Relating Social Media to Stock Movements

Emily Jin, Danny Tse, Tim Wu

1. Introduction

From the GameStop (GME) boom in early January in which a Reddit user’s hundreds of thousands of dollars became tens of millions, to the market crash in 2020 in which the S&P 500 index dropped 34%, it is clear that the stock market is an extremely volatile yet effective way to make money. Undoubtedly, as evidenced by Elon Musk’s tweets sending GME "to the moon", and the millions of users rallying on r/wallstreetbets hoping to drive DogeCoin’s price up, there is some correlation between market movement and rhetoric on social media. We apply machine learning techniques, particularly linear regression and neural networks, to model and predict the relationship between the sentiment regarding industries and companies on social media platforms such as Reddit, and the prices of the mentioned stocks. In our model, we provide social media posts regarding certain stocks, along with their like ratios and the mentioned stock’s price change in the last 5 trading days. Our model predicts the stock movement for the next 5 days, more specifically, a number 0-4 representing whether to buy, hold, or sell at varying confidences.

2. Motivation and Related Work

2.1. Motivation

The stock market is notoriously difficult to consistently predict; the Efficient Market Hypothesis, as discussed by (Nguyen et al., 2015), argues that out-performing the market in the long-run is near impossible because the constant search for predictable forecasts to exploit trading opportunities ensures that any such patterns will self-destruct in the long run when discovered by a large number of investors. Perhaps the most notable example of such a discovery by the masses is in the recent boom of r/wallstreetbets. Just a small yet robust discussion board a few years ago, the subreddit has seen itself balloon to 10 million subscribers to the most active subreddit in the world, winning over the attention of both the mainstream media and Wall Street. According to the Efficient Market Hypothesis, any advantages provided by such a message board are sure to have imploded by this point. By analyzing the sentiment of discussion in r/wallstreetbets over the past 10 months, tracking the subreddit from well before the GME boom to the present timeframe (May 14, 2021), we observe whether this Efficient Market Hypothesis holds through the discovery of r/wallstreetbets by the masses. Perhaps in doing so we even contribute to the Hypothesis itself.

2.2. Related Work

Using algorithms to understand the stock market is relatively common in modern finance. A quick Google search illuminates that approximately 70 – 80% of trading in the USA is algorithmic. Hedge funds like Two Sigma and Citadel have dedicated divisions of their workforce devoted to applying technological methods such as artificial intelligence and machine learning, for its trading strategies. Predicting the stock market, however, is an inherently nontrivial task. According to (Granger & Timmermann, 2002), degrees of accuracy of 56% hit rate are often reported as satisfying results for stock prediction. In this same project Nguyen et al. demonstrated how such a baseline hit rate could be exceeded through the use of sentiment; although this sentiment was "topic-sentiment" such as the company’s product and service rather than social media sentiment, this model had directional accuracy of 57.1% and thus demonstrated the relevance of sentiment data outside of pure stock technicals (Li et al., 2017).

A very early occurrence of the use of social posts for stock movement predictions was by (Schumaker & Chen, 2009), where they collected stock quotes from relevant financial news articles as input into a support vector machine. While this model was able to predict stock direction with a 57.2% hit rate, besting the 56% threshold mentioned above by Nguyen et al., due to the simplistic nature of the SVM model, significant improvements have been made in using Machine Learning to predict stock movement from social posts.

One famous model by (Qin et al., 2017) uses a dual-stage attention-based recurrent neural network (DA-RNN) to predict future price with the past stock movement. While this model does not employ any external social media data, such a model has served as a good baseline in the literature to
compare against and ensure relevant improvements to prediction can indeed be made through the inclusion of social media sentiment. A notable breakthrough in the use of social media for stock predictions occurred when (Wu et al., 2018) introduced the Cross-modal attention based Hybrid Recurrent Neural Network (CH-RNN) in order to predict stock price from social media data pulled from Twitter. This model could predict stock direction with an hit rate of 59.1% and consistently had higher average daily profits than the aforementioned DA-RNN model by Qin et al. Following this model, (Liu et al., 2019) used a transformer-based capsule network to predict stock prices based on social media post data also pulled from Twitter. This model blew CH-RNN out of the water, with a very impressive stock directional prediction accuracy of 64.2% that resulted in significantly higher revenues than CH-RNN.

While the above research demonstrates that social media has certainly been used in stock prediction, published research on the sentiment analysis of r/wallstreetbets is lacking. In addition, we noticed that while much previous research has focused on the binary task of prediction stock direction, few have followed the conventional financial model of rating stocks on a 1-5 scale the way our model does.

3. Dataset and Features

For data processing, we pulled data primarily from r/wallstreetbets. r/wallstreetbets was considered to be a significant driving factor in the GME rush early January, gaining over 1.5 million users a night. Currently, there are about 10 million subscribed users, and includes Daily Discussion Threads averaging over 17,000 comments a day and a daily influx of over 500 posts. We pulled comment data over 200 days worth of Daily Discussion threads, extracted company names and tickers from each comment, and labeled each comment with a forward 5-day stock price change from the next market open after the comment timestamps to the market close exactly 5 trading days (representing a traditional 7-day week) after this market open. Following standard economic practice, we then assigned these percentage changes to a number from 0-4. Under this system, 0 represented "strong buy", where the stock of the company mentioned had a percentage increase of over 3 percent, 1 represented "buy", where the stock increased between 1 and 3 percent, 2 represented "hold", where the stock increased between 0 and 1 percent, 3 represented "sell" where the stock decreased between 1 and 3 percent, and 4 represented "strong sell", where the stock decreased by over 3 percent. Furthermore, because we decided on experimenting with a multimodal sentiment analysis model, we added extra information, such as (1) the net upvotes/downvotes of the comment and (2) the percentage change of the stock of the mentioned company in the 5 trading days prior to the post.

The data was crawled through the use of the Reddit API. First, we created a new Reddit account (username 'CS229project') from which to request OAuth tokens and parse through the comments. In order to get the urls for all Daily Discussions over the past 200 trading days, we used the GoogleSearch API, pulling the first result when searching the terms "wallstreetbets daily discussion" concatenated by the date. Then, we used each of these urls as a submission for our Reddit API and observed all the comments. For each comments within each Daily Discussion thread, we recorded a lowercase list of each alphabetic word, the net number of upvotes/downvotes on that comment, all relevant stocks mentioned in the comment, and the previously-mentioned forward 5-day stock price change from the next market open after the comment timestamp to the market close exactly 5 trading days later for all mentioned stocks, as well as the trailing 5-day stock price change for all mentioned stocks.

In order to determine all stocks mentioned in a comment, we initially compiled a list of tickers of all stocks listed on the NYSE, NASDAQ, and S&P 500. However, after noticing the problematically high occurrences in overlap between ticker names and non-relevant mentions (ex: usually, when a Reddit user included the word "win" in their comment, they were not talking about Windfall Geotech Inc., which had the ticker "WIN") as well as the great skew in mention frequency to very few companies inherent in the nature of such a subreddit, we decided to pull a list of wallstreetbets’ 130 most talked-about companies. Whenever the company ticker (ex: GME) or frequently-associated name (ex: "gamestop" or "gamestonk" for GME) was mentioned in a comment, a line was added to our dataset associating the comment with this company. Then, in order to get our trailing and forward 5-day stock price change, we cross-referenced this ticker with the yahoo finance API’s historical data.

By looking at the timestamp of the comment, we could find the price of the company’s stock at all relevant dates and times (market open the first trading day after the comment was made, market close five trading days after this first day, market open five trading days before this first day, and market close on this first trading day) and thus complete each datapoint (lowercase list of words, mentioned ticker, upvotes/downvotes, trailing 5-day change) and label (forward 5-day change).

Overall, we had about 181,000 datapoints which we randomly split into train, dev, and test data with an 80/10/10 split.

4. Methods

As mentioned previously, we employ a multimodal approach to predict the stock’s growth through a classification setup. Instead of predicting the exact percent change, we create
Relating Social Media to Stock Movements

5 buckets for the stock’s growth: strong buy, buy, hold, sell, and strong sell. These correspond to a price change of more than 3%, between 1% and 3%, between −1% and 1%, between −3% and −1%, and less than −3%.

We hope to estimate the stock’s growth using sentiments of Reddit comments, as well as change over the past 5 days and net upvote/downvotes. To do this, we break our model down into three main components: (1) embedding/encoding layer, (2) sentiment analysis, (3) multimodal output layer.

4.1. Embedding

To preprocess our data, we convert raw text into vectors using pre-trained GloVe embeddings, and then we use these embeddings to build a vocabulary. We use GloVe embeddings with 6B tokens and 100-dimensional vectors, and by using these embeddings, we are able to build a recurrent model that considers the order of the words appearing within the text.

4.2. Sentiment Analysis

Our sentiment analysis model, SentimentLSTM, predicts whether a given text has negative (0) or positive (1) sentiment. It takes in the embedded text as input, and it then passes it through an embedding layer, encoder, predictor, and dropout layer. The encoder consists of 2 bidirectional LSTM (biLSTM) layers, and the predictor is a fully connected linear layer. Lastly, a dropout layer is used as a form of regularization (Yuan, 2020).¹

BiLSTM’s are an extension of regular LSTMs and can encode more information. A high level description of an LSTM is that it is a recurrent model composed of three “gates”, an input gate, output gate, and a forget gate, and can keep track of arbitrarily long-term dependencies in the input. A bidirectional LSTM simply trains two LSTMs on the input sequence, allowing us to encode information from the past and the future. This makes BiLSTMs particularly powerful for language processing, as shown by our high accuracy on our sentiment analysis (Li et al., 2019).

We utilized two approaches, to pretrain the sentiment model or not. In both cases, we used the last hidden state in the sentiment analysis before the classification output as the input into our final layers. In the pretraining case, we generated labels for the comments in our dataset using the VADER Sentiment Model, and we used these as true labels when training our own sentiment analysis model (Hutto & Gilbert, 2014). Since the sentiment prediction is a binary classification problem, we used Binary Cross Entropy (BCE) with Logits Loss in our pretraining, which is simply a sigmoid layer followed by BCELoss. The first sigmoid layer restricts the predictions so that they lie between 0 and 1, and the loss of a single sample $(x, y)$ is defined as:

$$
\text{BCEWithLogits} = -[y \cdot \log \sigma(x) + (1 - y) \cdot \log(1 - \sigma(x))]$$

In our pretraining case, we were able to achieve a sentiment prediction accuracy of 99.5%.

4.3. Multimodal Output Layer

Like the name suggests, our multimodal output layer takes our output of our sentiment analysis, and concatenates it with the score from the last five trading days along with net upvotes/downvotes. Our output layer architecture consists of two fully-connected (affine) layers, with an ELU non-linearity between, followed by a softmax with 5 neurons. To recall, we define our ELU non-linearity as

$$
\text{ELU}(x) = \begin{cases} 
  x & x \geq 0 \\
  \alpha (e^x - 1) & x < 0
\end{cases}
$$

where $\alpha$ is a hyperparameter in our neural network. Furthermore, our softmax layer is, given an input vector $\mathbf{z}$,

$$
\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}.
$$

In essence, the softmax function takes a vector of real value scores $\mathbf{z}$, reducing each entry of the vector into a probability value (all entries in the vector sum to 1). With this treatment, we are able to present this prediction task as a classification task, and interpreting the result of our model becomes natural. As we feed a 5-element vector into our softmax equation, we obtain probabilities for each score value; and the predicted value simply becomes the one corresponding to the largest probability score in the softmax output.

Our output layer can be visualized as follows:

![Figure 1. Diagram of Model Output Layer](https://github.com/yuanbit/sentimentLSTM)
As our model solves a multi-class classification problem, we trained by optimizing a multi-class classification loss function, namely the cross entropy loss function. Recall that the cross entropy loss function takes in a vector of probabilities \( \hat{x} \) (our output from the model) and our correct label \( l \), and is defined as:

\[
\text{CELoss}(\hat{x}, l) = -\log \left( \frac{\exp(\hat{x}[l])}{\sum_j \exp(\hat{x}[j])} \right) = -\hat{x}[l] + \log \left( \sum_j \exp(\hat{x}[j]) \right),
\]

where \( \hat{x}[l] \) represents the probability value in \( \hat{x} \) corresponding to label \( l \).

### 4.4. Baselines

We compare our model with two baselines: linear regression with bucketed outputs and a neural network without sentiment input.

#### 4.4.1. Linear Regression with Bucketed Output

For our naive baseline, we used a linear regression model that received a text matrix created with bag of words as input. This model disregards grammar and sentence structure, and merely observes the occurrences and densities of words in comments. The model predicted the percent change directly, and then assigned predictions to one of the five buckets defined previously (strong buy, buy, hold, sell, strong sell). We chose to use this as our baseline, rather than multinomial logistic regression, since the classes are ordinal.

#### 4.4.2. Neural Network Without Sentiment

To better understand the effect of sentiment analysis on the quality of predictions, we trained a baseline neural network that only received the change in the past 5 days and net upvote/downvote as input. The network architecture was the same as the output layer of our model, namely a linear layer, ELU non-linearity, linear layer, and Softmax layer.

### 5. Experiments, Results, and Discussion

#### 5.1. Evaluation Metrics

In order to evaluate dev and test results, we computed the dev/test loss, as well as two metrics we defined as "accuracy" and "closeness". We defined accuracy as the proportion of buckets predicted correctly, and defined closeness as the arithmetic mean of the precision and the proximity from the prediction bucket to the correct label. More formally, accuracy and closeness are defined as

\[
\text{acc} = \frac{\# \text{correct predictions}}{\# \text{total predictions}}
\]

\[
\text{cl} = \frac{1}{2} \left( \frac{\text{acc} + \left( 1 - \frac{\sum_{\text{pred}} |\text{predicted} - \text{actual bucket}|}{4(\# \text{total predictions})} \right)}{\text{acc}} \right)
\]

The accuracy score tells us exactly how precise our model is by only rewarding bucket predictions that were exactly correct, and the closeness score will also reward predictions for close proximity to the correct buckets. Since this is an ordinal classification problem, closeness provides an additional useful metric.

In addition to our self-defined accuracy and closeness scores, we also used the standard classification metrics: precision, recall, F1, and Matthews Correlation Coefficient (MCC). Formally, for each bucket \( i \in \{1, \ldots, 5\} \) we defined these four metrics as

\[
\text{precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i + 1}
\]

\[
\text{recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i + 1}
\]

\[
\text{F1}_i = \frac{2(\text{precision}_i)(\text{recall}_i) + 1}{\text{precision}_i + \text{recall}_i + 1}
\]

\[
\text{MCC}_i = \frac{\text{TP}_i(TN)_i - \text{FP}_i(FN)_i + 1}{\sqrt{(\text{TP}_i + \text{FP}_i)(\text{TP}_i + \text{FN}_i)(\text{TN}_i + \text{FP}_i)(\text{TN}_i + \text{FN}_i) + 1}}
\]

where TP, FP, FN, TN represent the number of true positive, false positive, false negative, and true negative, respectively.

#### 5.2. Hyperparameters

We tuned hyperparameters for both SentimentLSTM and the output layer. In SentimentLSTM, we used a standard batch-size of 128 and then tuned learning rate and dropout probability, which represents the probability that a neuron is dropped. We ran 20 epochs using learning rates of 0.001, 0.005, 0.01, and 0.05 and dropout probabilities of 0.3, 0.5, and 0.7. Then, we saved the model with the best sentiment prediction accuracy, which turned out to be a learning rate of 0.001 and dropout probability of 0.5.

Once the ideal dropout probability of 0.5 was determined, we went on to determine a reasonable learning rate and then tuned the \( \alpha \) hyperparameter, used as the coefficient of our ELU activation function. The results are recorded in table 1; as we can see, we decided on a learning rate of 10^{-3} and found that the ideal \( \alpha \) occurred at \( \alpha = 1.0 \).

#### 5.3. Experiments and Results

After training all of the models, we ran both baselines (linear regression and neural network without sentiment) and our proposed model (neural network with sentiment) using the optimal hyperparameters \( LR = 0.001 \), dropout probability 0.5, and \( \alpha = 1.0 \) on the test set.

We then calculated the precision, recall, F1, and MCC metrics using the predicted results, which are summarized in tables 2, 3, and 4 below:
<table>
<thead>
<tr>
<th>Results (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>$\alpha$</td>
<td>Loss</td>
<td>Accuracy</td>
<td>Closeness</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>0.2</td>
<td>1.483</td>
<td>0.3845</td>
<td>0.4617</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>0.2</td>
<td>1.463</td>
<td><strong>0.4119</strong></td>
<td><strong>0.4905</strong></td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>1.462</td>
<td>0.4113</td>
<td>0.4898</td>
<td></td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>0.5</td>
<td>1.464</td>
<td><strong>0.4193</strong></td>
<td><strong>0.4972</strong></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>1.0</td>
<td>1.468</td>
<td>0.4054</td>
<td>0.4848</td>
</tr>
</tbody>
</table>

Table 1. Evaluation Metrics with Different Learning Rates and $\alpha$

<table>
<thead>
<tr>
<th>Strong Buy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.313</td>
<td>0.245</td>
<td>0.481</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>Buy</td>
<td>0.582</td>
<td>0.009</td>
<td>0.501</td>
<td>0.394</td>
</tr>
<tr>
<td>Hold</td>
<td>0.296</td>
<td>0.013</td>
<td>0.304</td>
<td>-0.128</td>
</tr>
<tr>
<td>Sell</td>
<td>0.411</td>
<td>0.003</td>
<td>0.342</td>
<td>0.681</td>
</tr>
<tr>
<td>Strong Sell</td>
<td>0.139</td>
<td>0.001</td>
<td>0.501</td>
<td>0.427</td>
</tr>
</tbody>
</table>

Table 2. Linear Regression Baseline Metrics

<table>
<thead>
<tr>
<th>Strong Buy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.043</td>
<td>0.524</td>
<td>0.153</td>
<td>0.190</td>
<td></td>
</tr>
<tr>
<td>Buy</td>
<td>0.384</td>
<td>0.004</td>
<td>0.273</td>
<td>-0.451</td>
</tr>
<tr>
<td>Hold</td>
<td>0.591</td>
<td>0.001</td>
<td>0.501</td>
<td>-0.102</td>
</tr>
<tr>
<td>Sell</td>
<td>0.326</td>
<td>0.801</td>
<td>0.641</td>
<td>0.568</td>
</tr>
<tr>
<td>Strong Sell</td>
<td>0.219</td>
<td>0.001</td>
<td>0.501</td>
<td>0.391</td>
</tr>
</tbody>
</table>

Table 3. Neural Network w/out Sentiment Baseline Metrics

<table>
<thead>
<tr>
<th>Strong Buy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.474</td>
<td>0.419</td>
<td>0.733</td>
<td>0.590</td>
<td></td>
</tr>
<tr>
<td>Buy</td>
<td>0.494</td>
<td>0.406</td>
<td>0.505</td>
<td>1.0</td>
</tr>
<tr>
<td>Hold</td>
<td>0.612</td>
<td>0.351</td>
<td>0.602</td>
<td>1.0</td>
</tr>
<tr>
<td>Sell</td>
<td>0.5</td>
<td>0.034</td>
<td>0.668</td>
<td>-0.019</td>
</tr>
<tr>
<td>Strong Sell</td>
<td>0.363</td>
<td>0.671</td>
<td>0.731</td>
<td>0.441</td>
</tr>
</tbody>
</table>

Table 4. Neural Network With Sentiment Metrics

5.4. Discussion

By analyzing tables 2,3,4, we see that our neural network with sentiments outperformed the two baselines across the various metrics. By outperforming the neural network without sentiment, we reason that sentiments are useful in predicting stock growth.

However, the improvement in performance is not as significant as we had hoped. This could largely be due to the inherent noise in the comments that we used in our dataset. With Reddit API limits, we were only able to pull the top 1000 relevant comments off the r/wallstreetbets Daily Discussion Threads. Due to the nature of our webscraping method, we picked the most relevant comments off of the number of upvotes. At the time, we did not consider the inherent bias in the top comments - people are more likely to upvote comments regarding "interesting" stocks, or stocks with significant change. This resulted in our data being particularly polarized, and thus our model predicts the extremes (0 or 4) way more often.

6. Conclusion and Future Work

Inspired by previous work on relating social media to the stock market, we hypothesized that Reddit comments’ sentiments, as well as change over the past 5 days and the number of net upvote/downvotes, would be useful in predicting stock growth. Our proposed model was broken up into 3 components: an embedding layer that used GloVe embeddings, a biLSTM sentiment analysis model, and an FC-ELU-FC-Softmax output layer.

After comparing our model with two baselines, namely a bucketed linear regression model and neural network without sentiments, we noticed that our model outperformed the baselines. From this, we conclude that sentiments are useful in predicting stock growth. However, as mentioned in the discussion, our model’s performance was likely negatively impacted by the polarization in our data. In the future, we wish to either sample more comments per day for fewer days, or sample comments in random fashion, as opposed to our sequential sampling from the highest upvoted comment.

One other problem we found was that our loss function treated categorizations as equals; when training, we would penalize incorrect results equally. On the other hand, we considered "closeness" of our values, where given a true value 2, classifying 3 would be "closer" than classifying 4. One further improvement we could make to our model in the future would be to adjust the loss function (cross-entropy), adding an additional penalty for how far away two scores are. For example, given a true value of 2, the loss classifying 3 should be lower than for classifying 4.

Finally, we noticed that our sentiment analysis model had an accuracy of around 99 percent, it is unlikely that our sentiment analysis component is erroneous. Considering the unpredictable nature of the stock market and the lack of additional information we provide our model, we can improve our model by providing more information. For example, instead of providing a single Reddit post and mapping it to a label, we can feed in the concatenation of multiple Reddit posts from the same day. (Note that this would demand significantly more time and computational power, as this would imply that each Daily Discussion thread could contribute at most one data point for each ticker.) We could also provide additional information by including the trailing change over the past 30 days or frequency of mentions of the ticker in the Daily Discussion thread to the Multimodal Output Layer.
7. Contributions

We each contributed to a different aspect of the project initially, but helped each other out as we ran into difficulties. Tim focused on mining ~ 200 days of Daily Discussion Threads from r/wallstreetbets, pulling relevant comments mentioning tickers, the ticker, the ticker’s change in the past 7 days, and the label. Emily focused on selecting and pre-training a relevant sentiment analysis model and evaluating the model on the validation set. Danny focused on implementing the multimodal output layer and the train/test loops. We collectively drafted the writeup.

References


Yuan, B. Sentiment140 bi-lstm. 