# Predicting Stock Movement Using Form 8-K Transcripts

Wesley C. Olmsted
wolmsted@stanford.edu

Tarek W.F. Abdelghany
tabdel@stanford.edu

Kapil A. Kanagal
kkanagal@stanford.edu

## Abstract

We investigate deep learning techniques to predict stock movement based on Form 8-K transcripts. We explored a hybrid model that combined various embeddings with a feed-forward network and LSTM. Our best model predicted whether or not a stock price rose or fell the day following the Form 8-K transcript's release with an accuracy of 0.6208, significantly better than our Naive Bayes baseline.

## 1 Introduction

Our paper seeks to predict event-driven share price of publicly traded companies. Share price typically reflects the health of a business and serves as an insightful economic indicator for the viability of the overall economy.

Accordingly, events like mergers, bankruptcies, change in management, and asset movements impact a stock's share price. All of these events are captured in Form 8-K documents filed with the SEC. With the advancements in computing the past decade has seen, computers are now significantly more involved in the financial world. Our project seeks to use novel deep learning and natural language processing (NLP) techniques to more efficiently predict stock price movements resulting from these events.

## 2 Data and Features

### 2.1 Non-Text Features

Our non-text features include: a 60-day window of previous adjusted close prices, a 60-day window of the volatility index (VIX), a 60-day window of the S&P 500 data (tracked by the ticker $\hat{G}$SPC), and a many-hot feature vector containing an encoding of a company's 8-K event types. We used the difference of the adjusted close price the day after the 8-K was filed and the adjusted close price the day the 8-k was filed in order to label if a stock went UP or DOWN. We chose this indicator because of its relevance to the real world – we can purchase or sell stock after an 8-K is released, as many event-driven funds do.

In pre-processing this data, we excluded any 8-K events that occurred prior to the existence of 60 days of price data for a given stock. Thus, if a company's stock price history started on 12/1/10 and they filed an 8-K on 12/5/10, we would exclude that 8-K event. After this processing we ended up with 179,170 events. The VIX, Price History, and S&P vectors were then concatenated with the many-hot vector and our Y label. Y = 1 when the price on the following date is higher, and Y = 0 otherwise. This yielded a total of 215 non-text features and 2 classes of output. We split our data along an 80/10/10 temporal train/dev/test set basis. Since we are dealing with time-series data, we wanted to avoid a look-ahead bias and split our data so the most recent events would be in test set, followed by the dev and train sets.

### 2.2 8-K Transcripts

Below we show a snippet from an 8-K report of Google, Inc. under the Financial Statements and Exhibits category on February 1, 2005:

> *...Net income on a GAAP basis increased to $204 million or 19.8% of revenues in the fourth quarter of 2004 as compared to $27 million or 5.3% of revenues in the fourth quarter of 2003. Earnings on a diluted per share basis were $0.71 in the fourth quarter of 2004 as compared to $0.10 in the fourth quarter of 2003...*

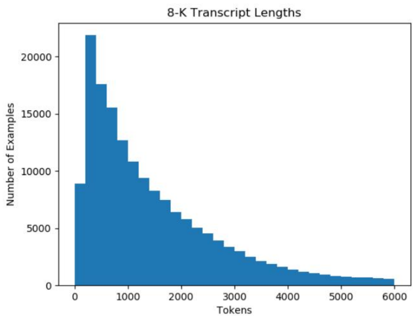

Figure 1: Histogram of 8-K Transcript Lengths

These transcripts vary widely in length, and large sequences would result in much larger training times for our models so we needed to choose a reasonable maximum length to cut them off at. We would then pad any 8-K transcripts to the max length if they were shorter. Based on the histogram in Figure 1, we chose 2000 as the max length because it fully contains a large portion of our data while also limiting the sequence length. We combed our corpus of 8-K transcripts to generate our vocabulary. We then created an embedding matrix using 300-dimensional pre-trained word embeddings trained from the Common Crawl [1]. We decided to remove number tokens from our vocabulary and data because they drastically increased our embedding matrix size. Any token that was not seen in the pre-trained embeddings was initialized randomly.

## 3 Model Structure and Metrics

### 3.1 Baseline Models

For our baseline, we evaluated two models: a random guess (with probability 50% of a correct guess) and a Naive Bayes model. Due to the difficult nature of this problem, a baseline of a 50% is reasonable, as predicting a stock's movement is difficult even for experts. Naive Bayes is a simplistic probability model and serves as a fundamental comparison to the more complex models we created.

### 3.2 Feed-Forward Model

The feed-forward model consisted of using all the non-text features concatenated in vectors and passing that through a series of fully connected layers. We used an input layer and six layers with ReLU activation functions for each of the fully connected layers. After the last layer, we used a softmax layer for binary classification. We also used the Adam optimizer for all variations of our model.

### 3.3 LSTM and Feed-Forward Model

For this model, we first passed in the tokens from the 8-K transcripts into the embedding layer. We also made this embedding layer trainable so that the embeddings are more representative of our problem. The padding tokens had a corresponding embedding so that our model could potentially gain insights into transcript length. From the embedding layer, our inputs are passed into a bidirectional LSTM and the final state of both the forward and backward LSTMs are concatenated. We used an LSTM because we can process variable lengths of text into a smaller encoding. We also chose to use a bidirectional LSTM to avoid any bias towards the beginning or end of the transcript. This section of the model is shown in red in Figure 2. This concatenated final state is then concatenated with the non-text features and passed into the feed-forward network described above.

### 3.4 LSTM, Ticker Embedding, and Feed-Forward Model

For this model, we made an addition to take the ticker into account. The ticker represents the company of the stock e.g. GOOG. We found that certain tickers tend to correlate with each other as well as the S&P. For example, two companies in the same industry would most likely see similar trends when an event related to that industry occurs. In order to add this behavior into our model, we created a new ticker embedding matrix initialized randomly for all tickers in our dataset. We would pass each ticker input through this embedding layer, and the embedding layer would learn the trends of that ticker. This addition is shown in blue in Figure 2.

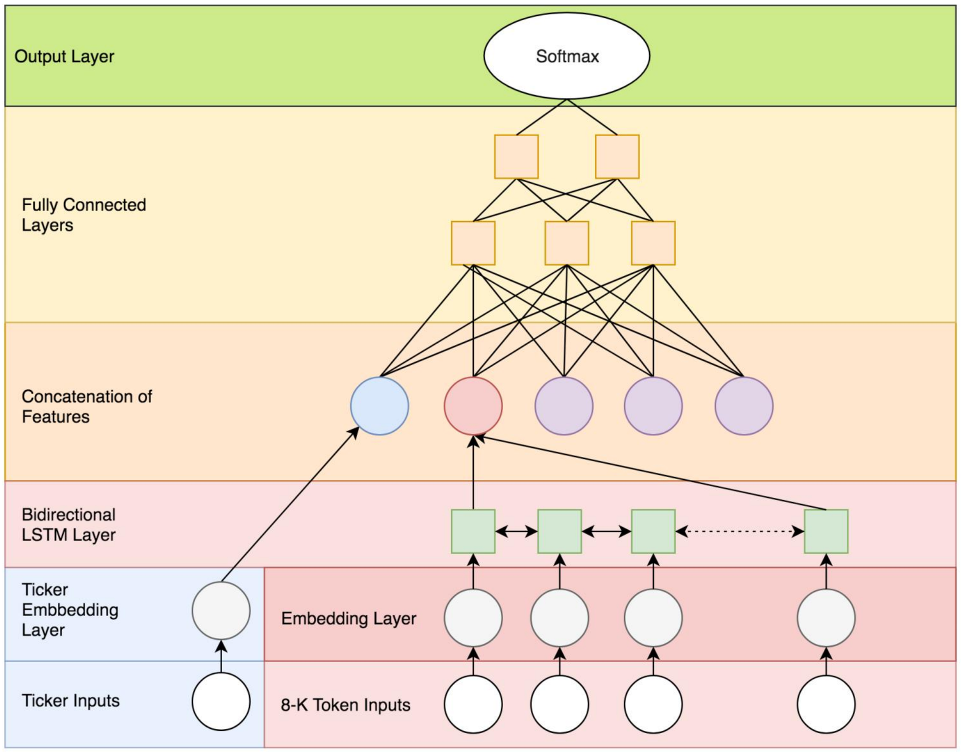

Figure 2: LSTM, Ticker Embedding, and feed-forward model.

### 3.5 Hyperparameters

We chose a learning rate of 0.00005 for all of our models except the feed-forward model which had a learning rate of 0.001. The models utilizing the language performed better with a very slow learning rate. When we used a higher learning rate, we saw train loss rise at later epochs. We also did not need to implement learning rate decay since we used the Adam optimizer. We applied 0.2 dropout to all layers not including the fully connected layers, which are shown in yellow in Figure 2. We chose not to apply dropout to these layers because our feed-forward model, which did not include any features from language or tickers, was unable to overfit our training set. Therefore, we knew this model did not have the complexity to warrant the application of dropout. We used a batch size of 256, which was the highest batch size we could use without running into memory issues. We also shuffled these batches before training and performed batch norm.

### 3.6 Evaluation Metrics

We evaluated our models using the following metrics: accuracy, precision, recall, and F1 score. The precision, recall, and F1 score were all calculated per each class (UP, DOWN). From a trading perspective, we want to optimize accuracy, as this tells us how well our model performs on both UP and DOWN samples. If our fund were long-only, we would seek to optimize precision for the UP class since we want a model that is correct more often when it predicts UP. On the other hand, if our fund were short-only, we would seek to optimize precision for the DOWN class so that we are more correct on predicting DOWN.

## 4 Results

The results of our experiments are shown in Table 1 below. Our tickers + LSTM + feed-forward model had the highest accuracy, UP precision, UP recall, and UP F1 score demonstrating our model's significant predictive value over the baseline. For financial prediction problems, a model that is over 55% accurate is typically considered successful, so our accuracy of 62% is a great result for this problem.

Table 1: Model F1 and EM Dev Set Scores

| **Model** | **Accuracy** | **UP Precision** | **UP Recall** | **UP F1** | **DOWN Precision** | **DOWN Recall** | **DOWN F1** |
|---|---|---|---|---|---|---|---|
| Guess Baseline | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.5000 |
| Naive Bayes Baseline | 0.4940 | 0.5016 | 0.1853 | 0.2706 | 0.4921 | **0.8109** | **0.6125** |
| Feed-Forward | 0.5754 | 0.5685 | 0.5967 | 0.5691 | **0.5821** | 0.4304 | 0.4948 |
| LSTM + Feed-Forward | 0.6083 | 0.6430 | 0.5953 | 0.6183 | 0.5745 | 0.4047 | 0.4749 |
| Tickers + LSTM + Feed-Forward | **0.6208** | **0.6758** | **0.6032** | **0.6375** | 0.5672 | 0.3968 | 0.4669 |

Below in Figure 3, we can see the smoothed accuracy (with a constant of 0.7) evaluated on the dev set over 10 epochs of our models. The feed-forward model's lack of complexity limited its capabilities of learning more over multiple epochs, while the other models had the requisite complexity to learn over multiple epochs.

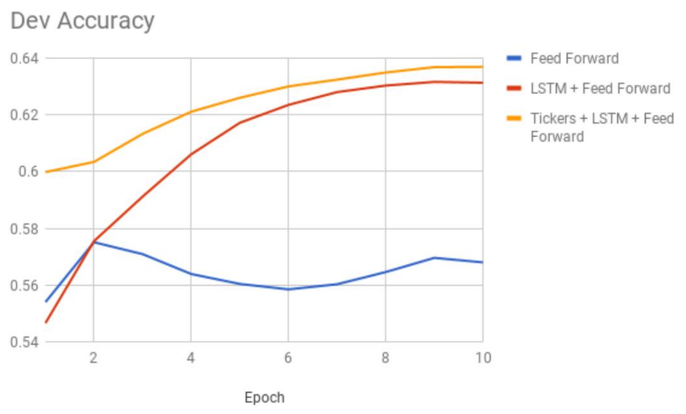

Figure 3: Dev Accuracies on LSTM, Ticker Embedding, and Feed-Forward Model

## 5 Discussion

Since our model performed the best on the UP metrics, it could feasibly be used for a short-term, long-only, momentum-based trading strategy. Many hedge funds use event-driven strategies and may trade around 8-K releases. Tools like ours can be used to help create probabilistic prediction models that assist fund managers in the trading process.

As we evaluate our results as the difference between the close price the day the 8-K is released and the following day, our model is able to detect the short-term momentum a stock derives from an 8-K release in the market using both our text and non-text features.

The high UP precision signifies that when our model outputs a positive prediction, the real movement of the stock tends to match our model close to 70% of the time. If we were a fund and our model output UP, we would open a long-position in the given stock.

One interesting note is on the performance of our baseline Naive Bayes model. Our Naive Bayes model has a very high DOWN recall. This is likely due to the fact that our model is not detecting many false positives. Thus, this baseline model will be highly biased to predict DOWN for the movement of a stock price. This is indeed the case and is validated by the very low UP recall, meaning that the model predicts a large number of false negatives.

An example of a win for our tickers + LSTM + feed-forward model comes from the following report filed by Netflix (NASDAQ: NFLX) on April 17, 2003. The feed-forward model predicts DOWN for this event, likely due to the fact that the S&P 500 data was trending downwards. However, our tickers + LSTM + feed-forward model predicts the correct value of UP, likely due to instances similar to the following text segment:

> *"According to Reed Hastings, founder and CEO of Netflix, 'The Company broadly outperformed its financial expectations for the quarter. Our strategic focus on improving the Netflix user experience, which produced record low churn and a lower gross margin this quarter...'"*

This qualitative data segment shows that our model's NLP feature augments it ability to correctly predict price movement. Our model is likely able to learn the relationships between various words in a given context and make a prediction based on similar sentiments in the training data. In particular, we note that our model might be attributing positive sentiment to phrases like "improving [...] user experience" and "low churn". Both of these text segments reflect a positive outlook on the company, and our model is likely picking up on phrases like this to help in its decision-making process. The recognition of these language features improves the accuracy and robustness of our model, reflecting its potential feasibility to be used in practice.

## 6 Future Work and Conclusion

If our team had another 6 months to work on the project, we would focus on improving a few aspects of our model.

First, we would implement a feature that would allow us to parse the Financial Statement events and compare the numbers in the 8-K to analyst expectations via earnings surprise. The EPS surprise plays an enormous role in a given stocks momentum, especially for a short-term, event-driven strategy.

Second, we would try to account for transaction costs. We could model transaction costs as a fixed percentage fee of our overall invested capital. We could calculate the number of long positions recommended by our model, compute the average notional trade size, multiply the two together and subtract a percentage of this as a trade fee. We would also want to run more backtests and determine metrics like Sharpe Ratio, Max Drawdown, and PnL, before implementing this as a live trading strategy.

Third, we could experiment with using attention to find relationships between different 8-K transcripts for a more holistic model. Finally, while we tuned our hyperparameters, we would allocate more time to this task. Hyperparameter tuning can very time-intensive and finding the optimal hyperparameters can be quite difficult.

Overall, our model successfully demonstrates the applications of NLP and deep learning to the event-driven trading of equities. We believe that our project is a stepping stone to further research in the area and look forward to the development of future NLP and deep learning models.

## 7 Contributions

Wesley: Implemented the three models. Worked on generation of word and ticker embeddings.

Tarek: Pre-processed data, added Tensorboard support, and implemented metrics.

Kapil: Implemented baseline (with metrics) and tuned hyperparameters. Generated poster content.

### Acknowledgments

Thank you to the CS230 teaching team, especially our project mentor, Suraj. Furthermore, this project would not have been possible without the help of Dan Jurafsky and Heeyoung Lee, who provided us with the 8-K transcripts, stock price, and VIX data as well as valuable insights.

### References

[1] Jeffrey Pennington, Richard Socher, Christopher D. Manning. "GloVe: Global Vectors for Word Representation."

[2] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, Dan Jurafsky. "On the Importance of Text Analysis for Stock Price Prediction."

[3] Tensorflow. https://github.com/tensorflow/tensorflow.

[4] Scikit-learn. https://github.com/scikit-learn/scikit-learn.

[5] Keras. https://github.com/keras-team/keras.

[6] Pandas. https://github.com/pandas-dev/pandas.

### GitHub Repo

https://github.com/twfabdel/8kStockPrediction

### Data

https://nlp.stanford.edu/pubs/stock-event.html