# Stock Price Trend Analysis Using Multiple Novel Models

Mengqi Wu
New York University
Brooklyn, NY
mw4259@nyu.edu

Yuan Li
New York University
Brooklyn, NY
yl6606@nyu.edu

Dongzi Qu
New York University
Brooklyn, NY
dq394@nyu.edu

## Abstract

*Stock market is a complicated system and many factors will make impacts on its performance. Using multiple factors to predict the trend of stock market is a classical and difficult problem. Some studies have applied long short-term memory networks to analyze financial news. However, only a few number of studies have explored novel networks, like graph convolutional neural networks (GCN) and BERT model for financial news. In this work, we research on the importance of textual data from different sources to predict the stock market trend using GCN and BERT model. For BERT model, we create word embeddings using a fixed-sized vocabulary. Then, the model is fine-tuned on a model trained on cased English text. For GCN model, we build a text graph for the whole documents based on word co-occurrence and document word relations. Then GCN models automatically learn the word embeddings and train itself with the fixed labels for documents. We experimented the performances of our two models and other baseline models on stock market trend predicting problem and text classification problem. The two models shows better accuracy on both tasks than baseline models. However, the accuracy of stock market trend prediction is rather low. In the end, we analyze the reason of low accuracy and provide possible methods to improve.*

## 1. Introduction

Predicting stock market is a fundamental and important problem in natural language processing (NLP). There are several traditional methods for stock trend predicting, which focus on historical prices. Xie [25] used semantic frames to predict change in stock price from financial news. Kogan [11] provided a regression model based on financial reports. Starting from Alexnet [12] in 2012, Deep learning methods have been widely used in natural language processing task. In paper [3], Battaglia found convolutional neural networks (CNN) and recurrent neural networks (RNN) can get semantic features in local word sequences using the ability of learning about entities, relations, and rules for composing these sequences. Vargas [21] used CNN to predict Standard & Poor's 500 index using financial news titles. Akita [1] combined price time series and miscellaneous events reported in newspapers. However, the shortage of capturing global words information restricts traditional networks like CNN and RNN to achieve better accuracy in stock trend predicting problem.

Recently, Bidirectional Encoder Representations from Transformers (BERT) [4] and its related versions have received wide attention from scientists. They can capture semantic and syntactic information in global documents. Sun [20] investigated different fine-tuning methods of BERT on text classification problem.

Besides BERT model, graph convolutional networks (GCN) also outperformed in multiple deep learning problems. Kipf [10] experimented semi-supervised classification with GCN. Yao [27] found GCN performed better than other baseline models even without complex preprocessing methods.

In our work, we combine stock news and 10-K financial reports to investigate the possible influence of financial reports and try to improve the performance of our models. We experiment two novel models, GCN and BERT, and analyze their performance on stock price trending predicting problem. The GCN model is built based on Yao's Text GCN model [27]. The BERT model is based on the transformers [24] package from huggingface. The two models achieve better accuracy on this task, although the accuracy is rather low. Then we analyze these results, and find the reasons for the low accuracy is the low correlation between stock trends and news sentiment. For future research, we think it can improve the accuracy by using joint features to generate labels, or using semi-supervised models. To summarize, our main contributions are as follows:

- We combine stock news and 10-K financial reports to predict the trend of stock price, and receive better accuracy than using them separately.
- We experiment two novel deep learning models, GCN

and BERT, on stock price trending predicting task. Based on our knowledge, there are still few researches that use these two models. The result shows that GCN and BERT perform better than other baseline models.

- We analyze the reasons for the low accuracy of the stock related data compared with other general datasets, and provide possible methods to improve the accuracy.

## 2. Related Work

### 2.1. Traditional Methods for Stock Analysis

There are already some works on predicting stock price trend based on textual features and algorithms. For textual features, Nassirtoussi [15] reviewed for NLFF (Natural Language based financial forecasting) stated several techniques for text mining for market prediction. Lee [14] and Kogan [11] use the companies' financial events reports to predict stock price trends. In addition, some traditional machine learning models have been considered, like the decision tree used in [14] and regression model used in [11]. However, these methods require complex features and preprocessing before training.

### 2.2. Deep Learning for Stock Analysis

Deep learning methods can also be divided into two categorizes. One part of studies mainly focused on the performance of different models. Nelson [16] and Althelaya [2] used long short-term memory networks for this task, as LSTM can caputre time related features without the vanishing gradient problem. As Alexnet [12] achieved a huge improvement in image classification task, some studies planned to migrate CNN to NLP related tasks, like [21] and [5]. Vargas's results [21] showed that CNN can be better than RNN on catching semantic from texts and RNN more focuses on catching the context information. Kim [9] further proposed feature fusion LSTM-CNN model, which outperforms a single model.

Another part of studies try to dig out the relationship between features. Xie's study showed that the relevance in local sequences has an important influence on models' performance [25]. Others combine textural information and price time series then fed them into two classifiers and merged the vectors in the end [1]. For financial reports, Lee [14] used similar 8-K reports, and Kang [8] used the same 10-K reports in our work. Part of results of Lee's paper is shown in table 1. Our work is similar to these methods, we also try to use dual network models to jointly learn the information from stock news and 10-K reports.

| Model | Accuracy |
|---|---|
| **Random guess** | 0.33 |
| **Majority class** | 0.349 |
| **Unigram model** | 0.544 |
| **NMF 50** | 0.547 |
| **NMF 100** | 0.554 |
| **NMF 200** | 0.553 |
| **Ensemble** | 0.555 |

Table 1. Results for some proposed models in [14]. The Unigram model uses 2319 unigram features in addition to the 21 financial features. NMF is non-negative matrix factorization. They used three different dimensions for NMF. The Ensemble model combines the three NMF models using majority voting.

### 2.3. Graph Convolutional Network

Graph Convolutional Network (GCN) is a novel model recently that has received a lot of attention in the field of Deep Learning. Different from the traditional CNN, some studies showed that GCN can work on fixed structure graph, which makes it possible for us to use GCN to solve text classification task. Kipf and Welling [10] provided the graph convolutional network (GCN), which got state-of-art results on a number of NLP related graph datasets. Based on GCN, Yao[27] then presented the Text Graph Convolutional Network (Text GCN) for a whole document classification. It can jointly generate the embeddings of text and text label together. Meanwhile, text GCN can catch information in the whole document, and request less data than traditional deep learning model for text classification. That might be one potential advantage using GCN on stock market dataset since the length and number of news for each company are not stable. This approach is mainly constructed by Text GCN with some dataset preprocessing.

### 2.4. BERT Model

BERT model is firstly introduced by Google in [4]. In the task of machine reading comprehension SQuAD1.1 [17], BERT model showed amazing results that it surpassed humans in all two measures. Three major points of BERT model are: (1) It replaces a small number of words with *mask* or another random word with a reduced probability; (2) It adds a loss for the prediction of the next sentence. (3) In order to receive direction features, the input representation of BERT is combined by the corresponding token, segment, and position embeddings. Figure 1 shows input representation of BERT. Some variant version of BERT are proposed later. Yang [26] used autoregressive to create a new two-way encoding for BERT. Lan [13] implemented cross-layer parameter sharing and factorized embedding parameterization.

## 3. Dataset

In order to implement one model predicting the trend of the stock market benefited from NLP techniques, we need to combine the numerical (price) data with textual data. The

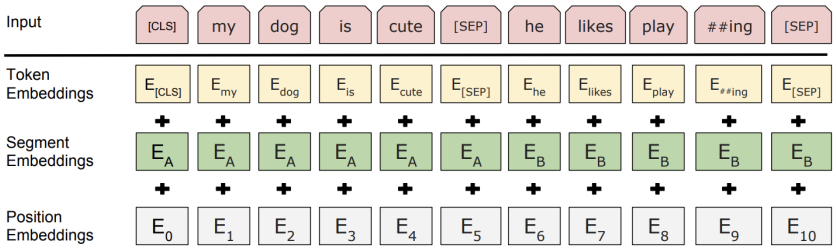


Figure 1. BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. Figure is from [4].

price data will be introduced in the third subsection. As for textual data, we mainly take use of news data and the objective evaluation from the third-party of one company to represent the potential sentiment. The news data will be introduced in the first subsection and second one for 10K data (objective evaluation).

### 3.1. News Data

Daily financial news data comes from Kaggle[1]. There are three columns in this dataset, which are news title, date and stock name. Specifically, more than six thousand companies are included and we can find the news headline for each company on each day from 2010 to 2020. We download this dataset and remove the hours+minute+second in date since we want to predict the influence of this news on the price change during the whole day.

### 3.2. 10-K Documents

We analyzed each company's financial report from Form 10-K dataset which is a document containing the disclose information on an ongoing basis for companies.

In order to make queries to retrieve the 10-K documents, we used the SecApi to get the data by setting a list of companies' index. The results retrieved by the SecApi are a series of URLs, so we needed to pull the data from these URLs and extracted the document sections as our final text data.

After get the raw dataset, we applied some preprocess operations on the dataset. First, we clean up the dataset by removing the html tags and all tables with numeric values. Second, made all the text lowercase. Besides, we lemmatized all the data and used stop words to exclude some common words from further analysis.

Besides, we want to perform sentiment analysis on the 10-K documents, so we will need to transform the textual values into tokenization forms. However, since the stock market is relating to financial topic, instead of using the default vocabulary from BERT, it's a better choice to use a financial related vocabulary to generate the embedding. Therefore, we will consider generate the embedding words by using a customer fixed vocabulary. Here, we used the Loughran-McDonald sentiment word list which is specifically built for financial textual analysis. There are six different sentiments in the word list. Since we only have three different labels (UP, DOWN, STAY), we select all the words refer to sentiment - positive, negative and uncertainty to form the vocabulary for further encoding.

The general Steps for processing 10-K Documents can be concluded in the following image:

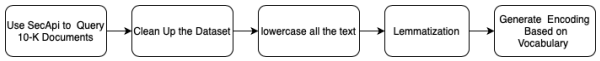


Figure 2. Steps for Processing 10-K Documents

### 3.3. Stock Price Data

Besides textual data, historical price data represents the change of price in a continuous period of time. We use stock market dataset[2] created by Boris Marjanovic. There are multiple columns inside this dataset, including date, open price, close price, volume etc. We need to generate the label for each day since we are dealing with the classification task.

Following Lee's work [14], we define three types of label (UP/STAY/DOWN) for both news data and 10K data. As for news data, we compute the daily label by subtracting the open price with close price and then divided by the open price which representing the price change on that day. 1.5% is our threshold to decide the real label. Moreover, for the 10K data, we compute the yearly label since each company only has one report in one year. However, the close price for the last day and the first day in one specific year will be used to calculate the price change and 10% is the new threshold.

[1]https://www.kaggle.com/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests

[2]https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

### 3.4. Dataset Combination

Until now, we have already built textual and numerical dataset. We need to combine them for the final training/test dataset. As for daily news with price, we regard stock+day as prime key to generate the larger dataset; while for the 10K text, we take use of stock+year as prime key to generate the other dataset. Based on the number of stocks and the length of news, we generate three groups of data:

- **RandomComp (RC):** we randomly select stocks from the whole company set and there are 50000 rows training data and 10000 for test data.
- **LimitedComp (LC):** we only focus on 200 companies selected in advance, this smaller dataset containing 26000 rows of training data and 7000 for test set.
- **LimitedCompLong (LCL):** based on the second dataset, we only select and merge those data which the number of news of one company is between 2 and 5. In this case, the length of news textual data is larger. There are 5000 rows for train and 1200 for test.

Since the first group of data contains more than 6000 stocks, which is hard to generate the corresponding 10K text for each company, we only generate 10K data for the second and the third one. The corresponding 10K data will be considered as an auxiliary sentiment representation.

## 4. Method

### 4.1. Graph Convolutional Network

A GCN has the similar structure as the normal convolutional networks. It contains multiple layers, where each layer is constructed based on the node graph data structure. Formally, assume of graph $G = (V, E)$, where $V$ ($|V| = n$) and $E$ denote the vertices set and edges set. Also, we introduce an adjacency matrix $A \in \mathbb{R}^{n\times n}$ and a diagonal matrix $D \in \mathbb{R}^{n\times n}$, where $D_{ii} = \sum_j A_{ij}$. Moreover, let $X \in \mathbb{R}^{n\times k}$ be a feature matrix (map) for all the vertices in $V$ of one layer, where $k$ is the size of feature space and each row $x_v \in \mathbb{R}^k$ is the feature vector for $v$. $X$ is the basic structure and each layer in GCN should be constructed by it. To get the new feature matrix for the $(i+1)^{th}$ layer ($L^{(i+1)}$), we can take use of the current result in $i^{th}$ layer ($L^{(i)}$) with some matrix multiplication and activation

$$L^{(i+1)} = \rho(\tilde{A}L^{(i)}W_i) \tag{1}$$

where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, which is same among all layers and $W_i$ is a weight matrix and its shape depends on the second entry of $L^{(i)}$ and the first entry of $L^{(i+1)}$. $\rho$ is a activation function, which we can use ReLu or tanh. Also, $L^{(0)} = X$, which is the initialized feature matrix. $W_i$ could be a rectangular matrix since the size of feature space might be different inside different layers.

Based on the structure of the basic GCN, we build a large and heterogeneous text graph which contains word nodes and document nodes. Therefore, the size of the node set is the number of documents (corpus size) plus the number of distinct words in a corpus. In this graph, edges exist between word to word or word to document. The edge between node and document can be calculated by Tf-idf of the word in that document. As for the word to word edge, the weight can be computed using point-wise mutual information (PMI), a popular measure for word associations. By use PMI, the graph will capture the correlation among some words with higher dependency. Also, for each node itself, we need to add a self-loop on it.

After building the text graph as in [10], assume we have $j$ layers, then the output $Z \in \mathbb{R}^{n\times F}$ of the last layer can be computed as

$$Z = \text{softmax}(\tilde{A}L^{(j-1)}W_{j-1}) \tag{2}$$

where $F$ is the number of categories and we use softmax function rather than ReLU is we want to make predictions among a group of labels. Let $j = 2$, then the whole model can be written as

$$Z = \text{softmax}(\tilde{A}\,\text{ReLU}(\tilde{A}XW_0)W_1) \tag{3}$$

The loss function is defined using cross entropy error over all labeled documents:

$$\mathcal{L} = -\sum_{i\in\mathbf{I_D}}\sum_{f=1}^{F} Y_{if}\ln Z_{if} \tag{4}$$

where $\mathbf{I_D}$ is the indices set for all documents and $F$ is the dimension of the output feature, which is equal to the number of classes. $Y$ is a label indicator matrix. Additionally, all the trainable parameters are coming from two huge matrix $W_0$ and $W_1$, which can be trained via gradient descent.

The architecture of our GCN training step is figure 3. The implementation of GCN is mainly based on GCN-TF2[3] repository.

### 4.2. BERT Model

The main part in BERT model is a transformer block from Vaswani's paper [22]. BERT model creates a multilayer bidirectional Transformer encoder based on the encoder part in [22]. Each layer is divided into two sub-layers: (1) multi-head self-attention layer, and (2) fully connected feed-forward network. A residual connection [6] is added to fix gradient disappearance problem, so there are 12 layers total. The output $y$ of each sub-layer is:

$$y = Norm(x + f_{layer}(x)) \tag{5}$$

[3] https://github.com/dragen1860/GCN-TF2

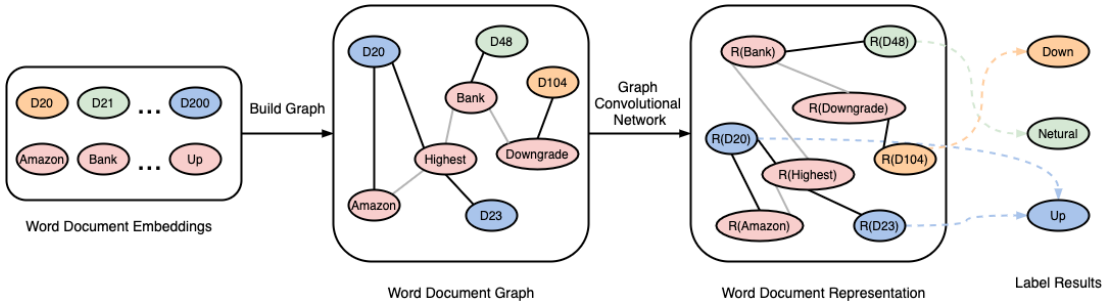


Figure 3. Steps for Training GCN. "D" means they are document nodes, which is a piece of title data for each. Others are word nodes. There are two edges: black edges are document-word edges and gray edges are word-word edges. $R(x)$ means the representation of $x$. Different colors are for different labels.

$Norm$ is normalization layer, and $f_{layer}$ is the corresponding function of each sub-layer. After normalization, the output size is $d_{out} = 768$.

The architecture of BERT model we used is figure 4.

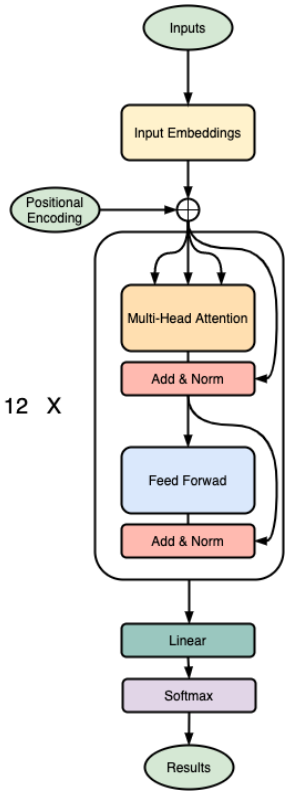


Figure 4. Architecture of BERT model. There are 12 blocks totally. Comparing the original transformer block [12], BERT model only uses the encoder part. A linear layer and softmax function is added after encoder part to get the classification results.

### 4.3. Data Combination

There are two parts of data we use – financial news and 10-K reports. We use the same architecture to train the two parts separately. After we get the outputs of each data, we merge them together with linear correction:

$$y = \frac{1}{\rho_r} y_r + \frac{1}{\rho_n} y_n \tag{6}$$

$y_r$ is the output of 10-K reports and $y_n$ is the output of financial news. $\rho_r$ and $\rho_n$ are adjustable coefficients.

## 5. Experiment

In this section, we evaluate TGCN and BERT model on two different experimental tasks. Specifically, in order to make a well prediction on stock market trend, we want to determine:

- Can TGCN model achieve satisfactory results on three different generated dataset, or higher accuracy than the baselines?
- Can BERT model achieve satisfactory results on same dataset? Furthermore, what if we add 10K dataset on it as an auxiliary sentiment representation?

**Baselines:** We compare TGCN and BERT models with multiple state-of-the-art text classification methods as follows:

- **TF-IDF+LR:** Bag-of-words model with term frequency-inverse document frequency weighting. Logistic Regression is used as the classifier.
- **fastText:** A simple and efficient text classification method [7], which treats the average of word/n-grams embeddings as document embeddings, then feeds document embeddings into a linear classifier. We evaluated it with and without bigrams.
- **SWEM:** Simple word embedding models [19], which employs simple pooling strategies operated over word embeddings.

- **LEAM:** Label-embedding attentive models [23], which embeds the words and labels in the same joint space for text classification. It utilizes label descriptions.

**Setting:** For baseline models, the parameters are following their original papers or the default settings. As for TGCN, the embedding size of the first convolutional layer is 200, and the learning rate is 0.01. In order to improve our model's robustness, we add dropout process in each convolutional layer with dropout rate 0.5. The $L_2$ loss weight is set as $5 \times 10^{-4}$. The maximum of training epochs is 200 with Adam. We randomly chose 10% data in training dataset as validation dataset for early stopping. The early stopping is set as 5 epochs that if there are no more loss decrease of validation dataset. The setting of parameters is mainly based on [10] and [27]. For BERT model, we use the BERT base architecture from [4]. The implementation is based on Transformers package [24]. It contains 12 transformer encoder layer. The hidden size is $d = 768$. The loss weight is $1 * 10^{-5}$. As the model is pre-trained, we fine-tune model with the strategy from [4] for 15 epochs.

**Performance:** Table 2 shows test accuracy of each model. There are no result when applying TGCN on 10K dataset because the the memory is not enough for the long vectors generated for 10K text. As shown, TGCN and BERT have better performance than other baselines on almost every dataset. Specifically, without the 10K dataset, the models have better results on smaller dataset (LC & LCL) than the larger dataset (RC). The reason might be the both TGCN and BERT have complex structure that there is no guarantee all the useful information inside the larger dataset will be captured. On the contrary, smaller datasets are more flexible. On the other hand, if we use 10K textual data as the auxiliary sentiment representation, the accuracy get decreased. Obviously, the additional information added from 10K makes prediction harder than before. Our inference for such a result is: 1) the 10K evaluation is too long for the model to capture the main semantic, so it's hard to focus on one specific topic; 2) 10K is more objective but there is no guarantee for daily news.

Although the overall accuracy is not so high (which will be analyzed in detail in the next section), TGCN and BERT still works well than other baselines. The main reasons are: 1) both text graph (TGCN) and bidirectional encoder + transformers (BERT) can capture the semantic information and compute the embeddings very well. 2) TGCN works a little bit better than BERT on larger dataset (RC) because graph neural network needs larger corpus to compute both global word-word relations and document-word relations. 3) BERT models has a slight advantage on smaller dataset (LCL) since it does not require too much training data to be fine-tuned.

# 6. Analysis

Our work shows that stock market trend prediction based on the text analysis of sentiment from the combination of different data sources can work well to some extent. However, the accuracy is rather low compared to our expectations. We think there are several possible reasons:

- The applicability of models. Whether GCN and BERT models are suitable for text classification problems?
- Accuracy of label. We set the labels for each textual data by calculating the stock trend in a certain time interval. Is this setting reasonable?
- Exclude factors other than news source. This is the most difficult and complicated part in stock market analysis. The external environment factors will have huge impacts on the stock market. For instance, the decline of the entire financial market will cause the stock price to decline even though the news are positive.

Detail discussion about the four reasons is given in the following part of this section.

## 6.1. Applicability of Models

Firstly, we want to measure whether our selected models are suitable for text classification problems. So we choose two general text datasets and record the test accuracy of GCN and BERT models on them. The there used models are:

- MR[4] is a movie review dataset for binary sentiment classification. This dataset has 5,331 positive and 5,331 negative reviews.
- R52 dataset[5](all-terms version), which is a subset of the Reuters 21578 dataset.
- R8 dataset[6](all-terms version), also from the Reuters 21578 dataset.

The results of test accuracy is in table 3. We can find that two models both achieve higher accuracy on general datasets than our stock news dataset. Which proves GCN and BERT models are appropriate for text classification problems.

However, the result of MR dataset is lower than other two datasets. We then analysis the summary statistics of these dataset. The statistics results of datasets are in table 4 (R8, R52 and MR results are from [27]). The summary shows that the number of words and total documents of MR

[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[5]https://www.cs.umb.edu/~smimarog/textmining/datasets/
[6]https://www.cs.umb.edu/~smimarog/textmining/datasets/

| Model | RC | LC | LC + 10-K | LCL | LCL + 10-K |
|---|---|---|---|---|---|
| **Tf-Idf + LR** | 0.5398 | 0.5714 | 0.5041 | 0.5693 | 0.5132 |
| **fastText** | 0.5110 | 0.5465 | 0.4857 | 0.5401 | 0.4813 |
| **fastText (bigrams)** | 0.5076 | 0.5579 | 0.5086 | 0.5384 | 0.4932 |
| **LEAM** | 0.5241 | 0.5653 | 0.5175 | 0.5503 | **0.5249** |
| **SWEM** | 0.5297 | 0.5714 | 0.5184 | 0.5628 | 0.5167 |
| **TGCN** | **0.5469** | **0.5782** | – | 0.5734 | – |
| **BERT** | 0.5427 | 0.5692 | **0.5283** | **0.5821** | 0.5201 |

Table 2. Test Accuracy on different dataset. They are mean results for 10 times. Because the vocabulary of 10-K reports is too large to make TGCN converge, there are no results of LC + 10-K and LCL + 10-K for TGCN.

| Model | R8 | R52 | MR |
|---|---|---|---|
| **BERT** | **0.9735** | 0.9248 | 0.7474 |
| **TGCN** | 0.9694 | **0.9361** | **0.7621** |

Table 3. Test Accuracy on general dataset. They are mean results for 10 times.

is larger than R8 and R52, which means MR is more complicated than other two datasets. This also explains the low accuracy of MR dataset. For our dataset, Total data and vocabulary volume are the largest in the four datasets, especially the number of words in vocabulary. It generates a huge vocabulary for BERT model and a huge words graph for GCN model, which may increase the complexity of the model, and constrain its convergence.

### 6.2. Accuracy of Label

We mark the sentiment of some text data manually, and select five pieces of text with different sentiment label and price trend label, shown in table 5.

Totally 100 random text data are marked, and there are 73 text that have the same two labels. This suggests that the correlation between the price trend label generated in our way and the text data is not high enough. The text data is a title of the corresponding company, which can not fully reflect the company's current price trend. For instance, the fifth text we show in table 5 mentioned Benzinga's upgrades and downgrades, but it doesn't say that upgrade or downgrade is more.

### 6.3. 10-K Report

Besides stock news, we want to analyze whether exclude factors can help the model make decisions. We applied BERT model on 10-K documents. The first step is to tokenize the input text data. In order to represent the data better, we used a specific financial vocabulary instead of the default vocabulary to do the tokenization. When the max length parameter for encoding was set to the same value 30, the accuracy comparison between these two vocabulary was shown in the figure 5. We can find in all experiments, the accuracy of using financial vocabulary is better than general vocabulary. This suggests that more professional preprocessing and features can be beneficial to model training to a certain extent.

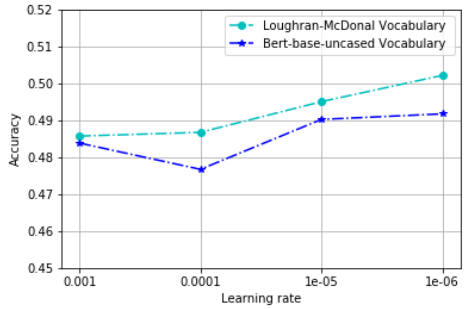


Figure 5. Accuracy for Different Vocabulary Comparison. The model used in this experiment is BERT. The dataset is 10-K only.

However, the results from table 2 show that adding 10-K features even makes the accuracy drop. So we want to modify hyperparameters to improve the performance. We experimented with the different combination for the learning rate and max length of a 10-k word embeddings. The relationship of different learning rate and max length for the word embeddings is shown in the figure 6. According to this result, we will pick learning rate as $1 * 10^{-5}$ and the max length for encoding as 30. The accuracy does improve after modifying hyperparameters, however, it's still lower than using LCL alone. We think the reason may be that 10-K report is only released once a year for each company, and the impact on the company's stock price is a few days to a few weeks after the release. But the data we used did not consider the time factor. Therefor, when predicting data with a long interval from the 10-K release, the inner features from 10-K are negligible. It becomes a disturbing factor instead.

## 7. Limitations

Following above analyze, we can find there are still several limitations and possible improvements in our system:

| Statistics | R8 | R52 | MR | RC | LC |
|---|---|---|---|---|---|
| **Docs** | 7,674 | 9,100 | 10,662 | 60,000 | 33,000 |
| **Training** | 5,485 | 6,532 | 7,108 | 50,000 | 26,000 |
| **Test** | 2,189 | 2,568 | 3,554 | 10,000 | 7,000 |
| **Words** | 7,688 | 8,892 | 18,764 | 26,202 | 25,109 |
| **Average Length** | 65.72 | 69.82 | 20.39 | 11.35 | 10.02 |

Table 4. Summary statistics of datasets.

| Text | Sentiment Label | Price Trend Label |
|---|---|---|
| IBM Shares Now Down 0.4% For Session | Negative | Stay |
| Five Overvalued Stocks With Earnings On The Way | Negative | Stay |
| Worst Performing Industries For October 19, 2015 | Negative | Up |
| JP Morgan Upgrades Diebold To Neutral | Positive | Stay |
| Benzinga's Top Upgrades, Downgrades For February 6, 2017 | Neutral | Down |

Table 5. Text with different sentiment label and price trend label. The sentiment label is marked by five people, and we choose the result of majority.

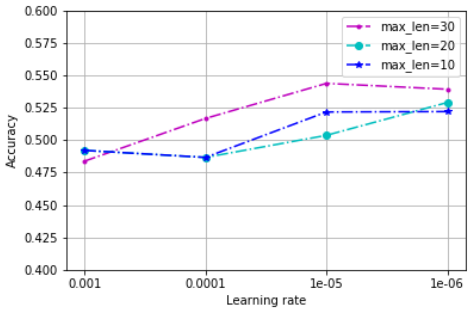


Figure 6. Accuracy for Different Parameter Combination Comparison. The model used in this experiment is BERT. The dataset is LCL + 10-K.

- Accuracy of label. The way we set label for each text data may cause the inconsistency between the real results with the label calculated by ourselves.
- Length for the text. LCL dataset receives a better performance than LC dataset, which means different length of data will have an influence on the prediction results. It is foreseeable that a longer length may improve the accuracy.
- Long time training. When training data is large and the model parameters are complicated, it will take a long time for training, especially BERT model, which has 110 million parameters in total. For instance, we may use a variant of BERT called DistilBERT model [18].

## 8. Conclusion

In this work, we tried to implement a forecast system for stock market based on the textual data combining from different sources. We built a corpus which aligns the financial report and news with the corresponding stock prices to train our model. Through our work, it shows that the textual information has great impact on the stock market trend predicting. Besides, we also proved that our system's predict results are better than the baseline. As has been mentioned above, our work is based on certain assumptions and there are still some limitations in our system, meaning that this research cannot be considered as a complete trading strategy. However, we still leveraged the relationship between text data and stock market. Therefore, the future tasks for our work are to find better way to improve the accuracy of our method and also optimize the training process to shorten the training time.

## 9. Acknowledgments

Thanks Hejie Cui[7] and Jiaxin Hu[8] for helping us mark the sentiment labels of text data. Thanks Jiaxin Hu for providing ideas on BERT model hyperparameter modification.

## References

[1] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, 2016.

[2] K. A. Althelaya, E. M. El-Alfy, and S. Mohammed. Evaluation of bidirectional lstm for short-and long-term stock market prediction. In *2018 9th International Conference on Information and Communication Systems (ICICS)*, pages 151–156, 2018.

[7] PhD student from department of computer science, Emory University
[8] Master student from school of software engineering, Tongji University

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[8] T. Kang, D.-H. Park, and I. Han. Beyond the numbers: The effect of 10-k tone on firms' performance predictions using text analytics. *Telematics and Informatics*, 35(2):370–381, 2018.

[9] T. Kim and H. Y. Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320, 2019.

[10] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[13] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[14] H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky. On the importance of text analysis for stock price prediction. In *LREC*, volume 2014, pages 1170–1175, 2014.

[15] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo. Text mining for market prediction: A systematic review. *Expert Systems with Applications*, 41(16):7653–7670, 2014.

[16] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426, 2017.

[17] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[18] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[19] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*, 2018.

[20] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

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

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[23] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics.

[25] B. Xie, R. Passonneau, L. Wu, and G. G. Creamer. Semantic frames to predict stock price movement. In *Proceedings of the 51st annual meeting of the association for computational linguistics*, pages 873–883, 2013.

[26] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[27] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.