**Company stock price change forecasting with text mining in financial news**

Name: YANG Haobo Sid: 20656441

1. **Abstract**

Texting mining is the process of deriving useful information and data from text. It involves the studying of word frequency, pattern recognition and information extraction, etc. The main idea is to convert the text into numeric data for further analysis. Texting mining can be widely used in financial applications, such as identifying customer sentiments, predicting financial risk and stock price forecasting. This project aims to construct a model that forecasting stock price changes of individual company with text mining in the company financial news. Finally, the cnn-lstm model can achieve 57.99% accuracy.

1. **Dataset**

The news dataset contains 106521 news from Reuters in the period of 20-Oct-2008 to 20-Nov-2013. Since the goal is to predict the stock price change of each company, the news about each company would be separated from all news using scripts which detect company name in news headlines. There are 74 companies’ news that the model will use to fit. Top 20 companies that have most news and the news numbers are listed in the Figure1 in appendix.

For stock price, historical stock price data is downloaded through Yahoo finance. Some scripts are written to select the certain company news from all news dataset and record the date of the news. Then, it will automatically check the stock price change on that day from the historical stock price data and combine them as one sample. After necessary filtering, there is 13179 samples used to train and test.

1. **Data preprocessing**

In the project, the different experiments show that how to do the data cleaning is very important. The first step is clean the text data. For all news, split the text into words, lowercase the words and remove characters. All stopwords are also removed since they have no sentiment meanings. After that, we can count the frequency of each word, and check whether remaining words help to sentiment analysis, and remove the useless words. In addition, using news headlines instead of news report will make better prediction performance, so the project focus on the headlines of the news report.

For stock price, considering the impact of most news would not influence in a long term. And for each company stock, the change range, mean value and variance is very different. Hence, the target value is set to the percentage change of each stock. Since most of the percentage change is close to 0 and the variance is also small, to reduce the train loss better, zero-mean normalization is applied to target data. And the accuracy is calculated as (the number of samples that prediction value times actual value is larger than 0) / (the number of total samples).

As mentioned in abstract, to do text mining, we should convert the text into numbers, vectors or matrix that can be used as inputs of common analysis tools like neural networks, SVM and others. Here I tried two different methods. First is to convert the text into tf-idf matrix. Tf-idf is a measure that how import a word is. After feed all text to the TfidfVectorizer and set minimum word frequency, each sample will be converted to a vector containing the tf-idf values of important words, hence the whole sample set will be a matrix, with 18193 dimensions in this case. The second method is to represent each word using pre-trained vector representations for words ‘Global Vectors for Word Representation’, the main idea is that we can find the neighbors and linear substructures of words better. In this project, the pre-trained word vectors of common crawl with each word representing by a 300 dimensions vector. Hence, each sample would become a matrix while each sample is only a vector in tf-idf measure. Since each example has different length, samples are fixed have 30 words length, remaining first 30 words if exceed and filled with <PAN> symbol if have short length.

1. **Model Construction**

Python Keras library is used to construct the neural networks in this project. When using tf-idf measure, since each sample (headline) is represented by an 18193-dimension, there is no need to use a complex network structure. A simple two-layer full connected network is used to predict the results (See Figure 2 in appendix).

When using pre-trained work vectors as model inputs, a cnn-lstm model is constructed. The model details can be viewed in Figure 3 and Figure 4. The model contains many different layers. There are three important layers. First, three 1-d convolutional layers with different kernel sizes are used to learn the information between close words. The initial idea comes from paper ‘Convolutional Neural Networks for Sentence Classification’ by Yoon Kim, 2014. After each convolutional layer, a lstm network is used to learn long-distance dependencies via RNN for sentiment analysis of short texts. Finally, the outputs are connected to a full connected dense layer and one node output layer for calculating the results.

1. **Results**

The two-layer model using tf-idf measure can achieve 55.8% accuracy while the cnn-lstm model using pre-trained word vectors can achieve 57.99% accuracy. The different pre-processing methods and structure of dense layer changes the performance little, so we would focus more on the cnn-lstm model.

The test set contains 1245 samples. The performance of testing accuracy is shown in Figure 6. Noted that during the training, some very similar news headlines are founded because there may be more than one reports on one certain event on one day. This will not only increase the accuracy, but also make some predictions perfectly close to the true values. Hence, to make the result fairer and more reasonable, one of any two headlines that jaccard-similarity is higher than 0.8 will be removed from the samples to ensure that the training and testing sets will not contain the same or very similar samples.

The testing samples that are predicted correctly are showed in Figure 6. We can observe that even without similar samples, some predictions are quite close to the actual values. Looking into the data, this may be caused by similar topic on one day. Many headlines describe the same thing, which make them contain same key words, then the predictions could be unreasonable accurate. But this situation does not happen every day, the remaining accurate predictions indicates that there is some relationship between company news headlines and its stock price change, though not high as it is expected to achieve.

Different model structures are also tested. The results can be concluded in 1. Deeper and more complex network will not increase the accuracy but take more time to train. 2. In a reasonable range, larger kernel size and more filters can increase the performance. 3. Testing accuracy basically achieve highest after 20-30 epochs and more epochs may cause in overfitting. 4. Text data cleaning influence the results a lot, there is still a lot can do to improve the performance.

1. **Further Work**

I think the performance can higher than 60%, if improve the pre-processing methods and modeling. But there is only limited time to finish this project. I also have two main idea of further work of this project.

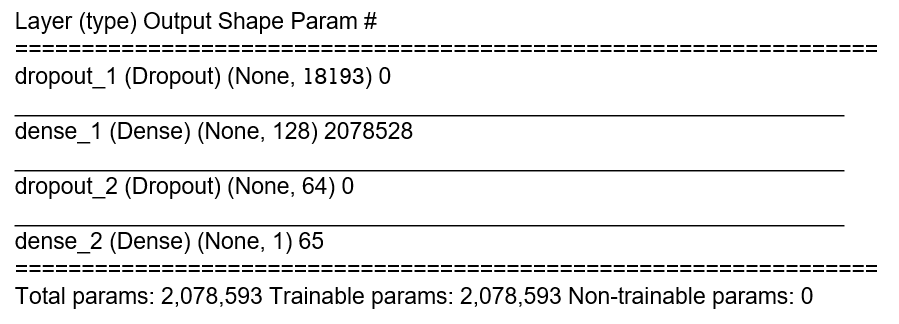
Firstly, what if the target value focus more on some general index like sp500, instead of company stocks. There is much more information about the general markets than induvial companies, the dataset would be larger. And the market index is more stable, unlike company stocks. If there is a huge change in sp500, then something important must have happened and it can be reflected on the financial news. There is another thing that for each company, the sentiment of the same words, topics and sentences may differ a lot due to the different types or markets of the companies. But if focus on one stock, the accuracy would be higher. Since there is not enough information for one company stock, the industrial index would be a perfect choose.

Secondly, I had also tried to use latent Dirichlet allocation to do topic modelling, but the performance is not good. But I still think it is a feasible way to get better results since I just start to know LDA. If we use LDA to get top 10 topics in all news headlines, the results would be in Figure 7. It shows some useful information. Topic 0 is most likely to contain banking companies. Topic 1 is likely to contains the news about company stores. Topic 7 is likely about the company profits and financial reports. Topic 9 is mainly about technology companies. I think it can solve the problem mentioned above to improve sentiment prediction using clustering.

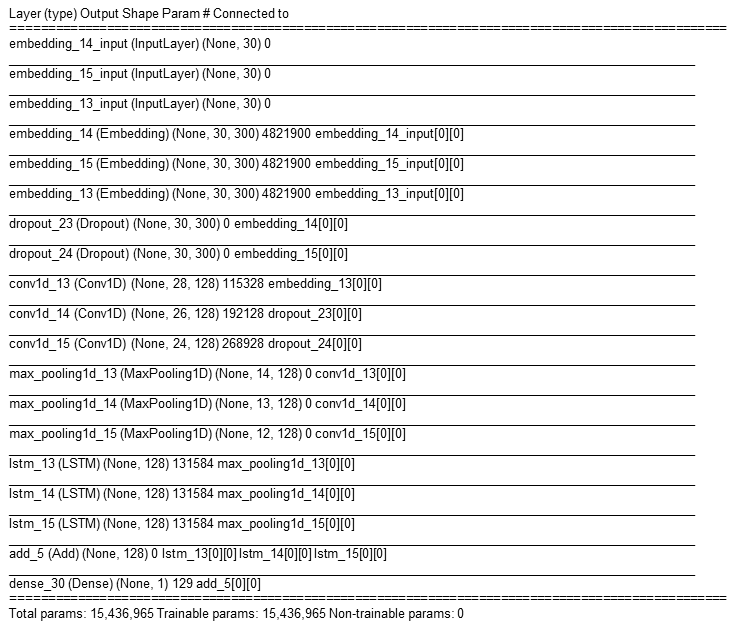
All files and codes can be viewed on: https://github.com/Dotafterfootball/Company-stock-price-change-forecasting-with-text-mining-in-financial-news

1. **Appendix**

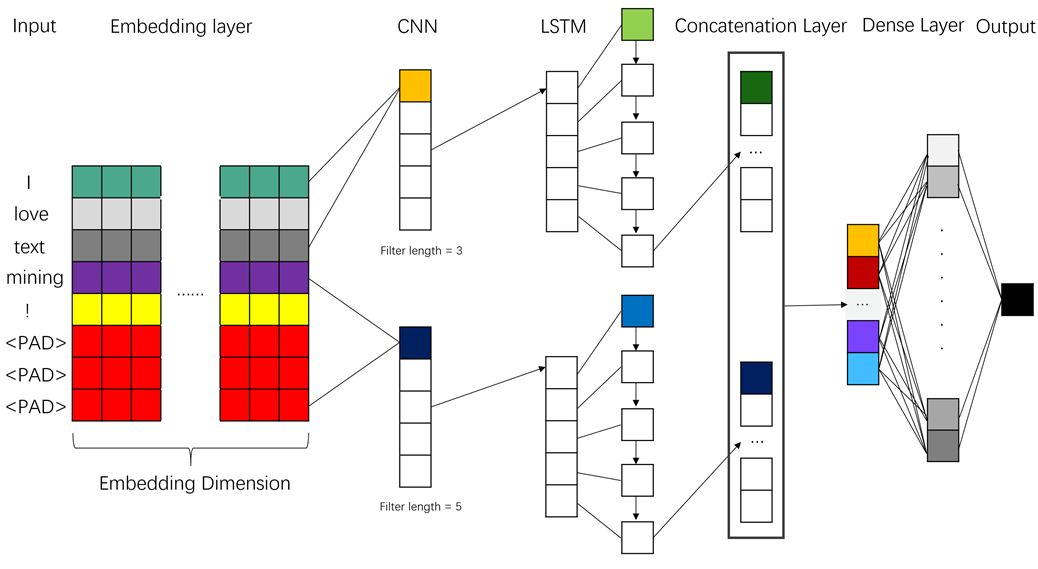
**Figure 1**

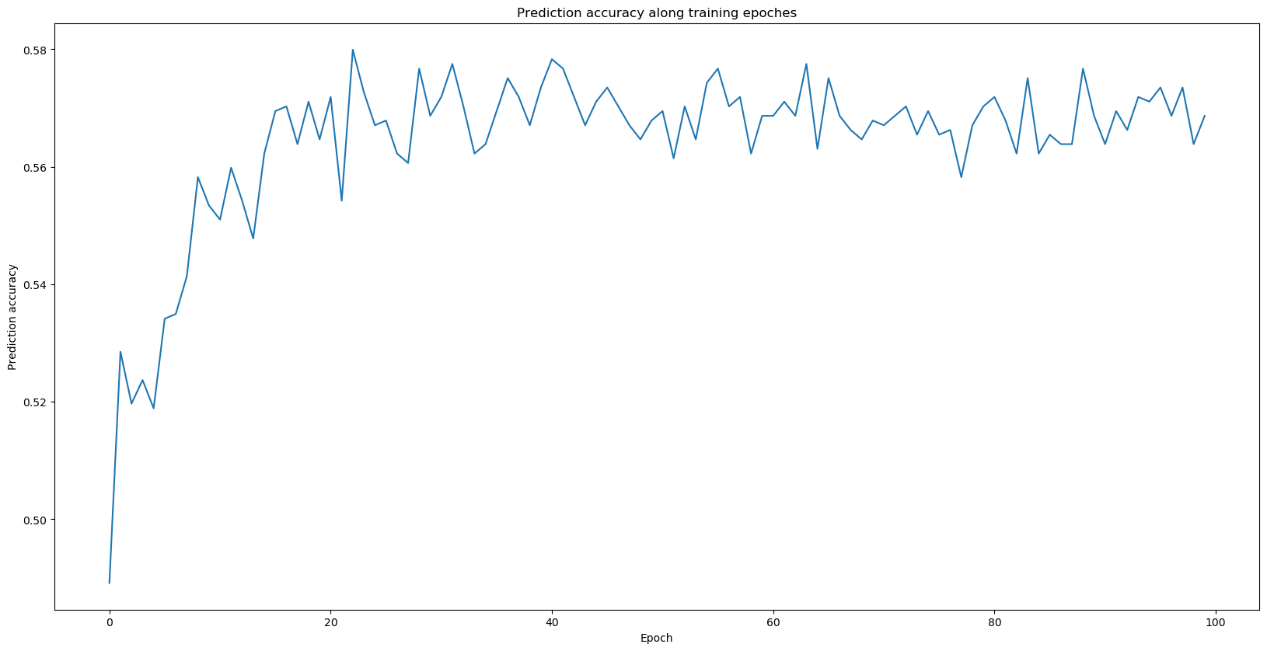


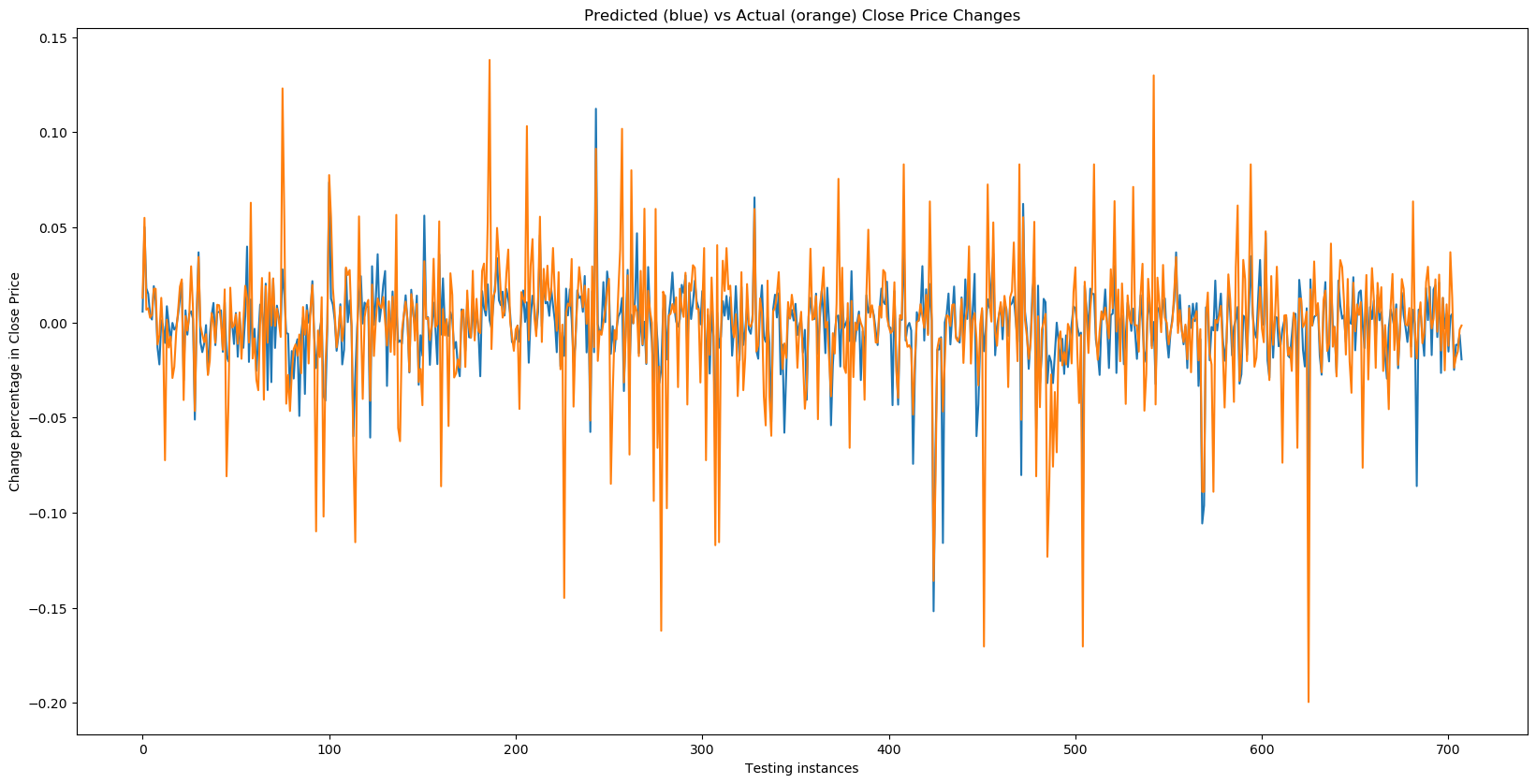
**Figure 2**

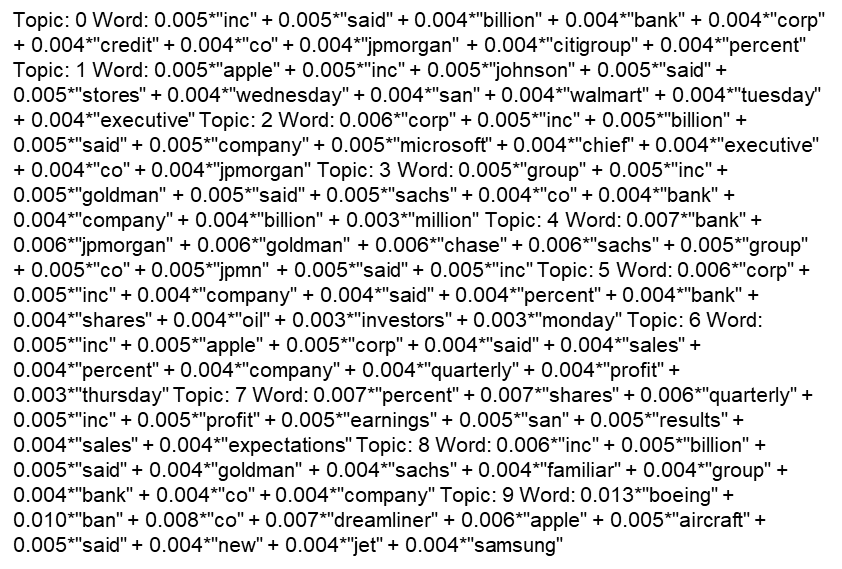


**Figure 3**

 **Figure 4**

 **Figure 5**

 **Figure 6**

 **Figure 7**

**Reference:**

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014.

*GloVe: Global Vectors for Word Representation*

Yoon Kim, 2014

*Convolutional Neural Networks for Sentence Classification*

*M. Day and C. Lee,*

*"Deep learning for financial sentiment analysis on finance news providers," 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), San Francisco, CA, 2016, pp. 1127-1134.*