Paper Trading From Sentiment Analysis on Twitter and Reddit Posts

Stanford CS224N Custom Project

Eden Wang
Department of Computer Science
Stanford University
eyw@stanford.edu

Chinmaya Andukuri
Department of Computer Science
Stanford University
andukuri@stanford.edu

Shobha Dasari
Department of Computer Science
Stanford University
sdasari1@stanford.edu

Abstract

Many papers develop NLP neural-network models to predict stock movement based on financial sentiment, typically using only Twitter data and predicting the output of a single pre-selected stock price. We seek to extend the approach to determine an intelligent portfolio strategy based on predictions for multiple stocks.

Our model takes as input clustered S-BERT embeddings (from Reddit and Twitter posts) and historical stock mean vector and covariance matrix over the past d days (from Yahoo Finance and NASDAQ data). Our model's outputs are the mean change in stock price and Cholesky factorization of the GT stock covariance matrix, both of which are estimated from Yahoo Finance and NASDAQ data. We evaluated three model architectures: a 4-layer linear model, a 6-layer linear model, and a final model with a 1D convolution layer followed by three linear layers.

To evaluate our model, we use the yield on investment of the predicted optimized investment spread to evaluate our paper trading predictions compared to the ground truth investment yields calculated from historical prices. All model architectures performed better than the baseline (uniform investment distribution across all stocks), with the 1D Convolution model architecture having the best returns.

1 Key Information to include

- Mentor: None
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Researchers have been investigating the validity of the widely-accepted Efficient Market Hypothesis, which states that stock market prices are driven by new information and follow a random-walk pattern. Therefore, stock prices cannot be predicted, since new information constantly enters the market. Solutions to this problem are of interest in financial engineering.

Investigating this hypothesis has become significantly easier in recent years given the accessibility of both large pre-trained word-embedding models and APIs for large social media platforms. Many papers develop NLP neural network models to predict stock movement based on financial sentiment

but fall short by focusing on predicting specific stock prices or the Dow Jones index. There is an opportunity to extend current approaches to the problem, to devise intelligent portfolio management strategies across multiple stocks, which is of interest in financial engineering.

Our goals are to investigate the potential applications of neural-network-based NLP methods to:

1. Evaluate and generalize community sentiment (in Reddit and Twitter comments) towards the most popular stock tickers.
2. Understand the potential influence of public sentiment and community discussion on multiple stocks' price movement.
3. Develop an intelligent portfolio strategy based on predictions for multiple stock tickers.

3 Related Work

Many research papers in this space have common characteristics. They generate social media data (such as Tweets) to either (1) perform general sentiment analysis under the assumption that general sentiment correlates with public sentiment or (2) specifically investigate financial sentiment by filtering for names of companies, their stock tickers, or discussion of financial current events in general. In either case, the papers may investigate a few particular stocks or instead choose the Dow Jones Industrial Average. Some papers use the pre-trained BERT model for word embeddings and build sentiment classifiers on top, while others rely on pre-trained sentiment libraries like TextBlob.

A particularly important contribution from a paper of interest, Thormann et al. (2021), is the use of both sentiment- and non-sentiment-based features to predict stock prices. Thormann et al. use metadata like the frequency of mention of particular stock tickers, the length of those tweets, volatility of the frequency of mentions, and more. These "content-unrelated features" provide additional information beyond the discrete categorization of tweets into a small number of predefined mood buckets, which made up the bulk if not all of the features in previous literature. The use of additional features beyond text serves as motivation to incorporate a wider variety of inputs or output formats in our research, including a covariance matrix discussed in the Approach section.

Thormann et al. (2021) also showed the convenience of an RNN framework for sequential data, including stock prices. The architecture allows context to inform current output, which is useful in incorporating previous stock information to the current state of the model. However, the paper only pulls Twitter data, limiting the expression of sentiment to pull from to 140 character units. Other public forums may provide a greater amount of rich text information, particularly those like Reddit known for speculation and discussion of the financial market. The paper also limited its outputs to a single stock price after choosing a stock of interest, and provided forecasts in pre-specified time intervals ahead. Therefore, there is an opportunity to extend the approach to determine an intelligent portfolio strategy based on predictions for multiple stocks.

4 Approach

To curate our dataset of Reddit comments and Tweets, we used open-source web scrapers to source comments about four stock tickers: $META, $GOOGL, $MSFT, and $AAPL. We leveraged SBERT to generate appropriate sentence representations of comments, then run k-means to generate representative embeddings.

For a specific trading day, we define our model to predict the mean vector and covariance matrix of the four discussed stocks, provided k sentence cluster embeddings for each stock. The model's inputs are the clustered sentence embeddings (computed by inputting comments into a distilBERT model and performing k-means on output embeddings) concatenated with historical stock mean vector and covariance matrix over the past d days (computed using data retrieved from Yahoo Finance and NASDAQ APIs). The model's outputs are the mean vector and Cholesky factorization of the GT covariance (to ensure a PSD covariance output).

To generate our output, we performed the following sequence of steps:

1. Curated a scraped and processed dataset of comments, obtained by applying sentence transformers to generate embeddings for each sentence.

2. We leveraged k-means clustering on the sentence embeddings for each stock and concatenated the cluster centers with historical stock data to form the input into our model.
3. We leveraged both processed embeddings and historical stock data to form the ground truth for our predictions.

We evaluated three model architectures in this project; a 4-layer linear model, a 6-layer linear model, and a model with both a 1D convolution layer and linear layers. Our baseline assumes a uniform allocation with equal and identical returns for all assets. We compared this baseline model against our SBERT-based models to evaluate overall trading performance.

5 Experiments

5.1 Data

For our dataset, we sourced our data from Reddit comments and Twitter tweets through open-source web scraping tools. For Twitter Tweets, we used Social Networking Scrape (snscrape), a general scraping library with a Twitter-specific sub-module (SNScrape). For Reddit comments, we used PRAW (Python Reddit API Wrapper), a Python package that allows for simple access to Reddit's API (Documentation).

Using these tools, we generated 146,466 total Tweets and Reddit posts over a 90-day period. Our scraping criteria were (1) comments that specifically mentioned the stock tickers $META, $GOOGL, $MSFT, and $AAPL and (2) were within the date range between December 1, 2022 and February 28, 2023 (Finance; API). We then sampled data from each data to curate batched X, Y data to curate train, dev, and test sets for our training flow. For days without tweets / comments, we replace the mean embedding with a zero vector of the same shape.

For our training labels, we sourced data from publicly-available APIs from financial data providers (NASDAQ and Yahoo Finance) to provide ground-truth mean and covariance data on the change in daily stock prices from December 1, 2022 to February 28, 2023 (5 years) for the stock tickers $META, $GOOGL, $MSFT, and $AAPL.

The model's inputs are clustered sentence embeddings (generated by a SBERT embedder and performing k-means), concatenated with historical stock percent change in price over the past three days. Figure 1 shows the structure of the model input.

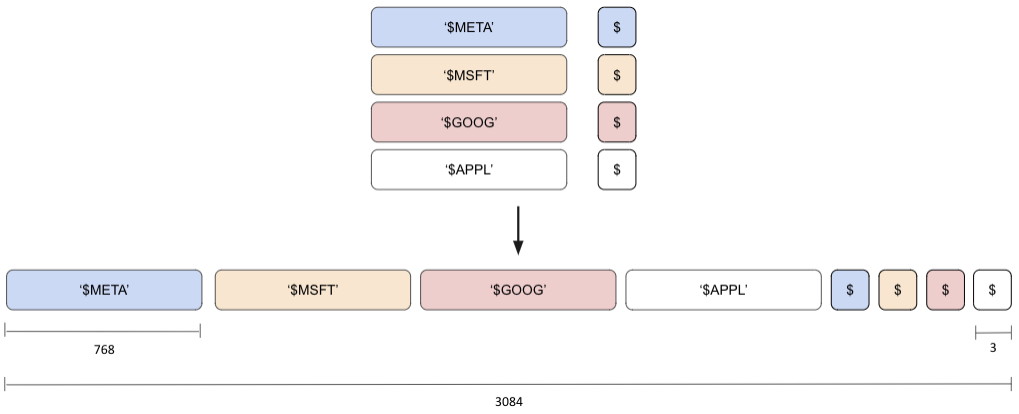

Figure 1: Model input structure, with clustered sentence embeddings and historical stock mean vector and covariance matrix.

5.2 Evaluation method

We evaluated the performance of our model by reserving a test set and using our outputted means and covariance matrix to optimize an automated paper trading setup. This setup leveraged convex optimization to determine the ideal allocation of funds, using the yield on investment and risk-reward tradeoff of the optimized investment spread to evaluate our paper trading predictions. We defined our own performance metrics to be the following:

Yield on Investment: We used the mean daily yield on investment, which is the average daily yield (i.e., the income returned on investment) expressed as a percentage over our withheld test set. This yield is defined by predicting the change in value, then evaluating the resulting asset allocation against the ground truth change in value. This indicates the effectiveness of our paper trading scheme to predict an optimal investment portfolio.

Risk-Reward Tradeoff: We used the risk-reward tradeoff, which is the correlation between the level of risk and the level of potential return on investment. To determine this metric, we vary a gamma value, which is a risk tolerance parameter, determine risk-reward profiles with varying degrees of variance.

Both metrics also account for a cash asset with zero risk and return, which allows our model to opt out of trading on days with overall negative portfolio value. We defined our baseline to be a uniform investment distribution across all stocks, which one can assume to be simply holding all stocks and cash in equal reserve. We compared our different model architectures and the baseline model performance against the yield on investment and risk-reward tradeoff of real-world stock trade performance.

5.3 Experimental Details

In our model, the input X, for a particular date, consisted of clustered sentence embeddings and vectors representing stock prices in the d days leading up to that date. The output is defined as concatenated vector containing the predicted mean and a decomposed covariance matrix (which ensures our resulting covariance is positive semi-definite), and is compared against labels Y, which are ground-truth stock mean and covariance matrices which we manually calculate.

For each experiment, the task is defined as predicting average stock prices and pairwise covariances for each of n companies, for day i, given (1) average embedding for a sample of online mentions for companies from day i and (2) stock prices for days $i-1, i-2, i-3$.

We trained and evaluated our model on our data with an AdamW optimizer and mean-squared error loss (MSE) between the ground-truth mean and covariance matrices and the analogous matrices generated by our model outputs for all examples in our test set. Specifically, for each test example i, we summed the element-wise squared differences between the label and the output matrices to generate a scalar squared error for i. We then took the mean of all these squared errors over all i.

We ran three experiments in this study, with varying model scales and architectures, which were each compared to the baseline. In each experiment, we evaluated one of the three following model architectures, as shown in Figure 2.

The hyperparameters used were the following:

```
n = 4 (number of stocks to predict)
k = 1 (number of k-means clusters)
d = 3 (number of days of historical data to use in training)
embed_dim = 768 (embedder embed size)
kernel_size = 3 (for 1D Conv layer)
p = 0.2 (dropout hyperparameter)
max_epochs = 100
lr = 1e-5
gamma = 0.875 (for learning rate scheduler)
betas = (0.9, 0.99) (for AdamW optimizer)
batch_size = 64
```

Figure 2: Model architectures evaluated in this study – Basic (4-layer linear model), 6LayerLinear (6-layer linear model), and 1DConvolution (1D convolution layers and 3 linear layers).

5.4 Results

During training and validation, loss convergence was achieved early in both train and validation sets, though training was continued for all 100 epochs and evaluation was performed with our last epoch. The model performed well with low MSE losses, as shown in Figure 3.

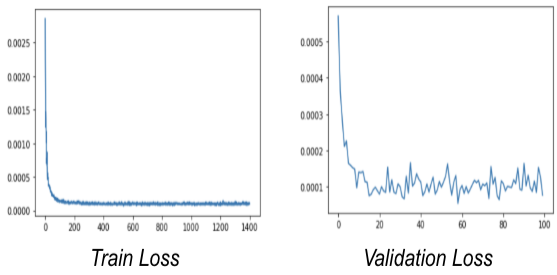

Figure 3: Loss convergence in the basic model architecture.

When evaluated on our withheld test set, the different model architectures achieved the following yield on investment metric values:

- Baseline (uniform investment distribution): 0.0059 mean daily returns
- Experiment 1 (4 linear layers): 0.0070 mean daily returns
- Experiment 2 (6 linear layers): 0.0078 mean daily returns
- Experiment 3 (1D convolution + 3 linear layers): 0.0081 mean daily returns

For our selected tickers, we believe our improvements over the baseline reflect our model's ability to learn from underlying stock sentiment and gauge stock volatility based on commercial sentiment. Our baseline approach, assigning equal portfolio distribution to each asset, retrieves a mean return of 5.92e-3 and variance of 2.22e-4.

6 Analysis

Compared to the baseline, the Basic Stock Prediction model produced higher mean daily returns. The 6LinearLayer model produced higher returns than the Basic model, and the 1DConvolutionLayer model produced the best returns of the three models. Figure 4 shows the annual returns on investment from the different model architectures.

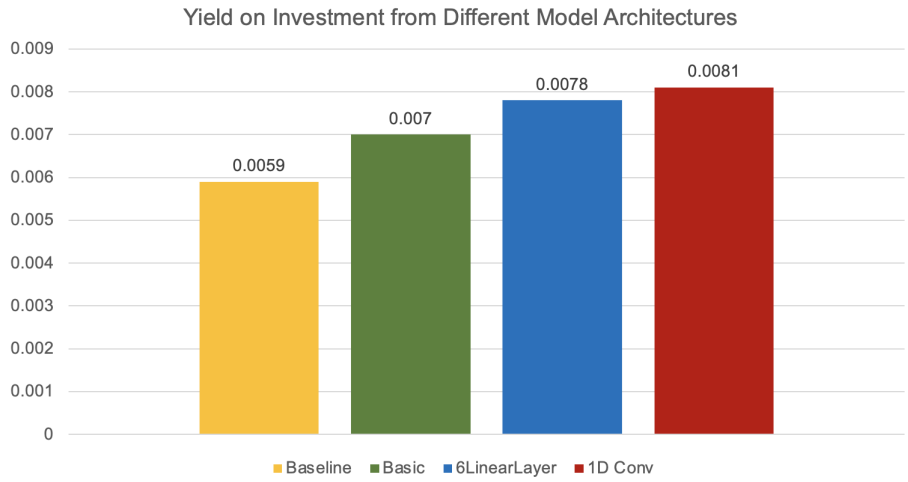

Figure 4: Mean yield on investment from the different model architectures.

In order to more rigorously analyze our model outputs, we perform several tests by varying the risk-return parameter during portfolio optimization on our withheld test set. We vary a parameter γ that correlates with relative risk aversion; higher values of γ correspond to greater risk aversion and push our optimal portfolio distribution towards less volatile assets. These experimental results are shown in Tables 1-3 below:

Gamma	Mean	Variance
0	0.00690	0.000328
1	0.00688	0.000325
2	0.00672	0.000315
3	0.00654	0.000313
4	0.00651	0.000313
5	0.00615	0.000303

Table 1: Exp. 1: 4 Linear Layer

Gamma	Mean	Variance
0	0.00758	0.000358
1	0.00746	0.000348
2	0.00716	0.000341
3	0.00678	0.000333
4	0.00647	0.000315
5	0.00614	0.000289

Table 2: Exp. 2: 6 Linear Layer

Gamma	Mean	Variance
0	0.00791	0.000358
1	0.00779	0.000345
2	0.00791	0.000352
3	0.00708	0.000330
4	0.00741	0.000319
5	0.00690	0.000293

Table 3: Exp. 3: 1D Conv + Linear

Note that linearly increasing parameter γ does not directly correlate with lower variance and mean; our optimization leverages our predicted means and covariances, which is a proxy for the true historical data. We observe that even with relatively high γ parameter values, our mean returns still outperforms our baseline, though with higher variance.

Interestingly, Tables 1-3 above show that a decrease in γ results in an increase in mean return on our test set in all cases except two ($\gamma = 2$ to $\gamma = 1$ and $\gamma = 4$ to $\gamma = 3$ in Table 3). Generally, returns decreased in all model architectures as risk tolerance decreased, with the 6LayerLinear model having the largest decrease in yield of all three model architectures, as shown in Figure 5 below.

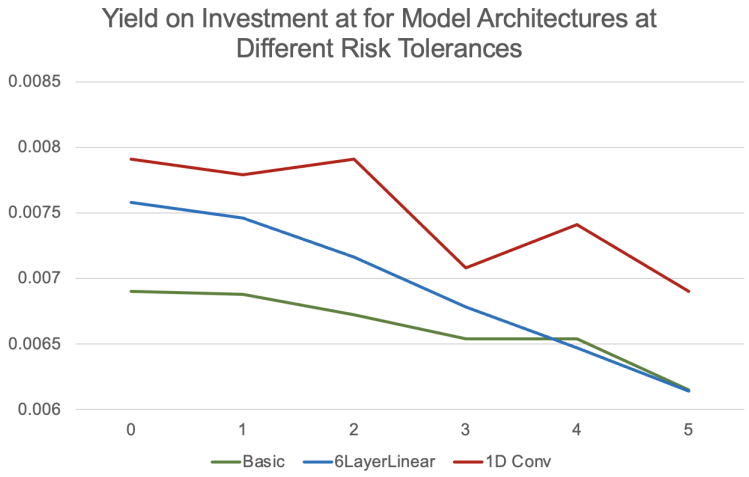

Figure 5: Mean yield on investment from the different model architectures at varying risk tolerances (higher gamma values mean lower risk tolerance).

It is unclear why this pattern emerged – one possibility is that the time period from which the data was pulled favored high-risk investments. The same weights might perform differently given data mere weeks after the existing data, in the face of the collapse of important financial institutions and changing labor market conditions. While the performance on the held-out test set is promising, we remain cautious and acknowledge that the model could benefit from a larger range of data from varying market conditions and online sources.

7 Conclusion

Overall, our results indicate that general trading sentiment and comment features can often be a predictor of stock performance. The results provide evidence against the Efficient Market Hypothesis: NLP neural network-based models can achieve greater returns than uniform investment strategies given only recent financial history and sentiment.

All model architectures performed better than the baseline (uniform investment distribution across all stocks), with the 1D Convolution model architecture having the best mean yield on investment. Our improvements over the baseline, even with a basic 4-layer linear model, reflect the emerging

paradigm that simple linear probes developed on top of large language models can yield powerful models for downstream tasks. In our case, even relatively simple models (with 100-300k parameters) are able to leverage latent information in our sentence embeddings to yield improvements in investing outcomes.

One limitation of this study was the availability of efficient and well-documented open-source scraping tools. As a result, we utilized weighted sampling necessary to combat sparsity in raw data when generating training pairs. This study was also limited by the restriction of the dataset to social media comments that directly mentioned the stock ticker name. The omission of social media comments that mentioned the company's name generally likely has implications for the performance of the model.

Extensions of this study could include the following:

- Extending stock prediction model to a wider variety of companies or stock indices through the generation of additional data.
- Expanding the search keywords through query engineering during the data collection process to increase the scope of inputs to the model.

References

NASDAQ Historical Stock Data API. *https://data.nasdaq.com/tools/api*.

Python Reddit API Wrapper (PRAW) 7.7.0 Documentation. *https://praw.readthedocs.io/en/stable/*.

Yahoo Finance. *https://finance.yahoo.com/*.

SNScrape. *https://github.com/JustAnotherArchivist/snscrape*.

Marah-Lisanne Thormann, Jan Farchmin, Christoph Weisser, Rene-Marcel Kruse, Benjamin Säfken, and Alexander Silbersdorff. 2021. Stock price predictions with lstm neural networks and twitter sentiment. *Statistics, Optimization Information Computing*, 9(2):268–287.