PSIMS: Predicting Stock by Inferencing Market Sentiment

Agnes Sharan Sahaya Raj Helan
asr647@nyu.edu

Nehemiah Dureus
nmd353@nyu.edu

Abstract

Accurately predicting the price of a stock into the future, even for a small period of time, can be exceptionally beneficial to those trading on the stock market. Machine learning has advanced to the point where learning from stock market with added context can produce novel results in stock prediction. We introduce PSIMS, a stock prediction tool comprised of two models, one to analyze individuals' sentiment around a particular stock, and one to use this sentiment along with previously seen prices to make a prediction about the next few prices movements. While our system does not take into account every aspect that leads to a stock's movement - a Sisyphean endeavor - it is able to reason about a completely unseen stock's price to a significant degree of accuracy.

1 Introduction

Stock market prediction has been of interest in the field of artificial intelligence because of machine learning's outsized capability of learning from data to make predictions about the future. If a machine learning model can predict the price of a stock, even for a limited amount of time into the future, the model can profoundly change how both financial institutions and everyday investors manage their portfolio.

Previous approaches to stock market prediction have primarily relied on using historical data. However, the stock market can be influenced by many factors and considering these can potentially help us anticipate future stock market movement. One such factor that could be leveraged is the sentiment of the market and its users. This is because much of the volatility of the market is driven by people's emotions.

Our model aims to analyze both market sentiment and historical stock price data, and use this to predict future pricing more accurately than a model that only uses historical stock price data.

2 Related Work

Due to the two-pronged approach of this project, there are two main topics that had to be explored: sentiment analysis and stock prediction.

2.1 Sentiment Analysis

Sentiment analysis has been a popular problem in Natural Language Processing due to its wide spread applications in different domains. An example of this is getting the opinion of the wider public about a product through social media discussions. This is beneficial as it replaces old techniques of using surveys and polls which may not be representative of the user demographics.

Sentiment classification problems are usually looked at as one of 3 types: document-level, sentence-level and aspect-level. For the purposes of this problem statement, the sentiment analysis mode can be considered to be document-level, where the documents are the tweets. It must be noted, however, that textual information is nuanced and thus needs an effective representation in order to be useful in classification tasks. This brought an array of feature selection techniques for problems of this type.

The first known feature selection methods in this domain consisted of using Bag-of-Words (BOW) (Aggarwal et al.) wherein the document is represented by a vector of frequencies of the individual words in it. This approach has two problems: firstly, it doesn't preserve the semantic relationships between the different words in the context of the sentence or document, which is necessary to inference the sentiment of the sentence or documents. Secondly, due to large sizes of language dictionaries, this will lead to a large and sparse vectors which when inference in needed on large amounts of data could prove perilous.

These limitations were overcome by embedding based models such as BERT. These employed a representation of the words by using latent factors.

Latent factors are units of an embedding vector which give information about a word in a multidimensional space, thereby preserving its semantic relationship between other words in the space. This makes these embeddings immensely popular for sentiment analysis tasks (Gao et al., 2019).

2.2 Stock Prediction

Stock market prediction has been a popular problem in the artificial intelligence community due to the immense benefits it stands to bring. Due to the efficient market hypothesis, it is often told that stock market predictions are akin to random walks and thus can't be predicted. However, recent studies suggests that it is a close approximation rather than exact random walk and thus opens the door to exploring factors that influence its movement.

Many studies have been performed on stock market prices using the historical data of a stock. This treats the problem as a time series analysis and attempts to predict the next step in the series. Popular models with this flavor of thinking include linear models such as AR, ARMA, ARIMA and non-linear models such as ARCH, GARCH. Various deep learning techniques were also used such as Multi-layer Perceptron (MLP), RNN, CNN and LSTM.

LSTMs showed most promise out of these methods due to its ability to learn long-term dependencies and thereby emulate the complex non-linear functions that are said to underline the stock market movement (Selvin et al., 2017). However, the usage of these networks in the stock market prediction context has been primarily seen with historical data which introduces the question of other factors that could be incorporated to improve their performance.

2.3 Sentiment Analysis and Stock Market Prediction

Using social media sentiment has been popular since the work of (Bollen and Mao, 2011). However, many of these methodologies rely upon using the pre-processed store of social media data, rather than querying them real time. What this means for these models is that they could be potentially dealing with stale data whose influence on the market is not as impactful. However, through our approach, we will be attempting to predict stock prices using tweets scraped real time and processing them. While this introduces new problems such as increased noisiness in the tweets due to lower processing time as well as possibility of lowered relevance of tweets, it also overcomes the problem of staleness of market information and thus can prove to be a valuable ground for future research.

3 Proposed Method

The architecture of both the sentiment model (for analyzing social media sentiment) and the prediction model (for predicting stock prices) have been discussed below. The combined architectural diagram of our methodology can be seen in figure 1.

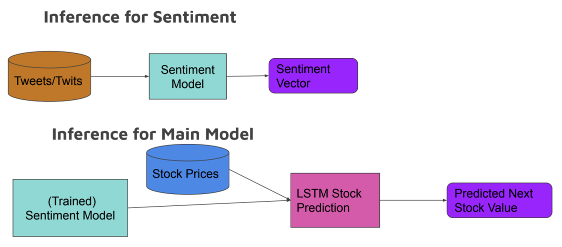


Figure 1: Overall architecture

3.1 Sentiment model

Dataset There were two datasets that were used during the course of the training and inferencing of the sentiment model. The first is the Twitter 140 dataset, This dataset contains over 1.6 million tweets from as well as their sentiments which is provides an useful collection of text similar to the real world data for training our model. These included numerous columns but only the tweet and sentiment columns were used.

The real world Twitter data was obtained from the Twitter developer API. This API although rate limited gives directed access to large social media network which makes it a good source for inferencing our sentiment. We used the search recent API along with the company ticker and returned the 30 most recent tweets which were then processed and sentiment was extracted.

Architecture The Sentiment model has the same architecture as the Tensorflow BERT for Sequence Classification model. This essentially uses a pretrained BERT transformer and adds a classification head on top of it. This is then finetuned without much change to the architecture using the Twitter 140 dataset.

3.2 Prediction model

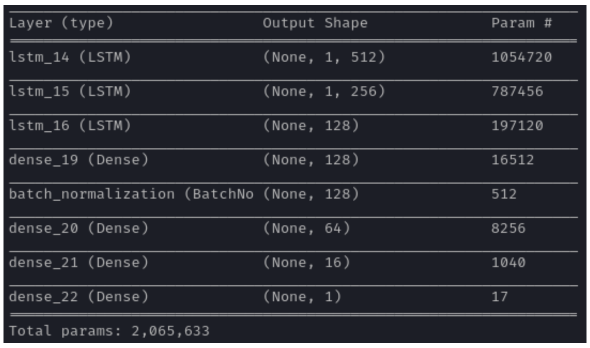
Dataset The data for the Prediction model was derived from the Yahoo finance API. While the API itself has little documentation, it is a good source for free and available stock data.

The API was also chosen over others, because it is not rate limited, offers a wide variety of data granularity, and supports other equities such as options and futures, which provide possibilities for Prediction model to be expanded upon.

The dataset for training consists of stock price series ranging in the thousands to tens of thousands, depending on the time period and range selected. Training is done on 10 different selected stocks. In standalone prediction model training, each stock's series was obtained over the past month, with each closing price being 2 minutes from each other. For training with the sentiment model, due to limitations with Twitter's API points were taken over the past week with a time period of 1 minute.

Architecture The Prediction model is based on LSTM layers which are trained using a previous stock price and an integer that serves as a representation of the sentiment that drives the next stock price. LSTM layers are chosen for Prediction because, unlike an RNN, it allows for key features of the stock series to be learned, while remembering enough of the past to keep learning new information.

Three LSTM layers are used, followed by four Dense layers, as can be seen in figure 2.



```
Layer (type)                 Output Shape              Param #
==============================================================
lstm_14 (LSTM)               (None, 1, 512)            1054720
______________________________________________________________
lstm_15 (LSTM)               (None, 1, 256)            787456
______________________________________________________________
lstm_16 (LSTM)               (None, 128)               197120
______________________________________________________________
dense_19 (Dense)             (None, 128)               16512
______________________________________________________________
batch_normalization (BatchNo (None, 128)               512
______________________________________________________________
dense_20 (Dense)             (None, 64)                8256
______________________________________________________________
dense_21 (Dense)             (None, 16)                1040
______________________________________________________________
dense_22 (Dense)             (None, 1)                 17
==============================================================
Total params: 2,065,633
```

Figure 2: The layers of the prediction model.

4 Implementation Details

4.1 Sentiment Model - Training

The Sentiment model was trained on GCP with P100 GPU. The tensorflow framework was used for the training process. The objective of this model is given a tweet to predict the whether the tweet was positive or negative. The dataset for this model was taken from Twitter 140 dataset and preprocessed.

Preprocessing of dataset For the preprocessing of the dataset, a couple of different techniques were used in combination. The first of these included the removal of any mentions of retweets or URLs in the tweets. Then the tweets were treated with a series of regular expression checks for replacing any existing abbreviations with their full forms. This included the replacement of commonly found colloquial conversation online, eg. brb = be right back.

It also included replacing some commonly found symbols such as $ with a word representative to better the performance of the sentiment model. Then the sentences were tokenized and their stop words were removed. Finally after lemmatizing the remaining words, a join of all the words is returned for training. This collection is then converted to a form that can be processed by the loaded BERT transformer.

4.2 Sentiment Model - Inference

Since this only loaded and used the trained model weights, this was done using Google Colab GPU mode. The objective of this section was to use take in a stock ticker, a list of UNIX timestamps, extract its corresponding tweets and use the trained model to predict the sentiment of these tweets. Then these sentiments were averaged for a query UNIX timestamp and a list of such averaged sentiments were returned.

Scraping and processing of tweets The raw tweets for a particular UNIX timestamp and stock ticker were obtained from Twitter Developer API. In order to ensure that the tweets were of the stock of interest, the ticker of the stock was supplied the API when making the call for tweets. Additionally, the tweets were restricted by their end date to that in the UNIX timestamp in order to get the most recent tweets related to the ticker. To account for the fact that keywords for each ticker symbol maybe different, the 30 most recent tweets were taken satisfying these criterion.

The processing of the raw tweets was done in a similar fashion as the preprocessing of dataset in the training process. This included expanding the abbreviations, removing URLs and retweets, removing stop words and punctuations and tokeninzing and lemmatizing the the words. This processed tweet is then supplied to the trained model and the inferred sentiment is obtained. The 30 sentiments for a UNIX timestep thus obtained are averaged together and supplied to the prediction model.

4.3 Prediction Model - Training

The Prediction model was also trained on GCP with P100 GPU, although a GPU with less compute can train this model in a reasonable amount of time. The tensorflow framework was used for the training process. The goal of this model is to predict the next time step of a stock given past sentiment and price. The dataset for this model was taken from Yahoo's Finance API.

preprocessing The main consideration for preprocessing the points is to ensure that they fit within the range $[0, 1]$, so that the LSTM model produces reasonable results. A Scaler is used on every point to scale each stock price down to this range. As a consequence, the model predictions can only fall within this range, limiting the inference capabilities. However, this will not affect the model's ability to predict, just the ability to inference. Therefore, if an inference suggests that the price is moving outside of the range, the Scaler will simply need to be adjusted.

overfitting Overfitting on the prediction model during training would result in a model that matches the stock price given by the training data, but would end up presenting strange, nonsensical behavior for unseen stock time series. In experimenting with different model architectures, overfitting presented as oscillating mean square accuracy.

It should be noted that a major challenge was keeping the model from over-fitting on the data. Many different techniques were tried to prevent this, with failed solutions including dropout regularization.

In the end, L_1 and L_2 regularization were applied to the dense layers, the learning rate was reduced to $\alpha = 0.00001$, and the number of epochs used to train the model was cut at a time before the loss function begins oscillating. With this structure the model was able to train the data points without suffering from over-fitting. The model architecture can be seen in figure 2.

model complexity Underfitting on the prediction model during training would result in a model whose prediction values blow up to either 1 or 0 after one or two time steps. This issue was due to two factors. First was that the model was too small, so it was unable to learn how to fit the data correctly to predict new stock information. The second was that it trained for a short period of time, reducing it's ability to learn meaningfully.

This was solved by adding more layers to the prediction model, and training for up to 75 epochs. Care had to be taken not to overfit, so the learning rate was reduced and batch normalization was added between the dense layers.

As a result of these model architecture considerations, the mean square error of the model was about 5 on the testing set, and the model was robust enough to predict stocks well into 5 timesteps.

4.4 Prediction Model - Inferencing

To extract information from the prediction model, we use a sentiment value and a stock price as the input. The model will then predict the price of the stock for the next time step.

To continue prediction, we feed this stock price back into the model, using the same value for sentiment. We must assume sentiment is the same for our predictions into the future, since, for our work, we do not predict how sentiment could change into the future. After a certain number of predictions, or interpolation steps, we obtain the current price of the stock and it's sentiment for it. This refreshing step is key for preventing prediction error from increasing, as shown in figure 3. In order to interpret the results, we compare the price of a stock not used for training, testing or validation to what the model would predict.

5 Results and Analysis

5.1 Sentiment Model

Although the sentiment model is not the focus of this project and rather a step along the way of making a better prediction model, it would give some context to discuss the results of the training of process of the sentiment model in this section. Due to the large size of the BERT embedding model and the resource allocation problems posed by it, the sentiment model was not trained for many epochs on the Twitter 140 dataset but still gave an accuracy of about 77%. However, the large size of the dataset along with the accuracy of the trained model on this dataset from the small number of epochs goes to show that the model has generalized well for our usecase and thus can be used as an inference engine for predicting sentiment for consumption by our stock prediction model.

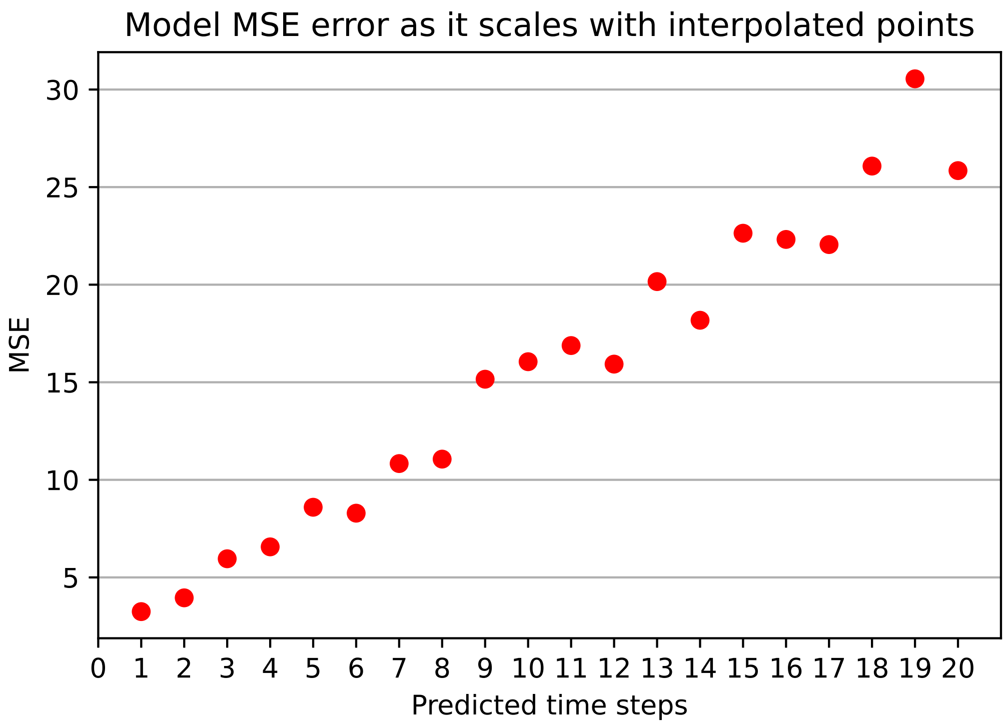


Figure 3: As the number of guesses from the prediction model using a previous stock price and sentiment increases, the mean square error increases linearly. Due to this, the model is best suited for predicting 2-5 time steps into the future. While a human may struggle to make trading decisions in a 4-10 minute window, a trading algorithm could utilize this extra data to help it attempt to beat the market.

5.2 Prediction Model

In figure 3 we can see how predicting further into the future affects MSE. There seems to be a linear relationship between the two metrics. However, keeping the amount of interpolation time steps to a minimum will keep the error to a minimum, and will still provide enough time to act on the model's inference.

Figure **??** and **??** show what the resulting time-series graphs look for predicting Google's stock, GOOG, into the future by two, three, and ten timesteps. Here each timestep is 2 minutes. Note that the error that compounds for 9 steps makes figure **??**'s timeseries significantly worse than figure **??**'s. The mean square error values for these graphs can be seen in figure 3. They are in line with the relative error values calculated in (Selvin et al., 2017).

5.3 Reproducibility

These results for the prediction model are reproducable, but due to weight randomization and other random factors that would effect the training of a model, during some training runs the error is significantly large. An example of these off results can be seen in figure 7.

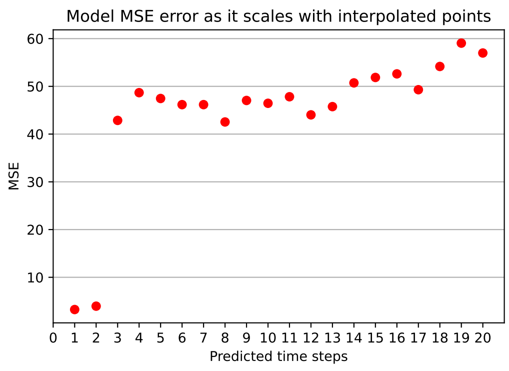


Figure 7: An example of what happens when prediction model's MSE is large, due to a bad training round. Note that the model can still predict 2 steps forward into the future, but anymore results in a high MSE.

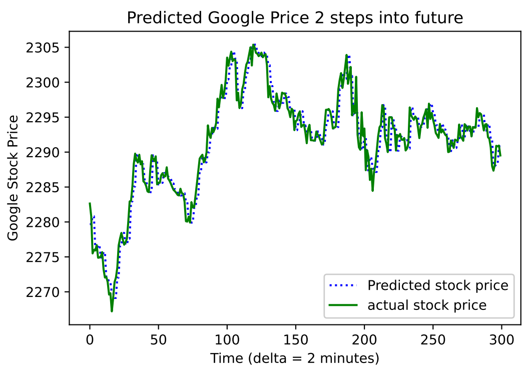


Figure 4: Predicting GOOG one timestep into the future

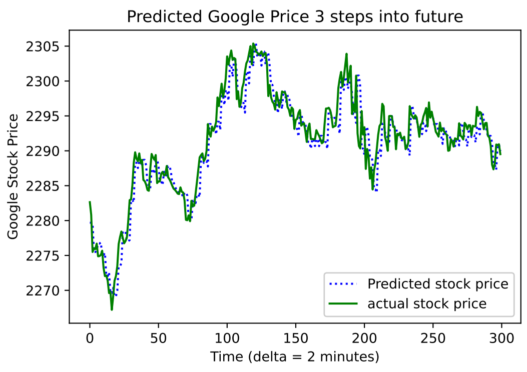


Figure 5: Predicting GOOG two timesteps into the future

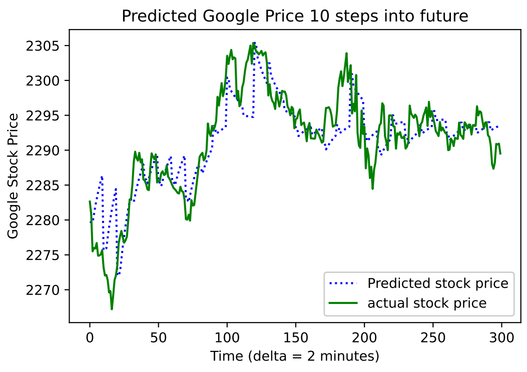


Figure 6: Predicting GOOG ten timesteps into the future

In (Selvin et al., 2017)'s work, they describe a relative error anywhere from 4-7% for using LSTMs to predict the price of shares. For looking 1-2 time steps ahead, our mean square error of 5-7 (at most 0.35% relative error on average) has beat this value.

6 Conclusion and Future Work

Using the interpolated technique we can achieve a stock prediction with single digit mean square error up to 4 time steps into the future. This is enough time for a computer to make some meaningful trading decisions.

This method requires that the real time stock data and sentiment is still used in between inferences of the model, to keep providing extra context to the model's decisions.

Sentiment analysis can be improved on, further expanding on tools like Finbert (Araci, 2019), but to get a fuller understanding of where a stock will go, more information beyond sentiment would be needed. Information related to the economy, geopolitical musings, data about a company, and more would describe a fuller picture for the heading of a derivative. Even then it would have to adhere to the same interpolated strategy outlined here.

It would be interesting to try transformers instead of LSTMs for prediction ((Li et al., 2019)). The current implementation is bound by the compute resources but in a production scale system, it would be interesting to see if adding transformers instead of LSTMs would be helpful or not.

References

Archit Aggarwal, Bhavya Gola, and Tushar Sankla. Analysis of feature selection methods for text classification using multiple datasets.

Dogu Araci. 2019. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv:1908.10063 [cs]*. ArXiv: 1908.10063.

Johan Bollen and Huina Mao. 2011. Twitter mood as a stock market predictor. *IEEE Annals of the History of Computing*, 44(10):91–94.

Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. 2019. Target-dependent sentiment classification with bert. *IEEE Access*, 7:154290–154299.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Advances in Neural Information Processing Systems*, 32.

Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE.