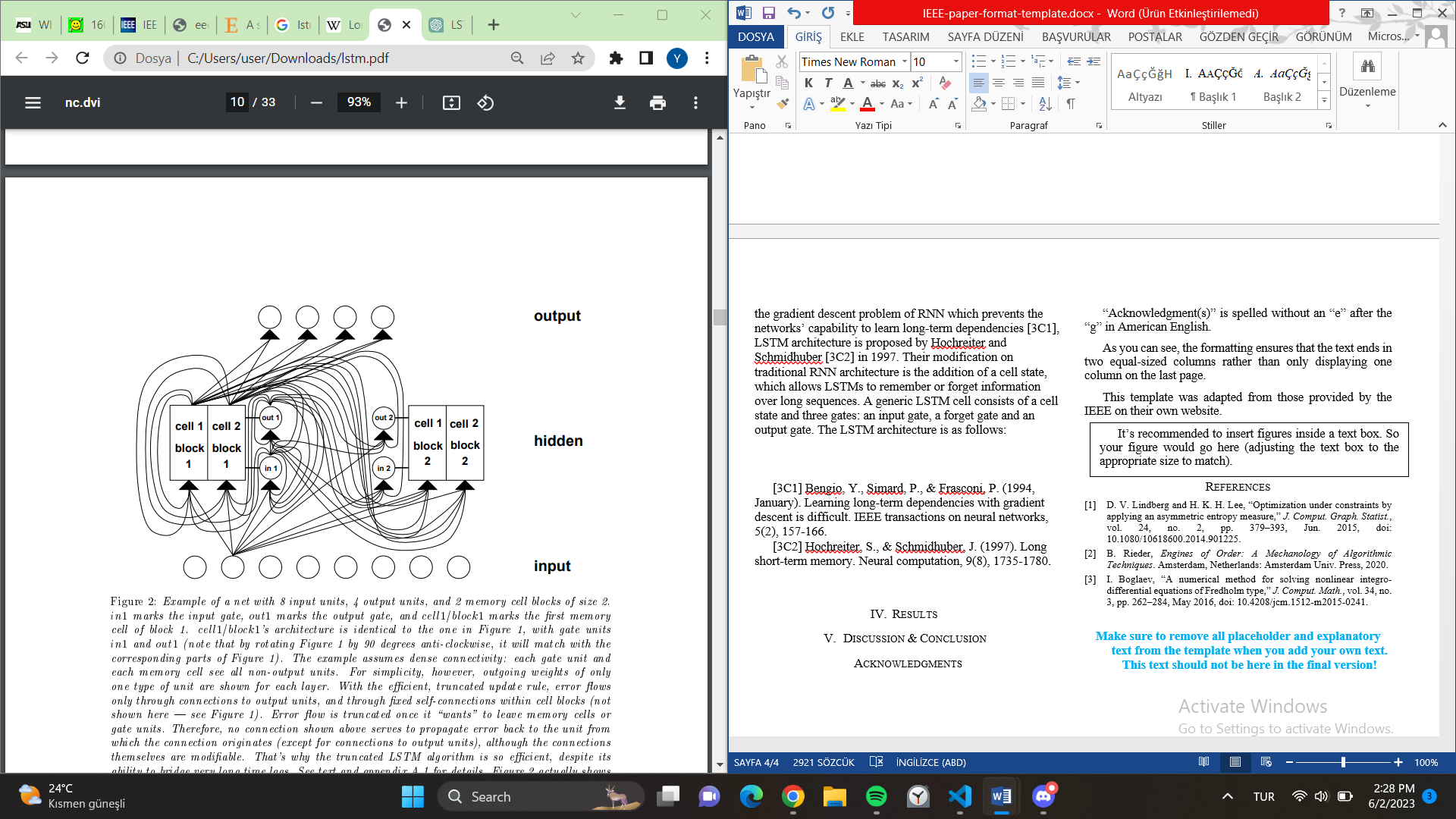


Fig 3. A LSTM network topology with 8 input, 4 output units and 2 memory cell blocks [17].

There exists a cell state and four gates in a generic LSTM memory cell block: an input gate, a forget gate, a cell state and an output gate.

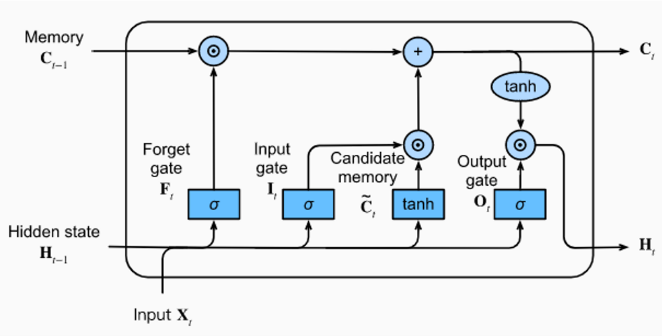


Fig 4. An LSTM memory cell block [18].

An LSTM unit receives three vectors as inputs, which means it takes a 3D tensor as input. Two vectors are generated from LSTM, specifically from cell state and hidden state for the instant t-1. The third one is the user input, in our procedure is called “X” containing time-series data of “High”, “Low”, “Open”, “Close”, and “Volume” values of Tesla stocks and “Sentiment Score” that we previously obtained as outputs from our language model by inputting Tesla Analysis articles. Since our purpose is to predict the next day’s “Closing Price ()”, we are inputting the historical financial and textual data () of the previous day. The reason for including financial information and sentiment scores other than closing price is to increase the robustness of the LSTM, which is proved to be the case according to Lindemann et al. [19].

Then, we defined our own model, namely “LSTM\_model(nn.Module)”, which inherits the base class “nn.Module” from PyTorch for all neural network modules. After initializing model parameters “input\_size”, “hidden\_size”, “num\_layers” and “num\_classes”, we fully connected the last layer of nn.LSTM module with nn.Linear module. Those modules from PyTorch library are utilized to apply a multi-layer LSTM RNN to an input sequence and a linear transformation to the output vector of LSTM layer. This object returns a decoded version of tensor of shape (seq\_lenght, batch\_size, hidden\_size). This module uses recent past information (short-term memory) and new information (input vector) to update the long-term memory (cell state). Then, long-term memory is used to update short-term memory (hidden state), which also is the output at instant t [17]. Noting that the vectors that will be part of the LSTM input are set in the t+1 instant, which allows to make predictions with this architecture.

The fitting and evaluating model is handled by a function “model\_fit\_eval”, which also uses the training object “LSTM\_model” for hyperparameter search and optimization. We defined “calc\_mse” function to calculate Mean Squared Error (MSE), which is decided to be our error function, inherited from “torch.nn.MSELoss()” module of PyTorch. MSE of regression is for evaluating loss function of the model to which indicates how accurate the model works. This error function is specifically chosen, since our input tensor includes vectors with small standard variation, which enhances the model to eliminate outlier predictions. To reduce MSE, PyTorch’s ADAM is chosen as optimizer, which is able to update attributes such as weights and learning rate over multiple epochs. Lastly, our model “LSTM\_model” is called for forward pass, and the result vector “y\_hat” inverse-transformed since the model takes an already scaled input in interval (0,1) for normalization, MinMaxScaler() of sklearn before training. Note that this normalization procedure is necessary to deal with vanishing gradient problem. After defining the training procedure, the “model\_fit\_eval()” function is called for hyperparameter search, in which RayTune is utilized. In this procedure, the optimizer tries hidden\_size = [2, 3, 5], learning\_rate = [0.0005, 0.001, 0.002], num\_epochs = [2000, 4000, 8000] and num\_layers = [1, 2, 4] with all combinations to obtain the best model. The results of error calculation, the best model parameters and the output after training and validation procedure will be will be presented in Results section.

# Results

In this section, results of language model and prediction models are going to be evaluated and their performance will be examined as well.

## Language Model

Before starting to give results of the model, to evaluate performance of the model, train/loss is plotted for steps.

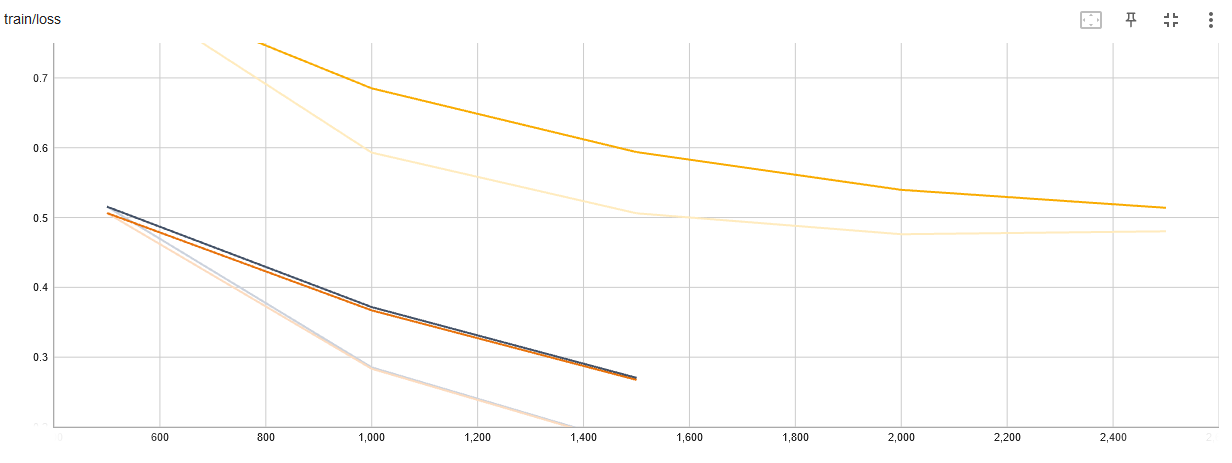


Fig. 5 Train/Loss for Steps Plot

There are three curves on the plot. The above one is the last model. The last model uses bigger, more accurate and better adjusted data as it can be seen from the plot. That model is used to obtain sentiment label and score of the news headlines.

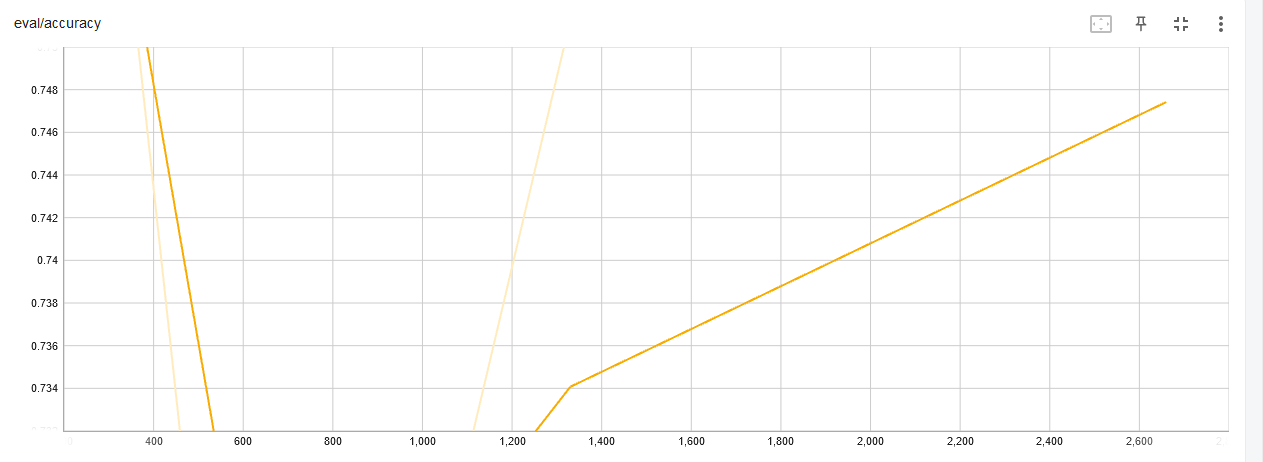


Fig. 6 Evaluation/Accuracy for Steps Plot

This plot shows that after 1200 steps, the model’s accuracy is increasing. This performance is expected because to have a good performing model, accuracy must increase.

1. traınıng results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Training Loss** | **Epoch** | **Step** | **Validation Loss** | **Accuracy** |
| 0.593 | 1.0 | 1330 | 0.5560 | 0.7513 |
| 0.4803 | 2.0 | 2660 | 0.5863 | 0.7631 |

To test whether the model is working correctly or not, some example inputs are given. Label \_0, Label\_1 and Label\_2 are neutral, positive and negative accordingly. Outputs are given below:

1. example headlınes results

|  |  |  |
| --- | --- | --- |
| **Headlines** | **Sentiment Label** | **Sentiment Score** |
| “Growth is strong and we have plenty of liquidity” | LABEL\_1 | 0.9050427079200745 |
| “Tesla goes bankrupt in next five years” | LABEL\_2 | 0.9084979891777039 |
| “Formulation patents might protect Vasotec to a limited extent” | LABEL\_0 | 0.969572126865387 |
| “Good reputation of tesla is damaged with recent events” | LABEL\_2 | 0.8681411147117615 |
| “Elon Musk is now owner of Twitter” | LABEL\_1 | 0.7057536244392395 |
| “Global efforts underway to combat climate change and reduce carbon emissions” | LABEL\_1 | 0.9698044657707214 |
| “Economists are discussing a new theory” | LABEL\_0 | 0.6796321272850037 |

When the table is examined the financial news headlines which have a precise language, it can be stated that scores and labels are correct. The model gives correct and desired outputs. However, the model is positive biased to degree and it can be developed for especially neutral news. The reason why the model has become mildly positive biased is that the training dataset tends to be positively biased. To develop the model, more training data which includes more neutral statements is needed. Therefore, the language model gives such outputs. Overall, the language model is working quite well in financial statements and news.

1. 5 News headlınes results

|  |  |  |
| --- | --- | --- |
| **Headlines** | **Sentiment Label** | **Sentiment Score** |
| “Auto Stock Roundup: GM, Honda, Tesla And More” | LABEL\_1 | 0.8789694905281067 |
| “Tesla CEO Musk Says Other Three Officers Should Be Charged In Floyd's Murder Case” | LABEL\_0 | 0.5824039578437805 |
| “Apple Vs. Tesla: Morgan Stanley Breaks Down The Parallels | LABEL\_2 | 0.8414359092712402 |
| “What to Expect From Tesla Shares After Joining QQQ (TSLA)” | LABEL\_1 | 0.8790620565414429 |
| “The Market In 5 Minutes: Tesla's Announcement, Major Earnings And Jobless Claims” | LABEL\_1 | 0.36116909980773926 |

The results from dataset are expected. The same bias problem can be seen here. The reason is the same as explained above. On the other hand, the news which uses precise and unambiguous language are labeled correctly and their score are logical.

## Prediction Model

In this paper, since the main aim is to predict movement of the stock by looking at the sentiment score. Therefore, as a first step, correlation between stock returns and sentiment score is analyzed.

**

Fig. 7 Correlation Between Daily Returns and Sentiment Scores

As it can be interpreted from the plot, there seems no general correlation between sentiment score and daily returns. Since it is quite hard to predict with just using sentiment analysis of the news. Other parameters added to the prediction model. The training has 10000 epochs and loses are shown below. The model’s best run minimum MSE is 0.368445825962461 with configurations hidden\_size is 5, learning\_rate is 0.002, num\_epochs is 2000 and num\_layers is 2.

1. epoch-loss table

|  |  |
| --- | --- |
| **Epoch** | **Loss** |
| 0 | 0.0375 |
| 1000 | 0.0008 |
| 2000 | 0.0007 |
| 3000 | 0.0007 |
| 4000 | 0.0007 |
| 5000 | 0.0007 |
| 6000 | 0.0006 |
| 7000 | 0.0006 |
| 8000 | 0.0006 |
| 9000 | 0.0006 |
| 10000 | 0.0006 |

To see training period’s performance, prediction and actual prices are plotted.

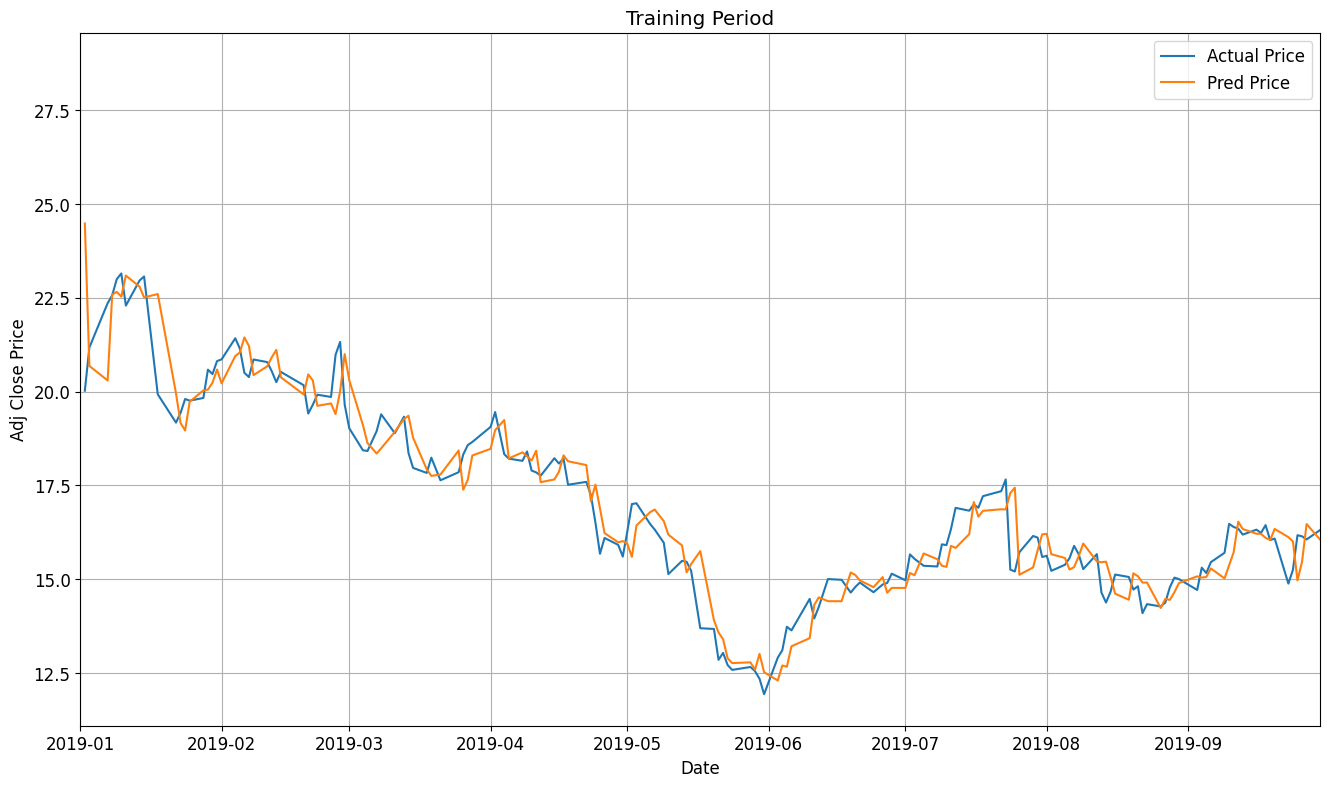


Fig. 8 Training Period Actual – Predicted Price

In training period sentiment score of nine months and stock indicators are use. In addition, MSE of the predicted and real stock price is 0.5837551690630792 which can be developed. In training period, since the trained data is the same with the predicted data, model is predicting the movements. It is almost fitted to real prices. This result is clearly expected. However, it should be considered that this prediction is done with the data which is already trained with. The MSE should have been much lower and the prices should have been closer. Thus, in evaluation step, it can be expected that accuracy will be quite low.

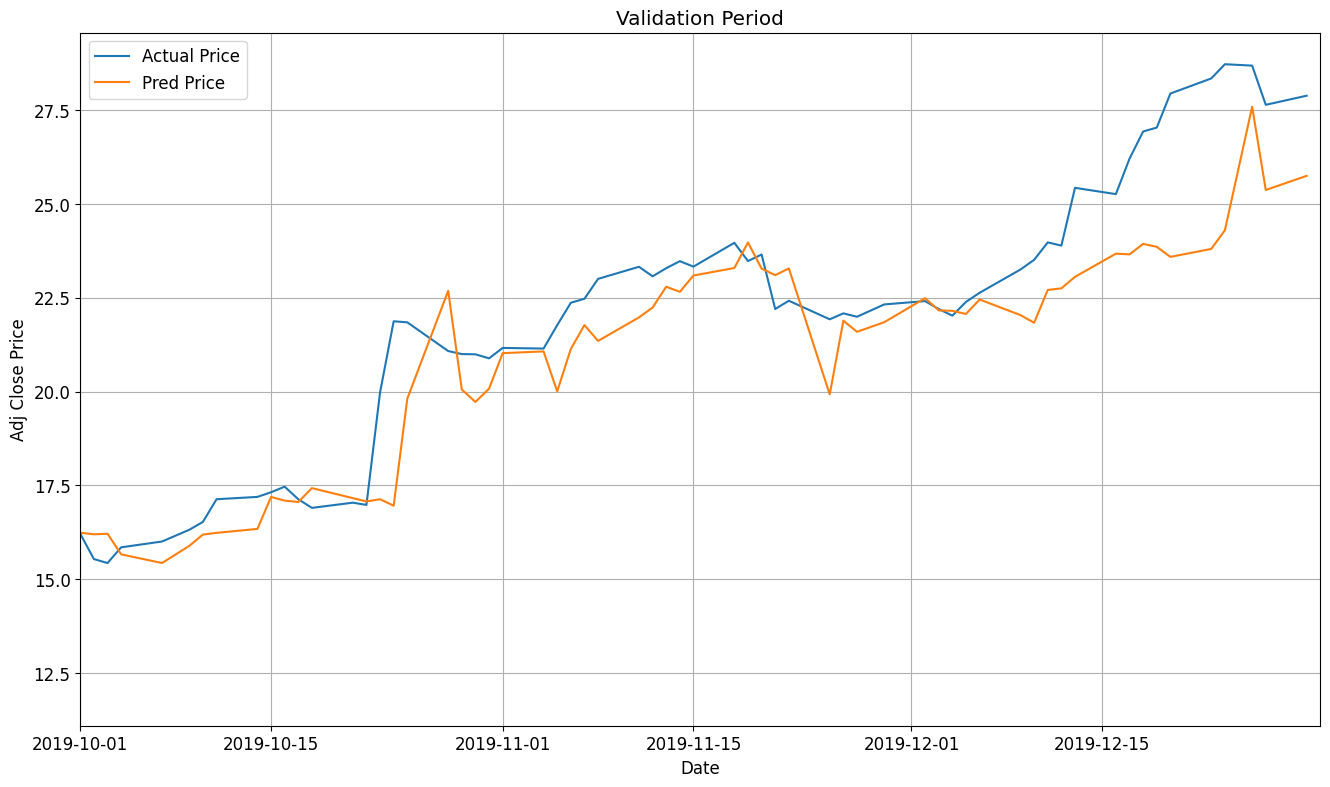


Fig. 9 Validation Period Actual – Predicted Price

In validation period, three months’ sentiment score and stock indicators are used. This plot indicates that predicted and actual price are close to each other and movement prediction is done quite well. On the other hand, the MSE of the predicted and real stock price is 2.266639753118555 in validation. This value is much higher than expected. The reason why plot looks quite well but MSE high is that every day when prediction model is corrected by the selected days closing stock price. This would mean that even if the model predicts higher or lower than actual price, to predict next day it uses last day’s real closing price. Therefore, the plot seems correct. In the next plot, it will be clear to see the real prediction accuracy.

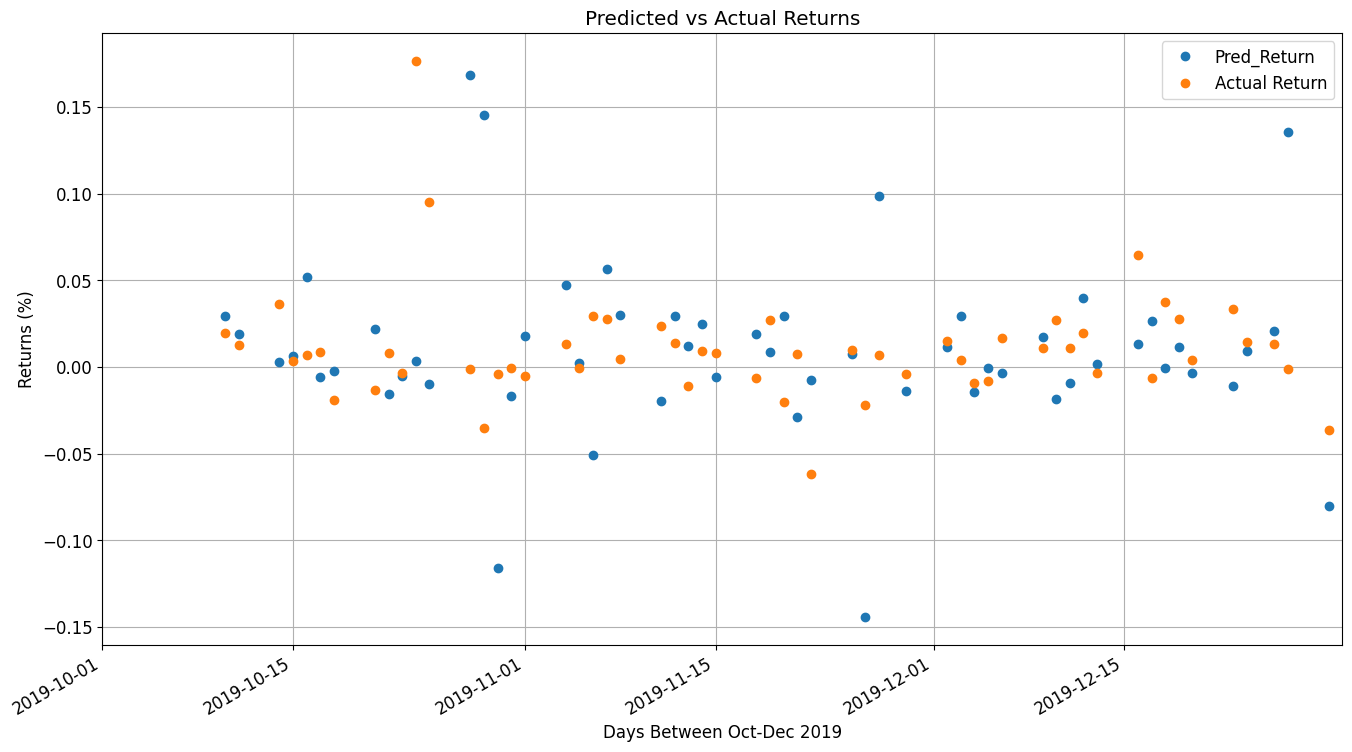


Fig. 10 Predicted and Actual Returns

As explained above paragraph, predictions have some error. In some days, the error is quite low but other days, is high. Actually, the model corrects itself with the last days closing price. Therefore, this feature prevents error to be much higher than the seen on the plot. The error is the result of actual stock prices are not just related with the inputs that given to the model. To minimize the error, number of stock indicators must be increased.

To better analyze the performance of the prediction model, the correlation between the actual returns and predicted returns are calculated and plotted.

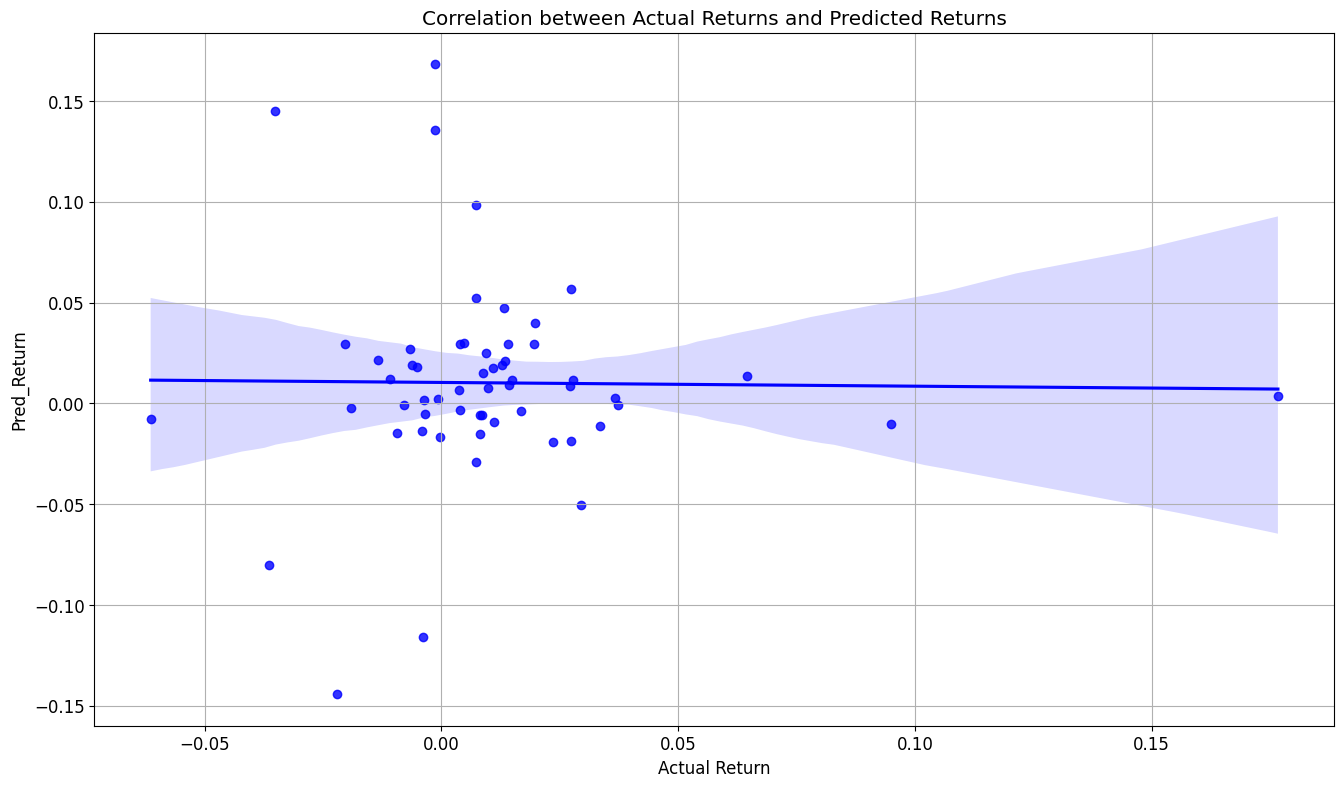


Fig. 11 Correlation Between Actual and Predicted Returns

When the plot is examined, it can be said that there is some correlation between the actual and predicted returns. This correlation might be cause of news which are especially effective. For example, some days, there are news more than just one. This news can be about influential event which is related to Tesla. Therefore, the model is working better for those days. As a result, the correlation can be explained with that type of days.

# Dıscussıons & Conclusıons

In the light of previous work, our model brought partial improvements to the field. The datasets that are used for training our models were merged and precisely preprocessed to prevent overfitting and obtain correct results. Instead of using generic news headlines, we used the analysis articles that are much more unambiguous in order to prevent false labeling. Instead of using sentiment analysis scores of several headlines from the same day, a new method is constructed for generating one sentiment score per day so that it can be used as a daily time-series input to the LSTM model. To reduce the bias in the language model for “positive” label, the inputs with a score smaller than 0.1 is also labeled as neutral. The LSTM model was not only fed with an initial closing price data and expected to predict the next day based on that initial condition. Instead, for the prediction of , the model is also fed with data to obtain more accurate next-day predictions and reduce the error in predicted returns.

Nevertheless, the final stock price predictions were far from perfect and there still exists a lot of room for improvement. For instance, the final sizes of the training datasets for both of the models could have been larger for better results. While similar sentence classification datasets such as Corpus of Linguistic Acceptability were 10000+ sequences, the training dataset of the language model consisted of 6568 sequences. The easily-accessible datasets overlapped greatly and the number of unique sequences was low. This problem occurred for the stock prediction model as well, where finding a news headline that was related to Tesla for each day of the year was an issue. Although we merged two very large datasets, no news article was found for 10 days within the year of 2019, and the number of input dates reduced to 252 after eliminating the weekends.

Another major problem with stock price prediction and news analysis is that real-life prices are not bound to move in a structural way and there are many more major factors that influence the stock price of a company other than the news headlines. The state of the economy and other external events may have much more significant results on the stocks compared to the news headlines. An attempt to partially address this issue was to limit the prediction to a day to day basis, with the assumption that news headlines only have a limited effective timespan of one day. Lastly, it is important to note that out LSTM model was trained using only the news headlines related to Tesla Inc., since different companies may have different sensitivities to news and speculations.

While Araci and Genç [7] and Yang et al. [9]’s work reached up to validation accuracies of 0.87 and 0.97, duplicate data entries were found when their datasets were analyzed and those were not removed in the codes either. It was quite expected that our accuracy metric is lower than theirs for the language model. Even if their accuracy is higher than our language model, the reason for this may have been the appearance of same sequences across their test and train datasets. This was an important issue that was addressed in our implementation. Comparing our final results from LSTM model with Mohan et al. [15] ’s work, we must state that our model still works well enough considering the deficiencies and limitations emerged from our conditions. Our modifications on their model such as using processed sentiment scores from a financial BERT model and other financial indicators as an input of LSTM model, instead of using polarity or text with only closing price compensated the positive effect of their remarkably larger dataset on training their model. The correlation rate between returns and scores also demonstrates the deficiency of our dataset and how neither news sentiment nor historical financial data is enough to obtain remarkably accurate prediction results, which also verifies the Mohan et al.’s findings. Considering that the result of returns of stock and the sentiment scores found to be uncorrelated in our study, our model’s ability of predicting closely the trending shape of the stock price line was clearly a valuable result. Our proposed methodology still stands as a bright alternative for further research, even if not as a benchmark in this field.

In conclusion, according to our results, it can be stated that direct correlation between daily sentiment score and returns is not found. However, when the data with stock indicators is fed to the stock prediction model, the predicted return and actual return can be partly related. As it has been addressed in literature review, stock prediction is a complex problem. There are many factors that affects its price. Therefore, in light of our results, there is not expected findings to add to literature because the results were anticipated. Overall, the methods that been used in the project worked. Certainly, there is a room to improve it. Even if the language model works quite well, for the prediction model is not performing in that level. At some dates, it gives prediction with quite high error. To improve these results, more extensive datasets must be used. The datasets must include more detailed stock indicators which are not available for public. In addition, the more financial news which are from different sources must be included to the datasets. This will solidly improve the model’s accuracy and performance. As a result, the project has given expected outcome and been successful as possible. Stock predictions is a popular topic in this field. Hence, there will be many researches about it. One of them could be try to combine LSTM with a different machine algorithm in order to predict stock prices in a period instead of daily.

##### References

1. P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, “Good debt or bad debt: Detecting semantic orientations in economic texts,” *Journal of the Association for Information Science and Technology*, vol. 65, no. 4, pp. 782–796, 2013. doi:10.1002/asi.23062
2. “Papers with code - fiqa dataset,” FIQA Dataset | Papers With Code, https://paperswithcode.com/dataset/fiqa-1 (accessed Jun. 3, 2023).
3. “Tesla, Inc. (TSLA) latest stock analysis,” Seeking Alpha, https://seekingalpha.com/symbol/TSLA/analysis?page=2 (accessed Jun. 3, 2023).
4. Benzinga, “Actionable trading ideas, real-time news, Financial Insight,” Benzinga, https://www.benzinga.com/ (accessed Jun. 3, 2023).
5. “Tesla, Inc. (TSLA) Stock Historical Prices & Data,” Yahoo! Finance, https://finance.yahoo.com/quote/TSLA/history/ (accessed Jun. 3, 2023).
6. C. Wang and B. Luo. “Predicting $GME Stock Price Movement Using Sentiment From Reddit r/wallstreetbets”. In Proceedings of the Third Workshop on Financial Technology and Natural Language Processing, pp. 22–30, 2021.
7. D. T. Araci, " FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," M.S. thesis, Faculty of Science, Univ. of Amsterdam, Amsterdam, June 2019. [Online]. Available: https://arxiv.org/pdf/1908.10063.pdf
8. P. Malo, A. Sinha, P. Korhonen, J. Wallenius ve P. Takala, “Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts,” *Journal of the American Society for Information Science and Technology*, Vol. 65, No. 4, pp. 782-796, 2013.
9. Y. Yang, M. C. S. Uy and A. Huang, "FinBERT: A Pretrained Language Model for Financial Communications," School of Business, Hong Kong University of Science and Technology, 15 Jun 2020.
10. Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross Mcdermott, Manel Zarrouk, Alexandra Balahur, and Ross Mc-Dermott. 2018. Companion of the The Web Conference 2018 on The Web Conference 2018, {WWW} 2018, Lyon , France, April 23-27, 2018. ACM. https://doi.org/10.1145/3184558
11. Y.-G. Song, Y.-L. Zhou, and R.-J. Han, "Neural networks for stock price prediction," 2018.
12. E. F. Fama, “Efficient Capital Markets: A review of theory and empirical work,” The Journal of Finance, vol. 25, no. 2, p. 383, 1970. doi:10.2307/2325486 Business, Hong Kong University of Science and Technology, 15 Jun 2020.
13. A. E. Khedr, S. E. Salama, and N. Yaseen, “Predicting stock market behavior using data mining technique and news sentiment analysis,” International Journal of Intelligent Systems and Applications, vol. 9, no.7, pp. 22–30, Jul. 2017. [Online]. https://www.researchgate.net/publication/318298991\_Predicting\_Stock\_Market\_Behavior\_using\_Data\_Mining\_Technique\_and\_News\_Sentiment\_Analysis
14. H. Alostad and H. Davulcu, “Directional prediction of stock prices using breaking news on Twitter,” Web Intelligence, vol. 15, no. 1, pp. 1–17, 2017. doi:10.3233/web-170349
15. S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia, and D. C. Anastasiu, “Stock price prediction using news sentiment analysis,” 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), 2019. doi:10.1109/bigdataservice.2019.00035
16. Bengio, Y., Simard, P., & Frasconi, P. (1994, January). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.
17. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
18. O. Calzone, " Architecture of a LSTM Unit," in "An intuitive explanation of LSTM," Medium, 2021. Available: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. [Accessed: June 2, 2023].
19. B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, “A survey on long short-term memory networks for time series prediction,” Procedia CIRP, vol. 99, pp. 650–655, 2021. doi:10.1016/j.procir.2021.03.088

##### APPENDICIES

APPENDIX – A

Every group member has a contribution to the project. At the start, work distribution was data collection, language model research and prediction model research. This is done by Alkın, Rıfat and Yigit accordingly. Then, in the stage of design of the models, language model is designed by Rıfat and prediction model is designed Alkın and Yigit. Thus, every member has completed their job.

APPENDIX – B

CODES:

Data merger.py

import pandas as pd

df = pd.read\_csv('data.csv') ## use your own customized dataset

df.head()

# %%

file1 = open("all-data.csv","r")

lines = file1.readlines()

file2 = open("all-data2.csv","w",encoding="utf-8")

labels ={"neutral":0,"positive":1,"negative":2}

for line in lines:

temp = line.split(",", 1)

if temp[1][0] == '"':

file2.write(str(temp[1]).strip()+ ",")

else:

file2.write('"'+str(temp[1]).strip()+ '",')

file2.write(str(labels[temp[0]])+ "\n")

file1.close()

file2.close()

#%%

file1 = open("data.csv","r", encoding="utf-8")

lines = file1.readlines()[1:]

file2 = open("data2.csv","w", encoding="utf-8")

file2.write("sentence,label\n")

labels ={"neutral":0,"positive":1,"negative":2}

for line in lines:

temp = line.rsplit(",", 1)

if temp[0][0] == '"':

file2.write(str(temp[0]).strip()+ ",")

else:

file2.write('"'+str(temp[0]).strip()+ '",')

file2.write(str(labels[temp[1].strip()])+ "\n")

file1.close()

file2.close()

#%%

file1 = open("data2.csv","r", encoding="utf-8")

file2 = open("all-data2.csv","r", encoding="utf-8")

file3 = open("BIG\_sentiment\_dataset\_NLP.csv","w", encoding="utf-8")

file3.write(file1.read())

file3.write(file2.read())

file1.close()

file2.close()

file3.close()

#%%

file1 = open("analyst\_ratings\_processed.csv","r", encoding="utf-8")

lines = file1.readlines()[1:]

file2 = open("analyst\_ratings\_processed2.csv","w", encoding="utf-8")

file2.write("title,date,stock\n")

count = 0

for line in lines:

print("line: ", count)

count += 1

temp = line.rsplit(",", 2)

temp2 = temp[0].split(",", 1)

try:

if temp2[1][0] == '"':

file2.write(str(temp2[1]).strip()+ ",")

else:

file2.write('"'+str(temp2[1]).strip()+ '",')

file2.write(str(temp[1].strip())+ "," + str(temp[2].strip()) + "\n")

except:

print("error")

continue

file1.close()

file2.close()

#%%

file1 = open("BIG\_sentiment\_dataset\_NLP.csv", "r", encoding="utf-8")

file2 = open("duplicates\_removed.csv", "w", encoding="utf-8")

lines = file1.readlines()

lines = list(dict.fromkeys(lines))

for line in lines:

file2.write(line)

file1.close()

file2.close()

# %%

df3 = pd.read\_csv('duplicates\_removed.csv') ## use your own customized dataset

df3.head()

#%%

import numpy as np

# %%

file1 = open("analyst\_ratings\_processed2.csv", "r", encoding="utf-8")

file2 = open("tesla-news-filtered.csv", "w", encoding="utf-8")

list = []

lines = file1.readlines()[1: ]

for line in lines:

temp = line.rsplit(",", 2)

list.append(temp)

newsarray = np.array(list)

headlines = newsarray[:,0]

dates = newsarray[:,1]

stocks = newsarray[:,2]

filtered\_headlines = []

filtered\_dates = []

filtered\_stocks = []

index = 0

counter = 0

for headline in headlines:

if (("tesla" or "elon" or "musk" or "electric car" or "tsla") in headline.lower()) or (("tsla" or "tslaq" or "tesla") in stocks[index].lower().strip()):

filtered\_headlines.append(headline)

filtered\_dates.append(dates[index])

filtered\_stocks.append(stocks[index])

counter += 1

index += 1

index2 = 0

for headline in filtered\_headlines:

file2.write(str(headline) + "," + str(filtered\_dates[index2]) + "," + str(filtered\_stocks[index2]))

index2 += 1

file1.close()

file2.close()

Stock\_data.py

#%%

import requests

# replace the "demo" apikey below with your own key from https://www.alphavantage.co/support/#api-key

url = 'https://www.alphavantage.co/query?function=TIME\_SERIES\_DAILY\_ADJUSTED&symbol=TSLA&outputsize=full&apikey=Q462BRIDL4V7EAFE'

r = requests.get(url)

data = r.json()

#%%

print(data[1])

# %%

import csv

import requests

CSV\_URL = 'https://www.alphavantage.co/query?function=TIME\_SERIES\_DAILY\_ADJUSTED&symbol=TSLA&outputsize=full&apikey='' &datatype=csv'

with requests.Session() as s:

download = s.get(CSV\_URL)

decoded\_content = download.content.decode('utf-8')

cr = csv.reader(decoded\_content.splitlines(), delimiter=',')

my\_list = list(cr)

for row in my\_list:

print(row)

# %%

len(my\_list)

# %%

file1 = open("tesla-news-filtered.csv","r", encoding="utf-8")

lines = file1.readlines()

def finding\_stock (my\_list,date):

for x in my\_list:

if date in x:

return [x[0] ,x[1], x[4],x[6]]

a=finding\_stock(my\_list,'2020-06-09')

file1.close()

# %%

file1 = open("language-model-outputs\_v2.csv","r", encoding="utf-8")

lines = file1.readlines()

tsla\_dates\_dict ={}

count\_dict ={}

for line in lines:

temp = line.rsplit(',',4)

date = temp[1].strip()

label = temp[3].strip()

score = temp[4].strip()

if date in tsla\_dates\_dict:

count\_dict[date]+=1

if label == "LABEL\_1":

tsla\_dates\_dict[date] += float(score)

elif label == "LABEL\_2":

tsla\_dates\_dict[date] -= float(score)

elif label == "LABEL\_0":

tsla\_dates\_dict[date] += 0\*float(score)

else:

tsla\_dates\_dict[date] = float(score)

count\_dict[date] = 1

for key in tsla\_dates\_dict:

tsla\_dates\_dict[key] /= count\_dict[key]

# %%

import csv

file1 = open("dataforsampleltsm\_v3.csv", "r", encoding="utf-8")

file62 = open("data\_for\_lstm\_v3\_rifat.csv", "w", encoding="utf-8")

labels = {'LABEL\_0': "neutral", 'LABEL\_1': "positive", 'LABEL\_2': "negative"}

csv\_writer = csv.writer(file62)

lines = file1.readlines()

date\_list = []

for line in lines:

temp = line.rsplit(',', 4)

date = temp[1].strip()

#label = temp[3].strip()

#label = labels.get(label)

if date not in date\_list:

date\_list.append(date)

try:

stock\_data = finding\_stock(my\_list, date)

open\_value = stock\_data[1]

close\_value = stock\_data[2]

volume = stock\_data[3]

score = tsla\_dates\_dict[date]

final\_val = float(close\_value) - float(open\_value)

if abs(score) < 0.10:

label = 'neutral'

elif score > 0 :

label = 'positive'

elif score < 0 :

label = 'negative'

else:

label = tsla\_dates\_dict(date)

print('SCORE IS NOT DETERMINED')

#score = temp[4]

except TypeError:

continue

file62.write(date+','+ label+ ','+ str(score) + ','+ str(final\_val)+ '\n')

#csv\_writer.writerow([open\_value, volume, label, score, close\_value])

file1.close()

file62.close()

# %%

import csv

file6 = open("dataforsampleltsm\_v3.csv", "w", encoding="utf-8")

with open('language-model-outputs\_v2.csv', 'r',encoding='utf-8') as file:

reader = csv.reader(file)

header = next(reader, None)

sorted\_data = sorted(reader, key=lambda row: row[1])

# Print the sorted data

for row in sorted\_data:

file6.write('"'+row[0]+'"'+','+row[1]+','+row[2]+','+row[3]+','+row[4]+'\n')

file6.close()

# %%

finding\_stock(my\_list, '2020-01-02')

finding\_stock(my\_list, '2020-12-31')

# %%