Project Report : CS 7643 — Predicting Future Equity Prices Using NLP

Josh Burdett
Georgia Institute of Technology
jburdett6@gatech.edu

Blaine Bursey
Georgia Institute of Technology
bbursey3@gatech.edu

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 elected to focus on a short time horizon due to the fickle nature of retail investor's sentiment. 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 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.

1. Introduction/Background/Motivation

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. Additionally, we experimented with directly training word embeddings (dense vectors) of the comment strings directly to the price. The last method was by far the most experimental and unfortunately proved the least promising. The most promising experiment we ran was combining price metadata with per-comment sentiment analysis, training a neural network on roughly 64k rows of comments, joined with back-filled daily price data.

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. LSTMs are favorable due in large part to their ability to limit exploding gradients, which occurs as time series data is persisted across training epochs. The ultimate objective of this project was to create a competitive advantage to public market equity traders using publicly available natural language text to train a neural network model for forecasting future equity prices.

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 [2]. One distinction worth noting is asset managers are quicker to share their qualitative investing strategies after a trade has been made. Often their qualitative strategies are used as pseudo-marketing strategies to attract more investors. 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 [5]. 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. Our approach differs from traditional NLP, which typically tries to use language based predictions like the next word in a sentence, machine translation, text-to-speech, etc. In our research we did not find an example of a firm or individual attempting to use raw word embeddings related to a stock to directly model the future price of a stock.

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 [3]. 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. Separately, 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. We derived the embeddings from scratch using the vocabulary of the words we scraped from Reddit, a custom dimension for the dense vector, and Tensorflow's library.

We used two disparate datasets for experimentation, with a total of 4 derivations of those datasets using different join methods and preprocessing 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 [3], 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 converting the strings to vectors for processing of a neural network.

We had roughly 64k comments and 2098 trading days (spanning back 10 years) related to AMC. Along with embeddings, this accounted for 9 total features to be used across our experiments to predict future close price.

2. Approach

We conducted experiments on four derivations from two disparate 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. There are several other researchers who have implemented an LSTM neural network to predict equity pricing using only price metadata, all of which was available from yahoo finance. We also found some scarce research about using sentiment analysis to predict stock prices, but have yet to see combining both sentiment scores and price data.

Other research papers used total market sentiment in these forums to try and predict total market movement versus an individual stock's movement [6]. We felt this was way too broad of an approach and losing specificity would hurt the model's performance. Our research narrows down on one individual stock. 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. There was also a very nice correlation between massive price swings and comment volume as can be seen in the periodicity graphs in Figure 1.

The things we kept constant during our experiments were the following: batch size of 34, lagging 5% testing data with 95% historical training data, 30 trailing days as input into the LSTM, predicting 1 day out in the future. The model architecture and epochs were both experimented with. Since we knew that LSTM was the most suitable base neural network architecture, we felt it was more important to standardize a set few variations across the four datasets versus focusing overly on hyperparameter tuning. We did notice that running 30 epochs versus 10 epochs was consistently

more performant across all datasets.

3. Experiments

In our first experiment, 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 experiment, we used the same price metadata from experiment 1, along with averaged comment sentiment scores from that day, resulting in open, high, low, volume, neutral score, positive score, negative score, and compound score only on trading days. Both the first and second experiments used 2098 rows of data for training. The third experiment used all of the features from the second experiment, but at the comment per row granularity, with back-filled pricing for all trading related data (this resulted in 64k rows for this training/test split). The fourth experiment, which was the most experimental, was converting the comments directly to embedded dense vectors to predict price. This experiment was performed at the comment embedding per price granularity, also resulting in 64k rows for the training/test split.

We found that using longer time windows to predict next day price movement would be harmful. We briefly experimented with 60 lagging days and noticed overfitting of the model with reduced performance on the train and validation set. 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. It likely still need to be shorter term, but predicting maybe 7-14 days out would seem like a reasonable next attempt for anyone that wants to build on this research or fork our attached code repository.

(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 back-fill 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.

4. Experiments and Results

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".

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, although it did show some promise.

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 of the graph 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, especially if the value in the forecasting is in being able to make a decision about the predicted future data before it happens.

We found several instances of using multiple LSTM layers with reduced units every layer and a dropout layer in between from other researchers attempting to solve time series problems[4]. We attempted that architecture as well as a single layer LSTM with 150 units, and obviously no dropout with the single layer architecture. 150 units seemed like a reasonable number across 30 historical trading days, or comments, depending on the nature of the dataset.

We closely followed Francois Cholet's Deep Learning with Python, 2nd Ed time series, text vectorization, and

embedding conventions [1]. The deviation we took from known convention was directly trying to model price from raw word embeddings, but the preprocessing followed industry best practices. We used a vocabulary of 5,000 from the 64k rows of comments, with dimension 64. We followed a standard convention for training a regression model using mean squared error as the loss metric. We also used adam as our optimizer, as is less prone to gradient issues that are commonly seen with stochastic gradient descent.

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 windows in the time series prediction graph that showed promise, particularly at the "every comment" granularity with back-filled 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 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 of Figure 4.

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. Lastly, Figures 12 and 13 show the time series results of training a model using only raw word embeddings to predict price, using all of the previous parameters noted for other experiments. The results were effectively useless for this attempt given our architecture decisions, but there may be room for iteration and improvement.

Given our experiments, we believe the most promising combination of data and model architecture are using the comment granularity data with backfilled price metadata, along with the three layer LSTM at 30 epochs. The validation loss floated around .35 (see Figure 7), and the time series prediction graph produced several actionable intraday trading predictions. One possible iteration on our research would be combining price metadata with the raw word embeddings to predict next day's price. This would eliminate the reliance on NLTK's architecture for predicting a sentiment, which then models price, and it would instead model price directly.

5. Appendix

5.1. Code Repository

https://github.com/jobu9395/stonks

5.2. Work Division

Data Wrangling	Burdett	Bursey
Data Preprocessing	Burdett	Bursey
Model Architecture	Burdett	
Embeddings		Bursey
Analysis	Burdett	Bursey

We jointly worked on data pre-processing. When we originally submitted for this project, we intended on using an already curated dataset, but we ended up scraping everything to use current data. We jointly set up the project architecture for web scraping, data cleaning, and pre-processing for the model. Burdet focused on getting working model code using Tensorflow, as well as creating the dataset for combining price metadata and comments at the comment per row granularity. Bursey created the aggregate functions for the sentiment analysis and embeddings experiments, along with the training pipeline for them respectively, leveraging the model architecture that Burdett created.

5.3. Illustrations, graphs, and photographs

References

[1] Francois Chollet. *Deep Learning with Python, 1st Edition*. Manning; 1st edition, 2017. 4

[2] Berkshire Hathaway. Shareholder letters. 1

[3] NLTK. Nltk documentation. 2

[4] Sidra Mehtab Jaydip Sen. A robust predictive model for stock price prediction using deep learning and natural language processing. 3

[5] Shobhit Seth. Basics of algorithmic trading: Concepts and examples. 1

[6] Mixu Xu. Nlp for stock market prediction with reddit data. 2

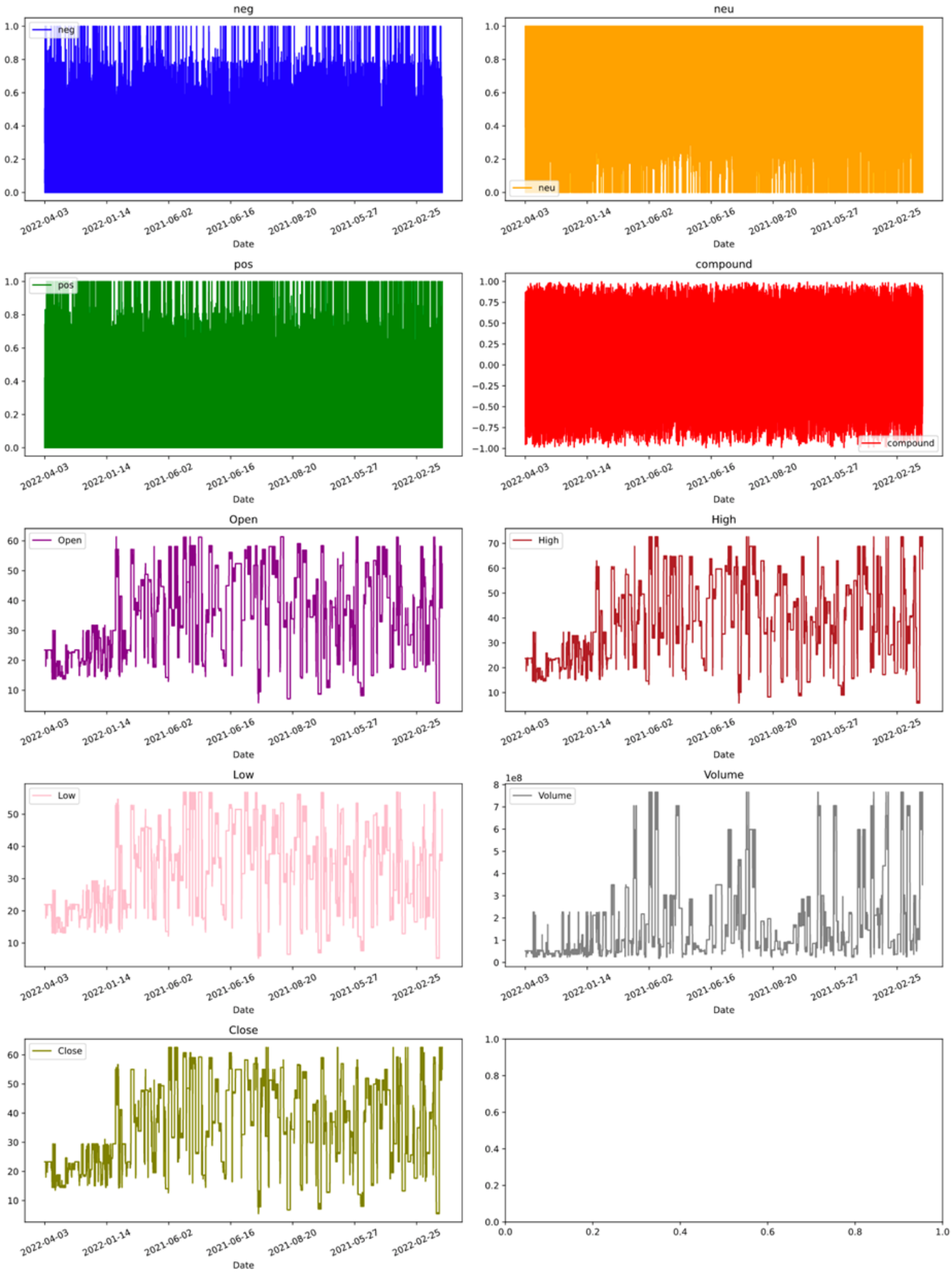

Figure 1. Periodicity of sentiment scores and price movement on days when comments where present related to the stock price of AMC

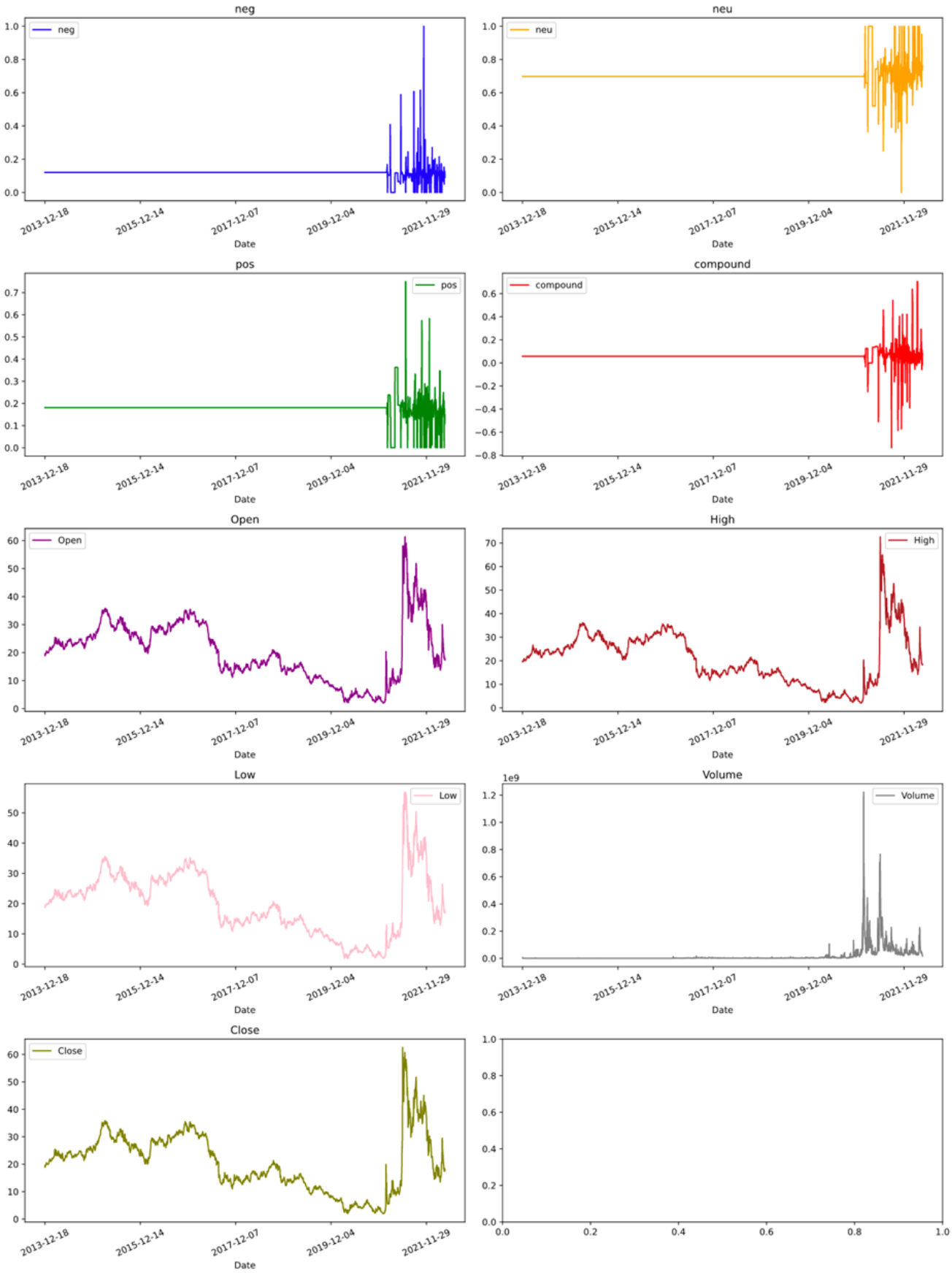

Figure 2. Periodicity of sentiment scores and price movement on trading days only, dating back 10 years (Apr 2012) for AMC

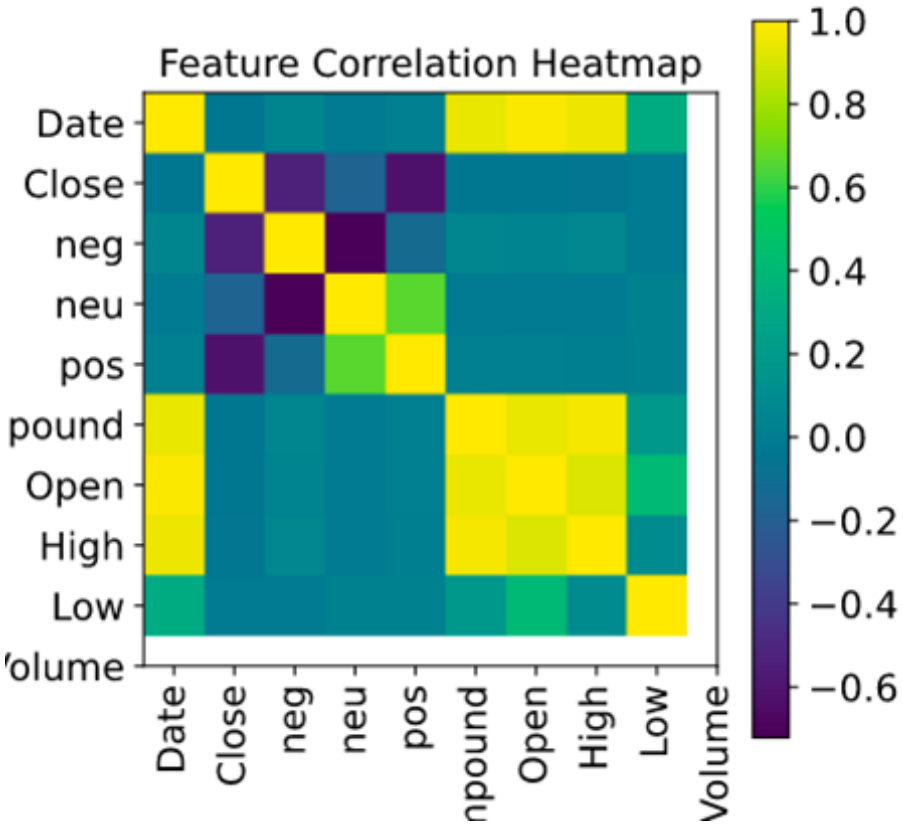

Figure 3. Correlation heat map of features, including price data and the four derivations of sentiment scores

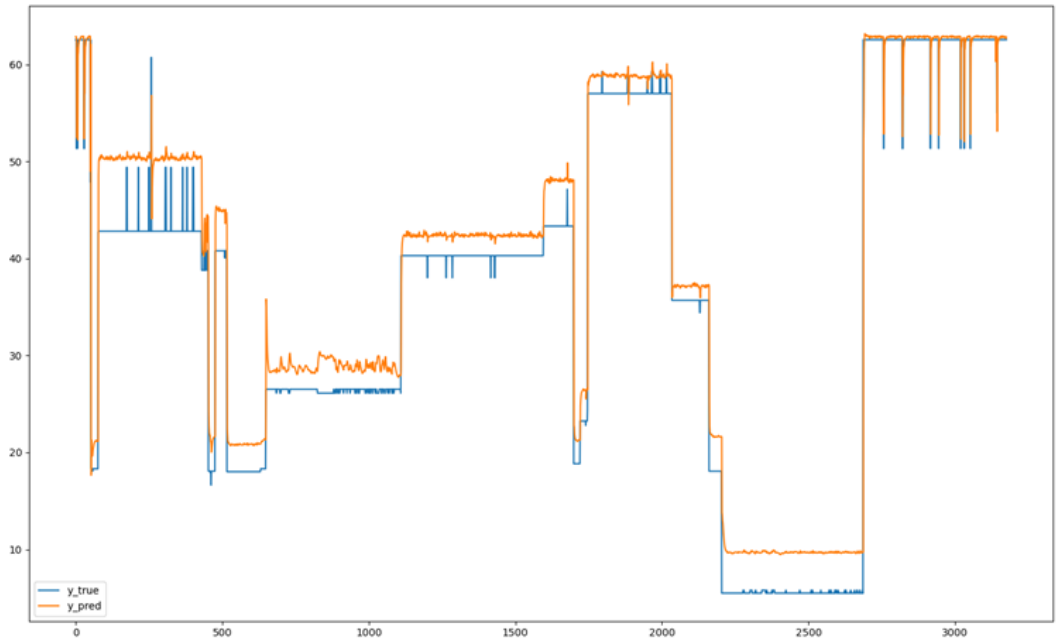

Figure 4. Time series prediction for test data, price is persisted horizontally for the count of comments that day related to AMC.

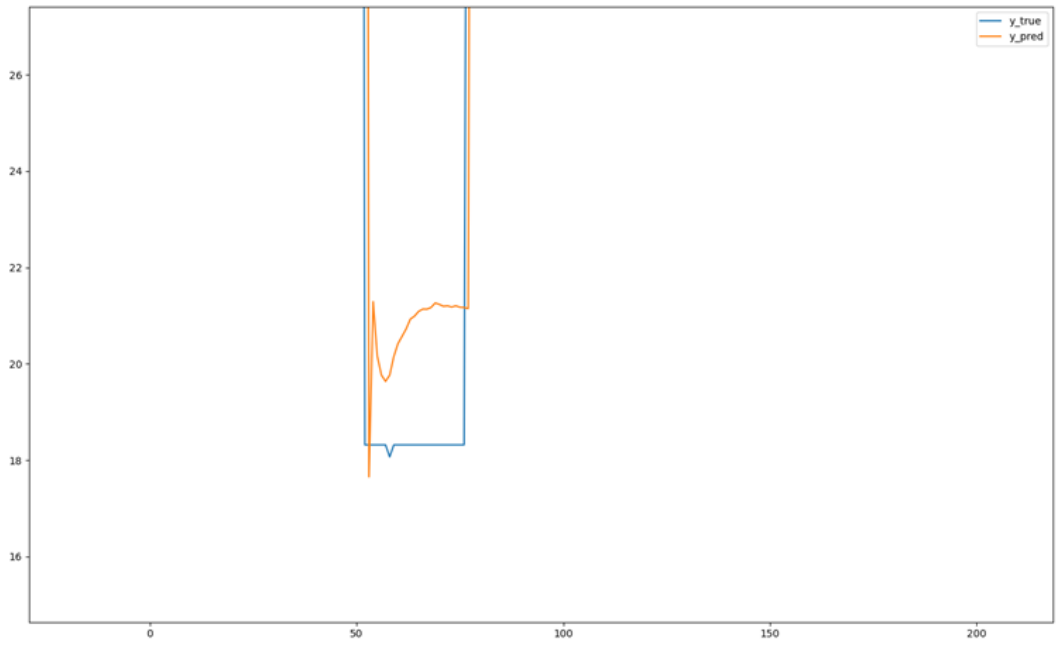

Figure 5. Zoom of Figure 4, where the model used sentiment and price to correctly predict next day price movement.

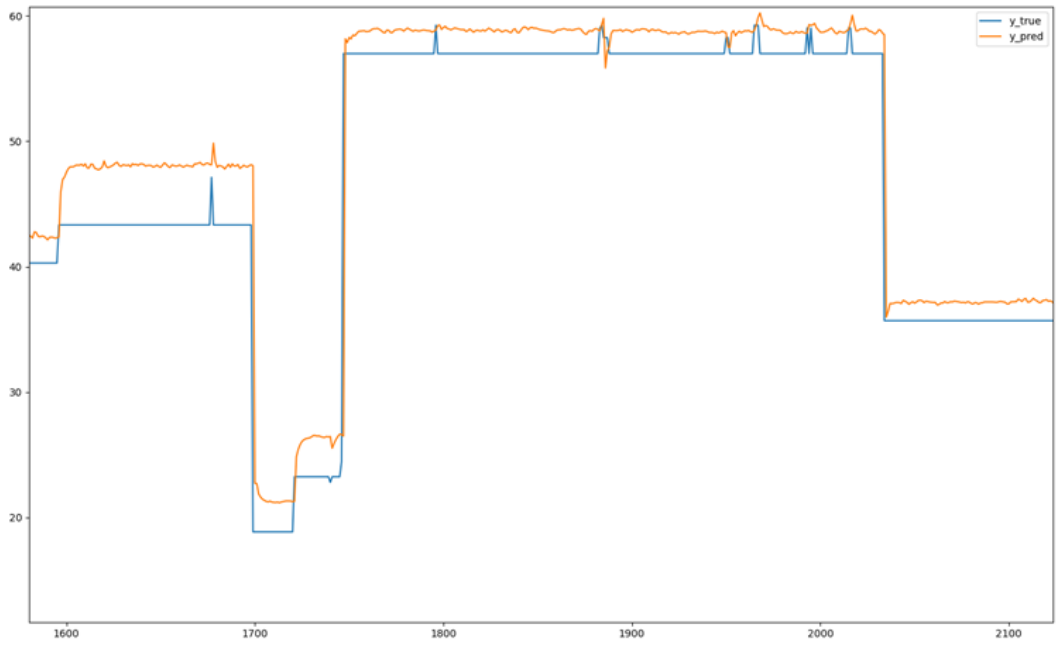

Figure 6. Zoom of Figure 4, where the model used sentiment and price to correctly predict next day price movement.

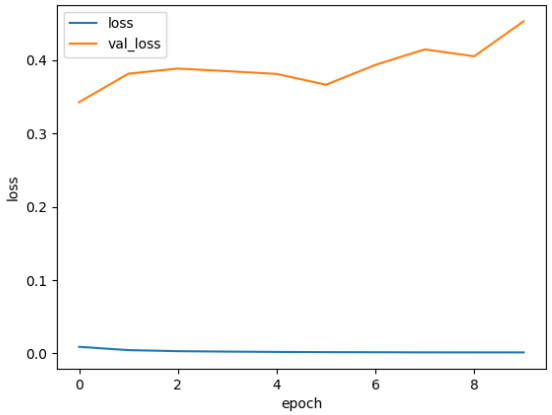

Figure 7. Loss curve of the model making the predictions in Figure 4.

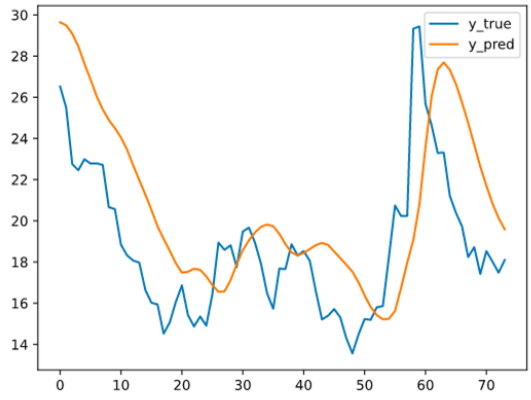

Figure 10.

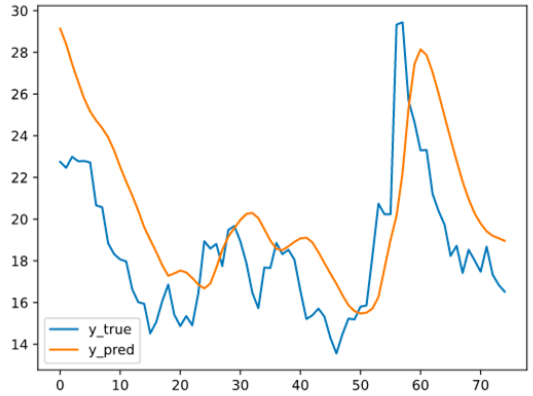

Figure 8. Time series prediction for test data. This is for 30 epochs at the trading day granularity, X axis represents last 70 trading days.

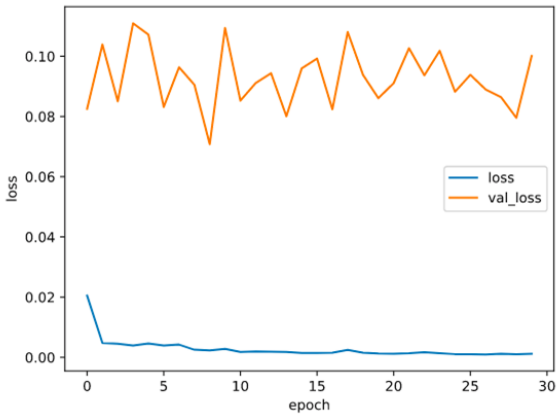

Figure 11.

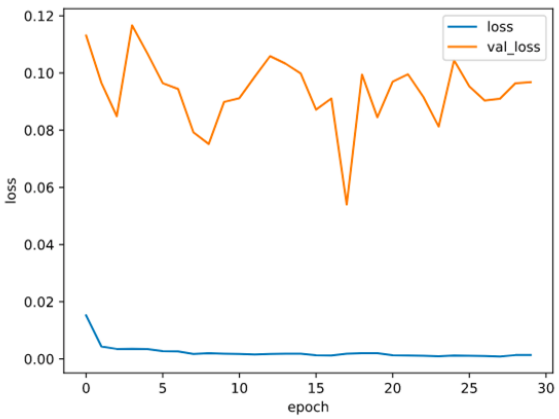

Figure 9. Loss curve for the model in Figure 8.

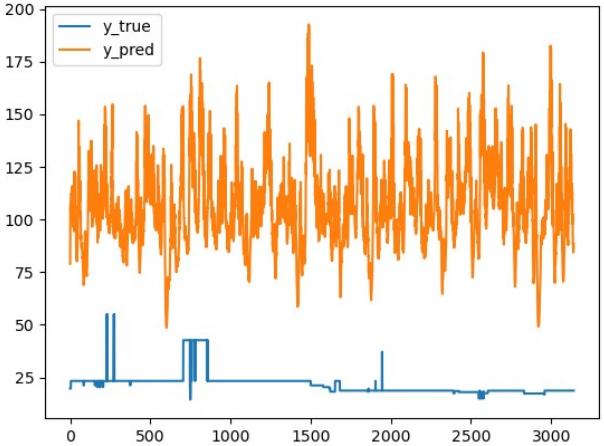

Figure 12. Time series prediction of the embeddings. This was with more regularization to combat exploding gradients.

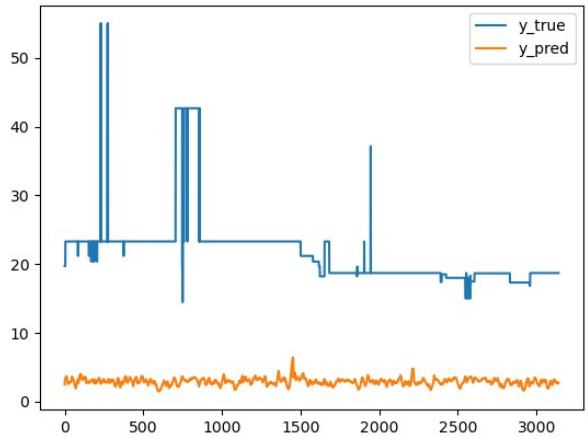

Figure 13. Time series prediction of the embeddings. This was with similar model structure as the other experiments. Notice how the structure of y-pred does not capture the shape of y-true.

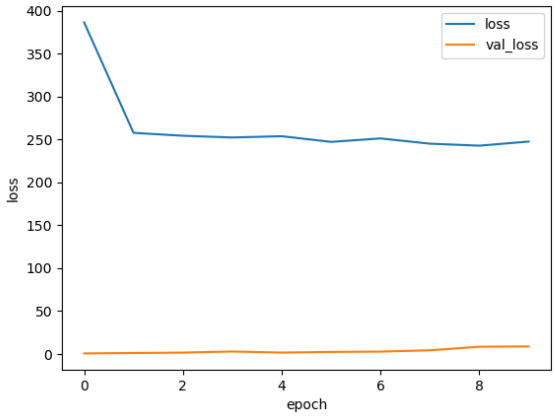

Figure 14. Training and validation plots for the embeddings loss.