Neural based event-driven stock rally prediction using SEC filings and Twitter data

Magdy Saleh
mksaleh@stanford.edu

Surag Nair
surag@stanford.edu

Abstract

In this work we present a deep learning based methodology for predicting event-driven stock rallies. We apply neural methods to the data presented in [1] as well as a dataset created for this work that includes tweets. We implement a combination of neural networks using both textual and non-textual features to predict stock movements on days of 8-K report filings. Our best model, on the dataset from [1] (Task 1) using GloVe embeddings with an LSTM and fully connected layers achieves an accuracy of 51%. Our model on the generated dataset (Task 2) performs at a 52.61% accuracy. In both tasks, we observe that adding textual information improves model performance. We also find that the attention distribution over full tweets yields exciting qualitative results, giving higher scores to tweets with rich information content.

1 Introduction

Natural language processing using neural methods has had a lasting impact on many problems including machine translation, question answering and sentiment classification. Recently one application that people have attempted is combining text based approaches to financial markets, specifically predicting stock price movement based on financial news and company results. The work by [2] used a combination of Recurrent Neural Networks (RNN's) and Convolutional Neural Networks (CNN's) on news articles from Reuters to predict whether, given a piece of news on a company, its stock price will increase the next day or not. [2] They produce a binary output of whether the price of the stock will increase and do not take into account overall movements in the market.

Stock performance prediction is an important problem in finance. Reliable estimates of stock price help guide investor strategies and manage their portfolios better. However, stock performance prediction is notorious for being an extremely challenging problem and has a high degree of randomness and external influences that are tough to account for. Traditionally, stock performance prediction algorithms are based on time series of various financial indicators of the stock. News events and internal changes in the company often drive changes in the stock price. There has been an increasing shift towards incorporating other sources of information, especially textual information such as news articles and tweets to infer how such events would impact the stock price. Inspired by [1], we attempt to quantify the benefit of incorporating textual information in a stock performance prediction pipeline in addition to other non-textual financial indicators. We primarily focus our attention textual information from SEC 8-K filings and tweets.

We observe that in general, adding textual information improves model performance. We also find that our model learns meaningful representations of tweets and is able to separate information rich tweets from non-specific and bot-generated tweets, despite not being trained for that directly. As is routine in stock performance prediction, we observe that often the model makes predictions that agree with the input data, but do not match the true label. This can be attributed to the randomness in the markets and other external factors that impact stock prices.

2 Related Work

This work is based on attempts by the aforementioned literature as well the work completed by Lee et al. in 2014 [1]. We base our task on their work on stock rally prediction using a combination of text-based and non-text based features. In comparison to the aforementioned studies, this paper computes whether given a company earnings announcement, the stock price of a company will move by over a certain margin. They decorrelate the stock movement from the overall market movement by accounting for the performance of an S&P 500 index tracker on the same day.

The paper sets up strong baselines for comparison. The first is a random guess baselines and the second is a majority baseline which yield a performance of 35%. Two more baselines that use financial features in addition increase the performance up to 50%. For the model that incorporates textual information from the 8-K documents, they used a simple bag of words model and retained words that appeared more than 10 times. The words were lemmatized and feature selection was based on Pointwise Mutual Information. In addition, they do a non-negative matrix factorization on the bag of words vectors to get a low dimensional representation of the document. They train a random forest model and increase the performance to 55%.

3 Approach

3.1 Task Description

For this work we consider two related tasks. We attempt to predict the movement of a stock on the day following an 8K filing. For the first task we consider this on the dataset presented by Lee et. al [1], and apply neural methods. For the second task we create a dataset for the years 2013-2018 to be able to incorporate twitter data.

3.2 Features

For both tasks in this work we use a combination of features. Some of which come from [1]. These are summarized in in Table 1. Particularly we focus more on the text based features in the data.

For the text in the 8-K filings, we use a simple regex to add whitespace around punctuation marks, e.g. "don't- no" $\rightarrow$ "don ' t - no". The sentence is then split on whitespace to obtain tokens.

We then construct a vocabulary from the processed 8-K texts and discard any word that occur less than two times in the vocabulary as well as limiting the vocabulary size to be less than 20000 words. Furthermore, an assumption we make in the feature engineering is that most important information in the 8K's occur in the first few thousand words of each filing (with a substantial number of filings containing less than a 1000 words). Thus we only consider the first 1000 or 2000 words of each filing.

We use a character level model for the tweets so they are processed differently. Since we find tweets by their tags (i.e. Google related tweets are all tweets mentioning '$GOOG') we remove the ticker mentions and replace it with a special character $<TKR>$.

Note that the earnings surprise feature is only used in task 1 due to the lack of publicly available earnings surprise data.

3.3 Models

3.3.1 Task 1:

Our key task is to model better ways to leverage the textual information from the 8-K filings. To this end, we implement two different ways to incorporate the text:

1. **Bag of words**: We use a simple unigram bag of words method with a vocabulary size of 5000, where each feature is 1 if it is present in the 8-K text, else 0. We concatenate this with VIX volatility and earnings surprise and pass through 2 feed forward layers.
2. **Embedding Based**: We use word embeddings and pass the 8-K text through a single-layer bidirectional LSTM. The outputs (h_n, c_n) from the LSTM are passed through a linear layer

Table 1: Features considered in this work.

Earnings surprise	The gap between consensus and reported earnings per share (EPS). Consensus EPS is the analysts' estimation of earnings per share, and reported EPS is the actual earnings per share reported by the company in the 8-K report
Trade Volume	The number of trades on given ticker that were made during the day. It gives an indication of the popularity of the stock on a given day, regardless whether the price movement is positive or negative.
Volatility S&P 500 index	The volatility index value (ticker: VIX) at the market close before the 8-K report is released. Volatility is a statistical measure of the variability of,returns for a given security or market index, typically defined as the,standard deviation of returns over some finite period.
Tweets	All tweets on the day of the 8-K filing. Character based embeddings are learned using convolutional operators.
8-K text features	Multiple embeddings of 8-K text including GloVe based and learned embeddings.

and then concatenated with the VIX volatility and earnings surprise and then passed through 4 feed forward layers.

3.3.2 Task 2:

For this task, we use raw tweets in addition to the 8-K filings text and market features (VIX volatility, trade volume, today's stock price movement). The model architecture with an example data point is shown in Figure 1. We process the 8-K text using the embedding based method, same as for Task 1—we use embedddings to pass the 8-K text through a single-layer bidirectional LSTM. The outputs are passed through a fully connected layer.

For the tweets, we use a character-level CNN network. For a given training example, we use text from one 8-K filing and 100 tweets (subsampled from a larger pool). Each of the tweets is individually passed through the same 2-layer 1D convolution network. This gives a vector representation for each tweet in the example. We consider 2 strategies for aggregating these vectors- 1) taking the average across all vector representations, 2) taking a weighted average using attention. To compute attention weights, we pass the vector representation of each tweet individually through a single hidden fully-connected layer. This yields attention values corresponding to each tweet. Attention weights are obtained by performing a softmax over the raw values. [3] The tweet vector representations are then aggregated by taking a weighted average.

The outputs of the 8-K filings and the tweet modules are the concatenated with the market features and passed through 2 fully connected layers to obtain the output.

For basic scaffolding, we borrow code from `https://github.com/victoresque/pytorch-template` and also we base our vocabulary class on the CS224N assignment 4 starter code.

3.4 Baseline

We use three main baselines for Task 1, following [1]. The first is a random classifier, and the second is a majority classifier that simply predicts the majority class in the dataset (`UP` in our case). The third baseline we use does not use textual features and only uses 2 scalars- the VIX volatility and the earnings surprise. Task 2 uses correspondingly similar baselines, except earnings surprise is replaced by current day's stock performance and the log of trading volume on current day.

4 Data

We use different datasets for the two tasks outlined above. Notably the second dataset was created for this work and includes twitter information.

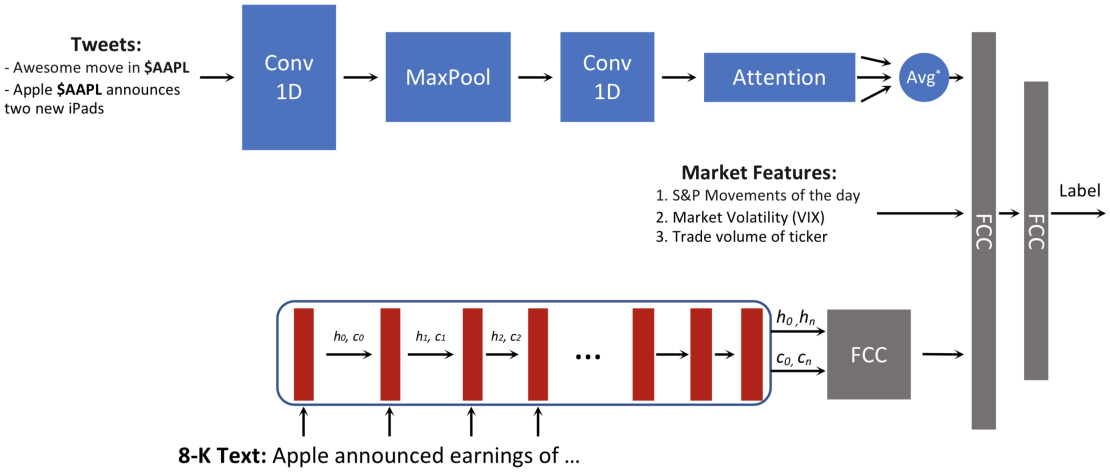

Figure 1: Main model used. For each data point we send a stack of 100 tweets through the CNN and then we compute an attention distribution over all the tweets passed in. We take a weighted average of the vector representation of each tweet by attention which we pass into the fully connected layer. Tweet based attention used here follows the work done in [4].

4.1 Task 1

For this task we use the dataset presented in [1]. It includes processed 8-K filings for 1500 companies as well as price history and earning surprise data that span from 2002 to 2012. This can be found here: nlp.stanford.edu/pubs/stock-event.html. In total, we obtain 17,300 training examples (2002-2008), 9,600 validation examples (2009-2010) and we hold out 10,000 test examples (2011-2012).

We process the text and we match each 8K filing with the correct earning surprise. We only consider the subset of 8K filings that announce earnings for a company. For each earning surprise we obtain the associated price movement of the company and get the next day normalized stock price movement as a label. We normalize based on the movement of the S&P500 index on the same day.

4.2 Task 2

For the second task we seek to incorporate more textual features in the form of twitter information. For this we set up a completely new dataset covering the years 2013-2018. We build a pipeline that allows for scraping the relevant price information and SEC fillings for any ticker symbol. We obtain the price information from yahoo finance using the scraper built by [5]. For SEC filings we use a scraper to pull all the 8-K filings. Within our pipeline we also process the received files to remove unnecessary metadata, figures and tables as well as html tags. After processing we keep the date, event tag and text for each filing. We also pull the volatility index of the market (VIX) on that date.

Furthermore, the pipeline we built is able to pull twitter data on the specific days of filing. We mainly chose 2013-2018 as this was a time when twitter was being used much more frequently. It does so by using a scraper for twitter data that can retrieve tweets on a given day that mention a given hashtag using the API specified in [6]. For each ticker in our list, we have the 8K filings, market features (volume of trades for the ticker, day's movement, VIX index) as well as all tweets that mention the ticker on that date. It is important to note that we could not collect earnings surprise data. This imposes a two-fold difference over Task 1: 1) the models do not use earnings surprise as an input feature, and 2) we are no longer restricted to earnings events and so we expand our events of interests beyond earnings results to managerial changes, bankruptcy, dividend notifications, etc.

We split our dataset as follows. $\sim 50,000$ data points covering Feb. 2013 until end of Dec. 2017 was our training set. Jan 2018 until June 2018 was our validation set ($\sim 5,000$) and we hold out a further $\sim 5,000$ data points from Jun. 2018 to Dec. 2018 as the test set.

5 Experiments

5.1 Training Details

We perform hyperparameter search over the following parameters- learning rate (0.001, 0.002), batch size (64, 128), max number of input tokens (1000, 2000), embedding dimension for 8K's (100, 200, 300), hidden dimension of LSTM (128. 256), L2 weight decay (0.01, 0.001, 0.0001). We use the Adam optimizer. Note that we only try certain combinations of these hyperparameters. For the CNN we used considered the number of tweets per event (50, 100). We used an embedding dimension of 100 for the twitter character embeddings. We use early stopping based on the validation accuracy. We note that one epoch took around 20 minutes.

5.2 Evaluation Method

Following [1], we split our labels into 3 categories— `UP` ($> 1\%$), `DOWN` ($< -1\%$) and `STAY`. We use accuracy as our primary metric.

We assign labels as follows:

$$l_i = \begin{cases} \texttt{UP} & \text{if } \delta_i - \delta_{snp} > 1\% \\ \texttt{DOWN} & \text{if } \delta_i - \delta_{snp} < -1\% \\ \texttt{STAY} & \text{else} \end{cases}$$

Where δ_i represents the percentage movement of the stock in a given 24 hour window and δ_{snp} represents the percentage movement of the S&P 500 over the given window (calculated based on the index tracker with ticker GSCP). This allows us to decorrelate company specific movements from market wide movements which introduce a substantial amount of randomness to the data. The big drawback of this approach is that it means we incorporate data from the next day to compute the label, making this a non-tradable strategy.

5.3 Results

5.3.1 Task 1:

The results are presented in Table 2. The validation accuracy is presented after averaging over 5 runs. We notice that the model without any textual features performs better than both the random guess and the majority classifier baselines. However, we notice that adding textual features as unigrams to the model decreases performance, contrary to what is reported in [1]. We believe that this anomaly is a consequence of incomplete hyperparameter search, as it should not be the case that adding more features to the model decreases its performance.

We note that our results are not directly comparable with those in [1], due to differences in data cleaning. We consider the label for a price movement to be how much the price moved 24 hours after the news came out. So if the filing came out after market hours on day n, we consider the price at the end of day $n + 1$. In [1] they consider the closest data point. We chose the larger window to try and decouple from reactionary swings in the data.

In addition, we see that adding textual information through embeddings confers an improvement to the performance over the model that does not use text features. This is promising since it is in concordance with the results in [1]. We believe that a better hyperparameter search will allow us to further increase the gap. However, we do not notice significant differences between the model that initializes the embeddings using pretrained GloVe embeddings and one that doesn't. This may be since the length of the 8-K filings is very high on average, and for our LSTM models we limit them to a length of 1000 or 2000 tokens.

5.3.2 Task 2:

We present the results for the models with twitter data in Table 3. As models took longer to train for the second task after incorporating the character level CNN on the tweets, results presented below are for single runs using the best combination of hyper-parameters from the search conducted.

We also note how important text based features are to our results, allowing for a **3.5%** increase in performance over our baselines. However, although we do successfully beat our baselines we find

Table 2: Performance of models for task 1

Features	Model	Validation Accuracy
-	Random Guess	33.33
-	Majority Classifier	41.78
VIX + Earnings Surprise	2-layer Feed Forward Network	49.29
VIX + Earnings Surprise + 8-K text	Bag of words for text + 2-layer Feed Forward Network	47.23
VIX + Earnings Surprise + 8-K text	LSTM + 2-layer Feed Forward Network	51.30
VIX + Earnings Surprise + 8-K text	LSTM + GloVe + 2-layer Feed Forward Network	**51.45**

that margin is not very high and this is due to the inherent noisiness of the data. An important point to note is that we did not have access to the earnings surprise data, and hence this increased the model's reliance on textual data. As a result, we saw that the ranking of the tweets and weighted averaging yielded exciting qualitative results. What is interesting to note is that tweets mentioning how the company's earnings performed against the estimates of analysts are consistently ranked highly.

Table 3: Performance of models for task 2

Features	Model	Validation Accuracy
-	Random Guess	33.33
-	Majority Classifier	49.13
VIX + Trade Volume	2-layer Feed Forward Network	49.56
Tweets + 8K + VIX + Trade Volume	LSTM + CharCNN + Dropout + 2-layer Feed Forward	50.92
Tweets + 8K + VIX + Trade Volume	LSTM + CharCNN with Attention + Dropout + 2-layer Feed Forward	**52.61**

6 Analysis

In this section, we present an analysis of our models. We focus on the model for Task 2 since the model takes as input only VIX index, 8K filings and tweets but does not take the earnings surprise. As a result, the model has increased reliance on textual features, and we hope that this will lead to meaningful importance attributions. For the analysis that follows, we use the Task 2 model with attention, as shown in Figure 1. Examples from the validation set have been used for the analysis.

6.1 Attention Analysis

For each data example, we provide the model 100 tweets, and the model then learns a vector representation for each tweet after running a character-level CNN. As a way to summarize these tweets, we initially took an average of the vector representations over the 100 tweets to condense the information into a manageable form. However, a better strategy was to calculate attention scores over tweets and then take a weighted average of the vector representations using the attention scores as weights. This makes sense because often tweets that mention a given ticker don't necessarily have useful information, and are often generated by bots.

For various examples, we rank our tweets by the attention scores provided by the model. Two of them are shown in Figure 2. In the first example, the top ranked tweets contain earnings results for the stock, which is a strong predictor of the price movement. The bottom ranked tweets are more general tweets that are bot generated and not specific to the stock in question. In the second example, the top ranked tweets discuss management changes and divident returns, which are also known to impact the stock price. As in the previous example, the bottom examples are non-specific tweets or tweets that do not have much useful information (e.g. "$<TKR>$'s sentiment is 1.02").

In general, we see that the model learns to correctly prioritize information rich tweets that are relevant to stock price movements- e.g. earnings results, changes in management, dividend, upgrade/downgrade ratings from analysts. The ability to rank tweets and filter those that are below a cutoff threshold can potentially be a useful tool for investors. Given the high volume of tweets for some tickers, it may not be possible for investors to follow all of them, and so filtering tweets and reporting the most important ones can be a useful application.

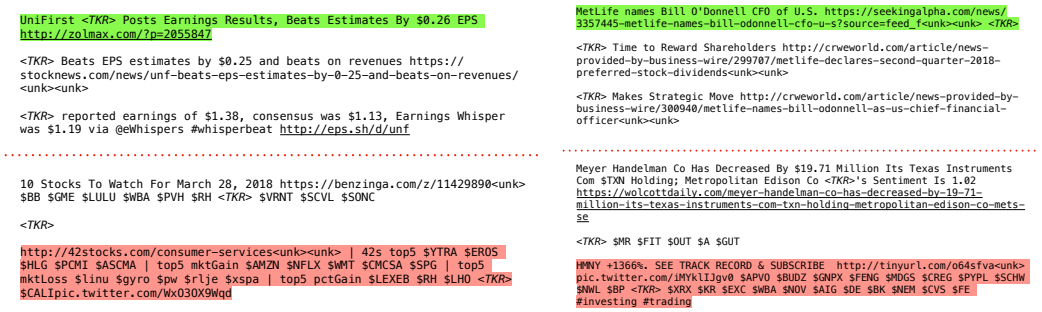

Figure 2: Tweets ranked by attention for two different examples. The tweets above the dashed red line show the top ranked tweets and the tweets below show the bottom ranked tweets. *<TKR>* is a placeholder for the ticker of the current company in question. The tweet in green highlight has the highest attention score, while that in red has the lowest.

6.2 Error Analysis

In order to understand the performance of the model, we manually examine some of the examples where the model performs correctly, and a few where it does not. Since the performance of the model is about 4% better than the performance of the majority classifier, we suspect that there is a high degree of randomness and external influences, as is routine is stock market prediction.

Since the most common label is `STAY`, we look at those examples for which the model predicts `UP` or `DOWN`. We then look at the top ranked tweets by attention for these examples. In general, the model is confident of its `UP` predictions when the tweets contain positive news on earnings, such as "MSA Safety beats by $0.37, beats on revenue", and confident of `DOWN` predictions when the tweets contain negative news on earnings, such as "EPR Properties misses by $0.13, misses on revenue". Interestingly, if the company beats on earnings but misses on revenue, the model generally predicts `DOWN`. In the examples in which the model is confident of its predictions but is wrong, the model prediction is usually still in line with the earnings result. However, the label is not as expected— e.g. "SNX Beats EPS estimates by $0.11 and beats on revenues" has a true label `DOWN`. Such cases exemplify the randomness and external influences that determine stock prices.

We next look at cases where the model predicts `STAY` but the true labels are `UP` or `DOWN`. In some cases, the model misses some signs— e.g. in "Jefferies Group Raises American Eagle Outfitters FY2018 Earnings Estimates to $1.17 EPS (Previously $1.16).", the model misses the obvious positive sentiment and instead predicts `STAY`. In most cases, however, there is no evidence in the tweets that hint at a significant movement in the stock price. This again highlights the high degree of randomness in the data randomness in the movements.

In the future, we would like to explore feature importance scores for 8K filing text to see how the 8K filings influence the model's decisions. Additionally we make an assumption about the main information of an 8K document is captured in the first 1000 words. We validate this qualitatively by going through many 8K's, however, we do inevitably lose out on information by reducing the text size.

7 Conclusion

Following [1], we attempt to examine the value of textual information to predict stock price movement following 8-K filings. We use market features such as VIX index, earnings surprise, trading volumes, and textual features in the form of raw 8-K filings as well as tweets. In Task 1, we incorporate 8-K filings by using a single-layer bidirectional LSTM to encode the input text. In Task 2, we use a character-level CNN to process a batch of tweets, followed by an attention weighted average of the tweets, in addition to the 8-K filings LSTM. Across both our tasks, we see that adding textual features leads to an improvement in the predictive performance of the model. We also qualitatively analyse the model— we perform an error analysis as well as look at the attention weights learned by the model. The attention weights reveal that the model learns to prioritise tweets that are information

rich and relevant to stock price movements. We also note that the model learns fairly intuitive rules to arrive at predictions, but is often incorrect due to the inherent randomness and external influences in stock prices. We believe that the power of the model can be further enhanced by training on a larger amount of data, and perhaps increasing the scope of events covered to those beyond 8-K filings. In conclusion, we see that incorporating textual features in analysis of stock price prediction yields improvements over solely using market based features.

References

[1] Heeyoung Lee, Mihai Surdeanu, Bill Maccartney, and Dan Jurafsky. On the Importance of Text Analysis for Stock Price Prediction. Technical report.

[2] Manuel R. Vargas, Beatriz S. L. P. de Lima, and Alexandre G. Evsukoff. Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 60–65. IEEE, 6 2017.

[3] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Technical report.

[4] Taro Miyazaki, Yuka Takei, Ichiro Yamada, Jun Goto, and Shin Toriumi. Extracting Important Tweets for News Writers using Recurrent Neural Network with Attention Mechanism and Multi-task Learning. Technical report.

[5] Łukasz Banasiak. Yahoo Finance API, 2015.

[6] kennethreitz. Twitter Scraper, 2018.