IE 598 **Final** Report

Text Based Stock Price Movement Prediction Models

Lingrui Gan; Yihe Wang
Fangwen Wu ; Wenjing Yin

Introduction

In algorithmic trading world, the main target is to find buy/short trading signals algorithmically. Given all the updated market information we received, whether we can predict the stock price movement, e.g., up or down, accurately is the problem of interest.

With the advent of machine learning in recent years, a great amount of effort has been made in this line of research and various machine learning models for the stock price movements have been proposed [1,2,3]. Considering their great capacity of capturing trends and patterns from massive amount of data, Machine Learning methods gained their popularity is unavoidable. Using historical data (e.g. stock prices before a certain time stamp), we can identify and predict the stock movement patterns with various machine learning frameworks, e.g., SVM, Boosting Trees, and traditionally, the numeric features of each stock, e.g., OHLCV (open, high, low, close, and volume), are used to build these models [4,5,6].

In our own experiences during trading, however, we found what affect the stock movement the most are actually text information like news, earning reports. If these text information can be leveraged and accurate predictive models based on that can be developed, that is more meaningful. In fact, researchers have already tried to incorporate text data into stock price prediction using classical machine learning models [9,10,11] and decent results have been demonstrated (~55 percent prediction accuracy).

In recent years, the frontier of NLP and text mining has being pushed forward with the trend of neural networks and deep learning [12,13,14,15,16]. Despite of this trend, we've only see little works in using deep learning methods to model the financial text information and predict the market [17,18]. It is of great curiosity for us to see if we could extract meaningful trading signals from the text using these advanced techniques.

With this thought, in this project, we try to take the initiative and explore this area, i.e. text mining and neural networks for financial data. We'd like to see if neural networks leveraged on text data can be useful in stock movement prediction, and whether they perform better in predictions than the classical methods.

In summary, the project goal is: we will try advanced NLP techniques(both non-neural network and neural network) to predict stock movement using news data as accurate as we can. Meanwhile, we'd like to compare deep learning models with classical models, and see if deep learning is really working here.
The project is specifically organized as follows. We first crawl news data from the Internet, e.g., websites Reuters and Reddit, and matched them with historical stock movements according to timepoints. Based on the text data we gathered, both classical machine learning models and different structures of neural networks are built. Performances of these models are then compared on these real data. In the remaining part of the report, we will discuss each part of the project separately following the same order when the project is done.

Datasets Descriptions

In this project, we analyzed news text data from two sources: Reddit and Reuters, and built models on these two datasets separately.

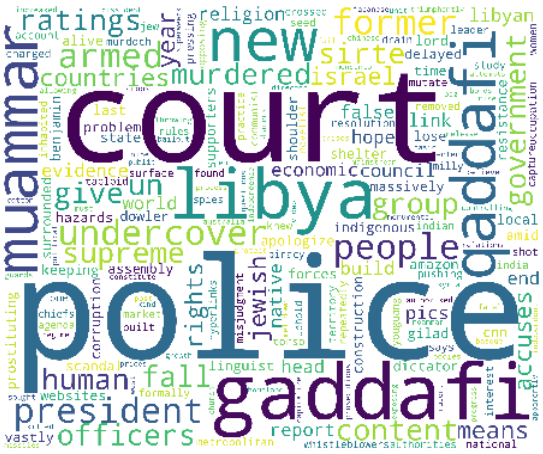
Reddit World News Data (3794 observations)

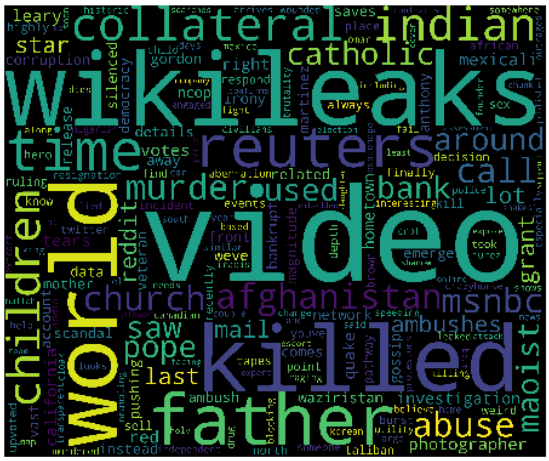
The first is the "Daily News for Stock Market Prediction" dataset from Kaggle. This data set contains the daily stock data as well as the corresponding daily news data from June 2008 to July 2016. The stock data comes from the Dow Jones Industrial Average (DJIA) while the news data is obtained from the Reddit World News Channel and in each day, only top 25 headlines ranked by the reddit's votes are considered.

Date	Label	Top1	Top2	Top3	Top4	Top5	Top6	Top7
2008-08-08	0	b"Georgia 'downs two Russian warplanes' as cou...	b'BREAKING: Musharraf to be impeached.'	b'Russia Today: Columns of troops roll into So...	b'Russian tanks are moving towards the capital...	b"Afghan children raped with 'impunity,' U.N. ...	b'150 Russian tanks have entered South Ossetia...	b"Breaking: Georgia invades South Ossetia, Rus...

The figure above shows the format of the data set with the label representing the rise or the fall of the Dow Jones Industrial Average index. Note that we have top 25 headlines in the actual data set.

This dataset is relatively small with the news and stock index data from 3793 days. In order to visualize the dataset and figure out the words that can be the indicator of the increase/decrease in stock price. We generate a positive word cloud on the bottom left and negative cloud on the bottom right.





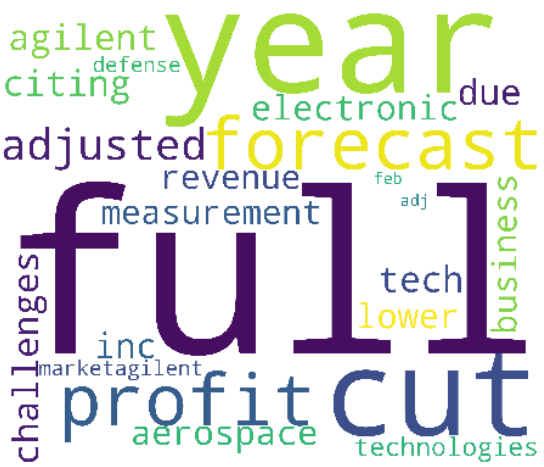
From the word cloud we can see the strong indicators of the increase in the stock price is related to the politics. Few words appeared on the word cloud have positive meanings such as ‘new’, on the other hand, the strong indicators of the decrease in the stock price is a mixture of word with negative meanings like ‘murder’, ‘abuse’, ‘killed’ and the neutral words such as ‘world’, ‘video’, ‘father’. We suspect the stock movement could be predictive based on news. Therefore, we continue to build models on the dataset.

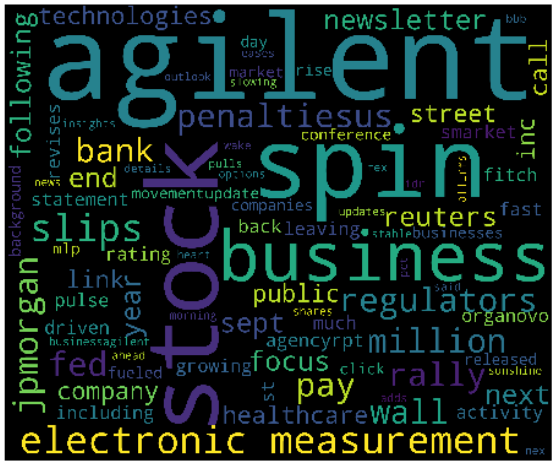
Reuters Financial News Data (60493 observations)

We made a larger dataset from crawling. We do this by first crawling the name of the ticker of the top 20% market cap companies from the NASDAQ website. Next, we crawl the date, headlines (title, subtitle) and the importance of financial news relevant to these top 20% market cap companies from Reuters. In addition, we crawl the company's’ stock prices on the day and the previous trading day of the news from Yahoo finance.

Ticker	Date	Diff	Title	Importanc	Prev
A	20130919	-0.03735	Agilent Tec	topStory	0.011214
A	20131122	-0.00463	Nov Agilen	normal	0.005561
A	20131216	-0.00018	Avago Tec	topStory	-0.00362
A	20140213	0.010541	Feb Agilen	normal	-0.00183
A	20160516	0.007715	Raises its f	normal	0.001644
A	20160627	-0.0319	Agilent Tec	normal	-0.00992
A	20160817	-0.00412	Q non gaa	normal	0.000207

After processing the collected data, we have our data set in the form shown above. Notice that in this data set we have 50:50 increase observations and decrease observations.





The word cloud generated from this data set shows the dominant words in the positive word cloud on the left is ‘full’, ’profit’, ’forecast’, ’cut’, ’year’, the dominant words in the negative word cloud on the right is ‘stock’.’spin’,’business’. It is worth noting that the word ‘ electronic’, ’technology’ appears in the both word cloud. The dominant word cloud words in this dataset is different from the previous dataset as we expected since we are using the financial news instead of the general word news.

We use the aforementioned two datasets to build models.

Models

To build models, we need to first project the text data into numeric features space through word embedding. In this project, we apply both classical methods, i.e., n-gram models and dimension reduction techniques like NMF (Nonnegative Matrix Factorization), and neural net methods, i.e., Word2Vec. The embedded features are then fed into machine learning models like neural nets and logistic regression for the predictions.

Embedding Methods

a. N-gram models:

In N-gram models, we take a contiguous N words from the text document and treated it as a feature. The text data will then be transformed to numeric matrix using the features just generated with one hot encoding. In our model, to be specific, we use bigram, i.e., N=2 , because we found it gives us the best results. For unigram with logistic regression, the prediction accuracy is only 42.5%.

After deleting the stopping words, which is defined as document frequency higher than 0.95 in our model, we also constrained the features matrix to only contain the top K frequent word

combinations to reduce noise and boost the performance, where K is chosen to be 30000 for the Reddit dataset and 300000 for the Reuters dataset.

b. NMF (Nonnegative Matrix Factorization):

One potential problem with the N-gram is the feature space is usually very large. It brings extra computation burden to the modeling and additional noises. One standard way is by dimension reduction technique, like nonnegative matrix factorization.
In nonnegative matrix factorization, we decompose the bigram model V into V = WH, and W will be used as the feature after dimension reduction. The W matrix is also often treated as the feature matrix with smaller noise. The results are , however, worse than those of original bigram features in our analysis. Because of this, we will not discuss more about the matrix factorization, and the results aren't presented in details. That means, in the report, only classical models combined with bigram results are presented.

c. Word2Vec:

The basic idea behind Word2Vec [25,26] model is: similar words have higher co-occurrence probability. For example, the word 'deep' and 'learning' are more likely to show up together in a deep learning paper. Thus, if one want to represent every word in vocabulary as vector, then similar words should have similar vector representation, therefore, higher mutual information. More rigorously, we want to represent each word in a low-dimensional, dense vector where similar words share similar vector representation.

We adopt idea in [25] where training corpus are W_1,..., W_N from a vocabulary set V. We want to minimize the average log probability. And associate with every word in vocabulary with an input vector of length d and output vector with length d. Also, window size defines how many previous words and following words we want to check. For each word, we now have a length d vector. Each news will then become a matrix with d columns and n rows. Where n is the number of words in the longest news. Shorter news will have padding with zeros. Now each document can be represented as a matrix with same dimension. The output of Word2Vec can be then used as input for CNN and RNN.

Embedding and Machine Learning Combinations Considered:

- **Classical Embedding Methods (N-gram) + Machine Learning Methods**

The classical embedding methods we are referring to are N-gram models. The features from bigram models discussed above will be feed to classical machine learning models, feed forward neural network (one hidden layer with Relu activation), LSTM (the specific structure is discussed in the later section), for stock movement classifications. The classical machine learning models

we considered here include: logistic regression with regularization, random forests and gradient boosting tree. Fine Tuning has been done for each model and results are as follows.

Reddit Dataset					
	Logistic Regression	Random Forest	Feedforward neural network	Boosting Tree	LSTM
Test Accuracy	56.3%	50.3%	60.3%	47.5%	55.0%
GPU Time	NA	NA	20 mins * 6 =2 hours (Tune with a parameter set size 6)	NA	20mins * 6 = 2 hours (Tune with a parameter size 6)
Tuning Parameters Selected	L_2 penalty strength = 1	Max_depth = 20	Hidden_unit = 5, Learning rate = 0.01	Max_depth = 5	Keep_prob = 0.5, Lstm_size= 128, Learning Rate = 0.01

It is surprising to see for one layer neural network, we can achieve 60.3% prediction accuracy. It is the best we've made in this dataset. One possible explanation is the "Occam's razor" that parsimonious models often perform surprisingly well.

Reuters Dataset						
	Logistic Regression	Random Forest	Feedforward neural network	Feedforward neural network with dropout (3 layer)	Boosting Tree	LSTM
Test Accuracy	53.3%	53.0%	53.5%	53.7%	47.5%	50.6%

GPU Time	NA	NA	2 hours	2 h *6 = 12h (Tune with a parameter set size 6)	NA	8 h * 6 =48h (Tune with a parameter size 6)
Tuning Parameters Selected	L_2 penalty strength = 1	Max_depth = 100	Hidden_unit = 20	Keep_prob = 0.9, Learning Rate = 0.01	Max_depth = 5	Keep_prob = 0.6, Lstm_size= 128, Learning Rate = 0.01

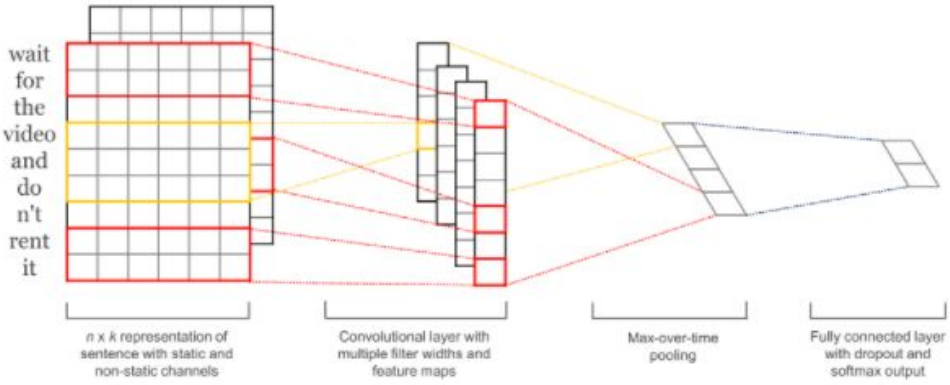
For the reuters data, the results can be a little worse. We believe the reason for that is: On Reuters, too many news are reported and many of them aren't important. That makes the signal-noise ratio in the news low and made the prediction harder.

- **Unsupervised Neural Net Embedding (Word2Vec) + Neural Network**

For the deep learning part of this project, we will consider convolutional neural network and recurrent neural network, each being applied to a different word embedding method.

a. CNN:

When Word2Vec is used, one way of building the predictive model is using CNN. This method, i.e., Word2Vec + CNN, is a re-implementation of the paper [12]. An illustrative visualization of the network we implemented is as below. Convolutional neural network has been shown to have the ability of capturing word dependence among sentences for a sentimental analysis on sentence level [12]. Since here we have news titles which can be also treated as sentences, convolutional neural network is a natural choice for classification task.



CNN Structure for Stock Movement Prediction

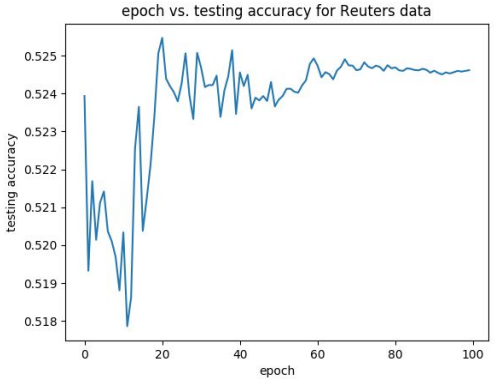
Structure Visualization credit to CNN for NLP tutorial Link

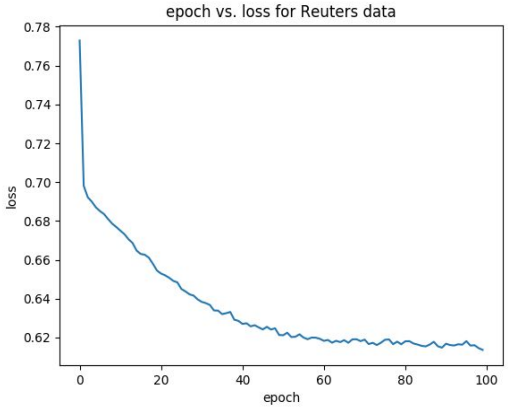
We first obtain an embedding matrix for each news title observation. Thus with word2vec embedding, convolutional neural networks is now a popular text classification architecture. Suppose we have n sentences(news titles in our case), we choose the embedding size of k. Then for each sentence, we will obtain a d by k matrix representation of the sentence with static and non-static channels where d is the maximum number of document length. Using 2 convolutional layers with stride size of 1, we are able to add filters to the feature map. After every 2 convolutional layers, we also add one layer of max pooling of kernel size s. Repeatedly doing this, we can construct very deep convolutional neural network with three fully connected layer at the end with dropout rate p. At last we will apply softmax function to get predicted classification with each observation.

The following table shows the cross validation result for choosing tuning parameters. We train each combination of the tuning parameters for 5 hours, and finally selected learning rate 0.001 with kernel size 5, dropout rate 0.2 using relu function.
At real coding stage, for the Reuters data, we follow two options to complete the convolutional neural network model. The first one is directly working with word embedding matrix representation of size d by k as input. This approach is straight-forward but limited by memory size of GPU. With slightly larger document length, embedding size and batch size, we will encounter a memory error problem in our program. Thus as mentioned before, we choose the embedding size as 50 compared to the maximize sentence length of 713. Alternatively, we may add an embedding layer particularly designed for word embeddings which in turn will largely decrease the size of our input. We tested out with adding one embedding layer and trained our convolutional neural network for about 50 hours (100 epochs for batch size 50 for 50000 training data) and obtain relatively similar result as we directly use the word2vec embedding matrix. Another challenge is that given a very deep convolutional neural network, we have numerous hyper-parameters for tuning. Unlike Yoon Kim mentioned in the paper, with very simple convolutional neural network and little hyper-parameter tuning, excellent results can be achieved[12]. In our case, due to the complexity of our data, many choices have to be made. The cross validation only considers a fraction of tuning parameters.

Training our CNN implementation without using embedding layer for another 50 hours, we obtain the following results. See the plots for an illustrative example for training accuracy, training loss, and test accuracy.

For the Reddit dataset, because the dataset size is rather small, our GPU training time is also smaller (40 mins for each parameter with a parameter set size 12).

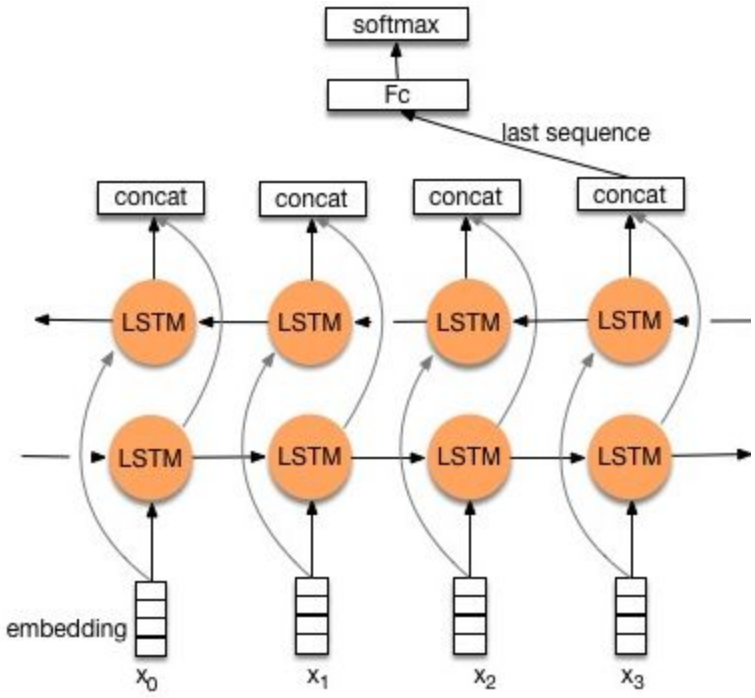




For the best model selected for Reddit data, the test accuracy is 57.50% and for the best model selected for Reuters data, the test accuracy is 56.36%. Using only stock news titles, we see that it is hard to accurately forecasting price change. The convolutional neural network has been verified to be a good method for sentiment analysis, however, since our news titles do not really convey information such as sentiments in reviews, it makes prediction harder.

b. RNN (LSTM):

The second method we implemented is recurrent neural network (LSTM) with a specific structure shown as in the visualization. Similar to CNN, we split the data into training set, validation set, and testing set.
The number of units in LSTM cells are chosen to be 64 and with 2 layers of LSTM for the Reddit data, and the test accuracy is 49.6% with GPU time 30 mins for each tuning parameter (we have a parameter set of size 6). The number of units in LSTM cells are chosen to be 64 and with 2 layers of LSTM for the Reddit data, and the test accuracy is 50.02% with GPU time 5 hours for each parameter (we have a parameter set size of 4). It turns out that varying combinations of parameters give similar results, so we simply choose learning rate 0.01 and dropout rate 0.5.



LSTM Structure for Stock Movement Prediction
Structure Visualization credit to Sentiment Analysis Tutorial Link

Illustrative Tuning Experiments

Here we present the comprehensive tuning results for Word2Vec + CNN for Reuters Data, to give a picture of our tuning process. The tuning results for other models are suppressed in our model, for the convenience and conciseness of our paper. Only the best selected models are presented.

Word2Vec + CNN for Reuters Data **GPU Time: 5 hours for each choice of tuning parameters**								
	Activation= relu				Activation = tanh			
Learning Rate		Kernel size = 1		Kernel size = 5		Kernel size = 1		Kernel size = 5	
	alpha = 0.1	Dropout Rate = 0.2	48.63%	Dropout Rate = 0.2	46.95%	Dropout Rate = 0.2	55.47%	Dropout Rate = 0.2	53.3%
		Dropout Rate = 0.1	50.36%	Dropout Rate = 0.1	51.32%	Dropout Rate = 0.1	53.16%	Dropout Rate = 0.1	51.32%
	alpha = 0.01	Dropout Rate = 0.2	56.2%	Dropout Rate = 0.2	52.2%	Dropout Rate = 0.2	53.88%	Dropout Rate = 0.2	53.76%
		Dropout Rate = 0.1	56.2%	Dropout Rate = 0.1	50.4%	Dropout Rate = 0.1	50.76%	Dropout Rate = 0.1	55.72%
	alpha = 0.001	Dropout Rate = 0.2	53.64%	Dropout Rate = 0.2	49%	Dropout Rate = 0.2	51.88%	Dropout Rate = 0.2	46.64%

		Dropout Rate = 0.1	53.8%	Dropout Rate = 0.1	56.36%	Dropout Rate = 0.1	49.32%	Dropout Rate = 0.1	54.72%

Results and Conclusions

To summarize, for all the methods (embedding methods + machine learning models) we implemented, the best selected models for the two datasets are as follows.

Reddit Data:

The Reddit Data is summarized as below. Because it is a small data, the overall GPU using time is also smaller (15 hours in total).

Reddit Dataset	
Methods	Test Accuracy
Bigram + Logistic Regression	56.3%
Bigram + Random Forest	50.3%
Bigram + Fully Connected NN (One layer)	60.3%
Bigram + Boosting Tree	47.5%
Bigram + LSTM	55.0%
Word2Vec + CNN [12]	57.5%
Word2Vec + LSTM	49.6%

Our best model is the bigram + one layer fully connected layer, and the test accuracy is quite good, 60.3%. We believe the reason is the dataset is small, a parsimonious model is enough to give us a good result. It also explains why complicated models don't perform well here.

Reuters Data:

The Reuters Data is summarized as below. The overall GPU using time is also smaller (182 hours in total).

Reuters Dataset	
Methods	Test Accuracy
Bigram + Logistic Regression	53.3%
Bigram + Random Forest	53.0%
Bigram + Fully Connected NN (One layer)	53.5%
Bigram + NN(With Dropout)	53.7%
Bigram + Boosting Tree	47.5%
Bigram + LSTM	50.6%
Word2Vec + CNN [12]	56.4%
Word2Vec + LSTM	50.0%

Our best model is Word2Vec + CNN, with a test accuracy 56.4%. The neural net structure [12] works. However, because the Reuters data is messier, the overall test accuracy is lower here compared with the Reddit data. We believe if we could figure out a way to get a better news dataset (a better signal-noise ratio dataset), or if we could find a systematic way to clean out some non-informative news, our results could get even better than what we had now.

Reference

[1] Dai, Yuqing, and Yuning Zhang. "Machine Learning in Stock Price Trend Forecasting." (2013).

[2] Khaidem, Luckyson, Snehanshu Saha, and Sudeepa Roy Dey. "Predicting the direction of stock market prices using random forest." ***arXiv preprint arXiv:1605.00003*** **(2016).**

[3] Dey, Shubharthi, et al. "Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting."

[4] Phan, Hung, Chien, Andrew, and Lim, Youngwhan. " A Framework for Stock Prediction."

[5] Agrawal, J. G., V. S. Chourasia, and A. K. Mittra. "State-of-the-art in stock prediction techniques." *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 2.4 (2013): 1360-1366

[6] Govindasamy, V., and P. Thambidurai. "Probabilistic fuzzy logic based stock price prediction." *International Journal of Computer Applications* 71.5 (2013).

[7] Längkvist, Martin, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling." *Pattern Recognition Letters* 42 (2014): 11-24.

[8] Bengio, Yoshua, Aaron C. Courville, and Pascal Vincent. "Unsupervised feature learning and deep learning: A review and new perspectives." *CoRR, abs/1206.5538* 1 (2012).

[9] Chakoumakos, Rowan, Stephen Trusheim, and Vikas Yendluri. "Automated Market Sentiment Analysis of Twitter for Options Trading."

[10] Lee, Heeyoung, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. "On the Importance of Text Analysis for Stock Price Prediction." In *LREC*, pp. 1170-1175. 2014.

[11] Sun, Andrew, Michael Lachanski, and Frank J. Fabozzi. "Trade the tweet: Social media text mining and sparse matrix factorization for stock market prediction." *International Review of Financial Analysis* 48 (2016): 272-281.

[12] Kim, Yoon. "Convolutional neural networks for sentence classification." In *In EMNLP*. 2014.

[13] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[14] Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[15] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315-323. 2011.

[16] Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645-6649. IEEE, 2013.

[17] Ding, Xiao, Yue Zhang, Ting Liu, and Junwen Duan. "Deep Learning for Event-Driven Stock Prediction." In Ijcai, pp. 2327-2333. 2015.

[18] Deng, Yue, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. "Deep direct reinforcement learning for financial signal representation and trading." ***IEEE transactions on neural networks and learning systems*** **28, no. 3 (2017): 653-664.**

[21] Fader, Anthony, Stephen Soderland, and Oren Etzioni. "Identifying relations for open information extraction." In ***Proceedings of the Conference on Empirical Methods in Natural Language Processing*****, pp. 1535-1545. Association for Computational Linguistics, 2011.**

[22] Zhang, Yue, and Stephen Clark. "Syntactic processing using the generalized perceptron and beam search." ***Computational linguistics*** **37, no. 1 (2011): 105-151.**

[23] Jiang, Zhengyao, Dixing Xu, and Jinjun Liang. "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem." ***arXiv preprint arXiv:1706.10059*** **(2017).**

[24] Lu, David W. "Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks." ***arXiv preprint arXiv:1707.07338*** **(2017).**

[25] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." ***arXiv preprint arXiv:1301.3781*** **(2013).**

[26] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." ***Proceedings of the 31st International Conference on Machine Learning (ICML-14)*****. 2014.**

[27]https://github.com/dennybritz/cnn-text-classification-tf
(http://www.nasdaq.com/screening/company-list.aspx)]