Sentiment Predictability for Stocks

Jordan Prosky[1], Andrew Tan[2], Xingyou Song[3], Michael Zhao[4],

***Abstract*— In this work, we present our findings and experiments for stock-market prediction using various textual sentiment analysis tools, such as mood analysis and event extraction, as well as prediction models, such as LSTMs and specific convolutional architectures.**

CONTENTS

*CS 294-136 Final Project

[1] jorpro@berkeley.edu
[2] andrewtan@berkeley.edu
[3] xsong@berkeley.edu
[4] m.zhao@berkeley.edu

I. INTRODUCTION

Stock market prediction is an important time-series learning problem in finance and economics. According to the semi-strong form of the efficient market hypothesis (EMH), markets are efficient and stock prices already reflect all publicly available information such as financial news and price data. This hypothesis incorporates the weak-form EMH, which implies that stock prices reflect all market information, a subset of public data. Semi-strong form EMH assumes that the market adjusts quickly to absorb new information and stock prices reflect all new available information. Given this assumption and the fact that investors purchase stocks after public information is released, an investor theoretically cannot benefit or outperform the market by trading on new information. Several studies contradict this hypothesis, and through our research, we show that we are able to outperform the market using a combination of deep learning, temporal data, and news articles.

Our report focuses on the use of various time-series and NLP techniques to extract information about an individual stock's chance of closing higher or lower than its opening price. We take advantage of the temporal nature of both stock price data and financial news data. In Section 2, we explore the current state-of-the-art in stock movement classifcation and the use of deep learning in financial prediction problems. Current advancements in sentiment analysis for stock markets focus on prediction for entire indexes. Section 3 outlines the problem of individual stock prediction and potential solutions to addressing these differences.

Our experiments, based on the state-of-the art models however show that perhaps more investigation (in network architecture, size of data, hyper-parameter tunings, and setup) is needed to produce better accuracies, and is an addition to the current literature on this topic.

II. RELEVANT AND PREVIOUS WORKS

Previous works in this area are sparse and may be dubious and untrustworthy. Furthermore, very few of the works specifically focus on individual companies and relative stocks. Nonetheless, we reviewed previous works. The first seminal paper relating the role of negative words in financial news documents to stock markets was [PCT08], where the author explored the relationship between a "pessimism"

index in papers to the Dow Jones Industrial Average. The most basic form of the "pessimism" was the number of negative words in the Wall Street Journal. The second work attempts to explain the relationship between negative words and individual stock pricing, found in [Tet07]. Further attempts under this work using more recent Natural Language Processing Developments were found using data such as Twitter sentiment.

A. Aggregate Twitter Prediction of Dow Jones

There has been previous work on the macro-level sentiment of Twitter posts and its relationship with the index of aggregate markets, such as the Dow Jones Average. [BMZ11], cited 2800 times, found that there was statistical significance between the Granger Causality of certain aggregate "mood" scores of 340,000 daily Twitter posts and the Dow Jones index approximately 3-4 days later. Using a neural network, they predicted the direction ("up" or "down) with 86% prediction rate.

Following this work, there were numerous works studying which individual stocks from the S+P 500 indices and the Dow Jones Averages, followed the Twitter sentiment prediction more closely. These include [RACM15], [RS12] However, this is perhaps highly similar to finding which stocks have high correlation to those two average indices. Furthermore, many of the Twitter posts were irrelevant to the stocks themselves.

Because of a still dearth of work on predicting individual stock return changes, this project was focused on experimenting with popular individual stock prediction. We explore individual stock prediction, and quantify some of these results with experiments.

B. Stock Movement Prediction Using News Article Titles with SVMs

There has been previous work on using news article titles and deep recurrent neural networks for predicting directional stock movement. Many studies contradict the efficient market hypothesis, including [RPS09], which achieved 57.1% accuracy in predicting the directional movement of the S&P 500 Index using Support Vector Machines (SVMs) trained on a feature vector containing both financial article data and stock price data over the same time period. Radial basis function (RBF) kernel SVMs are well-suited for classification problems involving numerical representations of text data and have performed well on stock market prediction based on news headlines. However, SVMs are unable to take advantage of the temporal nature of news data, which offers an avenue for improvement.

C. Deep Learning in Finance

Financial markets produce massive amounts of publicly available data every second. Financial prediction problems such as pricing securities, constructing portfolios, and managing risk often involve intricate data interactions that are difficult to interpret or speculate in a purely economic model. Applying deep learning models to these problems can produce more robust and useful results than traditional methods. In particular, deep learning can detect and exploit patterns in the data that are invisible to existing financial economic theory. The deep learning hierarchy has a number of advantages in prediction and classification: overfitting is more avoidable, non-linearities and complex data patterns can be accounted for, and input data can be expanded to include all possibly relevant features. In [HPW16], Heaton et al. introduced a variety of deep learning methods with applications in finance including dropout for model selection, autoencoders, and LSTM models. One application of deep learning they demonstrated was using an autoencoder to replicate and reconstruct a stock index with a subset of stocks. The sequential nature of financial data fits well with RNN models like LSTMs and GRUs. We want to extend upon existing research in deep time series analysis to create strategies for financial prediction.

III. Individual Stock Prediction

A. Difficulties

While the papers may state that they used many data sources, finding these sources is not an easy task. Also, prediction using relevant individual news articles to each stock is significantly more difficult, both practically and conceptually, than aggregate Twitter $\rightarrow$ aggregate stock prices.

The practical reasons include:

- It is difficult to have a reliable method of news article scraping. Data collection is both expensive monetarily, and in many cases, websites may block any basic web-crawling.
- Because news article collection is difficult, and news articles may be sparse, it is therefore difficult to get a sufficient number of articles per day.
- In terms of reading any HTML files collected, much of it may be unreliable due to the special encodings for many website.

The conceptual reasons also include:

- News articles do not typically follow normal sentiment patterns. Many articles are written objectively and without emotion, which makes judgement by an article's "happiness, sadness", etc. unreliable. Also, a specific news site may generate an underlying bias to the sentiment.
- Following this, most sentiment analysis does not keep track of a "importance" score. For example, one day's volume (i.e. total trades) may be based on a significant news event, but sentiment can only process mood. It is generally known that election days for the US significantly affect stock prices, but news reports may under-report this under a sentiment score.
- The sparsity of articles therefore makes statistical significance problematic - there may only be sparsely 1 or 2 articles per day, compared to the 340,000 Twitter posts that can be used per day.
- It is not known what is the correct pattern to focus on, and the usual parameters (time lag, meta-data of

stocks) are unknown. For example, one might approach this with: 1. Pos/Neg Sentiment vs. Daily (closing - opening) price, with days of lag, or 2. Intensity of article vs stock volume, or even perhaps 3. Immediate stock pricing, 1 hour after an release of an article.

B. Advanced Textual Information Approaches

There are few works that attack the single-stock problem, but require significantly more sophisticated methods. [DZLD15] notes that news titles are actually more useful for prediction than the news content themselves, with the reference [RDM12] showing these experimental results. The two references both use a similar method to encode events.

They attack the problem of learning from a title by representing each event as (O_1, O_2, P, T) tuple , where O_1, O_2 are two sets of objects, P is a relation of the objects where O_1 "acts" P on O_2, and T is a time interval. An example given is: Sep 3, 2013 - Microsoft agrees to buy Nokias mobile phone business for 7.2 billion. being modeled as: (Actor = Microsoft, Action = buy, Object = Nokias mobile phone business, Time = Sept 3, 2013). They use OpenIE to extract all possible event tuples from a title, and

They also noted many of the practical problems above, and addressed these issues:

- The most available textual information was from Bloomberg and Reuters. Furthermore, there was no gain in performance from using an entire article compared to using an article's summary according to [RDM12].
- Furthermore, the problem of sparse news was solved by encoding each summary into the event tuple III-B, and using StanfordNLP packages to convert an event to a "generalized event", where words such as verbs were classified into a category (e.g. "bought", "purchase", "auction" $\rightarrow$ "buy").
- It seemed that testing 1 day, 3 days, and 1-week lags were the most promising time delays from news article to stock price.
- Lastly, the problem of articles possessing more "dimensionality" in their meanings other than mood was solved by using event extraction for classification.

IV. Experiments

A. Stock Price and Financial News Data

We collected and preprocessed (normalized data through the min-max method) daily stock data using Google Finance for the 30 DJIA stocks from the last 15 years. We also collected data on the S&P 500 and Dow Jones Industrial Average indexes, which track entire industries. The daily features encompassed by our model include:

- Opening price: stock price at 9:30 AM ET
- Closing price: stock price at 4:00 PM ET
- High: highest price of the day
- Low: lowest price of the day
- Volume: number of shares traded during that day

For binary classification tasks, we generate labels that indicate whether a stock moved up or down intraday. We compare the opening and closing prices of the day: if the closing price was greater than or equal to the opening price, we set a label of 1, otherwise, it was given a label of 0. For our LSTM model using stock price data, our input data is a s x 5 matrix, where s represents our LSTM window size and 5 is the number of daily features we collected.

1) Websites Used: We attempted to use the following websites: `nytimes.com`, `cnn.com`, `reuters.com`, `wsj.com`. However, only `reuters.com` was accessible for large-scale scraping; all other websites either blocked the scraping request, or did not properly organize news into corresponding stocks.
Thus, we used only `reuters.com` news articles.

2) Sentiment Analyzer: We attempted to use the following sentiment analysis tools:

- VADER
- Google Sentiment API.

When using VADER [HG14], which is part of the NLTK for Python based on a work for internet sentiment trained on news corpus, this sentiment analysis tool has 4 scores: compound, neutral, positive, and negative. The last 3 scores sum to 1, while the compound score is a measure of the "intensity" of the input. Google Sentiment API [1] was also considered, however empirical tests showed that it displayed similar values with VADER, as well as being unavailable for free-to-use.

3) OpenIE Extractor and Word2Vec: StanfordNLP's OpenIE Extractor can be found in `https://stanfordnlp.github.io/CoreNLP/openie.html`. Once it converts each sentence into a event tuple,we then used the Word2Vec Skip-Gram Algorithm, found in `https://pypi.python.org/pypi/word2vec`.

4) Deep Learning Packages: TensorFlow was used to output the event embeddings, while Keras was used for LSTM-modelling. Lastly, PyTorch was used for the convolutional network for up-down prediction.

B. Setup

1) LSTM-Based Mood Prediction: We use NLTK sentiment scores as well as previous prices in order to create multi-step predictions. Our method is able to work for many different companies, but for this example, we focus on 600 days of Amazon (AMZN) returns. The forecasting LSTM we use can take in an arbitrary length input sequence and predict the closing prices for the next k days. We use a simple LSTM due to the small size of our data set, with only 8 hidden units, trained for 5 epochs. We dedicate the last 50 days to be our test set to evaluate the model trained on the previous time steps.

Although not presented, during many experiments we found that the model's predictions rely heavily on the length of the training set and the input sequence. The

[1] `https://cloud.google.com/natural-language/docs/analyzing-sentiment`

optimal length of training data varies over time and security, so we experimented with trial and error for the above results, where 600 days of data seemed to yield reasonable predictions.

2) Event Extraction using Neural Networks: In this section, we try to represent our article titles as a single vector that we can flexibly pass into any classifier or regressor. We start by learning 100-length word embedding vectors using the Word2Vec skip-gram algorithm for the words in our news titles. With the word embeddings, we could simply average the word embeddings of all words in a news title, but we decided to use a more complex approach that should produce vectors that better capture the relationships between words in the titles.

From StanfordNLP, we used OpenIE extraction, as seen from [DZLD15] and explained in III-B, to extract a sentence's underlying event structure, which for every news title yields between 0 and 10 possible event tuples. We then converted our event tuples into word embedding tuples. To get a single word embedding tuple E for each news title, we averaged the up to 10 possible word embedding tuples. At this point, we fed our word embedding tuples into a neural tensor network (below in figure 1) in order extract each brief summary into a single 100-length vector event embedding.

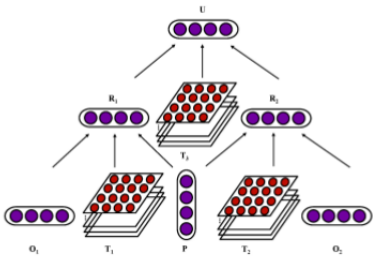


Fig. 1. Event embedding architecture found in [DZLD15].

This network attempts to learn the relationships between the actor O_1 and action P and between the action P and object O_2 to learn $R1$ and $R2$, respectively, then learn the relationship between $R1$ and $R2$ to produce the final predicted event embedding U which should capture the overall meaning of the article title. For every article, there is a learned event embedding M, which is a trainable parameter. To score an embedding, we take the inner product between M and the predicted event embedding U.

As in [DZLD15], we create corrupted event tuples E^r where we replace the actor embedding vector with the word embedding of a random possible word, which we also pass through our network to produce corrupted event embeddings U^r. We can then compute the margin loss between the actual event tuple and the corrupted one with ℓ_2 regularization to train our network.

$$MarginLoss(E, E^r) = max(0, 1 - M\dot{U} + M\dot{U^r})$$

$$\ell_2\text{-loss} = ||(T_1, T_2, T_3, W, b)||_2^2$$

We set our ℓ_2 regularization weight λ to 0.0001 and trained for 500 epochs to produce the best event embeddings as seen below in figure 2.

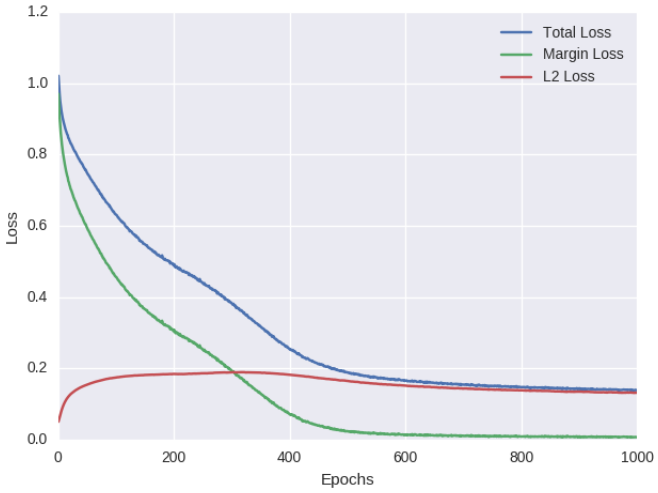


Fig. 2. Training losses for neural tensor network learning event embeddings.

Such event embeddings may be used in traditional classifiers: SVM, Naive Bayes, etc. as well.

3) LSTM using Event Embeddings: Here, we have sentence embeddings for news headlines, and we use them as features with the goal of predicting whether a given company's stock will go up or down the following day.

The data used consists of 8,000 sentence embeddings for different news article headlines, labeled with the future up or down movement of the company's stock. We use a simple LSTM model for binary classification, and see better-than-random results! We manage to consistently achieve 52% out-of-sample accuracy, while the training accuracy continues to increase as the model is trained. This leads us to believe that we simply do not have enough data to create a robust model, as evident by the drastic over-fitting. Nonetheless, we see that there is some promise in using LSTMs as a mapping between sentence encodings and stock price movements.

4) Convolutional Networks for Long, Medium, Short Term using Event Embeddings: (Found in NN_event_to_prediction.ipynb). We replicated the neural network experiments found in [DZLD15], in which the word embeddings $(U_{day}, \{U^i_{week}, i = 1, 2...\}, \{U^j_{month}, j = 1, 2, ...\})$ were injected into a feed-forward network consisting of convolutional filters for long-term $\{U^j_{month}, j = 1, 2, ...\} \rightarrow V^{month}$ (set of all events j in the month before day d), middle-term events $\{U^i_{week}, i = 1, 2, ...\} \rightarrow V^{week}$ (set of all events i in the week before day d), concatenated with a short term (day d's article $U_{day} = V^{day}$). The concatenation $V = \left[V^{month}, V^{week}.V^{day}\right]$ was inputted into a fully-connected hidden layer, with sigmoid transforms for both concatenation $\rightarrow$ hidden layer ($H = \sigma(W_1^T \cdot V)$), and hidden layer $\rightarrow$ output ($O = \sigma(W_2^T \cdot H)$). Because of the variability of the lengths of the monthly and weekly data, 0-padding

was used. A soft-max was also applied on the output for the final 2-length output ($y_{class}, class \in \{up, down\}$ where $y_{class} = Softmax(O)$), in order to interpret as probabilities. This was then used to predict the up-down classification for the price change between day d and day $d+1$.

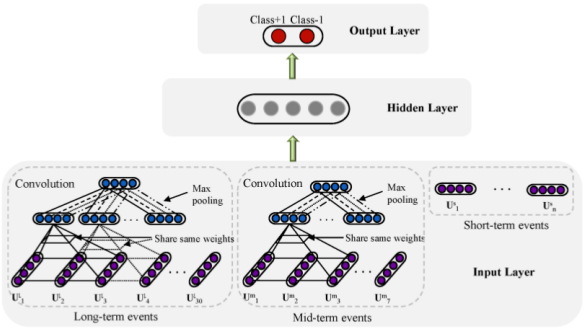


Fig. 3. Feed-forward architecture found in [DZLD15]. Note that the middle-term and long-term events will be inputted into convolutions and max-pools to extract the best features. The network was performed in PyTorch.

We also based a network purely on the short-term events, (without convolution on past events), as a reference.
Because of the need for sequential processing (i.e. we should not train on a random sample and test on samples whose timing is somewhere between the random samples), we used window-ed cross validation, where r of the first events were used for training, and the rest of the most recent events for testing, where $0 < r < 1$ is a ratio.

V. Results

A. Data Numbers

We present basic figures for our data. IBM and AMD were not used in the final experiments due to lack of clean data.

	Stock Name	Number of Articles
0	GOOGL	528
1	INTC	409
2	AAPL	2292
3	CSCO	229
4	AMD	23
5	QCOM	351
6	NVDA	54
7	AMZN	1062
8	MSFT	830
9	IBM	415

Fig. 4. Number of Articles Used.

	Stock Name	Price Up Percentage
0	GOOGL	49
1	INTC	49
2	AAPL	48
3	CSCO	60
4	AMD	50
5	QCOM	45
6	NVDA	72
7	AMZN	48
8	MSFT	51

Fig. 5. Percentage of news articles with stock price heading up the next day. (IBM inconclusive)

B. Basic Exploratory Analysis

The correlation between the Vader NLK sentiment scores on day d of an article and the changes in price from day d to the price on days $d+1, d+3, d+7$ were all inconclusive, as somewhat expected. Regardless of which score (pos, neg, neutral, or intensity), there was less than an absolute value of 0.6 correlation between all stocks and their scores, shown in 7.

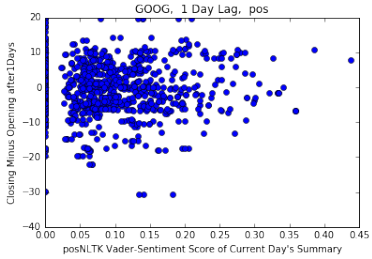


Fig. 6. Price Changes Compared with NLTK Positive Scores for GOOG.

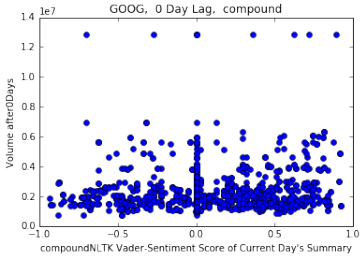


Fig. 7. Stock Volumes Compared with NLTK Compound (intensity) Scores for GOOG.

C. LSTM-Based Sentiment Score

Below, we present the full time series of the Amazon closing prices, as well as our predictions for the last 50 days (5-day future prediction at each of the last 50 days), denoted by the red lines.

For a clearer look at the behavior of our predictions, we zoom in on the above graph and present a close-up view of our predictions on the test set shown below:

Here, we see an upward bias of our predictions - likely due to the fact that the AMZN saw a huge price increase in the previous 100 days leading up to this snapshot.

D. Convolutional Network

We did not find the same results as [DZLD15] from replication. The hyperparameters (especially the size of the hidden layers in the convolutions and the fully-connected) were not given in the work, which made optimizing them difficult.

Roughly speaking, the parameters were:

TABLE I
HYPER PARAMETERS

Word-Embedding Vector Length	100
Final Hidden Layer	200
Short-Term Hidden Layer	150
(Monthly, Weekly) Convolutional Hidden Layer	(40,20)
(Monthly, Weekly) Convolution Window Size	(3,3)
Optimizer	ADAM
(Day, Week, Month) Lags	(1,7,30)
(Batch Size and Learning Rate)	(50,0.001)

The results are shown below when epoch size was 50:

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.788, 0.525)	(0.738, 0.548)	(0.775, 0.5)	(0.720, 0.480)	(0.725, 0.552)
1	INTC	(0.670, 0.481)	(0.751, 0.515)	(0.728, 0.5)	(0.595, 0.390)	(0.705, 0.5)
2	AAPL	(0.780, 0.476)	(0.766, 0.489)	(0.748, 0.491)	(0.800, 0.449)	(0.770, 0.512)
3	CSCO	(0.663, 0.604)	(0.654, 0.602)	(0.640, 0.6)	(0.659, 0.5)	(0.684, 0.722)
4	QCOM	(0.586, 0.568)	(0.676, 0.559)	(0.705, 0.608)	(0.596, 0.478)	(0.593, 0.391)
5	NVDA	(0.642, 0.571)	(0.705, 0.454)	(0.65, 0.5)	(0.652, 0.4)	(0.615, 0.5)
6	AMZN	(0.721, 0.523)	(0.681, 0.518)	(0.694, 0.522)	(0.693, 0.526)	(0.743, 0.4)
7	MSFT	(0.698, 0.503)	(0.704, 0.555)	(0.721, 0.505)	(0.729, 0.495)	(0.734, 0.5)

Fig. 8. (Training Accuracy, Test Accuracy) for Full (Month, Week, Day) Network with epoch of 50.

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.695, 0.515)	(0.527, 0.451)	(0.533, 0.448)	(0.594, 0.558)	(0.654, 0.447)
1	INTC	(0.602, 0.468)	(0.554, 0.437)	(0.524, 0.5)	(0.677, 0.406)	(0.650, 0.375)
2	AAPL	(0.590, 0.467)	(0.576, 0.520)	(0.586, 0.487)	(0.583, 0.449)	(0.582, 0.461)
3	CSCO	(0.663, 0.604)	(0.654, 0.602)	(0.640, 0.6)	(0.659, 0.5)	(0.618, 0.722)
4	QCOM	(0.586, 0.568)	(0.561, 0.591)	(0.546, 0.637)	(0.596, 0.478)	(0.593, 0.391)
5	NVDA	(0.642, 0.571)	(0.705, 0.454)	(0.65, 0.5)	(0.652, 0.4)	(0.615, 0.5)
6	AMZN	(0.657, 0.494)	(0.588, 0.471)	(0.641, 0.491)	(0.524, 0.613)	(0.632, 0.546)
7	MSFT	(0.495, 0.463)	(0.655, 0.547)	(0.639, 0.516)	(0.655, 0.561)	(0.641, 0.583)

Fig. 9. (Training Accuracy, Test Accuracy) for Short-Term (Day) Network Only with epoch of 50.

The epoch size was then increased to 500 to improve training accuracy:

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(1.0, 0.530)	(1.0, 0.529)	(0.996, 0.439)	(0.993, 0.467)	(0.994, 0.552)
1	INTC	(0.987, 0.512)	(0.994, 0.476)	(0.991, 0.447)	(0.968, 0.437)	(0.968, 0.468)
2	AAPL	(1.0, 0.453)	(1.0, 0.476)	(1.0, 0.474)	(0.998, 0.521)	(0.999, 0.435)
3	CSCO	(0.989, 0.505)	(0.981, 0.561)	(0.976, 0.545)	(0.965, 0.5)	(0.975, 0.611)
4	QCOM	(1.0, 0.551)	(0.985, 0.505)	(1.0, 0.550)	(0.983, 0.565)	(0.961, 0.478)
5	NVDA	(1.0, 0.5)	(1.0, 0.545)	(1.0, 0.75)	(1.0, 0.8)	(1.0, 1.0)
6	AMZN	(1.0, 0.569)	(0.993, 0.571)	(0.992, 0.539)	(0.988, 0.466)	(0.989, 0.573)
7	MSFT	(0.996, 0.480)	(1.0, 0.460)	(0.990, 0.532)	(0.997, 0.495)	(0.998, 0.466)

Fig. 10. (Training Accuracy, Test Accuracy) for Full (Month, Week, Day) Network with shuffling with epoch of 500.

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.943, 0.577)	(0.914, 0.483)	(0.915, 0.534)	(0.871, 0.532)	(0.88, 0.473)
1	INTC	(0.962, 0.493)	(0.953, 0.531)	(0.924, 0.531)	(0.902, 0.468)	(0.889, 0.437)
2	AAPL	(0.824, 0.485)	(0.815, 0.489)	(0.846, 0.495)	(0.85, 0.488)	(0.941, 0.480)
3	CSCO	(0.967, 0.604)	(0.945, 0.616)	(0.960, 0.545)	(0.952, 0.444)	(0.939, 0.5)
4	QCOM	(0.965, 0.543)	(0.964, 0.559)	(0.957, 0.550)	(0.962, 0.5)	(0.913, 0.565)
5	NVDA	(1.0, 0.428)	(1.0, 0.545)	(1.0, 0.5)	(1.0, 0.6)	(1.0, 1.0)
6	AMZN	(0.875, 0.521)	(0.847, 0.544)	(0.840, 0.561)	(0.849, 0.493)	(0.836, 0.52)
7	MSFT	(0.878, 0.493)	(0.885, 0.522)	(0.868, 0.489)	(0.856, 0.504)	(0.852, 0.433)

Fig. 11. (Training Accuracy, Test Accuracy) for Short-Term (Day) Network Only with shuffling with epoch of 500.

From above, both networks dramatically improved their training accuracy. However, only the short-term network showed a consistent increase in testing accuracy.

For the full-network, it did not necessarily improve testing accuracy, showing possibility of overfitting. Using a training ratio of 0.7, we provided a graph (under no shuffling) for the relationship using the AAPL stock:

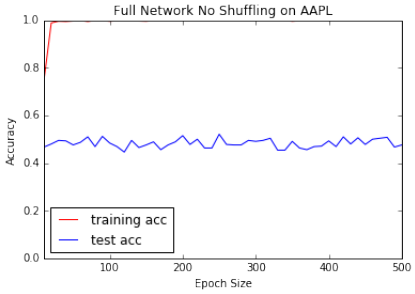


Fig. 12. (Training Accuracy, Test Accuracy) for Full (Month, Week, Day) Network with no shuffling with varying number of epochs.

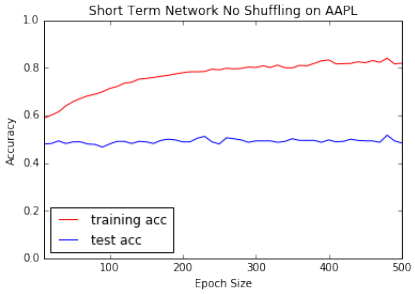


Fig. 13. (Training Accuracy, Test Accuracy) for Short-Term (Day) Network Only with no shuffling varying number of epochs.

It is clear that the full network can easily achieve close to 1.0 training accuracy very quickly and can strongly overfit. From the hyperparameters, the final hidden layer size asymptotically improved the training accuracy until there was a limit. Part of the reason for the full-network's overfitting could be due to lack of enough data points. We tested whether shuffling the training set made a difference as well, although experiments were first meant to be run sequentially. Furthermore, the LSTM-based models inherently uses the articles in chronological order.

The result was inconclusive, as there was not a general trend of improvement.

The results for epoch of 50 are shown:

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.762, 0.520)	(0.768, 0.516)	(0.786, 0.465)	(0.704, 0.519)	(0.697, 0.447)
1	INTC	(0.689, 0.5)	(0.777, 0.523)	(0.733, 0.531)	(0.723, 0.484)	(0.709, 0.5)
2	AAPL	(0.869, 0.475)	(0.919, 0.500)	(0.850, 0.459)	(0.919, 0.466)	(0.923, 0.5)
3	CSCO	(0.663, 0.604)	(0.654, 0.602)	(0.640, 0.6)	(0.659, 0.5)	(0.618, 0.722)
4	QCOM	(0.586, 0.568)	(0.618, 0.559)	(0.582, 0.579)	(0.596, 0.478)	(0.593, 0.391)
5	NVDA	(0.642, 0.571)	(0.705, 0.454)	(0.65, 0.5)	(0.652, 0.4)	(0.653, 0.5)
6	AMZN	(0.721, 0.574)	(0.626, 0.491)	(0.711, 0.557)	(0.666, 0.406)	(0.778, 0.453)
7	MSFT	(0.701, 0.536)	(0.743, 0.518)	(0.749, 0.538)	(0.745, 0.495)	(0.737, 0.483)

Fig. 14. (Training Accuracy, Test Accuracy) for Full (Month, Week, Day) Network with shuffling with epoch of 50.

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.737, 0.551)	(0.545, 0.458)	(0.680, 0.474)	(0.636, 0.467)	(0.631, 0.394)
1	INTC	(0.534, 0.468)	(0.544, 0.437)	(0.68, 0.510)	(0.587, 0.437)	(0.633, 0.437)
2	AAPL	(0.6, 0.501)	(0.588, 0.499)	(0.594, 0.484)	(0.589, 0.456)	(0.582, 0.461)
3	CSCO	(0.663, 0.604)	(0.654, 0.602)	(0.640, 0.6)	(0.659, 0.5)	(0.618, 0.722)
4	QCOM	(0.586, 0.568)	(0.561, 0.591)	(0.546, 0.637)	(0.596, 0.478)	(0.593, 0.391)
5	NVDA	(0.642, 0.571)	(0.705, 0.454)	(0.65, 0.5)	(0.652, 0.4)	(0.615, 0.5)
6	AMZN	(0.655, 0.494)	(0.619, 0.461)	(0.647, 0.5)	(0.555, 0.38)	(0.638, 0.573)
7	MSFT	(0.580, 0.476)	(0.674, 0.530)	(0.679, 0.532)	(0.651, 0.570)	(0.655, 0.566)

Fig. 15. (Training Accuracy, Test Accuracy) for Short-Term (Day) Network Only with shuffling with epoch of 50.

The results for an epoch of 500 are also shown:

	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(1.0, 0.556)	(1.0, 0.503)	(0.996, 0.474)	(0.996, 0.428)	(0.994, 0.5)
1	INTC	(0.987, 0.55)	(0.984, 0.570)	(0.977, 0.479)	(0.964, 0.453)	(0.975, 0.437)
2	AAPL	(1.0, 0.489)	(0.998, 0.495)	(1.0, 0.484)	(1.0, 0.491)	(1.0, 0.389)
3	CSCO	(0.989, 0.560)	(0.981, 0.520)	(0.968, 0.6)	(0.959, 0.527)	(0.963, 0.611)
4	QCOM	(1.0, 0.5)	(0.985, 0.505)	(0.969, 0.550)	(0.989, 0.521)	(0.990, 0.521)
5	NVDA	(1.0, 0.571)	(1.0, 0.545)	(1.0, 0.75)	(1.0, 0.8)	(1.0, 1.0)
6	AMZN	(0.992, 0.5)	(0.995, 0.561)	(0.992, 0.548)	(0.990, 0.486)	(0.992, 0.533)
7	MSFT	(1.0, 0.483)	(0.997, 0.514)	(0.990, 0.527)	(0.997, 0.520)	(0.994, 0.5)

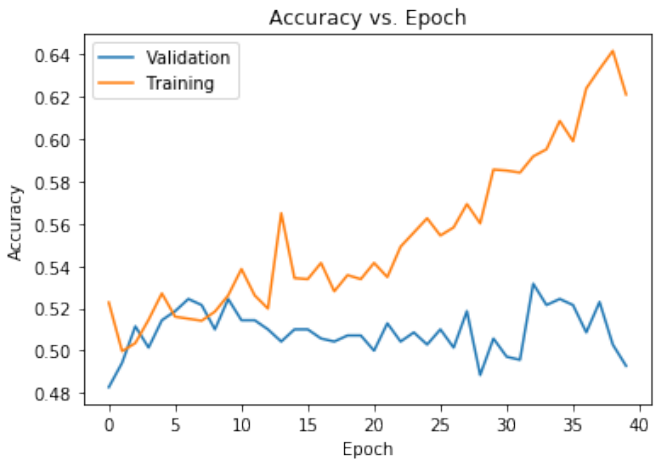
Fig. 16. (Training Accuracy, Test Accuracy) for Full (Month, Week, Day) Network with shuffling with epoch of 500.

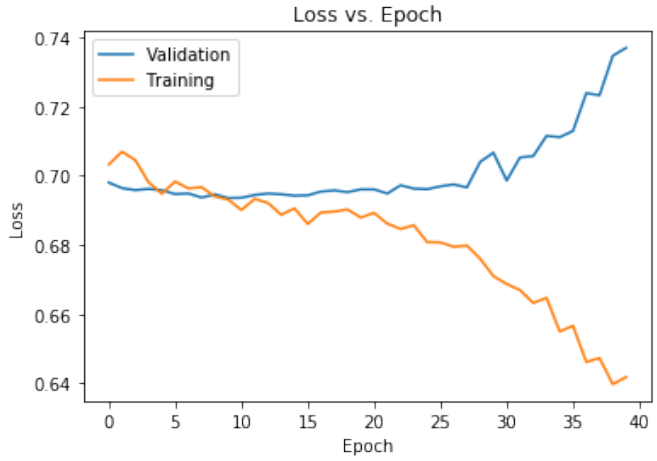
	Stock Name	0.5 Training Ratio	0.6 Training Ratio	0.7 Training Ratio	0.8 Training Ratio	0.9 Training Ratio
0	GOOGL	(0.943, 0.551)	(0.931, 0.548)	(0.908, 0.525)	(0.887, 0.493)	(0.888, 0.5)
1	INTC	(0.956, 0.562)	(0.948, 0.578)	(0.928, 0.552)	(0.922, 0.453)	(0.906, 0.343)
2	AAPL	(0.810, 0.475)	(0.8, 0.489)	(0.804, 0.476)	(0.805, 0.511)	(0.945, 0.480)
3	CSCO	(0.978, 0.604)	(0.945, 0.643)	(0.960, 0.509)	(0.945, 0.388)	(0.921, 0.611)
4	QCOM	(0.974, 0.543)	(0.992, 0.559)	(0.975, 0.608)	(0.967, 0.521)	(0.923, 0.434)
5	NVDA	(1.0, 0.428)	(1.0, 0.545)	(1.0, 0.5)	(1.0, 0.6)	(1.0, 1.0)
6	AMZN	(0.862, 0.510)	(0.856, 0.521)	(0.863, 0.544)	(0.827, 0.493)	(0.839, 0.56)
7	MSFT	(0.875, 0.529)	(0.893, 0.502)	(0.875, 0.543)	(0.852, 0.504)	(0.852, 0.4)

Fig. 17. (Training Accuracy, Test Accuracy) for Short-Term (Day) Network Only with shuffling with epoch of 500.

E. LSTM-Based Event Embeddings

We show learning curves (loss and accuracy) for our LSTM binary classifier on both the training and validation sets below:





This shows that the LSTM is very prone to overfitting similar to the convolutional network - as epoch increases, so does training accuracy, but validation accuracy stays constant, or validation loss increases.

VI. Conclusions

We summarized the available findings on the subject matter of individual stock prediction.

From our experiments, we noted essentially two patterns:

- Even with the current batch of event data, it was very easy to overfit for both LSTM and convolutional network models, implying much more data is needed.
- We did not achieve the strong results found in [DZLD15] with our convolutional re-implementations, which suggests either hyperparameters, data, or setup were slightly different.

Overall therefore, it is unknown whether deep learning within the sentiment analysis context may help with individual stock prediction. Possible future approaches include:

- We hope to see future work given more news data, more varying stocks, and different architectures. The current experimentation setup overall produced prediction rates close to 50%.
- It is possible that sentiment analysis on individual stocks may have better prediction rates with aspects of the stock other than price. For example, volume or liquidity may have better predictability because news articles are correlated with public opinion and tendency to trade.

VII. APPENDIX

The code and data may be found in `https://github.com/jorpro/DeepTimeSeries`. Essentially, the files are:

- NN_event_to_prediction.ipynb - This is the code for the neural network convolutional filter for up-down classification, found in IV-B.4.
- Stock_Data_Collection.ipynb - This collected daily stock prices and meta-data from Google Finance.
- nltk_scores.ipynb - This uses NLTK's VADER package to produce mood-based sentiment scores of sentences and visualizations based on the data.
- open_ie.ipynb - This uses Stanford NLP's OpenIE extractor for event extraction on a sentence.
- event_embedding.py - This produces event embeddings from the event tuple.
- cnn_classifier.py - Another version of the convolutional network from [DZLD15] in TensorFlow.
- multistep_pred.ipynb - Used VADER sentiment and previous prices for an LSTM.
- sentence_embedding_prediction.ipynb - Used the event embeddings into an LSTM.
- `https://github.com/jorpro/DeepTimeSeries/tree/master/Papers` - Relevant Papers used in references.

ACKNOWLEDGMENT

We thank Francois Belletti for suggesting various links and resources, and Wilson Cai for suggesting the project.

REFERENCES

[AYMU16] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information, 06 2016.

[Aza09] Pablo Azar. Sentiment analysis in financial news, harvard bachelor's thesis, April 2009.

[BMZ11] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *J. Comput. Science*, 2(1):1–8, 2011.

[DL16] Min-Yuh Day and Chia-Chou Lee. Deep learning for financial sentiment analysis on finance news providers, 08 2016.

[DZLD14] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425. Association for Computational Linguistics, 2014.

[DZLD15] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2327–2333. AAAI Press, 2015.

[HG14] Clayton J. Hutto and Eric Gilbert. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.*, 2014.

[HPW16] J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning in finance, 02 2016.

[IN17] Vasileios Iosifidis and Eirini Ntoutsi. Large scale sentiment learning with limited labels. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1823–1832, New York, NY, USA, 2017. ACM.

[PCT08] Sofus Macskassy Paul C. Tetlock, Maytal Saar-Tsechansy. More than words: Quantifying language to measure firms' fundamentals. In *The Journal of Finance*, 2008.

[RACM15] Gabriele Ranco, Darko Aleksovski, Guido Caldarelli, and Igor Mozetic. Investigating the relations between twitter sentiment and stock prices. *CoRR*, abs/1506.02431, 2015.

[RDM12] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 909–918, New York, NY, USA, 2012. ACM.

[RPS09] Hsinchun Chen Robert P. Schumaker. Textual analysis of stock market prediction using breaking financial news. In *ACM Transactions on Information Systems*, 02 2009.

[RS12] Tushar Rao and Saket Srivastava. Analyzing stock market movements using twitter sentiment analysis. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ASONAM '12, pages 119–123, Washington, DC, USA, 2012. IEEE Computer Society.

[Tet07] Paul C. Tetlock. Giving content to investor sentiment: The role of media in the stock market. In *The Journal of Finance*, 2007.