**Abstract**

The intent of this project was to explore time series forecasting as it relates to publicly available web forum data. As a baseline, we tested a well-known method of using historical price data only to predict future price. Throughout all of our experiments, to maintain consistency, we used trailing 30 days data to predict one day in the future. We tested two different custom Long short Term Memory (LSTM) neural networks with varying hyperparameters on 4 different datasets for a total of 12 experiments. The four datasets were comprised of historical price data only, historical price data with averaged daily sentiment scores of all comments related to that stock, historical price data with each comment’s sentiment score at the comment per price per row granularity, and finally raw comment embeddings only. The results show some promise but are inconclusive. There are instances where the models trained on historical price data combined with sentiment scores would give an advantage to an equity trader, but more robust testing would be needed to further validate.

**Introduction / Background / Motivation:   
o (5 points) What did you try to do? What problem did you try to solve? Articulate your objectives   
using absolutely no jargon.**

Time series forecasting is a valuable use case for machine learning. It can be used to inform revenue forecasting, weather predictions, and even equity pricing. This paper outlines several experiments we conducted regarding equity price forecasting using different combinations of natural language processing and metadata as training data sets. The features we used were price related data, namely daily price metadata: open, high, low and volume. We also experimented with sentiment analysis scores from public web forums designated to discussing trading strategies. To do this, we scraped data only related to this stock. The details about how we did this are discussed in the body of this paper.

Given the data is tabular and being windowed on a rolling basis, recurrent neural networks (RNNs) are the go-to architecture for these types of problems. Long-Short Term Memory (LSTM) networks are an improvement on RNN architectures and were the types of models we experimented with in this problem set.

**o (5 points) How is it done today, and what are the limits of current practice?**

Algorithmic time series forecasting is generally a guarded secret by banks, hedge funds, and other types of asset management institutions. Due to the nature of the markets, there is no benefit to sharing publicly what a financial institution’s technical or algorithmic trading strategy is. Qualitative investing strategies are harder to explain and directly replicate, and thus is more widely available for research and discovery. Warren Buffet, for example, publishes his shareholder letters every time his company, Berkshire Hathaway, holds their annual conference [1]. One distinction worth noting is asset managers are quicker to share their qualitative investing strategies after a trade has been made. Because qualitative analysis can change in a more fluid manner compared to algorithmic analysis and strategy, there’s less risk in sharing.

Algorithmic trading strategies can be anything ranging from heuristics to mathematical model-based [2]. Until recently with the advancement of deep learning techniques and hardware that supports it, these types of trading strategies relied on numeric and pricing only data with varying windows and algorithms to try and predict future prices. There is not a large amount of information related to how funds are currently using more widely available text data to try and predict future equity pricing.

**o (5 points) Who cares? If you are successful, what difference will it make?**

If this strategy is successful, it could open up a new style of research analysis for the investment sector – namely, using natural language processing related to public data about that stock to directly infer the future price of a stock. For our research, we used the open source NLTK Sentiment Intensity Analyzer Vader package to boil a comment string to a sentiment score ranging from 0 to 1. This package produces four additional features for augmenting: scores in the following category at the stock per day per comment granularity: neutral, positive, negative, and composite. Additionally, we developed a custom LSTM to train a model that used word embeddings of comments related to a stock on a public forum to predict the next day’s stock price.

**o (5 points) What data did you use? Provide details about your data, specifically choose the most   
important aspects of your data mentioned here: Datasheets for Datasets   
(https://arxiv.org/abs/1803.09010). Note that you do not have to choose all of them, just the most   
relevant.**

We used two different datasets for experimentation, with a total of 4 derivations of that dataset using different join methods. The first raw dataset was price only data from yahoo finance at the day granularity, going back 10 years for one stock.

The price categories are defined as follows (measured daily during trading days only):

* High: the highest price
* Low: the lowest price
* Volume: the number of trades executed
* Open: the starting price

The second raw dataset was from Reddit. Reddit has topic-specific forums known as “subreddits.” For our project, we developed scripts to narrow down on posts, comments, and related metadata made available by Reddit specific to one stock. Given the sheer volume of users in the subreddit we selected, our intent was to conduct “opinion data mining” from the masses as they tried to discuss the likelihood of a company’s stock rising or falling. There is a lot of metadata available from Reddit, we captured only what was relevant for our experiments. Our scraping script collects only posts that have at least a positive rating (Reddit scoring enables a user voting system to rate the post, where each user can cast an upvote or downvote). Inside of each post, we only collect comments that have a positive rating. Further, we clean the comments removing “stop words” as defined in NTLK’s corpus [4], as well as emojis, and then we enforce every word to be lowercase. This is a relatively standard approach for cleaning text data prior to vectorizing it for an ML model.

We built a fairly robust code base to automate scraping of post data, comment data, related metadata, daily price data, data preprocessing for time series for a specific equity, and model metric logging with related figures (loss, periodicity, correlation heatmap, and time series predictions true vs. actual).

**Approach:   
o (10 points) What did you do exactly? How did you solve the problem? Why did you think it would   
be successful? Is anything new in your approach?**

We conducted experiments on four derivations from a total of two datasets. The first dataset was from yahoo finance and was daily price metadata dating back 10 years from the time of writing this paper, ~April 2022. The second dataset was generated via Reddit’s PRAW library, which enables web scraping of posts, comments and metadata related to specific “subreddits.” The largest investing related subreddit we could find was “wallstreetbets”. This subreddit over the past 12 months grew to over 11 million users. Our hypothesis is that there is some correlation between the natural language from comments and posts related to future equity pricing. In order to test this hypothesis, we created a baseline error metric and plots for conducting further diagnostics.

Other research papers used total market sentiment in these forums to try and predict total market movement. Our research narrows down on specific stocks. We determine whether or not a post is related to a particular stock in our code by iterating through the post title and searching explicitly for the ticker. We used ‘AMC’ for our experiments for a few different reasons. AMC had a 10-year price history on yahoo finance, it had significant post volume, and the literal string ‘AMC’ cannot easily be confused with something else in the English language as we were scanning post titles.

In our first test, we modeled the following features (all daily price metadata for trading days only): open, high, low, and volume to predict close price the following day. In our second test, we used price data from the first test, along with average comment sentiment scores from that day, resulting in open, high, low, volume, neutral score, positive score, negative score, and compound score. The third test used all of the features from the second test but at the comment per row granularity, with backfilled pricing for all trading related data. The fourth test, which was the most experimental, was converting the comments directly to embedded dense vectors to predict price. This test was performed at the comment embedding per price granularity.

For all different variations of tests, we used a trailing window of 30 days. We have not yet seen in published research the attempt to use train embeddings produced directly from an independent corpus of comments using a long short term memory neural network. The closest approach we have seen is using sentiment analysis as a feature which we also tested.

We assumed that using longer time windows to predict next day price movement would be less useful. It also required significantly more compute as it produced more tensors that would be held in memory during training time. An interesting next step to improve upon this research would be using similar trailing windows for training days to predict further out in the future, rather than just one day out.

**o (5 points) What problems did you anticipate? What problems did you encounter? Did the very first   
thing you tried work?**

One problem we anticipated was defining the proper granularity of our data given the disparity of the frequency of data we collected from Reddit and Yahoo Finance. We overcame this by using a simple backfill imputing strategy where necessary. For example, if comments were produced on a non-trading day, we imputed the next nearest trading day’s price (in the future) for that row. For instances where there were multiple comments per day, we simply persisted the same price to different comments.

**(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data.**

Important: This section should be rigorous and thorough. Present detailed information about decision you made, why you made them, and any evidence/experimentation to back them up. This is especially true if you leveraged existing architectures, pre-trained models, and code (i.e. do not just show results of fine-tuning a pre-trained model without any analysis, claims/evidence, and conclusions, as that tends to not make a strong project).

As briefly mentioned in the abstract, we conducted 12 total experiments with four different derivations of datasets. For each dataset, we trained a model with the following hyperparameters and architecture:

* Single layer LSTM with 150 units, 10 epochs
* Single layer LSTM with 150 units, 30 epochs
* Three layer LSTM with 150 units, 100 units, and 50 units with $10 \%$ dropout following each LSTM layer

The four datasets we used are described in the previous section, “Introduction/Background/Motivation”. The loss used was “mean squared error” as this was a regression problem where we were trying to predict a continuous outcome, the future equity price. We experimented using PyTorch, Tensorflow, and Keras. We ultimately used the Keras framework due to some nice built in functionality it has for time series formatting, namely pre-configured Python generators.

One challenge with this experiment was that price data was more readily available than comment data. Until the great short squeeze of GME occurred in 2021, Reddit's "wallstreetbets" subreddit did not have 11 million members, and thus comment volume was much lower. Another interesting thing worth noting is that as we conducted exploratory data analysis related to the periodicity charts for price movement and sentiment scores of comment strings, there was massive variance in the sentiment scores compared to the price volatility. We believe that this introduced a large amount of noise and made the model more difficult to train.

See **figure 1** for the periodicity plots of the different price features along with the comment sentiment scores. Figure 1 is at the price per day granularity, with averaged sentiment scores of all comments for that day. **Figure 2** is at the comment per day granularity with daily price persisted for every comment (hence the horizontal continuation and higher density). Note that there was no comment data available for the first ~8 years of trading data on Figure 2, but Figure 1’s data is only persisted where there were comments. We did not plot periodicity with word embeddings as this wasn’t possible in a 2-D visualization. **Figure 3** shows a correlation heatmap of the features used with the dataset in Figure 1 and Figure 2.

Our decision to use LSTMs as the base architecture were due to the time series nature of the data. Every time series related research paper we could find that attempted to also predict future equity pricing used some form of RNN. A nuance regarding time series data is it takes quite a bit of qualitative evaluation of the true vs predicted curves to determine how well forecasting is being performed, in conjunction with quantitative evaluation of the validation and training loss.

A persistent issue that was not overcome was overfitting of the model. In most of our experiments, training loss would quickly converge to near zero, while validation loss would oscillate. Some of the better experiments had less oscillation and some performance, but the curves were fairly unstable and suggests that much more tweaking of the LSTM architecture and/or hyperparameters would be needed to increase the robustness of the model.

There were several instances that showed promise, particularly at the “every comment” granularity with persisted price. With this datasets particularly, the there were a few instances where the model accurately predicted the future price. See **Figure 4 and Figure 5**, where the daily price is persisted horizontally for the total of the comments, and the sentiment scores of the comments suggest that the price will go up intraday. The following day, the price skyrocketed from ~$17 to ~$42.

Similarly, in **Figure 6**, the model seems to both balance real world price predictions and predict yet another massive upswing in price from ~$21 to ~$55. There were certainly more instances where this model predicted future price increases accurately versus future price decreases. This could be because the NLTK sentiment analysis library did a better job correlating sentiment scores with positive comments correlated to price increase vs picking up on negative comments correlated to price decrease. **Figure 7** shows the loss curve for this particular model, which suggests there is certainly overfitting. The model seemed to preserve some predictive nature while simultaneously fitting to the wrong price day over day over sustained periods of time. See X axis ranges ~0 to ~100, ~2250 to ~2750.

**Figure 8** shows what the model trained on only historic price data and no sentiment scores produced. It’s clear to see that the price is not persisted horizontally because this data is at the aggregated day granularity. With the two datasets trained at the day granularity, we suggest there is less value with a 1-day out trading strategy. It’s much harder to clearly see a trading signal. **Figure 9** shows the loss curve related to this model.

**Figure 10** shows the model trained on historic price data with daily aggregated sentiment scores. The only clear difference between these two models is slight dampening of the loss curve, while the predictive time series curves appear very similar. See **Figure 11** for the associated loss curve.

**(5 points) Appropriate use of figures / tables / visualizations. Are the ideas presented with appropriate illustrations? Are the results presented clearly; are the important differences illustrated?**

* EDA
  + Periodicity
  + Loss curve
  + Prediction vs actual curve for validation set
  + 3 curves per 3 windows of 4 datasets (36 plots)
* Tables
  + Hyperparameters per experiment
  + Loss per experiment
  + For the more interesting plots, point out when the model would have been useful by diagnosing curve
* Link to code repo

**(5 points) Overall clarity. Is the manuscript self-contained? Can a peer who has also taken Deep Learning understands all of the points addressed above? Is sufficient detail provided?**

Yes

**(5 points) Finally, points will be distributed based on your understanding of how your project relates to Deep Learning. Here are some questions to think about:**

* What was the structure of your problem?
  + Tabular data, time series
* How did the structure of your model reflect the structure of your problem?
  + Very well. There is little to no debate we should be using LSTMs
* What parts of your model had learned parameters (e.g., convolution layers) and what parts did not (e.g., post-processing classifier probabilities into decisions)?
* What representations of input and output did the neural network expect? How was the data pre/post-processed?
* What was the loss function?
  + MSE (standard for regression)
* Did the model overfit? How well did the approach generalize?
  + Need to model loss curve for validation vs train
* What hyperparameters did the model have? How were they chosen? How did they affect performance?
  + Model architecture
* What optimizer was used?
  + Adam
* What Deep Learning framework did you use?
  + Tensorflow
* What existing code or models did you start with and how did these starting points help?
  + NLTK

**References**

1. <https://www.berkshirehathaway.com/letters/letters.html>
2. <https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp>
3. <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3502624>
4. <https://www.nltk.org/howto/corpus.html>
5. <https://web.stanford.edu/class/cs224n/reports/final_reports/report030.pdf>

**Figures**

A picture containing calendar

Description automatically generated

Figure

**Diagram, engineering drawing

Description automatically generated**

Figure

**Chart, treemap chart

Description automatically generated**

Figure

Chart

Description automatically generated

Figure Chart, box and whisker chart

Description automatically generated

Figure (zoomed from figure 4)

A picture containing chart

Description automatically generated

Figure (zoomed from figure 4)

Chart, line chart

Description automatically generated

Figure

Chart, line chart, histogram

Description automatically generated

Figure

Chart, line chart

Description automatically generated

Figure

Chart, line chart, histogram

Description automatically generated

Figure

Chart

Description automatically generated with low confidence

Figure