Using Natural Language Processing and Deep Learning Techniques on 8-K reports for stock-price movement prediction

Srinidhi Bhat
SBU ID: 112584858
Computer Science Department
sbrahmavar@cs.stonybrook.edu

Anurag Dutt
SBU ID: 113019209
Applied Mathematics
Statistics
anurag.dutt@stonybrook.edu

Apurva Lodha
SBU ID:112964966
Computer Science Department
aplodha@cs.stonybrook.edu

Abstract

We examine the significance of text analysis for stock value prediction. Specifically, we present a framework that gauges organizations' stock value changes (UP, DOWN, STAY) because of financial events detailed in 8-K records. Our outcomes demonstrate that utilizing text boosts prediction accuracy around 71 percent (relative) over a strong baseline that incorporates many financially-rooted features. This effect is generally temporarily (i.e., the following day after the financial events) yet endures for as long as five days.

1 Introduction

A huge amount of new data associated with organizations recorded on the stock exchange shows up continually, with prompt effect on costs of stock. Checking such data continuously is easier for big establishments yet out of reach of the individual financial specialist. We present a news monitoring and stock forecast framework, planned from the situation of the individual financial specialist without access to real-time trading tools. The commitments of our work are:

i)The 8-K financial reports, are documented by every publicly listed U.S. organization when major occasions happen, swaying the stock cost of the comparing organization for a few days.

ii)We implement a model that predicts the following days stock development by fusing applicable financial report, for example, ongoing stock value development and earning surprise, and textual data from these financial reports. We exhibit that the model which incorporates financial records performs essentially superior to a model with financial data alone.

The baseline paper has used various Machine Learning techniques such as the Random Forest and Support Vector Machine (SVM) to predict the short-term changes in stocks(UP, DOWN and STAY) [4]. On the other hand, we have implemented various Neural Network models such as the Multi-layer perceptron, Convolution Neural Network(CNN), Recurrent Neural Network (RNN) and a Unified CNN-RNN Framework. Moreover, Bidirectional LSTM has been incorporated for generating the appropriate the POS tagging after parsing [2][6].

Talking about our approaches, first of all, the baseline implementation of ours has used the Uni-gram model. The uni-gram model is highly skewed towards the frequency of words. It also ignores the rare words occurring within the sentences. Moreover, the uni-gram model considers each words as an independent entity which may lead to loss of essential information encoded over a span of a sentence. To resolve this issue we have used glove embedding. Glove enforce the word vectors to capture sub-linear relationships in the vector space. It adds more practical meaning into word vectors by considering the relationships between word pair and word pair rather than word and word. Lastly, Glove gives lower weight for highly frequent word pairs to prevent the meaningless stop words like “a”, “an”, ”the” to dominate in the training process.

Secondly, the classical Machine Learning techniques used in the baseline implementation needs to be provided with feature selection information. The 8-K filing reports encodes information about growth rates, profit margins, sentiments, etc. It would be better if the model selects it's own features and thus Neural Network comes for the rescue. The 8-K reports have data related to busi-

ness events, including bankruptcies, layoffs, the election of a director, a change in credit, trading suspension, Amendments to company Governance Policies, SEC investigations and internal reviews, Modifications to shareholder rights and many other things. We have shown how various Neural Networks have incorporated the necessary information to act and react when proved the appropriate text.

To test our implementation we have used the parameters such as loss, accuracy and accuracy-roc. By comparing the results with the baseline model of Random Forest we saw that almost all Neural Network such as Multi-layer perceptron, Convolutional Neural Network, Recurrent Neural Network and unified CNN-RNN framework have gained a better accuracy than the baseline. We have been using the cross entropy loss function. For further evidence please refer the Results section.

Talking about the data-set, we have used the 8 K filing reports provided by the Securities and Exchange Commission (SEC). The report has to be published by every publicly listed companies. It contains the information about major events such as change in CEO, new product launch, IPO opening etc which has a significant impact on the prices of stock (for short-term). We have considered 40 reports for 473 companies and passed the relevant information to our models.
Major outcomes:

1. Neural Network clearly shows a better accuracy than any Machine Learning techniques used before. One of the reasons could be the feature selection is done by the NN algorithm. The best accuracy received by our unified RNN-CNN model was –% as compared to –% gained by the Random Forest model.

2. The glove embedding used instead of unigram model helps to encapsulate the overall document information and makes it easier for relevant model well-equipped to perform.

3. The parser used enhances the input information by adding in the POS tags and thus making it easier for the underlying model to understand the data better.

2 Your Task

Our task is to predict the movement of the market solely based on analyst reports present in the 8K filings of the fortune 500 companies. In our training task we have ensure that no numerical values have been used in order to understand the current position of the company.

2.1 The Issues

As states in the Introduction section, the baseline paper uses the uni-gram model for feeding the data set as an input. This leads to a number of disadvantages such as ignorance of rare words, missing of context information present in the sentence, loss of meaning extraction over a number sentences. Moreover, judging the document with the frequency of words present isn't an optimal measure. We instead used Glove embedding which could make sure that the above problems are addressed. Glove embedding ignores the stop-words like 'a','an', 'the' etc. Moreover, it is able to represent similar sentences with the same sub-linear vector representation format. Also, the document consist of a lot of information and it becomes difficult for humans to handpick certain features by itself. Glove embedding can comprise the information in a neat manner making it easier for the underlying models to have better representation of data. For 473 companies 40 reports are extracted and each report is given as a glove embedding after pre-processing steps.

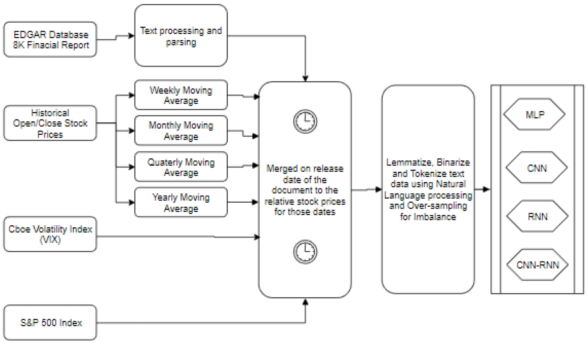
Secondly, various Neural Network incorporated over here such as Multi-layer perceptron, CNN, RNN and unified CNN-RNN are able to capture better generic information and feature if provided with appropriately formatted input data. In the below sections we talked about the advantages of filter in CNN, restoring of sequential information by RNN models. The statistics shows that the best generalization is done by the Multi-layer perceptron on the training data however it doesn't perform very well on testing as well as validation data. The unified CNN-RNN model is also implemented by us to see how could we benefit from the best of both worlds i.e. maintaining the sequential information and then applying appropriate filters on it.

Finally, after obtaining successful results we have tried to see what changes comes in efficacy

and efficiency of the model when provided with better information about the text i.e. POS tags. A bidirectional LSTM has been used to parse the sentences in both the direction and generate the appropriate tags. The relevant tag information is then coupled with the input vector and provided to the CNN-RNN model of ours.

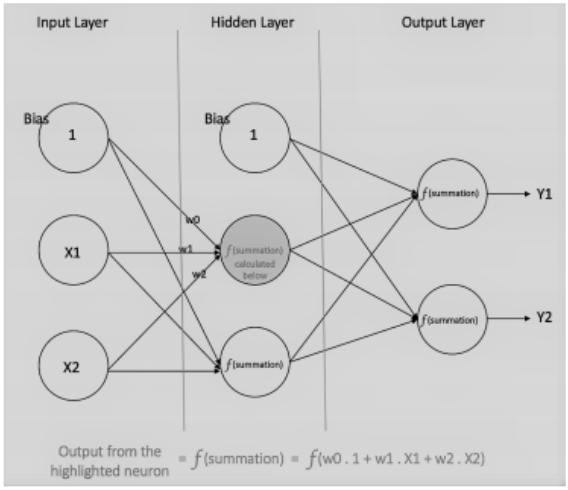
3 Approach and Models Used

Like any Machine learning project we have first of all collected the raw data and made it undergo the necessary pre-processing steps. The pre-processing includes converting the documents to glove embedding and encapsulating it with VIX(Volatility Index). The necessay analysis, parsing and reduction is done before giving it as an input vector to our models. The below figure better represnts the flow:



3.1 Model 1: Multilayer Perceptron

Multi-layer Perceptron (MLP) is a supervised learning technique that learns a capacity via preparing on a data-set, where m is the quantity of measurements for information and o is the quantity of measurements for yield. Given a input feature set with X=(x_1,x_2,...,x_n) and an objective y, the MLP can get familiar with a non-linear capacity approximator for either regression or classification. In between the input and the output layer, there can be at least one hidden layers. Also, all the connections between the inner and the hidden layer and the hidden and the output layer have a weight attached to them.



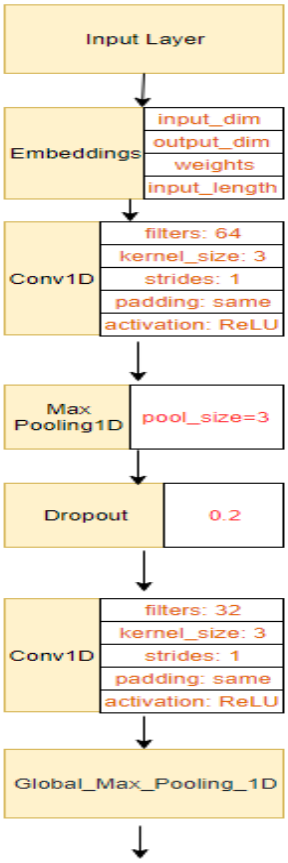
Input layer: No computation is performed at this layer. The main objective is to pass the glove-embedding provided to the input layer to the hidden layer.

Hidden layer: Various activation functions and computation are associated with this layer. We have use the sigmoid activation function in our dense layer. The altercation and alteration in weights and biased between the connection are heavily dominated by this layer. It finally passes the intermediate-embedding to the output layer.

Output layer: The values reflected in the output nodes are the output of the multi-layer perceptron.

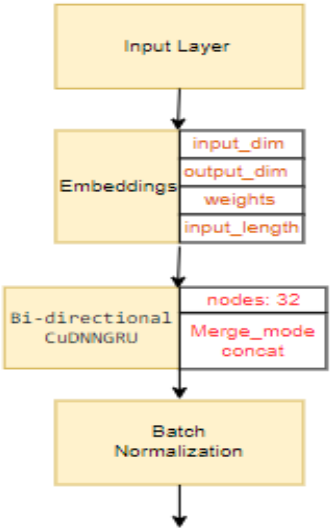
3.2 Model 2: Convolutional Neural Networks

Following the advancement of word embedding and its capacity to represent the words in a dispersed space, the need emerged for a compelling component work that gets more high level highlights from comprising words or n-grams. These abstract features are utilized for various NLP undertakings, for example, sentiment analysis, machine translation, Question-Answering(QA) etc. Convolutional Neural Networks (CNN) ended up being best suited technique for all the above problems due to it's filtering capacity. A CNN basically a neural-based approach which represents feature function that is applied to input words or the n-grams to extract the appropriate higher-level features.



The objective of their technique was to change words into a vector representation by means of a lookup table, which brought about word embedding approach that learns loads during the preparation of the system. So as to perform sentence modeling with an essential CNN, sentences are first tokenized into words which are then changed to a word embedding matrix (for example input embedding layer) of d dimension. At that point, convolution filter are applied to the input embedding layer which comprises of applying a filters of all conceivable window sizes to deliver what is known as a feature map. This is then trailed by a maximum pooling activity which applies a maximum activity on each channel to get a fixed length yield and diminish the dimensionality of the yield. Also, that methodology creates the final output representation. Nonetheless, CNNs neglect to demonstrate long distance dependencies, which is significant for our problem. By and large, CNNs are compelling in light of the fact that they can mine semantic hints in logical windows, yet they battle to protect sequential order and model long-distance contextual data.

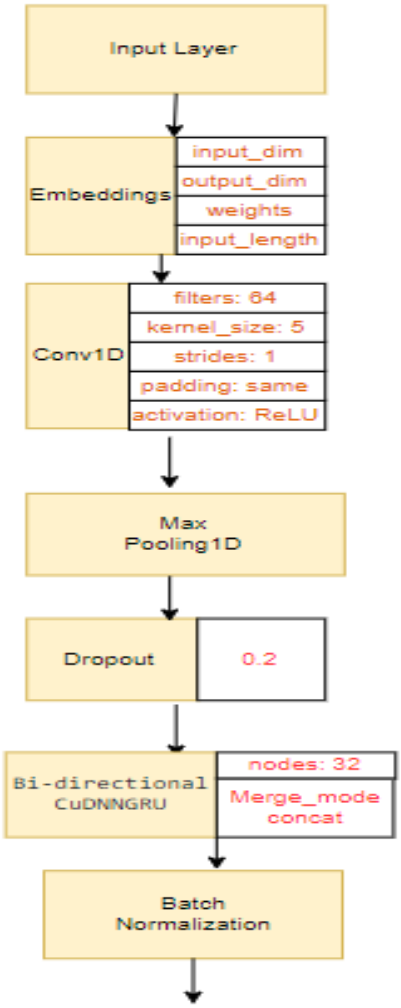
3.3 Model 3: Recurrent Neural Network



Recurrent Neural Networks are particular neural-based methodologies that has efficacy and efficiency at processing sequentially data. The expression ”recurrent” applies as they play out a similar undertaking over each occasion of the succession with the end goal that the yield is reliant on the past calculations and results. These successions are commonly represented to by a fixed-size vector of tokens which are fed successively (individually) to a repetitive unit. The fundamental quality of a RNN is its ability to retain the outcomes from the past calculations and use it for the present calculation. This makes RNN a decent contender for demonstrating setting conditions for contributions of arbitrary length. RNNs have been utilized to contemplate different NLP errands, for example, machine translation, image captioning, and language modeling, among others. Straightforward RNNs experience the ill effects of the infamous vanishing gradient issue, which makes it extremely difficult to learn and tune the parameters of the previous layers in the system.

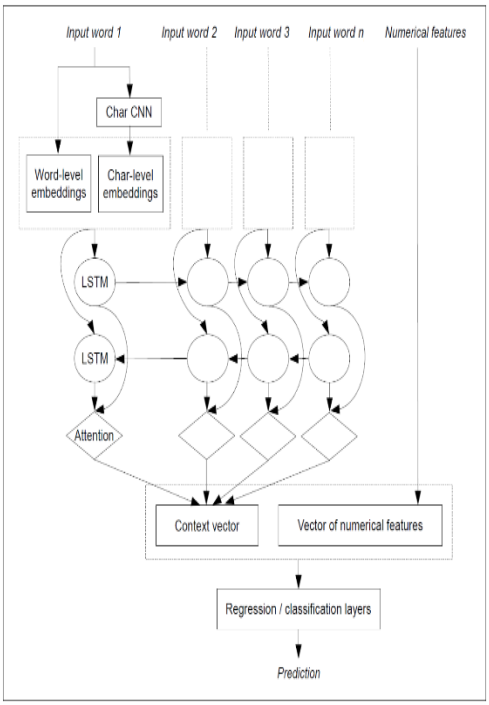
3.4 Model 4: Unified CNN-RNN model

CNN’s pooling activity on nearby words can hold the neighborhood highlights and their consecutive relations in a sentence. Plus, RNN can learn the long-term dependencies and the positional relation of features as well as the global features of the whole sentence.



3.5 Model 5: Using Dependency Parser to obtain POS tags using Bi-LSTM's

After passing the input word embedding to various Neural Network we did achieve an accuracy more than that of classical Machine Learning Techniques. However, we wanted to see what impact the POS tags would bring to the models. Hence we implemented a bidirectional Long Short Term Model which was able to parse the sentence and assign relevant tags to each word. The bidirectional nature of the model is able to see the sentence inside out and with the help of forget gate we are able to decide the span capacity for the Neural Network. The methodology produces an incremental increase in predicting power on various stock price prediciton by analysing the stock-related textual information better.



Due to scarcity for high computational resources we have passed the POS tags information to our best models only.

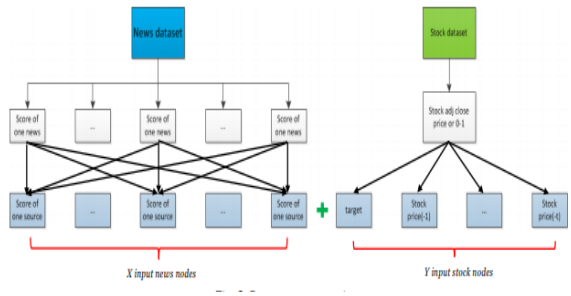
4 Evaluation

Evaluation was done based on model's test accuracy, loss and Area Under Curve characteristic. Figure -1 depicts a table which has all the values we have obtaine during our experiment.

4.1 Dataset Details

4.1.1 Reading data from the SEC EDGAR and AlphaVantage

We download for SP 500 companies as of December 2019 the list of 8-K document links from the SEC EDGAR website. Then for each link, we download the associated text reports from the SEC EDGAR website using BeautifulSoup. We then parse metadata such as the date and time of document release and the categories of disclosures made were extracted, while tables and charts were discarded. We also gather the historical stock prices (close, open, adjusted-close).

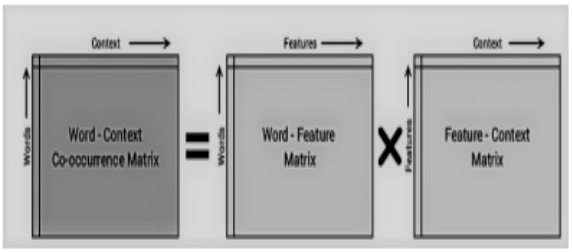


The Neural Network tries to grasp any important event details such as Material definitive agreements, Bankruptcies, Director is elected, Director departs, Asset movement: acquisition or sale, Result of operations and financial condition, Material Direct, Financial obligations (bonds, debentures), Triggering events that accelerate material obligations (defaults on a loan), Exit or disposal plans, Material impairments, Delisting or transfer exchange notices, Unregistered equity sales, Modifications to shareholder rights, Change in accountant, SEC investigations and internal reviews, Financial non-reliance notices, Changes in control of the company, Changes in executive management, Departure or appointment of, company officers, Amendments to company Governance Policies, Trading suspension, Change in credit, Change in company status and other such events.

We pre processed the text by removing all the HTML tags such as $< p >$ or $< td >$.We removed all tables with numeric values, which are typically accounting figures. Next we ignored the stopwords, white-spaces and punctuation. Finally, performing stemming lemmatization. An example of the pre-processed data is as below: "On November 15, 2011, the Board of Directors (the "Board") of Apple Inc. (the "Company") appointed Robert A. Iger to the Board. Mr. Iger will serve on the Audit and Finance Committee of the Board"

4.1.2 Glove

The GloVe model represents Global Vectors which is an unsupervised learning model which can be utilized to acquire dense word vectors like Word2Vec. The training is performed on an aggregated global word-word co-occurrence matrix, giving us a vector space with important sub-structures. The fundamental philosophy of the GloVe model is to initially make a colossal word context co-event grid comprising of (word, setting) matches with the end goal that each component in this framework speaks to how regularly a word happens with the specific circumstance (which can be an arrangement of words). The thought at that point is to apply matrix factorization to estimated this grid as shown in the figure below.



The Word-Context (WC) matrix, Word-Feature (WF) matrix Feature-Context (FC) matrix, are factorized as WC = WF x FC. We reconstruct WC from WF and FC by multiplying them. Thus to implement this, we initialize WF and FC with some random weights and attempt to multiply them to obtain WC' (an equivalence of WC) and calculate how close it is to our original WC. We perform this couple of times using Stochastic Gradient Descent (SGD) to minimize the error. Finally, the Word-Feature matrix (WF) outputs the word embedding for each word where F can be presented to a specific number of dimensions.

4.2 Evaluation Measures

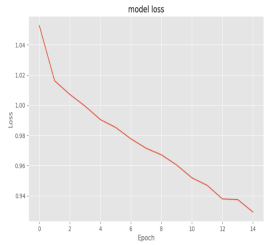
4.3 Baselines

For all our models (MLP,RNN,CNN and RNN-CNN) we kept some of the metrics common to consider a common ground for comparison.
Common Parameters in the experiment

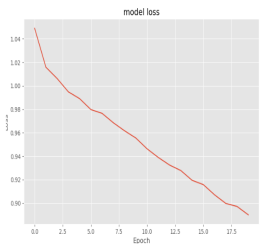
- Batch size - 32
- Loss Function - Cross Entrpopy Loss Function
- Optimisation Function - Stochastic Gradient Descent. (With Learning rate 0.001)
- Model Dropout = 0.2
- Epochs - All models underwent training for 15 epochs and 20 epochs.

Based on the results in the prelimenary stage, we then added the dependency parser model over the best model to refine our results. As per our experiments RNN-CNN was the best model based on the experiments, we applied the Dependency Parser only for the RNN-CNN model.

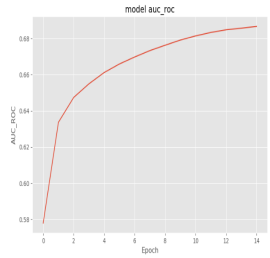
4.4 Results



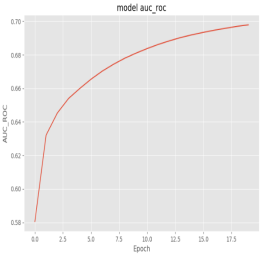
Model: CNN-RNN Epocs:15



Model: CNN-RNN Epocs:20



Model: CNN-RNN Epocs:15



Model: CNN-RNN Epocs:20

Table - 1 shows a detailed list of the results we obtained in our experiments. We see that our RNN-CNN model performs the best. This is expected because RNN is able to capture the sequential information in a sentence (of considerable length up to some extent). The RNN must have captured the sequence of events from the 8-K report filings. Applying a CNN architecture over that, the CNN must have captured the pattern based information in the entire corpus giving us a superior result overall.

We see that on Applying our Bi-LSTM based dependency parser over then RNN-CNN, there is improovement to some extent in the performance of the model further.

4.5 Analysis

1. Our model performs better than the base model proposed by Jurafsky et al [4] and it is expected because Deep Learning model can represent and capture patterns in complex data in a better way then classical machine learning model.

2. The involvement of Glove Embedding over Uni-grams also turned out to be a better idea. This is expected because Embeddings give a much better representation of words according to [7] [3]. and hence this could be a reason we are getting better results.

3. Another interesting observation is that during training MLP shows the highest train accuracy but during test it doesn't perform so well. This is reflected by the fact that MLP is not able to generalise the data that well as it the data representation is complex and the MLP is not able to generalise it so well.

4. Future scopes include using Fast Dependency Parser as a parser over the Bi-LSTM to see if there is a performance difference. We can also try to integrate Arc labels as part of our training as suggested by Nivre [5] and Chenn and Manning [1]

4.6 Limitation

- Currently our tasks only predict the motion of the stock based on previous data from the 8-K filing. Instead can we build a simulation over this as to how the stock will behave in future?

- We have not considered several other financial features which can effect the performance of a stock or the company on a long term basis, like Price/Earning ratio, Dept ratio, Bonus issuing, Dividend, etc. We can thing of integrating these features in the future models.

- We could have tried different type of parser like Fast Neural Parser, Constituency Parser and its different types to see what kind of effect it has on this task while integrating POS tags. Another idea which stems here is to use src labels as well, for this task.

- Another limitation is that this is doesn't take into consideration the various factors in world economy which can effect the stock price. Like recession, Oil price, war ,etc and somehoe we can think of integrating such features as well.

Model	Batch Size	Epochs	Loss	Accuracy	AUC score
RNN	32	15	2.30257	0.36181474	0.60088783
RNN	32	20	2.337947	0.23100189	0.54320165
CNN	32	15	0.908357	0.59281664	0.74156837
CNN	32	20	0.964679	0.5584121	0.69183949
MLP	32	15	1.599439	0.68620038	0.80427078
MLP	32	20	1.866408	0.68355388	0.80108476
RNN-CNN	32	15	0.928935	0.56824197	0.72933794
RNN-CNN	32	20	0.940398	0.56824197	0.71147662

Figure 1: Change in Metrics observed when POS tags where used in the task using Bi-LSTM

Model	Parser	Accuracy	AUC Score	Epochs
RNN-CNN	Bi-LSTM	0.67	0.82	15
RNN-CNN	Bi-LSTM	0.71	0.79	20

Figure 2: Change in Metrics observed when POS tags where used in the task using Bi-LSTM

4.7 Code

Repository Link to the Project
https://github.com/anuragdutt/spred_nlp

1. Most of the code has been written from scartch, including pre-processing, scraping ground truth from the internet using API's, Scraping the Edgar Database for 8-K filings and the entire model. However, we would like to acknowledge https://github.com/YuxianMeng/bilstm-dependency-parser for provding some hints regarding Bi-LSTM based dependency parser which we used in our module at a later stage.

2. The main file is the model file, it takes inline parameter of the model name, batch size and epochs. An example to run it is

   python model.py –model=rnn-cnn –batch-size=32 –epochs=20

3. Packages and their Versions - Python3, Keras 2.2.6, Tensorflow, 1.10, Numpy, Seaborn, Matplotlib, tqdm, Ipython.

5 Conclusions

In this work we saw how financial reports can be utilized to research the significance of text analysis for stock price prediction. Our corpus adjusts portrayals of money related occasions announced in 8-K records with the relating stock costs, which encourages the improvement of stock price estimation frameworks which joins textual and financial data. Utilizing the corpus, we demonstrate that using textual data has a significant impact especially for short-term periods.(the two days promptly following the occasion). For instance, our trials show that anticipating following day's value development is improved by –% (relative) if content is considered.

References

[1] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[2] Tomer Geva and Jacob Zahavi. Empirical evaluation of an automated intraday stock recommendation system incorporating both market data and textual news. *Decision support systems*, 57:212–223, 2014.

[3] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[4] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. On the importance of text analysis for stock price prediction. In *LREC*, pages 1170–1175, 2014.

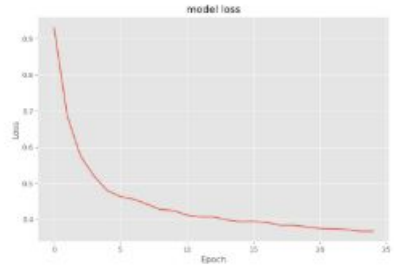
[5] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*, volume 6, pages 2216–2219, 2006.

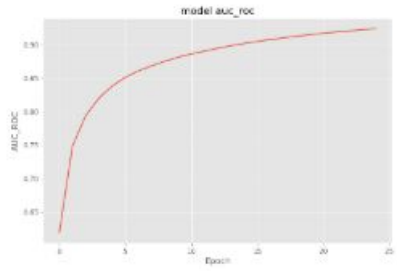
[6] Oleg. A text-based forecasting model for equity trading. *Decision support systems*, 57:212–223, 2014.

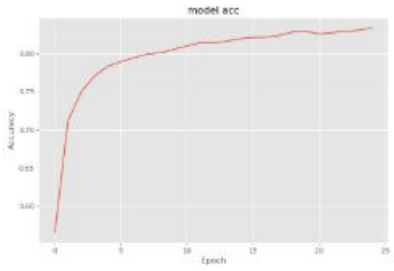
[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Appendix

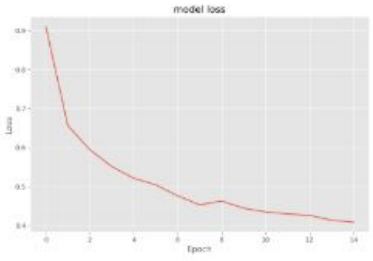
a) Multi-layer Perceptron epoch 25

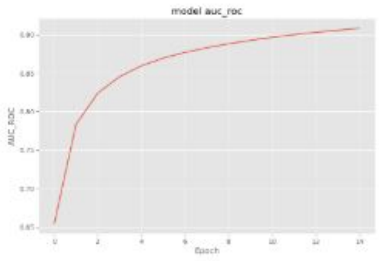


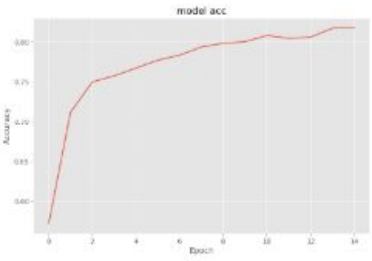




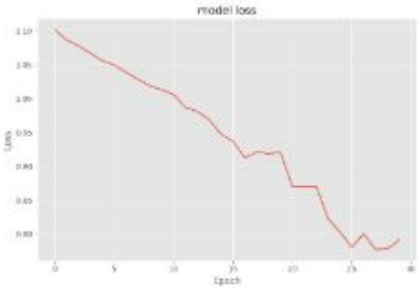
b) MLP epoch 15

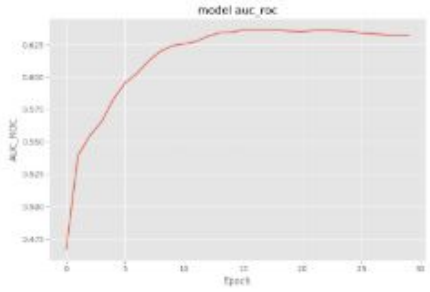


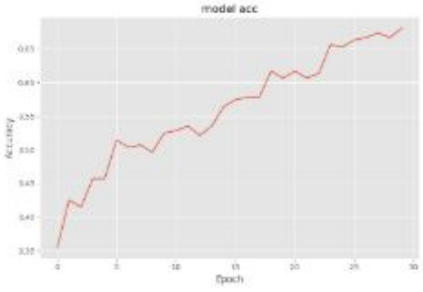




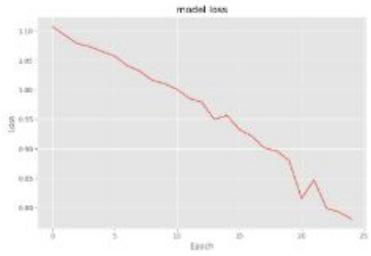
c) RNN epoch 30

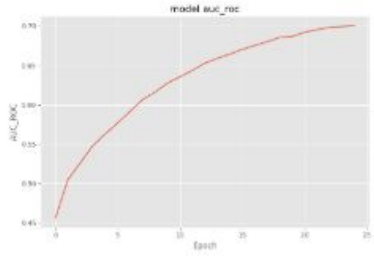


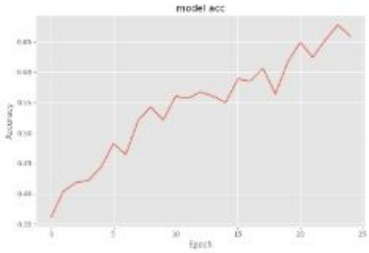




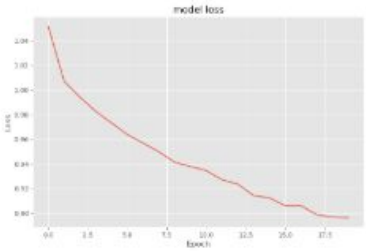
d) RNN epoch 25

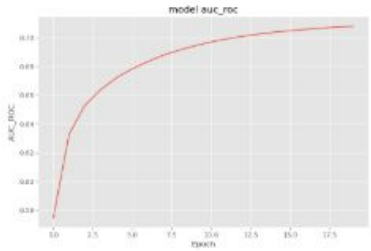


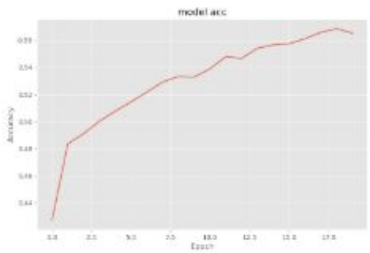




e) CNN epoch 20







f) CNN epoch 15