April 2021

Combination of Time Series Analysis and Sentiment Analysis for Stock Market Forecasting

Hsiao-Chuan Chou
University of South Florida

Combination of Time Series Analysis and Sentiment Analysis for Stock Market Forecasting

by

Hsiao-Chuan Chou

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Art in Statistics
Department of Mathematics and Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Kandethody M. Ramachandran, Ph.D.
Manish Agrawal, Ph.D.
Lu Lu, Ph.D.

Date of Approval:
April 5, 2021

Keywords: Long short-term memory, word embedding, neural networks.

Dedication

To my mother Li-Ping and family.

Acknowledgements

Foremost, I sincerely appreciate my thesis advisor Dr. Kandethody M. Ramachandran for continuous advises and support for the entire journey. His guidance and experiences help me not just with master thesis but also with learning process in the future. Through the process of writing thesis, I have confidence in learning new theories and code techniques. My thanks also go to committee members Dr. Manish Agrawal and Dr. Lu Lu for amount of their time to accommodate my defense. All the suggestions from my defense are precious. I also owe thanks to faculties of department of Mathematics and Statistics for time-saving paperwork.

I would never have gone through graduate school without the support and encouragement from my mother and family. My family deserves credit for making my work reachable.

Table of Contents

List of Figures

List of Tables

Abstract

The goal of this research is to build a model to predict trend of financial asset price using sentiment from news headlines and financial indicators of the asset. Objective of the model is to conclude good results but also to minimize the difference between predicted values and actual values. Unlike previous approaches where the sentiments are usually calculated into score, we focus on combination of word embedding of news and financial indicators due to nonavailability of sentiment lexicon.

One idea is that the sentiment of news headline should have impact on financial asset values. In other words, it would be crucial how we extract information from news headlines. Another idea is that price data through time series analysis is also useful to predict trend of financial asset prices. Hence, improvement should be made with combination of sentiment analysis of news headlines and time series analysis.

Compared to time series models and word embedding models, our combined model shows smaller or similar small MAPE, MAE, and RMSE with time series models, and reduces lag in graphs.

Chapter One:

Introduction

Financial asset forecasting is a charming and challenging problem at all times. There are bunch of factors affecting trend direction, and they could be reasonable or unreasonable. News and historical price are commonly considered as two of these important factors affecting trend of financial asset. Hence, the idea to this thesis is to apply proper analytical approaches to them to reach better predictions of financial asset prices.

Sentiment is a personally subjective attitude toward a subject, which means that sentiment is an opinion to depict emotion. Sentiment analysis known as text mining is capable to extract subjectivity from reviews, customer feedbacks, or texts. Sentiment analysis is being applied in many fields, especial business field, to gather valuable information. Adjustment of business strategy, for instance, can be made from sentiments of customers' reviews, and that's why sentiment analysis is quite popular.

Sentiment analysis can be performed on not only reviews of customers but also tweets, news and any personal text, and many researches related to stock market forecasting with sentiments have been done. With combination of moving average of financial indicator and sentence level sentiment score of Really Simple Syndication (RSS) news, trend of stock price for specific company is performed [1].

Since historical price of financial asset is time series data, many approaches for time series are applied to predict stock price. The autoregressive integrated moving average (ARIMA) has been explored in literature for stock price prediction [90]. Artificial neural networks (ANN) as widely used forecasting model are also applied to solve non-linear problem such as stock price prediction [3]. Hybrid strategy, combination of ARIMA and the support vector machine (SVM) shows the great improvement compared with single ARIMA and single SVM [2]. Besides, recurrent neural networks (RNNs) as one of the eyes-catching and popular neural networks family constructs a model for trend of financial assets [4]. Volatility of financial asset market leads a difficult situation to predict trend of market.   Combination of several machine learning approaches is being used in prediction [13].

The brief outline of this thesis is as follows. An overview of dataset in Chapter is given in two and procedure of data preprocessing would be introduced in Chapter three. Methods and procedure to build model are discussed in Chapter four. Results and evaluation would be presented in Chapter five, and we finally would summarize and conclude this thesis in Chapter six.

Chapter Two:

Literature Review

Stock price prediction has been a difficult and challenging task due to it volatility. Efficient market hypothesis as an economic hypothesis played a critical role for stock prediction. The efficient market hypothesis (EMH) is a hypothesis that asset prices reflect all available information [37]. EMH can be categorized into “weak-form”, “semi-strong-form”, and “strong-form”. Weak-form efficiency is that previous asset prices cannot be applied to predict present asset price. Semi-strong-form efficiency is that present asset price has already reflected all public information. Strong-form efficiency is that prediction for financial asset cannot base on neither public nor private information. According to EMH, stock prediction is a time series problem, also called random walk. However, it was also shown that textual information or investors’ sentiment is correlated to stock market direction against efficient market hypothesis [38, 48, 63].

Most of investment decisions are made based on fundamental analysis (financial statements), technical analysis (financial indicators), and textual information (news, opinions, tweets). The focus of fundamental analysis is on value investing which is estimated by fundamental attributes such as earning per share [49]. The most popular method is technical analysis which states that prediction based on historical prices and volume, but it might be less profitable after mid-1980s [50].

Over the past few decades, many researchers have used machine learning approaches to analyze financial information including financial time series data and textual data [42]. For time series data, there are various approaches such as artificial neural network (ANN), autoregressive integrated moving average (ARIMA), k nearest neighbor (KNN), recurrent neural network (RNN), support vector regression (SVR), and so on. ANN as one member of the famous neural network family has been used for time series forecasting for over 20 years for its ability to tackle nonlinear patterns, but it is sensitive to parameter selection [51,52]. ARIMA which is introduced in 1970 has also been being used to analyze financial data due to its high ability for linear time series [3]. Besides, ARIMA is still an option as one component of hybrid approaches because it does not work well for nonlinear time series. It usually combines machine learning approaches with better ability to handle nonlinear time series such as ANN or XGBoost [53, 54]. K nearest neighbor (KNN) is introduced as non-parametric classification approach in 1951, and expanded as regression approach in 1992 [55, 56]. Some researchers combine SVM as classification approach and KNN as regression [57]. Support vector machine (SVM) is approach for two-group classification problem, and it works well with strong mathematical support [58], which means that SVM can provide global optimal solution. With the mathematical support, it is often combined with other approaches such as ARIMA or KNN [57, 61]. With similar principle and the same researches, support vector regression (SVR) which uses SVM to do regression task is proposed [59]. It is powerful for financial time series [40, 41, 60, 62] despite sensitivity to its hyperparameter. Recurrent neural networks is well-suited to solve both classification and regression problems with series data. Long short-term memory (LSTM) is the most fashionable member of it because LSTM is capable to exploit the patterns in data and remember information for long time[43]. With memory property, it is wildly used for series data such as financial time series data [44, 45], and text mining task because text is considered as a series of words. A combined approach is proposed with combination of convolutional neural network (CNN) and LSTM for gold price time-series analysis [65]. CNN can be applied to reduce dimension, and it also works for series data due to its sliding processing.

Many researchers focus on the relationship between stock market and investors' sentiment based on textual data such as news, tweets, and opinions. Sentiment analysis is necessary to the task, which is capable to extract investors' subjectivity to textual data. Empirical evidence shows strong correlation between investors' sentiment derived from microblogging platform and stock return. Moreover, investors' sentiment plays a less important role while companies with larger capitalization [64]. Combined approach of support vector machine and bag of words is applied with textual information to predict price and direction [39]. Sentiment analysis approaches is rarely used to predict stock price alone because it extracts sentiment polarity from given text. Nevertheless, these approaches are applied to sentiment classification problems by labeling given texts based on stock price change. Recently, an increasing number of approaches based on two or more machine learning approaches is introduced into stock price prediction and direction. Neural networks with great performance to time series data are often combined with other machine learning approaches. Price prediction of multiple companies based on neural network with concatenation of news and historical prices data of multiple companies is proposed [46]. Price prediction models based on dictionary-based text mining, time series analysis, and word embedding are performed with different combination of independent variables including news, polarity to news, historical prices [47]. SVR is also combined with sentiment analysis of tweets that work well on price forecasting tasks [66]. Other researchers use textual data and historical prices to predict price, approaches are performed for price movement prediction by mixing TF-IDF language model and three classifiers which are Naïve Bayes, KNN and SVM [67].

Chapter Three:

Data Description

Stock price as known as share price is the price to trade a share of a stock in the market. The term "market" actually means both the primary market and the secondary market. The primary market is the place where companies sell new stock to the public for the first time such as initial public offering (IPO), and stock price would not be affected by any factor each IPO. The secondary market is the place that investors trade stocks among themselves such as New York Stock Exchange (NYSE), and stock price is fluctuant and affected by many factors which could be economic, political, or related to investors' sentiment. All the prices in this study are share prices in secondary market which is directly affected by investors' sentiment.

Common sources for sentiment analysis in finance field are tweets, opinions, or news, etc. There are huge amounts of comprehensive tweets related to stock market post by any users who can be individual investors, financial analyst, or companies etc. Thus, tweets are less reliable, and spam detection algorithm are usually applied with sentiment analysis. Investing.com is one of the top three global financial websites with more than 46 million monthly users, and over

400 million sessions [17]. In other words, news or opinions on Investing.com are more reliable and influential to investors' sentiment. In this thesis, we choose to obtain news and opinion headlines of six stocks from different sectors, which are Bank of America Corporation (BAC), The Boeing Company (BA), Exxon Mobil Corporation (XOM), Uber Technologies, Inc. (UBER), Johnson & Johnson (JNJ), and Apple Inc. (AAPL) on Investing.com, and data overview is shown in the figure 3.1.

id	ticker	title	category	release_date	provider	url	article_id
308080	BA	ANALYSIS Industry bailouts risk unfair trade c...	news	2008-10-29	Reuters	https://www.investing.com/news/forex-news/anal...	1108.0
308081	BA	ANALYSIS US defense market seen facing rising ...	news	2008-11-03	Reuters	https://www.investing.com/news/forex-news/anal...	2897.0
308082	BA	Critics urge ouster of GM CEO but allies rally	news	2008-12-09	Reuters	https://www.investing.com/news/forex-news/crit...	12545.0

Figure 3.1. General View of News and Opinions Data.

The frequent words in news titles are also visualized for the six stocks respectively so we can observe frequent words which might bring less information for stock price forecasting. Word is more frequent when it is larger in figure:

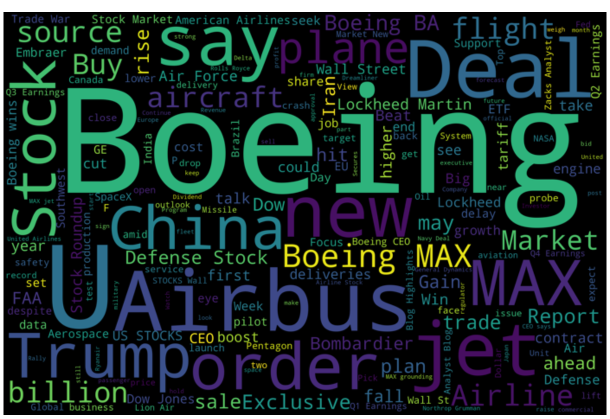

Figure 3.2 Word Cloud of BA News Title.

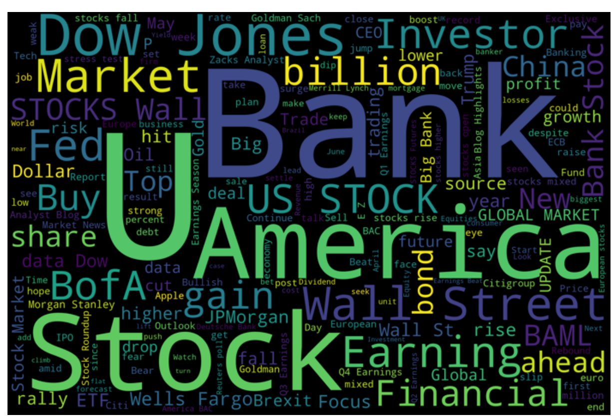

Figure 3.3 Word Cloud of BAC News Title.

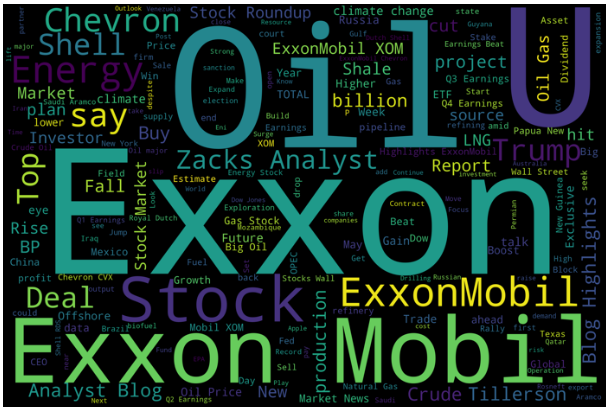

Figure 3.4 Word Cloud of XOM News Title.

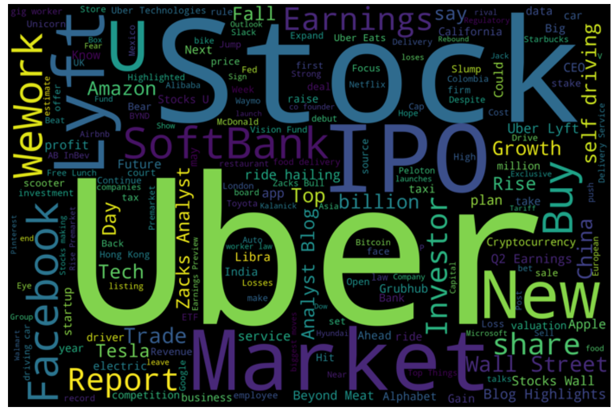

Figure 3.5 Word Cloud of UBER News Title.

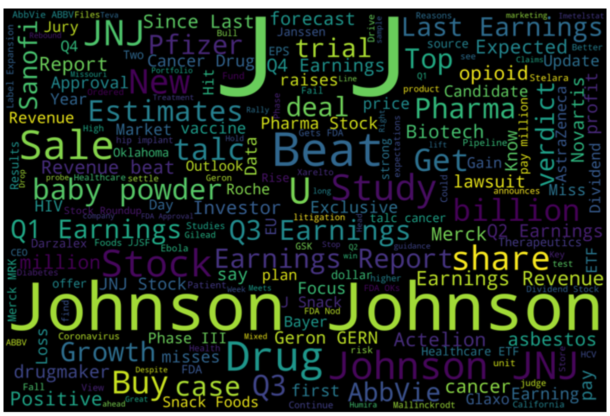

Figure 3.6 Word Cloud of JNJ News Title.

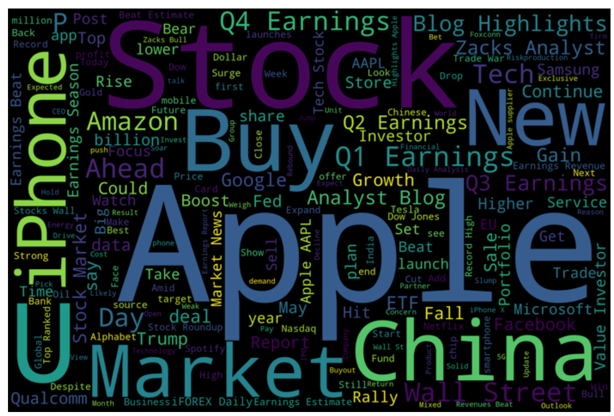

Figure 3.7 Word Cloud of AAPL News Title.

It is obvious that company name is always frequent, and further stopwords list is adjusted based on this property in Chapter 4.2. Besides, the textual dataset we use covers the various period depending on stock. Reason to choose this dataset is because there are trading days without news for each stock (See Table 3.1.), which is more realistic. Even for huge company Apple Inc., there are still 158 trading days without any news.

Table 3.1 Numbers of News, Days , and Days Without News.

Ticker	Period		# of news	# of days without news	# of total days
BA	2008-10-30	2020-01-30	5773	1460	2831
BAC	2008-10-08	2020-01-28	7234	825	2845
XOM	2009-05-21	2020-02-12	2763	1740	2701
UBER	2019-05-14	2020-01-21	1195	2	174
JNJ	2012-07-23	2020-02-10	750	1456	1900
AAPL	2012-07-17	2020-01-27	19972	158	1894

It happens practically so stock price forecasting requires other information, and we choose historical prices because historical prices are time series data and easy to obtain. Historical prices are critical due to lack of news and opinion for days, and combination of two types of data is helpful to show the individual influence of news and opinions. We obtain historical prices of the six stocks on Yahoo! Finance website [18] over the same period in order to complement news and opinions data. The numerical data set of historically daily price is series of close prices of the six stocks, and chart of historical prices are able to show relative stability of stock trend.

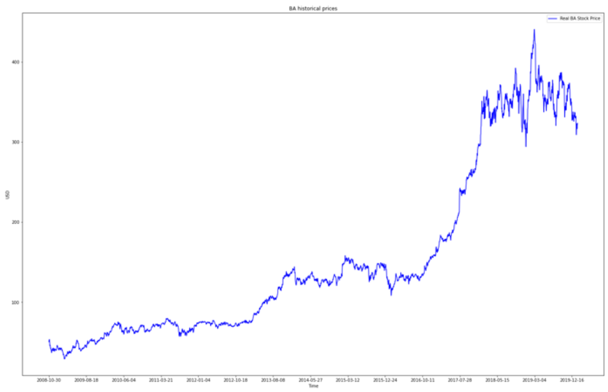

Figure 3.8 Chart of Historical Prices of BA.

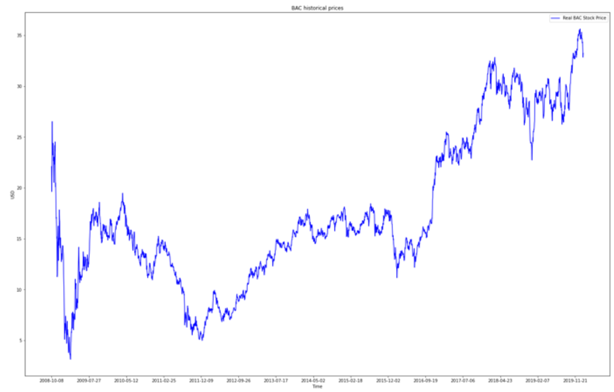

Figure 3.9 Chart of Historical Prices of BAC.

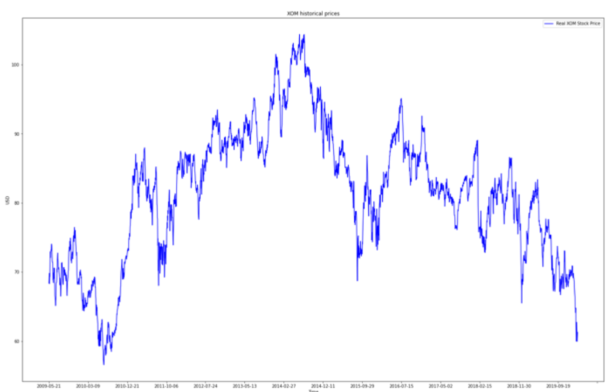

Figure 3.10 Chart of Historical Prices of XOM.

Figure 3.11 Chart of Prices of UBER.

Figure 3.12 Chart of Historical Prices of JNJ.

Figure 3.13 Chart of Prices of AAPL.

For time series analysis, dataset is split into training set and test set without shuffle. We can observe that chart of historical prices of BA shows difference after 2018. We might need transformation method to ease the difference. Moreover, the reason to choose companies in different sectors is because we are interested in whether sector type affects result of time series analysis and sensitivity to investors' sentiment. These sectors are Financial services for BAC, Industrials for BA, Energy for XOM, Healthcare for JNJ, and Technology for UBER and AAPL. We are also interested in relationship between stock volatility and capitalization.

Table 3.2 Stocks Overview.

Company	Ticker	Capitalization	Sector
Bank of America Corporation	BAC	302 B	Financial services
The Boeing Company	BA	127 B	Industrials
Exxon Mobil Corporation	XOM	221 B	Energy
Uber Technologies, Inc.	UBER	102 B	Technology
Johnson & Johnson	JNJ	426 B	Healthcare
Apple Inc.	AAPL	2120 B	Technology

Chapter Four:

Data Preprocessing

Preprocessing for textual dataset belongs to natural language processing (NLP), which is an area of research and application that explores how computers can be used to understand and manipulate natural language text [21]. It plays a critical role in sentiment analysis because machine learning algorithms show better performance when text is transformed into a more digitally digestible form [20, 22]. In this thesis, we apply word tokenization, stop-words removal, stemming, and part-of-speech tagger for textual dataset, and data transformation for numerical dataset prior to further models for better performance.

4.1 Word Tokenization

Word tokenization is the procedure that input text is split into a sequence of words. It is necessary for sentiment analysis because words are basic unit to sentiment analysis. The words split by word tokenization are called tokens. In general, word tokenization is to "tell" computer that the basic unit is a word but a sentence or a letter. In this thesis, a headline is considered as a single unit by computer before word tokenization, and then a sequence of words which is transformed from the headline would be considered as input tokens by computer. Besides, tokenization sometimes includes normalization such as lowercasing. Lowercasing is the simplest preprocessing technique which consists of lowercasing each single token of the input text [22], which transforms every word of text into lowercase. Normalization also includes abbreviation replacing which replace abbreviation with its words. “Isn’t”, for instance, is replaced by “is not”.

For better information extraction from textual data, we apply word tokenization, lowercasing, and abbreviation replacement.

4.2 Stop-words Removal

We consider stopwords removal as crucial step to reduce noise of textual data. Stop words, referred to as function words, consist of high-frequency words that usually include little useful information. We need to remove stop words because these words include little information and slow down computational speed such as “the”, “in”, and “a”. Besides, more words lead more parameters, which increases overfitting risk. Stop words include pronouns, determiners, and so on. “She”, for instance, is usually a stop word, and it does not bring much information to further analysis. Some sentiment classifiers show an improvement in accuracy when stopwords removal is applied as a preprocessing step [23].

We construct our stop-words list based on the idea Fox (1989) proposes, and we also adjust list depending on input data. Fox (1989) shows how to generate a stop-words list for general text based on Brown Corpus which contains 500 samples of English-language text, about one million words[6]. Several arbitrary decisions were made for compiling list of the most frequently occurring word. Firstly, a cut-off point has been chosen, which is the size of the list. The size of the stopword list bases on observation to count and browse Brown corpus. The way to count words is to count word lemmas by hands. Words such as “go”, “went”, “goes”, “gone” are counted as the same word, but words which can be noun or verb would be counted as two words. For instance, noun keep and verb to keep are counted different. They observe a situation that many words, including words as important as index items, occur at rate of one or two hundred per million in English. With the observation, the size of stop list should be less than 300 words. Furthermore, stop words are added into the list because many words traditionally appearing in stop list did not in the preliminary list. Words such as “above”, “sure”, and “whether” less than 300 times are added.

Our stop-words list is originally from Natural Language Toolkit (NLTK) which is a platform for building Python programs to work with human language data. The list contains 127 stopwords, and we add new stop words based on word frequency.

For instance, the word “apple” is extremely frequented due to news related to Apple Inc. It means that “apple” brings little sentiment information so we add apple into our stopwords list. Besides, our data consists of financial news or opinion, and words related to direction or trend include up, under, below, etc. are commonly used. These words should be crucial to investors’ sentiment so we remove them from list.

4.3 Stemming

Stemming algorithm is necessary to our data because number of total words in text affects the computational speed and overfitting risk. For grammatical reasons, text would use different forms of a word, such as “compute,” “computed,” and “computing”. Additionally, there are families of derivationally related words which are words with a same root. “Computer”, “computation”, and “computational”, for instance, these derivationally related words usually bring similar sentiment polarity so they should generally belong to a stem to reduce the size and complexity of input data. Stem is the form of a word before inflectional affixes are added. Stemmer, or so-called stemming algorithm, is the process to cut off words’ inflectional affixes to its stem form. All the words in example can be stemmed to “comput” by Porter stemmer.

Porter stemmer, as most popular rule-based stemmer, brings a fast computation based on vowel and consonant [8, 14]. It removes suffix for better information retrieval but linguistic readability so that sentence after Porter stemming might not be readable. Porter stemmer has only 5 if-then steps, and it practically works well.

4.4 Part-of-speech Tagging

Part-of-speech (POS) tagging is a process to classify words on the basis of part of speech category such as noun, verb, adverb, and adjectives. POS tagger is required to some lexicon-based scoring system. POS tags describe the characteristic structure of words within a text, and the information is useful for accurate sentiment score which is scaling system to give an associated score to words having a negative, neutral, or positive sentiment. POS tagger in this thesis bases on WordNet which is a large lexical database of English, which is dictionary-based [9].

4.5 Data Transformation

Since our numerical data is obtained from Yahoo! Finance, there is no missing values and it is decent. One problem is wide range among stocks. BA stock prices, for instance, shows an extremely wide range as shown:

```
count    2831.000000
mean      152.886697
std       107.245339
min        29.360001
25%        71.379997
50%       127.139999
75%       179.345001
max       440.619995
```

Figure 4.1 Descriptive Statistics of BA Prices.

Data transformation is to apply mathematical function to each sample point in a dataset. The purpose of data transformation is to make data closely meet the assumptions of statistical inference. Common transformations are logarithm, and MinMaxScaler, and the transformation we apply varies among the six stocks depending on error.

One advantage of log transformation is to make highly skewed distribution less skewed. According to previous researches, natural logarithm and logarithm with 10 of base have been applied. There is no huge error different between the two logarithms, and we decide to apply logarithm with 10 of base. Besides, MinMaxScaler is to rescale features to a given range, e.g., between zero and one, by computation minus minimum value and then divided by the difference between maximum and minimum value:

$$X\_rescaled \ = \frac{(X - X\_min)}{(X\_max \ - \ X - min)}$$

Our experiments without data transformation shows much more higher error measure than experiments with transformation. Thus, transformation is always applied in our experiments.

Chapter Five:

Methods and Procedures

5.1 Sentiment Score

Words are assigned an associated score by scaling system, and which represents the scale of sentiment polarity [24]. The sentiment score represents the scale of polarity of a given text, which is the one purpose of sentiment analysis. The range of a scale varies, but it always represents the scale from extremely negative to extremely positive polarity. Based on scaling system, researchers are able to sophisticatedly understand sentiment polarity of a target text by sentiment scores of words in the target text [25, 26]. Practically, there are several ways to determine polarity of a target text. With negative score to negative words and positive score to positive words, polarity of a sentence can be computed by sum of scores of words of a sentence representing sentimental polarity. A sentence is considered as positive sentiment with positive score of sums, and a sentence is considered as negative sentiment with negative score. We estimate polarity of daily news in score based on the scaling systems, and use these scores to predict stock price because sentiment score might be highly correlated to direction of stock price movement. Two approaches to construct scaling system are introduced.

5.1.1 Pointwise Mutual Information and Information Retrieval

The sentiment polarity of individual words, also known as semantic orientation, can be calculated by Semantic Orientation from Pointwise Mutual Information and Information Retrieval (SO-PMI-IR) [10]. SO-PMI-IR bases on seven opposing pairs, called as seed words, to infer semantic orientation. Seven opposing pairs include seven positive words (good, nice, excellent, positive, fortunate, correct, and superior) and seven negative words (bad, nasty, poor, negative, unfortunate, wrong, and inferior). With assumption of independence between words, they applied pointwise mutual information (PMI) to compute strength of the semantic association between words:

$$PMI(word1, word2) = \log_2 \frac{P(word1, word2)}{P(word1)P(word2)}$$
$$= \log_2 \frac{\frac{C(word1, word2)}{N}}{\frac{C(word1)}{N} \cdot \frac{C(word2)}{N}}$$

P(word1) in formula represents the probability of word1 in text, which is calculated by numbers of word1 in text, denoted as C(word1), divided by number of words in text, denoted as N. Furthermore, semantic orientation of word1 is calculated by:

$$score(word1) = SO - PMI - IR(word1) =$$

$$PMI(word1, \{positiveparadigms\}) - PMI(word1, \{negative\ paradigms\})$$

where {positive paradigms} represents set of the seven positive words, and {negative paradigms} represents set of the seven negative words. Based on SO-PMI-IR, sentiment lexicon is created by utilizing large twitter corpora [15, 27]. 78 seed words (32 positive and 36 negative) are chosen from hashtagged emotional words from 775,000 tweets. The sentiment score for a word is calculated as shown above.

5.1.2 SentiWordNet

SentiWordNet, another scaling system to scale of sentiment polarity, is constructed with the similar idea to previous system. Firstly, the mentioned seven opposing pairs in 5.1.1 are considered as original set of seed words with manually labelled. The original sets of positive words and negative words, denoted L_p and L_n, are then iteratively expanded in K iterations into final training set, denoted Tr_p^k and Tr_n^k. At each iteration step k, two sets Tr_p^k and Tr_n^k are generated, where $Tr_p^k \supset Tr_p^{k-1} \supset \cdots \supset Tr_p^1 = L_p$ and $Tr_n^k \supset Tr_n^{k-1} \supset \cdots \supset Tr_n^1 = L_n$. In other words, sets after each iteration are added new words and also contain previous words. Iteration step k is from 0 to 6, and new words are decided by direct antonymy, similarity, derived-form, pertain-to, attribute, and also-see respectively from other resource. Besides, a set for objective words, denoted L_o, is constructed by words that do not have either positive or negative characteristics in General Inquirer lexicon [81], and L_o always consists of 17,530 synsets. Afterward, a ternary classifier is applied, which includes two binary classifiers. One is to classify words to positive or not positive, and the other one is to classify words to negative or not negative. Words are considered as positive when positive by former classifier and not negative by latter classifier. Words are considered as negative when not positive by former classifier and negative by latter classifier. Words are considered as objective when positive by former classifier and negative by latter classifier, or when not positive by former classifier and not negative by latter classifier. For training classifier, four training sets are determined when iteration step k=0, 2, 4, 6. Two alternative classifiers are Rocchio [84] and Support vector machines packages [83]. Finally, sentiment scores to each synset are obtained by the ternary classifiers, and then are normalized to 1.0.

5.2 Artificial Neural Networks

Artificial neural networks (ANNs), usually called neural networks (NNs), are nothing more than nonlinear regression and discriminant models [36]. NN is critical to our NLP tasks because we apply it to map words form the vocabulary to vectors of real numbers, which is called word embedding and introduced in Section 5.3. Besides, two members of NN family, convolutional neural networks and recurrent neural networks, are used to avoid overfitting and to perform time series analysis due to their characteristics respectively, which are introduced in the further Sections 5.4 and 5.5, respectively. A neuron as simplest and basic unit of neural networks is just linear regression. Neural networks are able to do non-linear algorithm by combination of linear regressions. Given an input data matrix X with n observations and k features, the input for a neuron is linear combination of X and parameter vector w.

$$X \in F^{(k+1)\times n}$$

$$w \in F^{(k+1)\times 1}$$

The unit works in the following way:

$$y = f(u)$$

where u is a scaler number, which is input of the neuron. Number u is defined as:

$$u = w^T X$$

The size of X depends on how many data we like to feed algorithm each time. The output of a neuron is derived from activation function *f*. Popular activation functions include Heaviside function (ReLU function), hyperbolic function (tanh function) and logistic function (sigmoid function) as shown as:

$$logistic\ function\colon\ f(u) = \frac{1}{1 + e^{-u}}$$

$$hyperbolic\ function\colon f(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

$$Heaviside\ function\colon\ f(u) = \begin{cases} u & if\ u > 0 \\ 0 & otherwise \end{cases}$$

Neural networks are popular due to its further derivation, multi-layer perceptron (MLP). An MLP is composed of one input layer, one or more layers (so called hidden layers), and one output layer. Every layer includes one or more neurons. Every layer except the output layer includes bias neuron and is fully connected to next layer. A neural network is called deep neural network (DNN) when it includes two or more hidden layer. Furthermore, softmax function is usually applied for classification task, which estimates the probability of each output value of output layer. Given M classes, softmax function is calculated as:

$$softmax\ function\colon\ f(O_j) = \frac{e^{O_j}}{\sum_{j=1}^{M} O_j}$$

For error measure, cross entropy as following equation is commonly applied for classification task. Unlike mean square error, cross entropy stands for the difference between predicted probability and target probability from given dataset.

$$H(t,p) = -\sum_{x} t(x)\ log\ p(x)$$

where t is target distribution and p is predicted distribution.

Furthermore, backpropagation training algorithm, known as Gradient Descent (GD) with chain rule, is applied to update weights to perform optimization. GD is a very generic, basic and common algorithm to optimize neural networks with differentiable loss function. The idea of GD is to tweak parameters iteratively in order to minimize object function, which is loss function for machine learning approaches [12, 32]. Gradient is the partial derivatives of the function, which means a vector with increasing direction. It is explanation why there is a minus in gradient descent step as following equation. There are also two hyperparameters, learning rate η and value of parameters θ (weights in NN) at beginning.

$$\theta^{(new)} = \theta^{(old)} - \eta \nabla_{\theta} L(\theta)$$

where $\nabla_{\theta} L(\theta)$ is gradient vector and $L(\theta)$ is loss function. The gradient vector is computed as:

$$\nabla_{\theta} L(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} L(\theta) \\ \frac{\partial}{\partial \theta_1} L(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} L(\theta) \end{pmatrix}$$

Gradient descent algorithm cannot work at all when the initial value of parameters are zero. Moreover, gradient vector often become smaller and smaller values when the algorithm progresses down to lower layers, which means parameters in lower layer keep unchanged, called vanishing gradients problem. Weight Initialization strongly affect the efficiency to reach local minima. A popular solution is to random initialization with mean of zero and standard deviation of one [33]. We also apply random initialization to avoid vanishing gradient problem.

Besides, the number of hidden layers and the number of neurons per hidden layer are still somewhat of a black art. All we know is that number of hidden layers is related to how complexity of target model. Higher level structures are required higher hidden layers. A practical approach is to pick a model with more layers and neurons as possible, then apply regularization such as dropout or early stopping.

For better optimization, some gradient descent-based algorithms also consider combination of learning rate, gradient, and momentum, etc. These algorithms include GD with momentum, RMSProp, Adam and so on [36, 37]. All of them focus on the width of step of gradient descent. Firstly, gradient descent with momentum bases on an idea that gradient is related to previous gradients by adding a momentum vector. Gradient descent with momentum gets fast, if gradients are in the same direction by adding a new hyperparameter β_1, called the momentum, as shown:

$$m^{(new)} = \beta_1 m^{(old)} - \eta \nabla_\theta L(\theta)$$

$$\theta^{(new)} = \theta^{(old)} + m^{(new)}$$

Secondly, gradient descent ideally should be fast for steep dimensions and it should be slow for gentle slope. For the purpose, adaptive learning rate is applied, which means learning rate of each parameter is divided by the root mean square of its previous derivatives. Besides, decay rate β_2 is added to slow down gradient descent. The algorithm based on combination of adaptive learning rate and decay rate is called RMSProp as shown:

$$s^{(new)} = \beta_2 s^{(old)} + (1 - \beta_2) \nabla_\theta L(\theta) \otimes \nabla_\theta L(\theta)$$

$$\theta^{(new)} = \theta^{(old)} - \eta \nabla_\theta L(\theta) \oslash \sqrt{s^{(new)} + \varepsilon}$$

where $\otimes$ is element-wise multiplication, $\oslash$ is element-wise division, and ε is a number to avoid division by zero. RMSProp is created by Tijmen Tieleman and Geoffrey Hinton in 2012, and presented in his Coursera class on neural networks and no paper is written for it. Furthermore, Kingma and Ba propose adaptive moment estimation algorithm (adam) by combining momentum and RMSProp as following equation [35]. Since adam requires less tuning learning rate, we applied it for our experiments.

$$m^{(new)} = \beta_1 m^{(old)} - (1-\beta_1)\eta\nabla_\theta L(\theta)$$

$$s^{(new)} = \beta_2 s^{(old)} + (1-\beta_2)\nabla_\theta L(\theta)\otimes\nabla_\theta L(\theta)$$

$$\widehat{m} = \frac{m^{(new)}}{1-{\beta_1}^t}$$

$$\hat{s} = \frac{s^{(new)}}{1-{\beta_2}^t}$$

$$\theta^{(new)} = \theta^{(old)} - \eta\widehat{m} \oslash \sqrt{\hat{s}+\varepsilon}$$

where t is the iteration number. Since m and s are initialized at 0, they are divided by small numbers to counteract their bias to 0 especially for first few iterations.

Since NN usually has so many parameters to fit a huge variety of complex datasets, it is likely to overfit. Regularization is quite useful to ease the overfitting situation. Penalty can be added into loss function. Otherwise, dropout is a simple but effective technique to prevent NN from overfitting [34]. Dropout is that every neuron in neural network has probability p of being entirely ignored during training step at every training step. The probability p is 0.5 in our experiments.

5.3 Word Embedding

For every language analysis, the first and most important is to find a great word representation method which is able to represent semantic and contextual meaning, and further to obtain sentence/text information. In our study, one of our concerns is the potential relationship between news headlines information and stock price. Pretrained word embedding is applied to our news data due to our small size of textual data.

5.3.1 Word2vec

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing (NLP) where vocabulary words are mapped to vectorial representation of real numbers. The key idea for word embedding is dimension reduction, which means that mathematical embedding from a space with many dimensions per word to a continuous vector space with much lower dimension [28]. Methods to generate vector representation include neural networks (NN), word co-occurrence matrix, and so on. Tomas Mikolov and colleagues propose two NN-based model architectures to learn vector representations of words, which are continuous bag-of-words model (CBOW) and continuous skip-gram model (Skip-gram). Given a sentence, denoted S, including n words as training dataset, S = ($w_1, w_2, \ldots, w_n$) where w_1 stands for first order word in the sentence, w_2 stands for second order word, and so on [11].

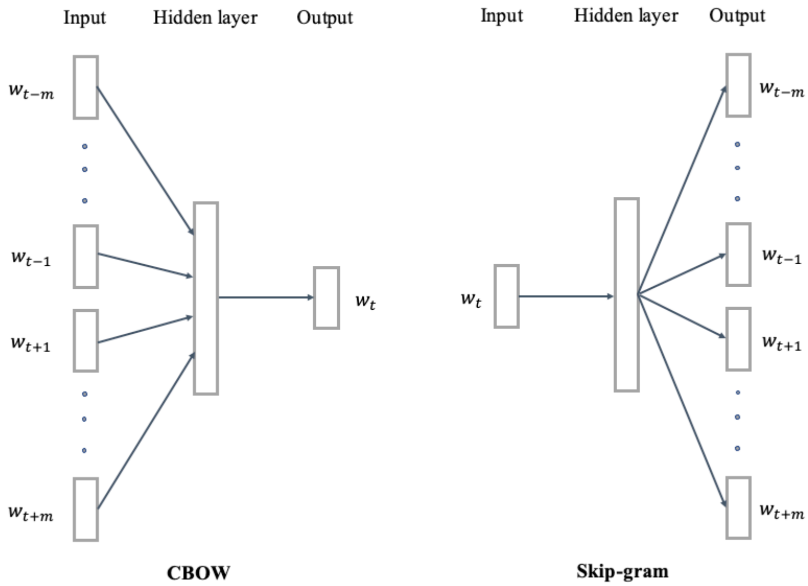

Figure 5.1 Two Model Architectures, CBOW and Skip-gram.

The CBOW predicts the current word based on surrounding 2m words, and the Skip-gram predicts 2m surrounding words based on the current words. With mathematical detail, CBOW is shown as:

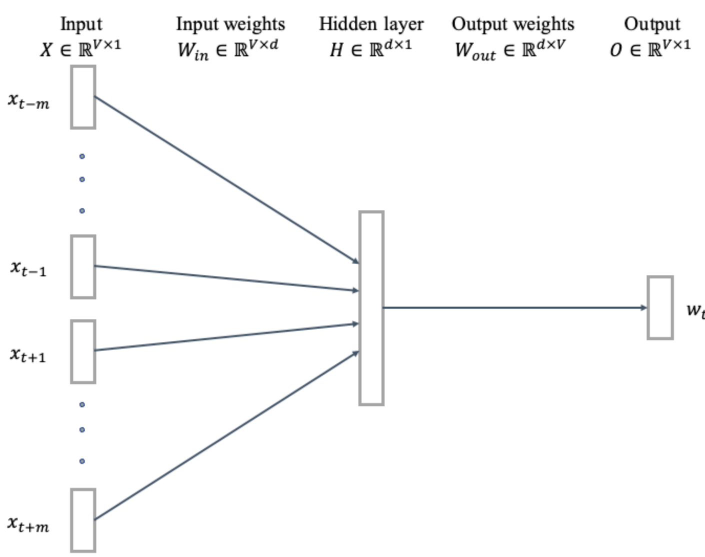

Figure 5.2 CBOW with Dimensions.

The input vectors are one-hot encoded, which means only one out of V units will be 1, and all other units are 0 for a given input context word. V is the number of distinct words in the given training texts. The weight between input layer and hidden layer is a V×d matrix W_{in}, where d is a hyperparameter. The "d" is number of dimensions of new word vector. Each row of W_{in} is the d-dimension vector representation of the associated word ω_i of the input layer, denoted $v_{\omega_i}^T$, where $i = 1, \ldots, V$. Because input vectors are one-hot encoded, given a context, hidden layer output is computed as:

$$h = \frac{1}{2m} W^T (x_{t-m} + \cdots + x_{t-1} + x_{t+1} + \cdots + x_{t+m})$$

$$= \frac{1}{2m} \left(v_{\omega_{t-m}} + \cdots + v_{\omega_{t-1}} + v_{\omega_{t+1}} + \cdots + v_{\omega_{t+m}} \right)^T$$

where m is window size we have chosen. From hidden layer to output layer, a score u_j for each word can be computed as:

$$u_j = {v'_{\omega_j}}^T h$$

where v'_{ω_j} is the j-th column of the weight matrix W_{out}. Furthermore, softmax function is applied to obtain the posterior distribution of words as:

$$p(\omega_t \mid \omega_{t-m} \cdots \omega_{t-1} \omega_{t+1} \cdots \omega_{t+m}) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})}$$

The loss function is maximum of the conditional probability of actual output word ω_t given its surrounding words $\omega_{t-m} \cdots \omega_{t-1} \omega_{t+1} \cdots \omega_{t+m}$ [30]:

$$max\ p(\omega_t \mid \omega_{t-m} \cdots \omega_{t-1} \omega_{t+1} \cdots \omega_{t+m})$$

The skip-gram model is opposite to CBOW model, which means surrounding words are predicted based on current word.

Vector representations of words by the two model architectures are capable to be measured similarity between words by cosine similarity. Given vector representations of two words, $\vec{a}$ and $\vec{b}$, the similarity between them, $\cos(\theta)$, is represented as:

$$similarity = cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \, ||\vec{b}||} = \frac{\sum_i a_i \cdot b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}}$$

where $\vec{a} \cdot \vec{b}$ is dot product of the two vector, and $a_i \; and \; b_i$ are components of vector $\vec{a}$ and $\vec{b}$ respectively. Moreover, the impressive thing is that relationship between words is characterized by a relation-specific vector offset. The famous example is that vector("king") – vector("man") + vector("woman") result in a vector which is closest to vector representation of word "queen" [11, 29]. These representations capture syntactic and semantic regularities in English.

Since our dataset is much smaller than other dataset to pretrained word2vec model, we applied Google's trained model which is trained on roughly 100 billion words from a Google News dataset [31].

5.3.2 Bidirectional Encoder Representations from Transformers

Bidirectional encoder representations from transformers (BERT) is a language representation model from unlabeled text [86], and it is a pre-training transformer-based machine learning approach for NLP task by Google. Pre-training BERT is used for headlines embedding due to our small size of distinct words in our textual data, our limitedly computational power, and its notable achievements for many natural language processing tasks. We generally know what structure it is from its name, which is encoders from transformer as shown in figure 5.3. The encoder consists of a stack of 6 identical layers. Each layer contains two sublayers, multi-head self-attention mechanism, and fully connected feed-forward network. The outputs of dimension, denoted d_{model} for all sub-layers including embedding layers is 512 in original paper [85].

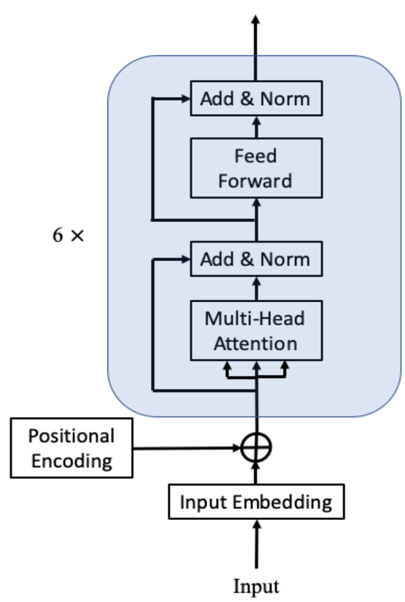

Figure 5.3 Architecture of Encoder From Transformer.

where $\oplus$ represents concatenation. Firstly, positional encoding is applied to obtain relative or absolute position information because order of sequence of input data is unimportant for models with self-attention mechanism. Position information can be provided by one-hot encoding, and in paper to encoder, sine and cosine functions of different frequencies are used for positional embedding (PE) as shown:

$$For\ even\ number\ of\ positions: PE_{(pos,2i)} = sin({}^{pos}\!/_{10000^{2i/d_{model}}})$$

$$For\ odd\ number\ of\ positions: PE_{(pos,2i)} = cos({}^{pos}\!/_{10000^{2i/d_{model}}})$$

where pos is the position and *i* is the dimension. These functions are chosen because they would allow model to easily learn. Afterward, positional embedding is concatenated with input embedding vector as new input vector for multi-head self-attention mechanism.

Since multi-head self-attention mechanism consists of multiple self-attention function, self-attention function should be explained. Self-attention function is to map a query and key-value pairs to an output [85]. In other words, three vectors, query, key, and value, are created by three learnable weight matrices, denoted $W_Q, W_K, and\ W_V$ respectively, and then scale of importance is calculated by element-wise multiplication of query and key vector. The output is produced by element-wise multiplication of value and scale of importance after softmax function. In mathematical detail, given a input matrix a, self-attention function is shown as:

$$Q = W_Q^T a$$

$$K = W_K^T a$$

$$V = W_V^T a$$

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-head self-attention is actually multiple self-attention mechanism, which has h sets of $W_Q, W_K, and\ W_V$, and h is hyperparameter. The output of multi-head self-attention is concatenation of outputs of all self-attention.

Afterward, output of multi-head self-attention is concatenated with input vector, and followed by layer normalization. Layer normalization is a technique to normalize summed input into standard normal distributed to each hidden layer. Layer normalization is able to reduce covariate shift problem which means that changes in the output of one layer will tend to cause highly correlated changes in the summed inputs to the next layer [88].

5.4 Convolutional Neural Networks

Convolutional neural network (CNN) is an important approach to cope with our textual data and numerical data. Unlike long short-term memory introduced in Section 5.5, CNN extracts information of subsequence of our series data through whole dataset by its sliding filters. Given *m* of window size, for our historical prices data, prices of *m* days are converted into one new representation form. For textual data, matrix of *m* of words are also converted into one new form. Besides, CNN is popular in hybrid approaches such as convolutional long short-term memory due to its ability to extract information.

CNN is one member of neural networks family proposed by Lecun et al. in 1998 [71], and originally designed to perform image-driven pattern classification problems. Afterward, it is also used to perform NLP task such as sentiment classification [70] based on its architecture and mechanism, and its key characteristic is combined with notable long short-term memory for time series analysis such as financial asset price forecasting [65, 72].

CNN is analogous to ANN in the idea that they consist of layers of neurons which optimize through gradient descent. The obvious difference between CNN and ANN is that CNN is able to reduce overfitting risk than ANN. For pattern recognition within images, ANN requires much more parameters to compute image data than CNN, which leads that neural networks is unable to extract pattern of data effectively. Thus, two new types of layers, convolutional layer and pooling layer, are added into ANN architecture to reduce computational complexity, which is named convolutional neural network (CNN). A simple CNN architecture, comprised of one convolutional layer with 4 kernels first, one pooling layer, and artificial neural network (See figure 5.4). Firstly, the image is converted into matrix form of pixel values. Secondly, the convolutional layer determines the output of neurons of which are connected to local regions of the input matrix through the calculation of element-wise product between learnable kernel (weights) and the region connected to the input matrix. Thirdly, the pooling layer downsamples along the spatial dimensionality of the given input, which reduces number of parameters. Lastly, traditional artificial neural network is performed [73] for further classification or regression task.

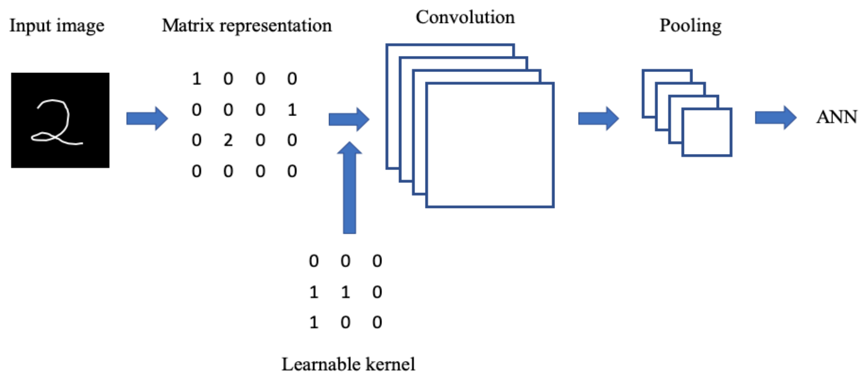

Figure 5.4 An Simple CNN Architecture

The convolutional layer is the critical and crucial layer for CNN operation, and the focus of convolutional layer is the learnable kernels as known as filters. These filters are always smaller than input matrix in spatial dimensionality, and produce convolutional feature map by element-wisely multiplying connected regions across the spatial dimensionality of the input matrix. That's how convolutional layer is able to reduce complexity of input data through the optimization of its output. The optimization bases on four hyperparameters, the number of filters, the size of filter, the stride, and zero-padding. The learnable filters are actual parameters of neural networks in matrix forms so that overfitting risk increases when the number of filters increases.

Similarly, overfitting risk increases with the large size of filter due to increasing number of parameters. Besides, large size of filter leads to the smaller feature map. The stride is the depth that kernels slide across the spatial dimensionality of the input matrix. Feature map becomes smaller when stride increases. Zero-padding is the process of padding border of the input matrix with zero, which is able to maintain the size of feature map as input matrix. Given the same conditions of figure 5.4, the size of kernel is 3 by 3 the stride is 1, number of kernel is 1, computation of convolutional layer is shown as:

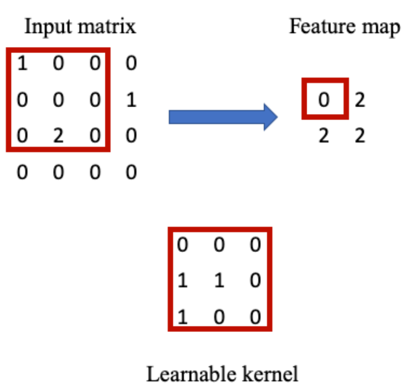

Figure 5.5 Computation of Convolutional Layer.

The learnable kernel element-wisely multiplies region across the spatial dimensionality of the input matrix. The output of convolutional layer as known as feature map, would be a 2 by 2 matrix.

Pooling layer is the next step which also reduce dimensionality of input matrix and complexity of the neural network by downsampling. There are two common pooling layers, max-pooling layer and average-pooling layers. Pooling layers also require a kernel size and stride of the kernel along the spatial dimensions of input. The average-pooling layer computes the average of elements of corresponding regions of input matrix as output. The max-pooling layer computes the maximum value of elements of corresponding regions of input matrix. Given the condition of Figure 5.4.2 and 2 by 2 of kernel size, output of max-pooling layer and average-pooling layer are respectively shown as:

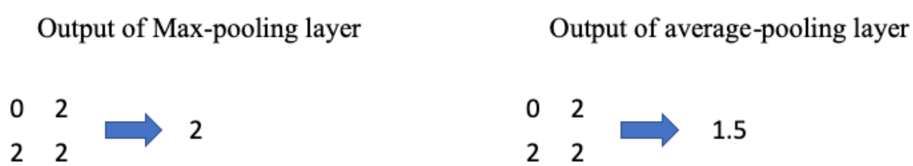

Figure 5.6 Computations of Max-pooling Layer and Average-pooling Layer.

In our example, 4 by 4 of input matrix becomes one value after convolutional layer and pooling layer. That's why CNN is able to cope with high dimension of image classification problems. Besides, CNN is also able to cope with sentiment classification problems based on its sliding property [70]. Since words can be represented in vector form after word embedding, a sentence as a series of words is represented in matrix form. Afterward, the matrix of a sentence can be considered as input matrix to CNN as shown. Given conditions that sentence contains $|s|$ of words, each word is represented in 1 by *d* of vector, filter size is *d* by *m*, architecture of CNN for sentiment classification problem is shown as:

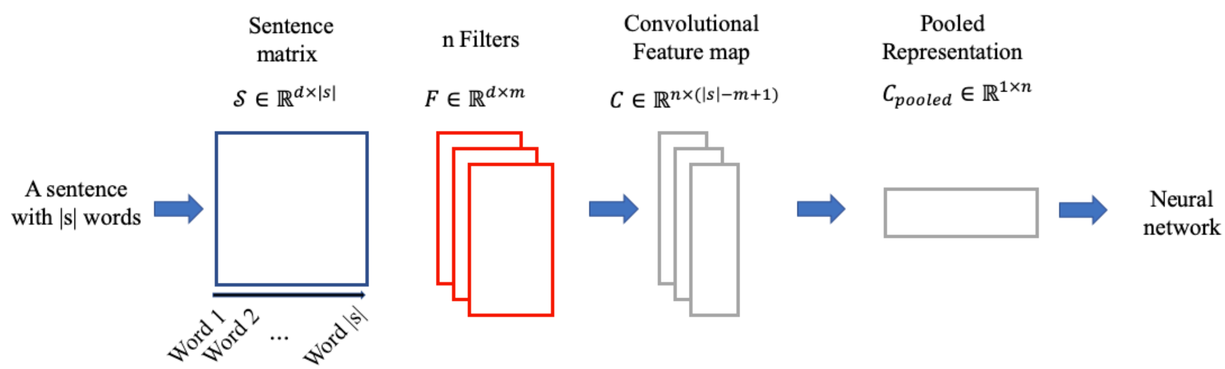

Figure 5.7 The Architecture of CNN Model for Sentiment Classification.

Every sentence can be represented in matrix form by combination of series of its words vector, and then it is considered as input matrix of convolutional layer. Filter (F) as known as kernel always has the same length (*d*) with length of word vector in natural language process. The width m of filter is decided by us. Afterward, element-wise multiplication between filter and regions of input matrix results in convolutional feature map, and spatial dimensionality of feature map further decreases by pooling layer. Lastly, any classifier can be performed for classification task.

Moreover, CNN of sliding characteristic with recurrent neural networks is also powerful and useful to time series analysis. Unlike traditional time series analysis approach, its sliding characteristic is able to extract days' information each stride. Thus, convolutional layer is often combined with recurrent neural networks.

5.5 Recurrent Neural networks (Long Short-term memory)

Recurrent neural network (RNN) was referred to neural networks structure with repeated loops conditionally allowing information moving from one state to afterward states. RNN is the main tool in our study due to its moving characteristic and high performance in sentiment analysis and time series analysis. It can be used not only to forecast stock price on next day but also to obtain news headlines vector because text is considered as a series of words and historical prices are considered as a time series of prices. Given input data at time step t, denoted $X_{(t)}$, and output (cell's state) at time step t, denoted $h_{(t)}$, RNN is as shown:

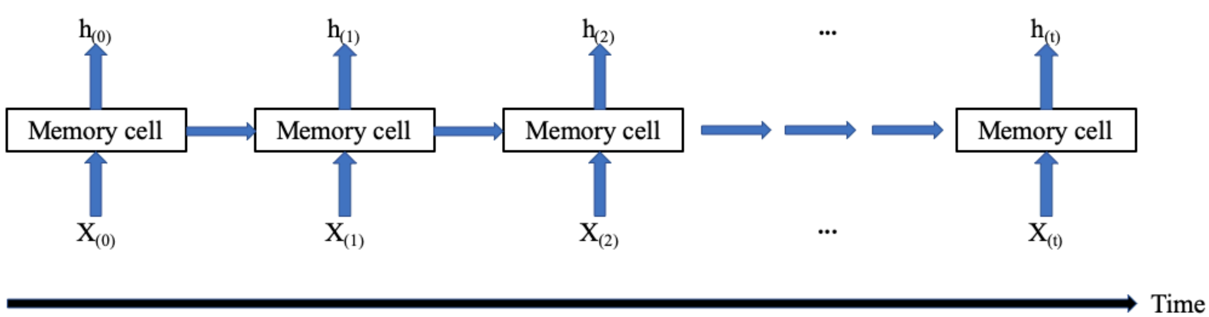

Figure. 5.8 An Unrolled Recurrent Neural Network Through Time.

where memory cell can be a single neuron, or a layer. RNN is capable to analyze series data such as time series and text which seems a series of in NLP. However, RNN is totally superseded by one special kind of RNN, named Long short-term memory (LSTMs), nowadays because of its vanishing gradient problem. LSTM is designed to store useful information for long period and to forget unnecessary information. Given input X at time step *t*, denoted $X_{(t)}$, and output Y at time step *t*, denoted $y_{(t)}$, computation of a LSTM cell is shown as:

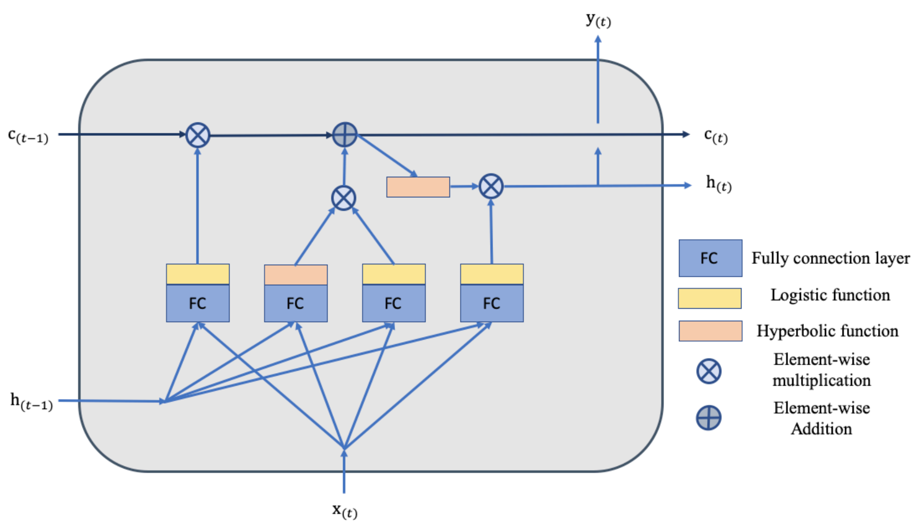

Figure 5.9 Long Short-term Memory Cell [12].

The structure of LSTM cell shows the three key points:

1. How LSTMs store long-term information, which is cell state $c_{(t)}$.
2. How LSTMs forget information, which is controlled by forget gate $f_{(t)}$.
3. How LSTMs get short-term output,, which is hidden state $h_{(t)}$.

The first key point is what scale of long-term information would be dropped, and the decision is controlled by forget gate layer. The output range of forget gate layer is (0,1) because logistic function is activation function, which represents the dropping percentage. The output can be computed as:

$$f(t) = \sigma(W_{x_f}^T X_{(t)} + W_{h_f}^T h_{(t-1)} + b_f)$$

where σ is logistic function $X_{(t)}$ is input vector at time step *t*, and $h_{(t-1)}$ is output vector at time step *t-1*. Besides, new information might be party added into cell state, and it is controlled by input gate layer and tanh layer. The input gate layer works as same as forget gate layer, which decides how much information is added into cell state due to (0,1) of range. The activation function of the *tanh* layer is *tanh* function as known as hyperbolic function, and it is a rescaling function due to its (-1,1) of range. Element-wise multiplication between them decides new information added into cell state:

$$i(t) = \sigma(W_{x_i}^T X_{(t)} + W_{h_i}^T h_{(t-1)} + b_i)$$

$$g(t) = tanh(W_{x_g}^T X_{(t)} + W_{h_g}^T h_{(t-1)} + b_i)$$

where *tanh* is hyperbolic function. Thus, the procedure of updating cell state is shown as:

$$c(t) = f(t) \otimes c(t-1) + i(t) \otimes g(t)$$

The last part is output for current time step, which is decided by current cell state, hidden state and input. Current cell state is rescaled by *tanh* function, and then multiplied by percentage which is decided by current input and last hidden state as shown:

$$o(t) = \sigma(W_{x_o}^T X_{(t)} + W_{h_o}^T h_{(t-1)} + b_i)$$

$$y(t) = h_{(t)} = o(t) \otimes tanh(c(t))$$

LSTMs as one kind of neural networks also use gradient descent to update parameters which are all W above.

5.6 Support Vector Regression

Support vector regression (SVR) is a notable global optimization method in nonlinear regression estimation with mathematically theorical support which tries to locate a hyperplane by transforming input data into a higher dimension space [76]. It has been successful applied in time series forecasting in financial market [75] so that we consider SVR as our baseline model.

Stock price prediction is to establish an optimal prediction function based on historical data and other technical indicators to forecast stock price. Suppose we are given training data $\{(x_1, y_1),\ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes the space of the input data. For example, these might be historical stock prices measured at subsequent stock price. The goal is to find a function f(x) having at most ε deviation from the actual targets y_i, and at the same time is flat as possible. In other words, those points with error less than ε are ignored. The function f is shown as:

$$f(x) = w^T x_i + b, \qquad w \in \mathcal{X}, b \in \mathbb{R}$$

The goal is considered as a convex optimization problem as:

$$\begin{cases} \min\limits_{w,b} \frac{1}{2} w^T w \\ subject\ to \begin{cases} y_i - (w^T x_i + b) \le \varepsilon \\ w^T x_i + b - y_i \le \varepsilon \end{cases} \end{cases}$$

Furthermore, slack variables ξ_i, ξ_i^* are introduced for otherwise infeasible constraints of the optimization problem [76], and problem can be stated as:

$$\begin{cases} \min\limits_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ subject\ to \begin{cases} y_i - (w^T x_i + b) \le \varepsilon + \xi_i \\ w^T x_i + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{cases}$$

where *C* is a regularization. A larger *C* gives more weight to minimize the error. By using Lagrangian, the constrained optimization problem can be solved as:

$$\min_{w,b,\xi_i,\xi_i^*} L(w, b, \xi_i, \xi_i^*) = \min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) - \sum_{i=1}^{\ell} (\eta_i \xi_i + \eta_i^* \xi_i^*)$$

$$- \sum_{i=1}^{\ell} \alpha_i (\varepsilon + \xi_i - y_i + w^T x_i + b)) - \sum_{i=1}^{\ell} \alpha_i^* (\varepsilon + \xi_i^* + y_i - w^T x_i - b))$$

The partial derivatives of *L* with respect to variables(w, b, ξ_i, ξ_i^*) equal to zero are applied for finding minimization of *L* as:

$$\frac{\partial L(w,b,\xi_i,\xi_i^*)}{\partial b} = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0$$

$$\frac{\partial L(w,b,\xi_i,\xi_i^*)}{\partial w} = w - \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i = 0$$

$$\frac{\partial L(w,b,\xi_i,\xi_i^*)}{\partial \xi_i} = C - \alpha_i - \eta_i = 0$$

$$\frac{\partial L(w,b,\xi_i,\xi_i^*)}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0$$

Also, *Karush-Kuhn-Tucker* (KKT) condition states the product of the Lagrange multipliers and the constraints is equal to zero as shown:

$$\alpha_i(\varepsilon + \xi_i - y_i + w^T x_i + b) = 0$$

$$\alpha_i^*(\varepsilon + \xi_i^* + y_i - w^T x_i - b)0$$

$$\eta_i \xi_i = 0$$

$$\eta_i^* \xi_i^* = 0$$

By substituting these equations, the optimization problem yields the dual optimization problem:

$$\begin{cases} max \ -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*){x_i}^T x_j - \varepsilon\sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \\ \qquad\qquad subject\ to \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_j, \alpha_j^* \le C \end{cases} \end{cases}$$

Thus, the goal function *f* is shown as:

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\, x_i^T x + b$$

KKT condition which state that the product between dual variables and constraints must vanish at point of the solution. Training data with error larger than ε will have nonzero $\alpha_i \ or \ \alpha_i^*$. Points with error less than ε, $\xi_i = 0$, and so does ξ_i^*. Therefore, b can be calculated by KKT condition:

$$b = y_i - w^T x_i - \varepsilon$$

$$b = -y_i + w^T x_i - \varepsilon$$

Furthermore, kernels are introduced which preprocesses features to yield nonlinearity by mapping transformation. Definition of kernel function is:

$$K(x, x') = \phi(x)^T \phi(x') =< \phi(x), \phi(x') >$$

$$\forall x, x' \in \mathrm{X}, \exists \phi: x \to Z$$

$$subject\ to\ K(x, x') = \phi(x)^T \phi(x')$$

The optimization problem would be restated as:

$$\begin{cases} max\ -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x, x') - \varepsilon\sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \\ \qquad subject\ to \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_j, \alpha_j^* \le C \end{cases} \end{cases}$$

The goal function *f* would be restated as:

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\, K(x, x') + b$$

Common kernel functions are linear, polynomial, Gaussian RBF, and sigmoid:

$Linear: K(x, x') = x^T x'$

$Polynomial: K(x, x') = (\gamma x^T x' + r)^d$

$Gaussian\ RBF: K(x, x') = exp(-\gamma \|x - x'\|^2)$

$Sigmoid: K(x, x') = tanh(\gamma x^T x' + r)$

Moreover, hyperparameters such as *C*, gamma γ, ε are critical to the regression result. *C* is regularization term which is a penalty. Small *C* leads larger margin, and vice versa. Besides, ε decides the width of margin, which is margin of tolerance, and data points in the margin would be ignored to compute error. In other words, smaller ε means that less data points would be ignored, and overfitting risk increases. Gamma is actually a parameter of kernel function. It affects how original data points are projected into higher-dimension feature space. Larger gamma tends to increase overfitting risk [79].

5.7 Evaluating the model

Forecast error is the measure to estimate how good the model is, and it is the difference between an actual prices and predicted prices in many ways. In this thesis, three error measures are applied, which are root mean squared error (RMSE), mean absolute error (MAE), and Mean absolute percentage error (MAPE). MAE and RMSE measure difference between predicted prices and actual prices as shown:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i'\right|$$

The two measures are popular because they are intuitive, but they might not work well in stock forecasting because of high volatility of stock market. It happens that stock price boosts more than 100% in short term such as GameStop stock price which boosts 500% in three days. They are sensitive to these outliers. MAPE is more intuitive than previous two measures. It shows the difference between predicted and actual prices in average percentage:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i' - y_i}{y_i'}\right|$$

MAPE resembles daily return which is percentage change between prices of two days. Despite various prices among our data, it is easy to compare one stock with other five stocks. For instance, one of notorious disadvantages of MAPE is that its denominator cannot be zero or MAPE does not exist, and it becomes advantage because it is hardly happens for stock price prediction. For easier readability, MAPE would multiply by 100 in result.

Chapter Six:

Models

For experimental setting, numerical dataset and textual dataset would be joint based on date so there are days without news (see table 3.1). Afterward, joint dataset would split into training and test dataset by 80/20 without shuffle because we prefer to keep it a time series of prices and news headlines. Series of data would be converted into generic data for computational reason.

We focus on the combination of time series analysis and sentiment analysis to better forecast stock price, and our combination approach includes four part:

1. Find a time series model which lead smallest error. Predicted prices by the model are considered as new input data.
2. Use current word embedding approach to obtain representation matrix of daily news headlines for each stock as new input data.
3. Apply lexicon-based sentiment approach to obtain the polarity of each daily news headline in score as new input data.

4. Perform neural networks with these new input data for stock price forecasting.

For time series analysis, window size is a key factor, which is number of price of the past days for current price prediction. Intuitively, everyone believes that there is somehow relation between historical stock prices and present price. We experiment different window sizes (3, 4, 5, 10, 15, 20, 30) and window size is decided by the model with smallest error (MAPE, MAE, and RMSE). Secondly, we would perform all models 6.1 to 6.5, and the one with smallest error is used to further predict stock prices.

Time series models includes Model_1, Model_2a, Model_2b, Model_3, Model_4a, and Model_4b which are introduced from Section 6.1 to 6.5. Word embedding models include Model_5a, Model_5b, and Model_5c which are introduced in Section 6.6. Combined models include Model_6, Model_7, Model_8, and Model_9 which are introduced from Section 6.7 to 6.10.

6.1 Support Vector Regression Models (Denoted Model_1)

Experiment based on Support vector regression (SVR) as baseline models from previous researches [77]. Instead of the training error at the minimum, SVR tends to minimize the training error and regularization, and shows potential alternative to stock price forecasting. All the hyperparameters (C, ε, gamma) and kernel function are decided by experiments that result in smallest error.

The input data for model_1 is a time series of closing prices, $P_1\ P_2 ... P_t ... P_n$, for n trading days, and the SVR is applied to learns the pattern of input data by:

$$\widehat{P_t} = f(P_{t-4}, P_{t-3}, P_{t-2}, P_{t-1})$$

where P_t is the real closing price at time *t*, and $\widehat{P_t}$ is the predicted value at time *t* [77]. For better performance, kernel function and hyperparameters are selected by grid search to reach the smallest as possible.

6.2 Unidirectional Long Short-term Memory (Denoted Model_2a)

6.2.1 Single Layer of Long Short-term Memory

Kumar and Ningombam perform long short-term memory to implement technical analysis for stock price forecasting of APPL in 2018 [44]. They only use series of close prices from Jan 1 2013 to May 1 2018. Close prices are rescaled with Min-Max scaler into range of (0,1), which make outlier less effective to model. Their training details include dropout of 0.25, learning rate is 0.001, optimizer is adam with first order moment estimates 0.9 and second order moment estimates 0.999, and epochs of 50. Their output layer activation is linear activation. Given series of t rescaled close prices $P_{(t)}$ and window size *w*, their model is shown as:

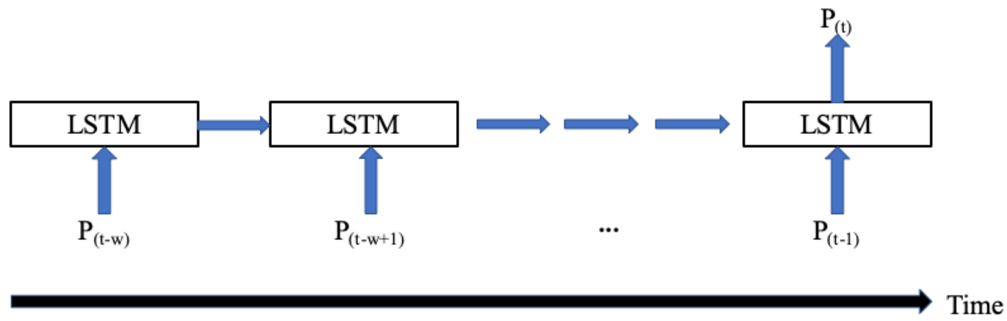

Figure. 6.1 Price Forecasting Model Using LSTM.

We consider their single layer of recurrent neural network as our baseline model, and window size would be selected depending on error.

6.2.2 Multilayer LSTM

The architecture of multilayer LSTM is to stack several LSTM layers. The design is able to capture more complicated patterns of data, but it also increases overfitting risk. We are interested in how many layers of LSTM is better to capture information so that LSMT models with different layers are performed to lead smallest error. The number of layers is from one layer to five layers, and the model with the smallest error is denoted model_2a.

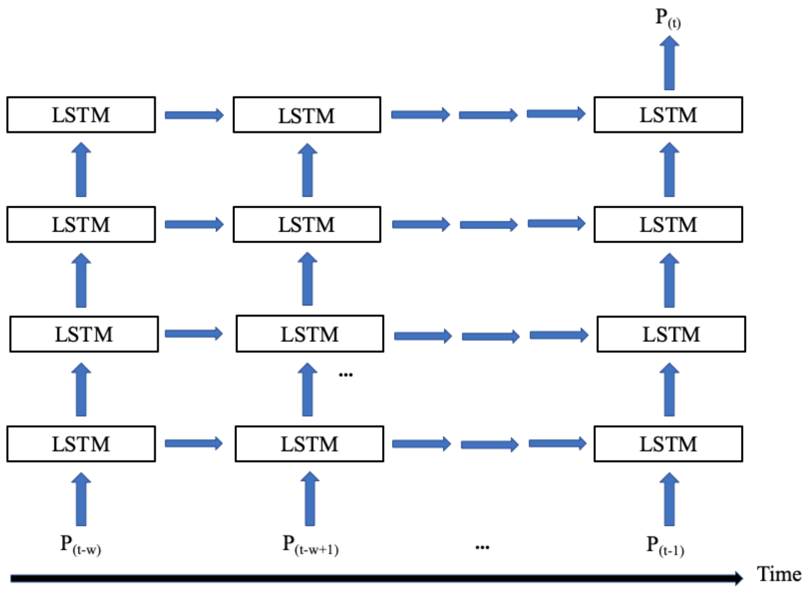

Figure 6.2 Four Layers of Long Short-term Memory

6.3 Bidirectional LSTM (Denoted Model_2b)

According to an analysis of forecasting financial time series by Siami-Namini et al, Bidirectional LSTM results in smaller error than LSTM and ARIMA on NASDAQ index, Nikki 225 index, S&P 500 commodity price index, Dow Jones industrial average index, and IBM stock [69]. Bidirectional LSTM firstly is applied on the input sequence (from past to present) and then on the reverse of input sequence (from present to past) so that it improves the shortcoming of unidirectional LSTM which is moving information from the past. For example, given a series of t daily stock prices and window size *w*, computation of bidirectional LSTM is shown as:

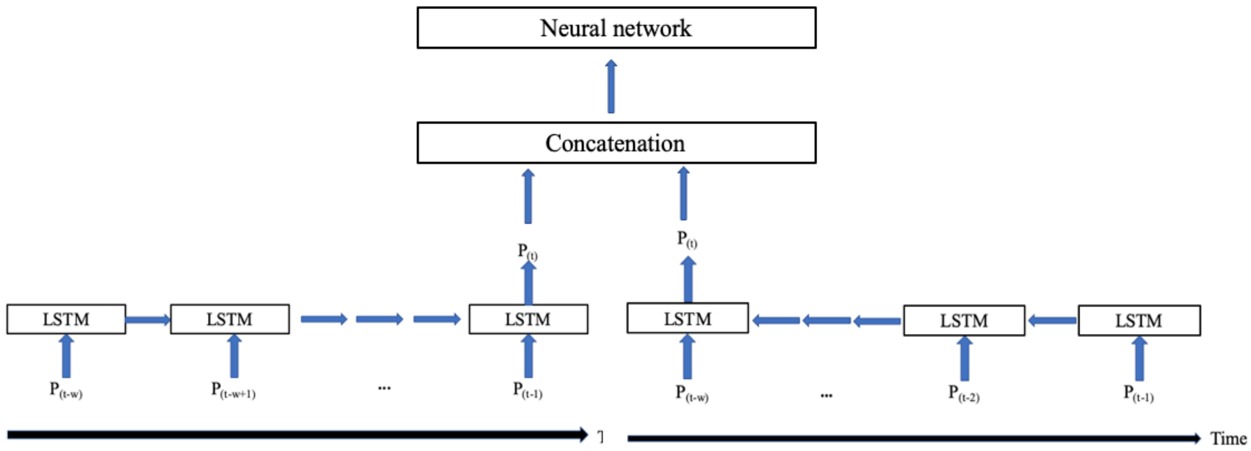

Figure 6.3 Bidirectional Long Short-term Memory.

6.4 Convolutional Neural Network (Denoted Model_3)

The Convolutional neural network (CNN) model is based on the idea that price at a certain time is less affected by the prices a long time ago. Recurrent neural networks must go through prices day by day, but on the contrast, CNN model goes through prices subseries days by subseries days. The baseline CNN model consists of one input layer, two convolutional layers, one pooling layer, and a hidden layer. The size of convolutional filter is 1×7, and pooling size is 1×2, and there are 6 and 12 filters in two convolutional layers respectively [80]. Given n days of time series of prices, the procedure of whole CNN model is shown as:

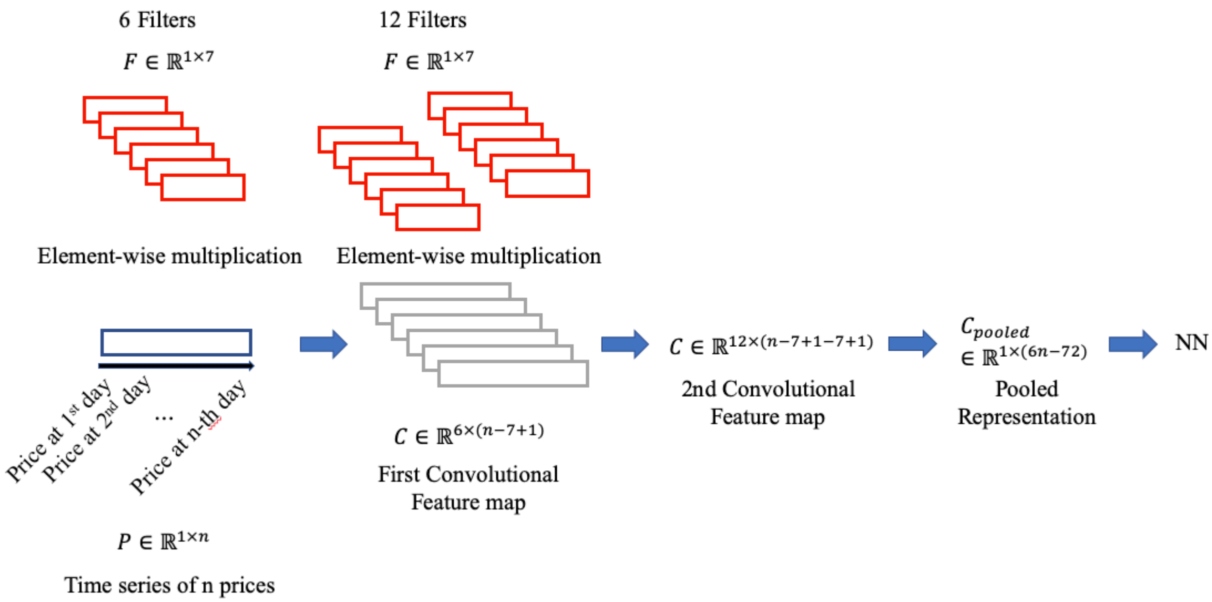

Figure 6.4 Architecture of CNN Model for Time Series Analysis.

6.5 CNN-LSTM (Denoted Model_4a and Model_4b)

Recurrent neural networks may capture sequence pattern information, but they are unable to filter out noise of input. In contrast, convolutional neural networks may filter out noise and extract more valuable feature, but they are designed to cope with spatial data. Therefore, combination of the two neural networks might lead better performance on time series data which is series with noise. Proposed LSTMs with convolutional layer (CNN-LSTM) provide a boost in prediction performance for gold price in 2020 [65]. Raw data is converted into new feature values by element-wise multiplication with kernel in convolutional layers, which aims at filtering out noise. These feature values are subsampled in pooling layer for lower dimension. Their first architecture of CNN-LSTM consists of two convolutional layers of 32 and 64 filters of size (2,), respectively, followed by a max-pooling layer with size (2,), a LSTM layer of 100 units, and a output layer of one neuron,, denoted Model_4a. The second architecture consists of two convolutional layers of 64 and 128 filters of size (2,), respectively, followed by a max-pooling layer with size (2,), a LSTM layer of 200 units, a hidden layer of 32 neurons and a output layer of one neuron, denoted Model_4b.

6.6 Models Based on News Headlines

Inspired by prediction based on sentiment score of news headlines and historical prices, polarity of news headlines is informative so that new headlines vector might also provide useful information. We convert new headlines into vector by pretrained word2vect, BERT and LSTM, and then use news headline vectors as input and actual price as target.

6.6.1 Pretrained Word2vec Model (Denoted Model_5a)

Since our textual datasets for six stock are much smaller than any corpus for training language models, pre-trained language model is preferred to produce vectorial representation of words. Thus, we apply package gensim, pre-trained word2vec model to convert word into vector, and afterward, the series of word vectors are fed into recurrent neural network to extract information (see Figure 6.5). The pre-trained Word2vec bases on amount of the training data including First billion characters form Wikipedia, Latest Wikipedia dump, WMT11 site, dataset from “One Billion Word Language Modeling Benchmark”, UMBC webbase corpus, and text data at statmt.org and in the Polyglot project [91].

Figure 6.5 Price Forecast by Pre-trained Word2vec Model.

6.6.2 Word Embedding Model Based on Our Textual Dataset (Denoted Model_5b)

Unlike usage of pre-training language models, we train our word embedding model based on our textual data, and word vectors are produced by our embedding model.

6.6.3 BERT (Denoted Model_5c)

We apply pre-training language model (BERT-base-uncased) on BookCorpus which consists of 11,038 books and English Wikipedia. It is trained by masked language modeling. Besides lowercase does not make different because of uncased. In other words, there is no difference between Headline and headline. [87,89]. Every news headline is converted into headline embedding with 1×768 of dimension, and then support vector regression with grid search is applied to forecast stock price as shown:

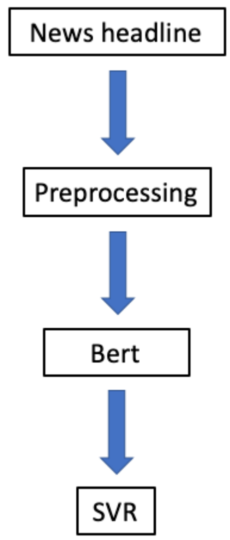

Figure 6.6 Price Forecast by Pre-trained BERT Model.

6.7 Model Based on Historical Prices and Sentiment Score (Denoted Model_6)

Mohan et al combined lexicon-based sentiment analysis and LSTM for stock price prediction [47]. Words of news headline is converted into sentiment score, and then pair of historical price and sentiment score is used to forecast price. Sentiment score is computed as shown:

$$Score_i = (+/-)max(abs(N_i, P_i))$$

$$Score = \frac{1}{k}\sum_{i=1}^{k} Score_i$$

where N_i and P_i are negative and positive values to words in the i-th of k news headlines, and abs is absolute. Afterward, stock price at time t, denoted $Price_{(t)}$, is predicted by pairs ($Price_{(t-1)}, Score_{(t)}$),…,($Price_{(t-m)}, Score_{(t-m-1)}$), where m is window size. Window size m is decided by error.

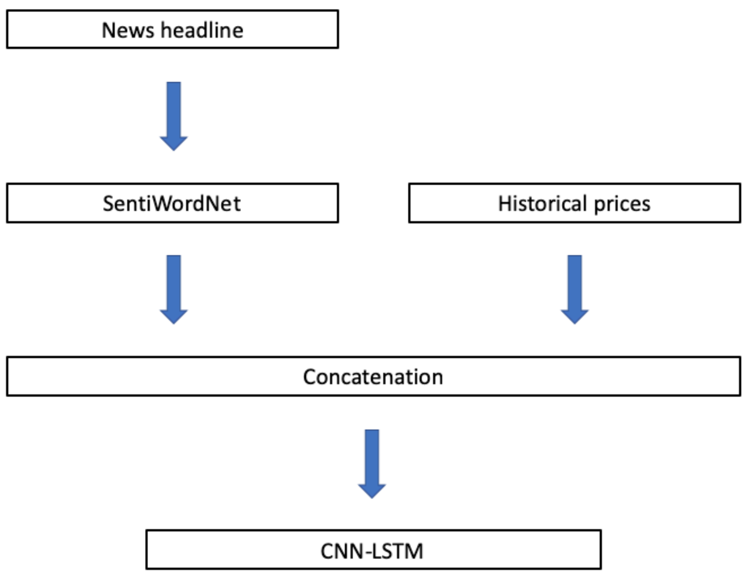

Figure 6.7 Price Forecast Based on Sentiment Score and Historical Prices.

6.8 Model Based on News Headlines and Predicted Prices (Model_7)

Instead of identifying the polarity of news headlines, news headline is processed in Section 6.6 by word embedding model, and then concatenated with predicted prices by time series model for further analysis. There are two situations for word2vec and BERT respectively. In the first situation, headline is converted into headline vector by BERT, concatenated with predicted prices by timeseries model, and then support vector regression is applied for prediction when Model_5c is better than other embedding models (see Figure 6.8). In second situation, headline is converted into a sequence of word vectors, and the converted into headline vector by LSTM. Afterward, the headline vector is concatenated with predicted prices by time series model, and the neural network is applied for prediction, when Model_5a and Model_5b are better than Model_5c (see Figure 6.9).

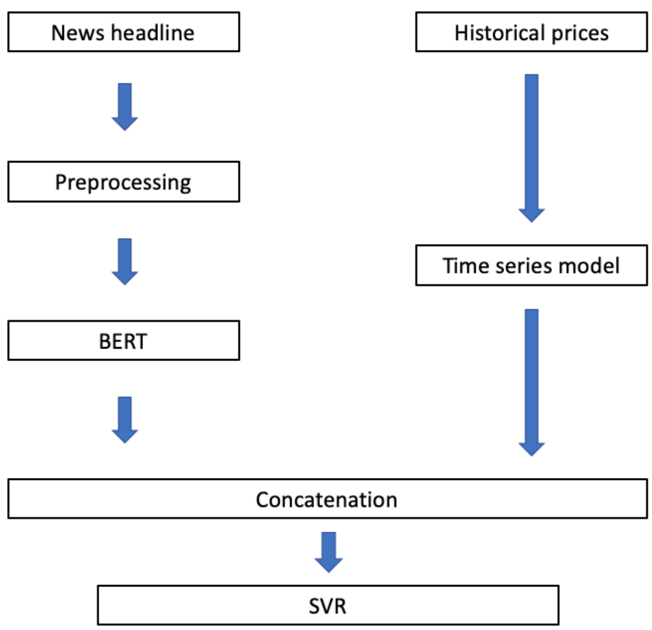

Figure 6.8 Price Forecast Based on BERT and Predicted Prices.

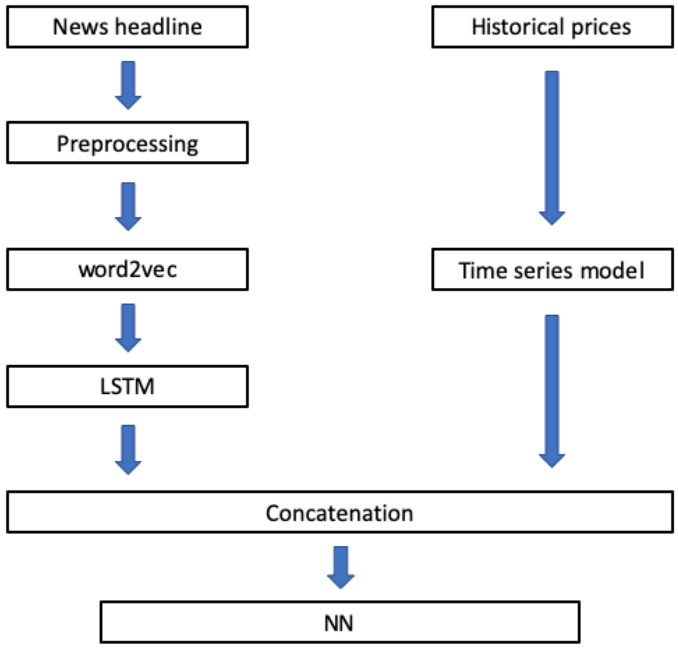

Figure 6.9 Price Forecast Based on Word Embedding and Predicted Prices.

6.9 Model Based on Historical Prices, Sentiment Scores, and Predicted Prices

This model, denoted Model_8 is constructed on the idea of Model_6, which bases on pairs of sentiment score and historical prices. For Model_6, stock price at time *t* is predicted by pairs of historical price at time *t-1* and sentiment score at time *t*. On the contrast, for Model_8, stock price at time *t* is predicted by pairs of historical price at time *t-1*, sentiment score at time *t*, and predicted price at time *t*. The predicted prices base on the BERT language model which is Model_5c in Section 6.6.3. Afterward, convolutional LSTM (CNN-LSTM) is applied for price forecasting.

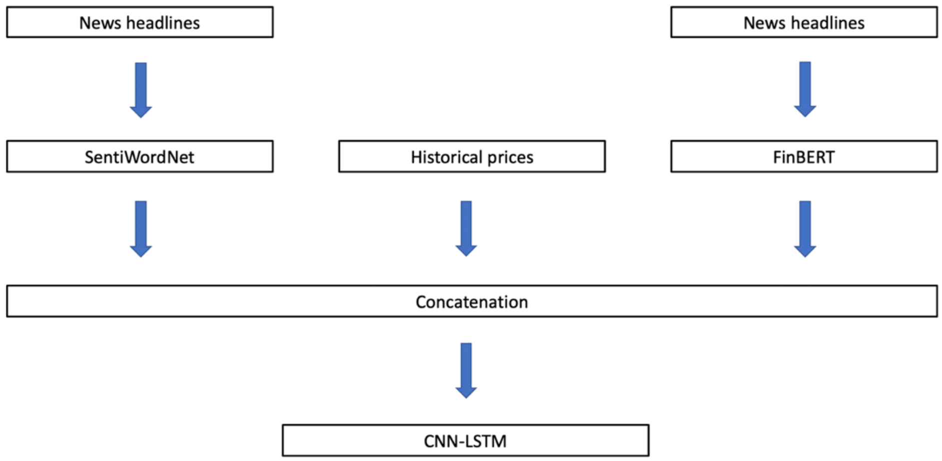

Figure 6.10 Architecture of CNN-LSTM Model.

6.10 Model Based on News Headlines Vector, Sentiment Score, and Predicted Prices

The last model is extension of model_7 which combines sentiment score, headline vector and predicted price, and this model is denoted Model_9. We assume that sentiment scores by lexicon and headline vector should be complementarily informative. When Model_5c is better than Model_5a and Model_5b, headline vector by BERT is concatenated with sentiment score by lexicon and predicted price by time series model, and support vector regression is applied for price forecasting (see Figure 6.11). When Model_5a and Model_5b are better than Model_5c, headline is converted into headline vector by word embedding model and LSTM, the headline vector is further concatenated with sentiment score by lexicon and predicted price by time series model, and then neural network is applied for price prediction (see Figure 6.12).

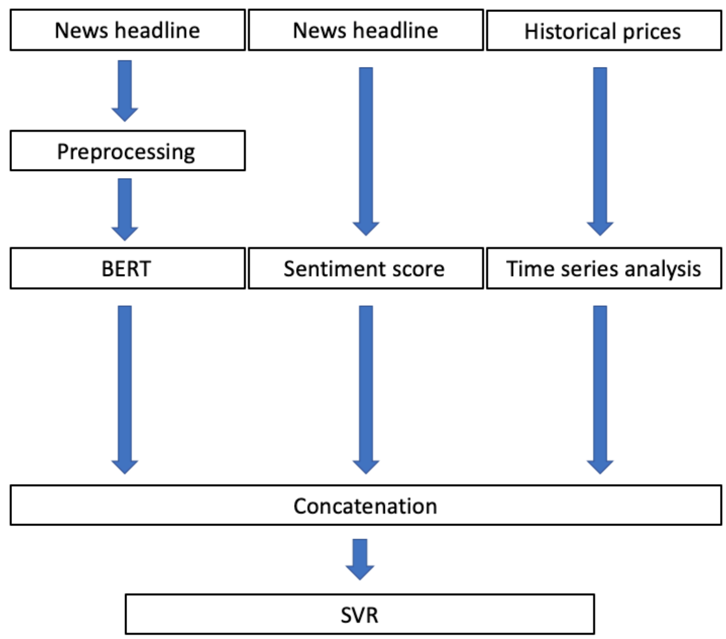

Figure 6.11 Model_9 in Situation 1.

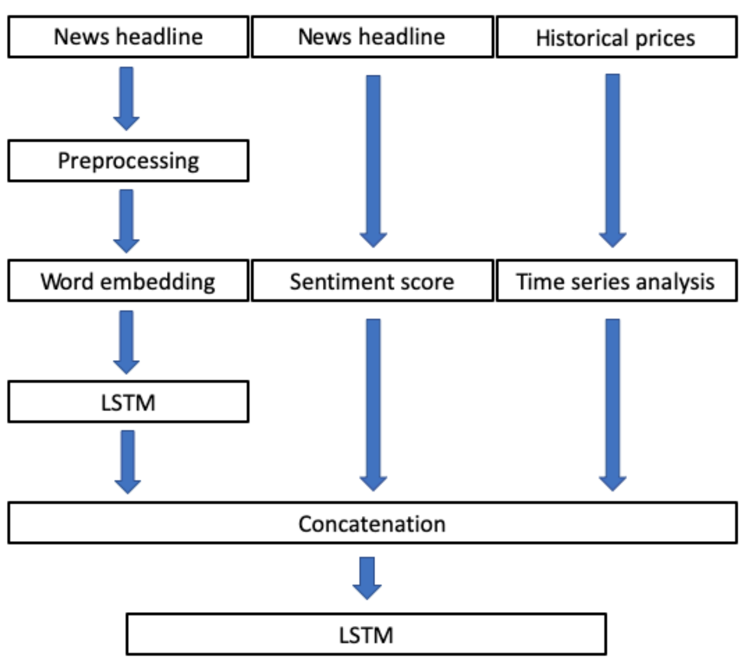

Figure 6.12 Model_9 in Situation 2.

Chapter Seven:

Result and Performance

7.1 Window Size

Window size as an important issue to time series analysis can quite influence the performance of time series performance, and it is greatly determined by the structure of prediction model. Window size is actually related to the difficulty to determine the scaling region and the proper number of sample. For neural network, the optimal time window size depends on dataset and task, which might neglect important information with too small size or might lead overfitting with large window size. Despite ability to learn long-term information, recurrent neural networks still is under overfitting risk by the size of how long-term the window covers. In this study, Grid search is used for finding out proper window size and hyperparameters. For time series analysis, a set of window size (3, 4, 5, 10, 15, 30, 60) is selected from previous researches, and proper window is decided based on smallest MAE, RMSE, and MAPE among models for the six stocks. Besides, data transformation also influences prediction performance, especially for wide range of historical prices so that either logarithm and MinMaxScaler transformations might be applied to rescale dataset. The following tables (see Table 7.1) show the proper window size for the chosen six stocks.

Table 7.1 Better Window Size and Error Measure for Six Stocks.

Stock	Window size	MAPE	MAE	RMSE
BA	3	1.9164	6.7913	9.1339
BAC	10	1.8153	0.5271	0.6446
XOM	3	1.3685	1.0141	1.3214
UBER	3	2.7847	0.8476	1.0531
JNJ	3	1.2789	1.7449	2.3005
AAPL	4	2.0712	1.1166	1.3669

Window size for BA, BAC, XOM, UBER, JNJ, and AAPL are respectively 3, 10, 3, 3, 3, and 4, and the result confirms that window sizes depend on task and data. Generally, window size should be 3 or 4.

7.2 Layers of Long Short-term Memory

Long short-term memory uses information from previous lags and predicts the future price, and stock market is highly dynamic and volatile. Multilayer of long short-term memory might work better than single layer because Multilayer of long short-term memory use information not only from previous lags but also information from previous layers. In other words, multilayer of LSTMs might better recognize pattern from training data. Based on the idea, error should decrease once the number of layer increases, but error would not decrease forever. These experiments show relationship between number of layers of LSTM and prediction performance. For easy comparison, we fixe window size and transformation method and perform different numbers of layers of LSTM.

Table 7.2 Different Number of Layers of LSTM for BA.

Number of layers	MAPE	MAE	RMSE
1	1.9164	6.7913	9.1339
2	2.1505	7.7570	9.6390
3	2.7687	9.7242	12.3683
4	3.7149	13.8079	17.4910
5	4.2101	15.6775	19.6143

Table 7.3 Different Number of Layers of LSTM for BAC.

Number of layers	MAPE	MAE	RMSE
1	1.8153	0.5271	0.6446
2	1.9433	0.5614	0.7038
3	1.6162	0.4621	0.6258
4	1.9229	0.5562	0.7274
5	3.1085	0.9245	1.0910

Table 7.4 Different Number of Layers of LSTM for XOM.

Number of layers	MAPE	MAE	RMSE
1	1.3685	1.0141	1.3214
2	1.4400	1.0639	1.3996
3	1.5632	1.1538	1.5029
4	1.5409	1.1456	1.4766
5	2.2152	1.6344	2.0873

Table 7.5 Different Number of Layers of LSTM for UBER.

Number of layers	MAPE	MAE	RMSE
1	2.7085	0.8476	1.0531
2	2.5714	0.8154	1.0296
3	3.1322	1.0086	1.2712
4	3.1210	0.9984	1.2493
5	3.2270	1.0315	1.2879

Table 7.6 Different Number of Layers of LSTM for JNJ.

Number of layers	MAPE	MAE	RMSE
1	1.2789	1.7449	2.3005
2	1.6781	2.3002	2.7912
3	1.3550	1.85513	2.4729
4	1.5462	2.1180	2.7180
5	1.2921	1.7447	2.4921

Table 7.7 Different Number of Layers of LSTM for AAPL.

Number of layers	MAPE	MAE	RMSE
1	2.0712	1.1166	1.3669
2	3.6187	1.9810	2.3459
3	2.4294	1.2966	1.7728
4	4.6721	2.7262	3.8571
5	5.7976	3.4918	5.4482

With fixed window sizes respectively, numbers of layers of LSTM to better prediction performance for BA, BAC, XOM, UBER, JNJ, and AAPL are 1, 3, 1, 2, 1, and 1 of layers. Only two stocks out of six have better performance with increasing layers of LSTM.

7.3 Predicted Prices for Further Experiments

Our proposed approaches require combination of time series analysis and sentiment analysis, and the way is to use time series model to produce predicted prices and then concatenation of predicted prices with further information such as sentiment scores and sentence embedding vector. The following tables show part of the predicted prices by the time series models we mentioned with smallest error, and BERT model (Model_5c) which is used as new input in Model_8. Split rate to training dataset and test set is 80/20 but number of predicted prices is different. UBER, for instance, has IPO after 2019 so there is only 174 trading days of prices into 140 training set and 34 test set (predicted prices).

Table 7.8 Predicted Prices for BA

By time series model	Actual prices	Predicted prices by BERT
55.90502	53.619999	94.45471746
57.016468	49.549999	94.45471746
56.136967	45.720001	94.45471746
53.562096	46.580002	94.45471746
50.71529	46.139999	94.45471746
49.383514	43.970001	94.45471746
48.83146	42.52	94.45471746
47.318375	43.16	94.45471746
46.094635	41.040001	94.45471746
45.06647	41.18	94.45471746
44.53123	39.560001	94.45471746
43.205296	37.48	94.45471746
41.93259	37.110001	94.45471746
40.34494	39.580002	94.45471746

Table 7.8 (Continued)

By time series mode	Actual prices	Predicted prices by BERT
40.189342	40.75	94.45471746
41.40405	40.18	94.45471746
42.660007	41.279999	94.45471746
43.26565	42.630001	94.45471746
43.92046	39.880001	94.45471746

Table 7.9 Predicted Prices for BAC

By time series mode	Actual prices	Predicted prices by BERT
23.653166	22.66	13.3624329
22.76592	23	13.3624329
22.677708	21.07	13.3624329
21.312956	20.530001	13.3624329
20.420527	23.02	13.3624329
22.08265	22.32	13.3624329
22.445232	22.780001	13.3624329
22.777256	24.17	13.2389584
23.805891	23.610001	13.3624329
23.683867	24.530001	14.0655187
24.147396	21.75	13.7834306
22.191664	20.120001	13.3624329
20.152433	20.49	13.3624329
19.860502	19.48	13.3624329
19.378168	18.690001	14.757032
18.794096	17	13.3624329
17.45471	17.1	13.3624329
17.031914	16.42	13.4661974
16.52534	15.03	13.9857656

Table 7.10 Predicted Prices for XOM

By time series mode	Actual prices	Predicted prices by BERT
68.75148	69.230003	83.238681
68.92249	69.349998	83.238681
69.122444	71.760002	83.238681
70.883804	72.919998	83.238681
72.3832	72.080002	83.238681
72.402885	72.980003	83.238681
72.679146	72.970001	83.238681
72.83158	73.169998	83.238681
73.08029	73.120003	83.238681
73.06102	73.839996	83.238681
73.53398	74.050003	83.238681
73.89136	73.779999	83.238681
73.843956	72.809998	83.238681
73.09559	71.629997	83.238681
71.97472	71.419998	83.238681
71.391464	71.440002	83.238681
71.294586	71.050003	83.238681
71.095604	68.839996	83.238681
69.54818	68.949997	83.6726548

Table 7.11 Predicted Prices for UBER

By time series mode	Actual prices	Predicted prices by BERT
41.42359	41.91	39.8690397
41.983772	41.59	39.5427463
41.88908	41.5	39.4716121
41.444958	41.25	39.2139419
41.22674	40.470001	38.4278436
40.796894	41.509998	39.4741452
40.852325	40.950001	38.9102186
40.817997	39.939999	37.9004557

Table 7.11 (Continued)

By time series mode	Actual prices	Predicted prices by BERT
40.47547	39.799999	37.7555056
39.895496	40.41	38.3795847
39.822906	41.25	39.2064611
40.34874	42.75	40.7065307
41.425518	45	42.9667674
43.133774	44.919998	42.8781939
44.370842	44.16	42.1181574
44.628384	42.610001	40.5690943
43.675083	42.450001	40.4125572
42.81097	42.169998	40.133007

Table 7.12 Predicted Prices for JNJ

By time series mode l	Actual prices	Predicted prices by BERT
68.14055	68.739998	102.078387
68.58469	69.519997	102.078387
69.37894	69.449997	102.078387
69.85374	69.220001	102.078387
69.88141	69.379997	102.078387
69.85879	68.449997	102.078387
69.46047	69.120003	102.078387
69.46586	68.839996	102.078387
69.39138	68.290001	102.078387
69.18478	68.349998	102.078387
68.966965	68.32	102.078387
68.866714	68.639999	102.078387
69.00861	68.459999	102.078387
69.03386	68.639999	102.078387
69.11361	68.349998	102.078387
68.38765	67.779999	102.078387
69.01299	68.199997	102.078387

Table 7.12 (Continued)

By time series mode l	Actual prices	Predicted prices by BERT
68.8826	67.800003	102.078387
68.59859	67.699997	102.078387

Table 7.13 Predicted Prices for AAPL

By time series mode	Actual prices	Predicted prices by BERT
21.506311	21.582144	18.546224
21.452919	21.565357	17.0180009
21.369572	21.461428	18.0456217
20.965223	20.534643	18.0456217
20.665445	20.531429	17.2525287
20.583292	20.898571	18.0632034
20.703457	21.251072	20.609269
21.084103	21.812857	18.0456217
21.32057	21.671785	19.6596934
21.453775	21.706785	18.0456217
21.601538	21.989286	18.0456217
21.762096	22.233929	19.1443206
21.878563	22.175358	18.0456217
21.934334	22.137857	18.0456217
21.951183	22.168928	17.6171127
21.957548	22.203571	18.0456217
22.074627	22.5	19.7312202

Predictions based on time series models are likely either under or above actual prices. Besides, word embedding model likely produce the same or volatile prediction, and it might be explained by news occurrence.

7.4 Performance Comparison Among Models

7.4.1 Models Summary

Table 7.14 Models Summary.

Symbol	Description	Input data
Model_1	Support vector regression	Historical prices
Model_2a	Multilayer LSTM	Historical prices
Model_2b	Bidirectional LSTM	Historical prices
Model_3	CNN	Historical prices
Model_4_a	CNN-LSTM architecture 1	Historical prices
Model_4_b	CNN-LSTM architecture 2	Historical prices
Model_5_a	Pre-trained word2vec and LSTM	News headlines
Model_5_b	Self-trained word2vec and LSTM	News headlines
Model_5_c	BERT and SVR	News headlines
Model_6	SentiWordNet & CNN-LSTM	Sentiment scores & historical prices
Model_7	Word embedding model	News vector & predicted prices
Model_8	Combined model of 6.9	Historical prices, sentiment scores & predicted prices
Model_9	Combined model of 6.10	Headline vectors, sentiment scores & predicted prices

All models are introduced in Chapter 6, and comparisons of models based on stocks are shown for better presentation. Since we want to boost prediction performance by combination of two analysis, the results of time series models are firstly showed, and then results by three different word embedding models would be shown for relation between news headlines and stock prices. Furthermore, results by combined models would lastly be presented.

Table 7.14 show description and input data for each model. Model_1, Model_2a, Model_2b, Model_3, Model_4a, and Model_4b are time series analysis based on different approaches, which are support vector regression, multilayer of LSTM, bidirectional LSTM, convolutional neural network, and convolutional LSTM respectively with historical prices. Besides, Model_5a, Model_5b, and Model_5c base on Pre-trained word2vec, self-trained word embedding, and BERT to exploit the pattern between news headlines and prices. Model_6 bases on SentiWordNet, a lexicon-based scaling system, and convolutional LSTM to exploit sentiment of news headlines and historical prices. Model_7 is our focus on information of textual representation vectors and predicted prices by time series model. Model_8 is a model combining sentiment score of news headlines, historical prices, and predicted prices of BERT. Model_9 states the potential based on headline vectors, sentiment scores, and predicted prices by time series model.

Since error measures sometimes does not look ag good as small values, we also use graphs for our decision-making. In each section, we would firstly present one graph by all models, followed by an table of error measure and two graphs which are best models among time series models, word embedding models, and combined models, and best model with smallest error.

7.4.2 The Boeing Company (BA)

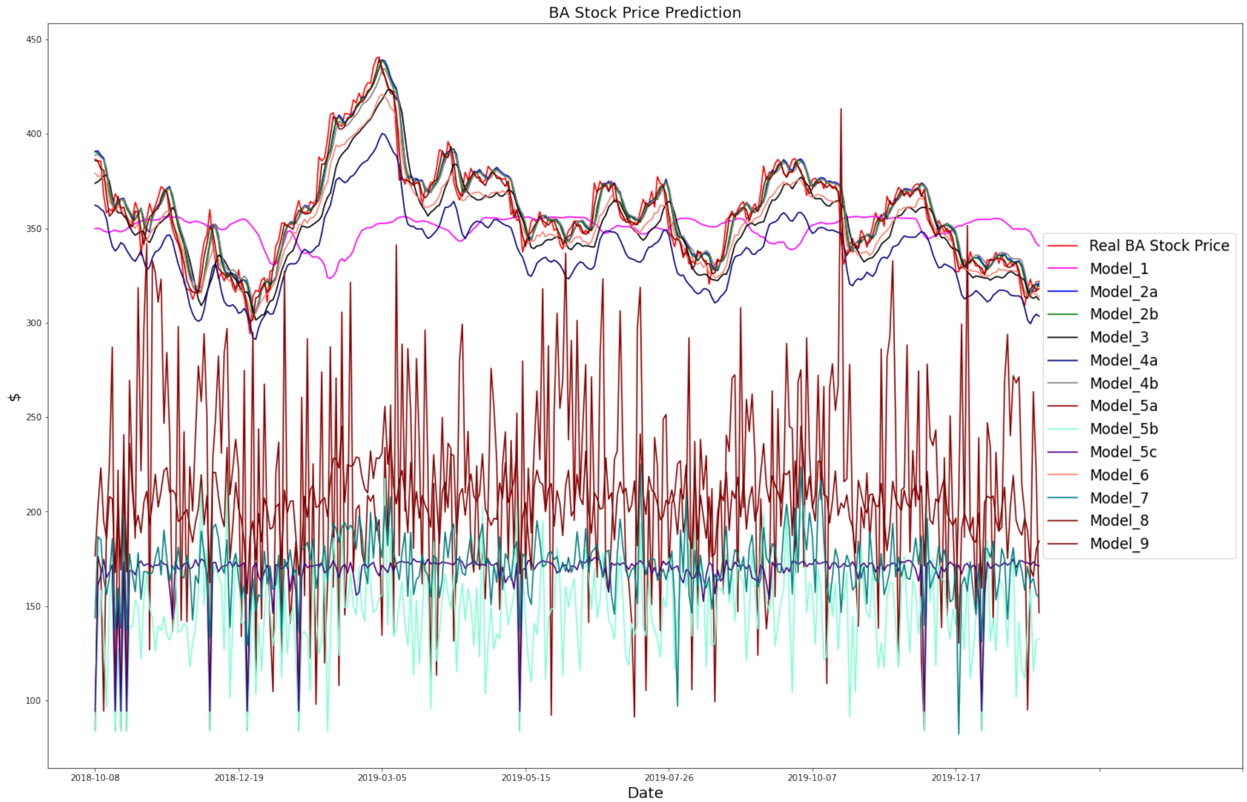

Figure 7.1 BA: Stock Prices Plotted on Zero Axis.

Table 7.15 Performance Comparison Based on MAPE, MAE and RSME for BA

Model	MAPE	MAE	RMSE
Model_1	6.1905	22.7480	29.7917
Model_2a	1.9157	6.7867	9.1399
Model_2b	1.7766	6.2972	8.4879
Model_3	3.2922	11.9305	14.2742
Model_4a	5.9260	21.6058	23.5110
Model_4b	2.0090	7.1192	9.5257
Model_5a	40.2427	145.4781	160.0848
Model_5b	58.9057	211.8250	214.5503
Model_5c	53.0341	191.0317	193.2290
Model_6	2.3790	8.6836	10.4685

Table 7.15 (Continued)

Model	MAPE	MAE	RMSE
Model_7	52.1324	187.4539	128.1867
Model_8	**0.9521**	**3.3952**	**4.5568**
Model_9	42.9119	154.4238	156.1739

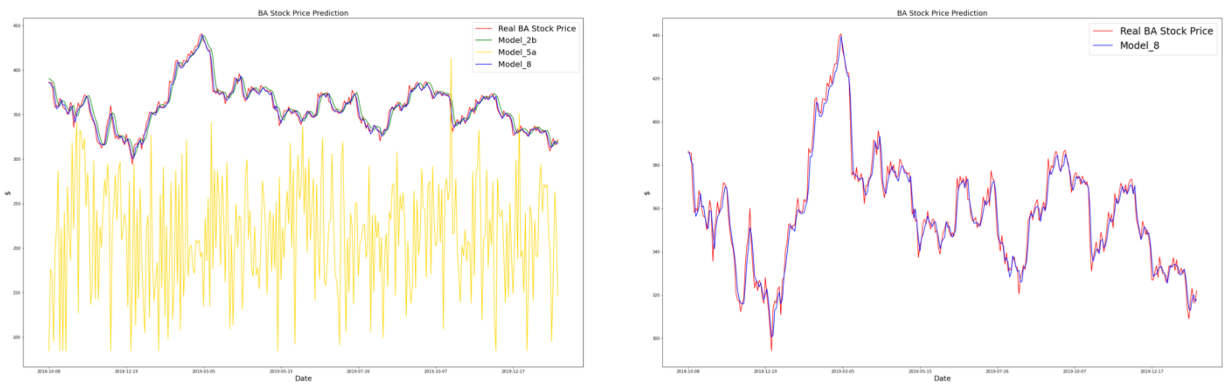

Figure 7.2 Left: Three Better Models. Right: Our Best Model.

Among the time series models (Model_1, Model_2a, Model_2b, Model_3, Model_4a, and Model_4b), Bidirectional LSTM (Model_2b) produces smallest error which are MAPE of 1.7766, MAE of 6.2972, and RMSE of 8.4879. Among word embedding models (Model_5a, Model_5b, and Model_5c), it looks like None of embedding models is able to catch pattern and predictions based on embedding model are unreliable. However, compared with Model_6 which consists of CNN-LSTM with sentiment score of news headline, and historical prices, Model_8 shows an improvement in 1.4269 of decrease in MAPE, 5.2884 of decrease in MAE, and 5.9117 of decrease in RMSE, which means that predicted prices by BERT are informative. From Figure

7.2, regression line by Model_2b resembles moving average line which shows lag between regression line by Model_2b and actual prices. Combined model Model_8 shows less lag than Model_2b with similarly small errors. With sentiment scores as new input feature, error decreases from Model_7 to Model_9, but combined model_7 and Model_9 are still less reliable than Model_6 and Model_8.

7.4.3 Bank of America Corporation (BAC)

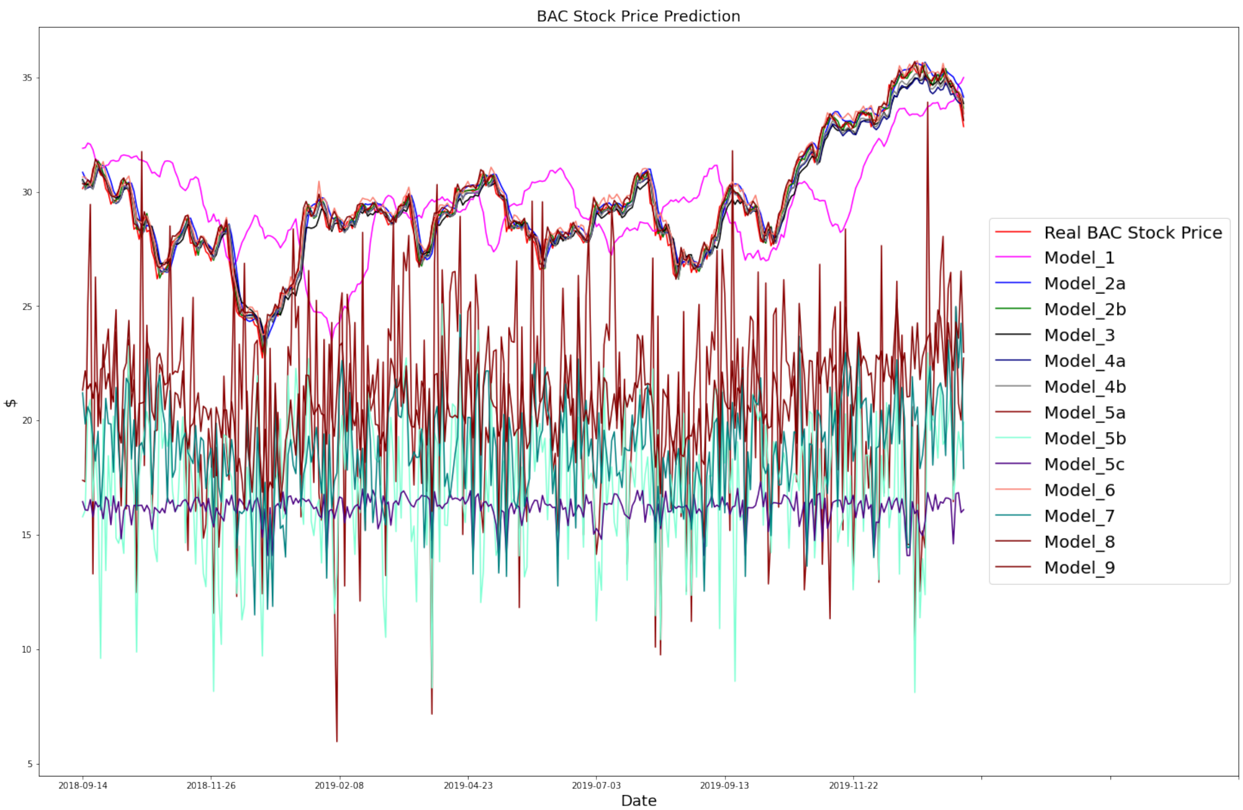

Figure 7.3 BAC: Stock Prices Plotted on Zero Axis.

Table 7.16 Performance Comparison Based on MAPE, MAE and RSME for BAC

Model	MAPE	MAE	RMSE
Model_1	7.0236	2.0168	2.3744
Model_2a	1.5338	0.4401	0.5925
Model_2b	1.1405	0.3282	0.4383
Model_3	1.5294	0.4421	0.5643
Model_4a	1.3321	0.3877	0.5016
Model_4b	1.2309	0.3564	0.4687
Model_5a	29.8403	8.9151	10.2604
Model_5b	41.3456	12.2594	12.8196
Model_5c	44.6574	13.2564	13.5243
Model_6	1.2356	0.3558	0.4437
Model_7	36.1119	10.6882	11.0415
Model_8	**0.6809**	**0.1956**	**0.2397**
Model_9	28.6205	8.4836	8.7312

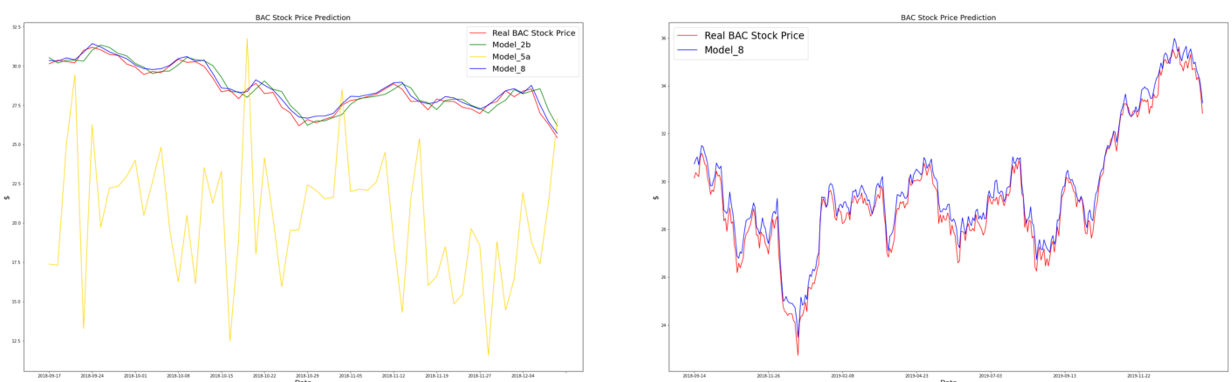

Figure 7.4 Left: Three Better Models. Right: Our Best Model.

Among the time series models, bidirectional LSTM gives smallest errors which are MAPE of 1.1405, MAE of 0.3282, and RMSE of 0.4383 than others. None of embedding models seem functional because of smallest MAPE by embedding models is 29.8403. In other words, stock price predictions based on embedding models are volatile. Besides, combined models, Model_6 and Model_8, produce similarly small errors as time series model. Especially, errors by Model_8 are smaller than by bidirectional LSTM, which also shows more reliability and less lag in Figure 7.4. Despite smaller error by Model_9 than by Model_7, Model_5a outperforms Model_7 and Model_9, which is not reasonable. The possible reason could be the way to combine headline vector and predicted prices.

7.4.4 Exxon Mobile Corporation (XOM)

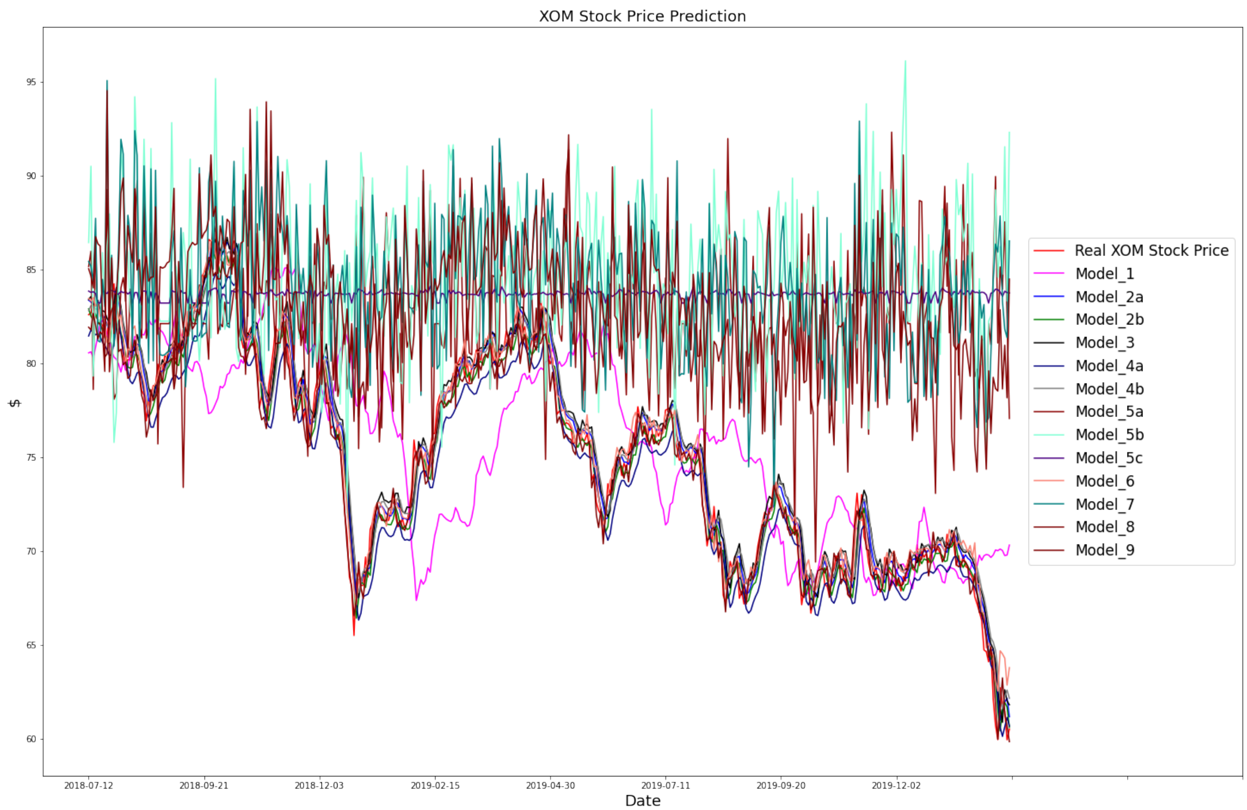

Figure 7.5 XOM: Stock Prices Plotted on Zero Axis.

Table 7.17 Performance Comparison Based on MAPE, MAE and RSME for XOM

Model	**MAPE**	**MAE**	**RMSE**
Model_1	5.0748	3.7610	4.6909
Model_2a	1.2073	0.8922	1.1568
Model_2b	1.0858	0.8098	1.0289
Model_3	1.4269	1.0513	1.3528
Model_4a	1.7493	1.3182	1.5955
Model_4b	1.4663	1.0785	1.3994
Model_5a	11.1683	7.9629	9.8078
Model_5b	13.9731	10.0166	11.8135
Model_5c	12.3759	8.8043	10.3865
Model_6	0.9327	0.6789	0.8942
Model_7	12.9858	9.3518	10.7769
Model_8	**0.6887**	**0.5212**	**0.6407**
Model_9	11.6688	8.4274	9.5500

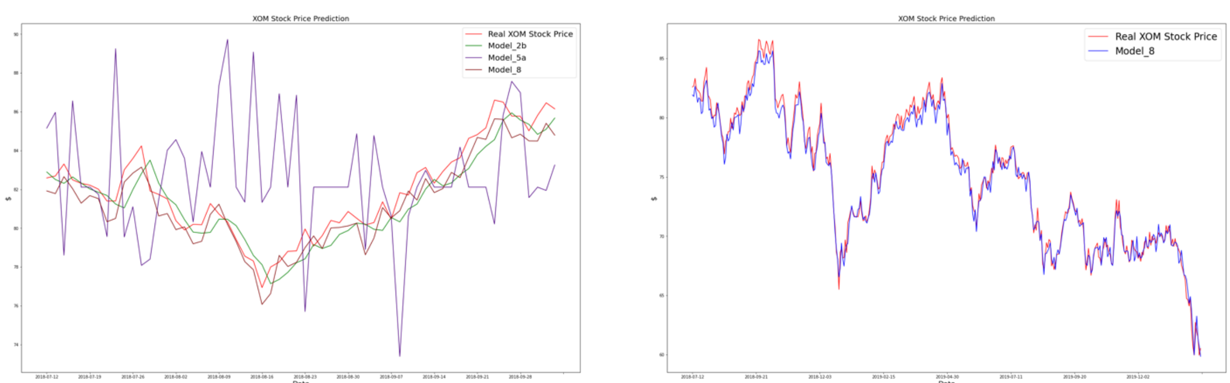

Figure 7.6 Left: Three Better Models. Right: Our Best Model.

Overall, time series models give great performance excluding support vector regression model, and Bidirectional LSTM leads smallest error than other time series models, which is MAPE of 1.0858, MAE of 0.8098, and RMSE of 1.0289. All of word embedding models look much more functional than in BA, BAC cases, which give rough MAPE of 11 under conditions without news during 1740 out of 2701 days, With such informatively numerical and textual data, there is no doubt that Model_6 and Model_8 show small MAPE. Besides, Model_7 and Model_9 seem no improvement to embedding model, and reason could be word2vec might not extract important information.

7.4.5 Uber Technologies, Inc. (UBER)

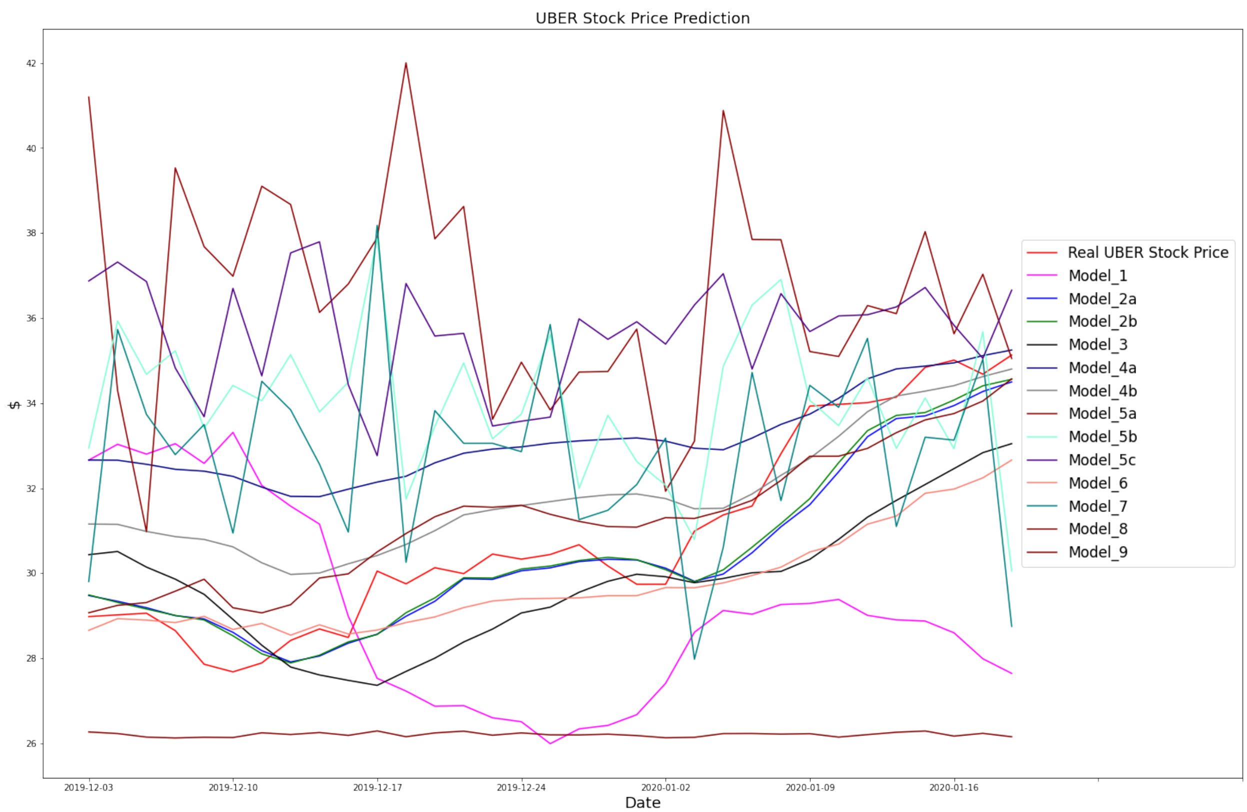

Figure 7.7 UBER: Stock Prices Plotted on Zero Axis.

Table 7.18 Performance Comparison Based on MAPE, MAE and RSME for UBER

Model	MAPE	MAE	RMSE
Model_1	13.0715	4.1223	4.4782
Model_2a	2.5365	0.8111	1.0336
Model_2b	**2.3639**	**0.7551**	**0.9751**
Model_3	5.3941	1.7214	1.9704
Model_4a	7.7479	2.2953	2.6787
Model_4b	4.2463	1.2725	1.5279
Model_5a	19.3778	5.8230	6.8904
Model_5b	12.9310	3.9197	4.6411
Model_5c	16.0682	4.8024	5.3988
Model_6	4.1442	1.3402	1.7054
Model_7	10.6323	3.2549	4.0323
Model_8	3.0765	0.9387	1.0554
Model_9	15.1156	4.8640	5.4916

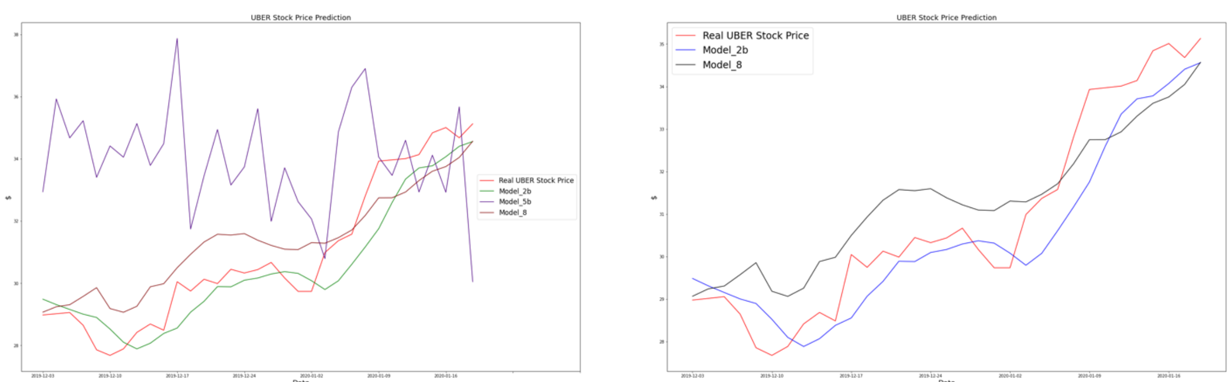

Figure 7.8 Left: Three Better Models. Right: Our best two Models.

Prediction for Uber price is much worse than for other stocks, all models for UBER give much larger errors than for other stocks. Small sample might cause the result. There are 140 days for training and 34 for test. Bidirectional LSTM still leads to smallest errors which is MAPE of

2.3639 among time series models. Surprisingly, embedding models for UBER give small error, and self-trained embedding model produces smallest errors among embedding models. The reason might be quality and quantity of news headlines. Besides, Model_8 is able to give similar small error with Model_2b. Despite larger error, Model_8 shows less lag than Model_2b in Figure 7.8. Model_9 shows no improvement after adding sentiment scores as new feature, and reason could be small sample size and stochastic characteristic of neural network.

7.4.6 Johnson & Johnson (JNJ)

Figure 7.9 JNJ: Stock prices plotted on zero axis.

Table 7.19 Performance Comparison Based on MAPE, MAE and RSME for JNJ

Model	MAPE	MAE	RMSE
Model_1	5.5753	7.6714	8.6684
Model_2a	1.0044	1.3571	2.0105
Model_2b	0.9566	1.2965	1.8897
Model_3	1.4535	1.9818	2.6458
Model_4a	1.6421	2.2583	2.7734
Model_4b	1.3241	1.7789	2.5504
Model_5a	19.7436	27.1313	29.7720
Model_5b	20.1212	27.6287	30.1566
Model_5c	18.9405	26.0472	28.2778
Model_6	1.0636	1.4393	2.0482
Model_7	3.6074	4.9771	5.3382
Model_8	**0.7580**	**1.0371**	**1.3188**
Model_9	3.6029	4.9709	5.3326

Figure 7.10 Left: Three Better Models. Right: Our Best Model.

Model_2b based on bidirectional LSTM shows better performance among time series models again, and unidirectional LSTM also gives similar errors. Since there are days without news, embedding models produce large errors and are less reliable than time series models and combined models. Model_6 as combined model with sentiment score and historical price leads to similar performance with time series models based on neural networks, and Model_8 produces smallest errors among all models. Since Model_5c outperforms Model_5a and Model_5b, support vector regression is applied for prediction. Comparison of Model_7 and Model_9, it seems that headline vector by BERT is more informative than by word2vec and LSTM.

7.4.7 Apple Inc. (AAPL)

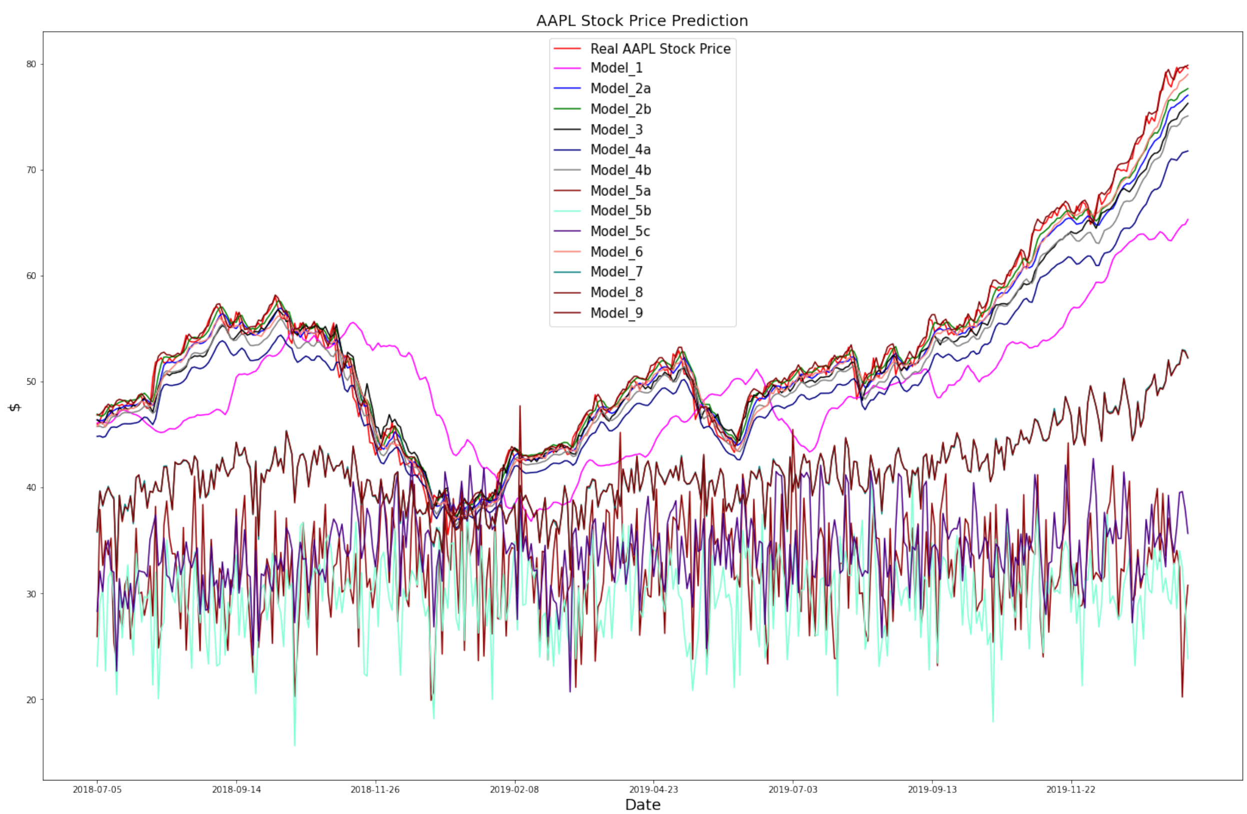

Figure 7.11 AAPL: Stock Prices Plotted on Zero Axis.

Table 7.20 Performance Comparison Based on MAPE, MAE and RSME for AAPL

Model	MAPE	MAE	RMSE
Model_1	10.6527	5.6437	6.5346
Model_2a	2.0027	1.0736	1.3920
Model_2b	1.6852	0.8788	1.1741
Model_3	2.4920	1.2638	1.7805
Model_4a	4.9625	2.7769	3.3100
Model_4b	3.0298	1.6806	2.0872
Model_5a	36.7100	20.1947	22.5825
Model_5b	41.9674	22.8933	24.9113
Model_5c	33.1868	18.4446	20.8859
Model_6	1.8301	0.9519	1.1928
Model_7	20.0284	11.2110	12.8617
Model_8	**1.0405**	**0.5222**	**0.6774**
Model_9	20.0526	11.2261	12.8853

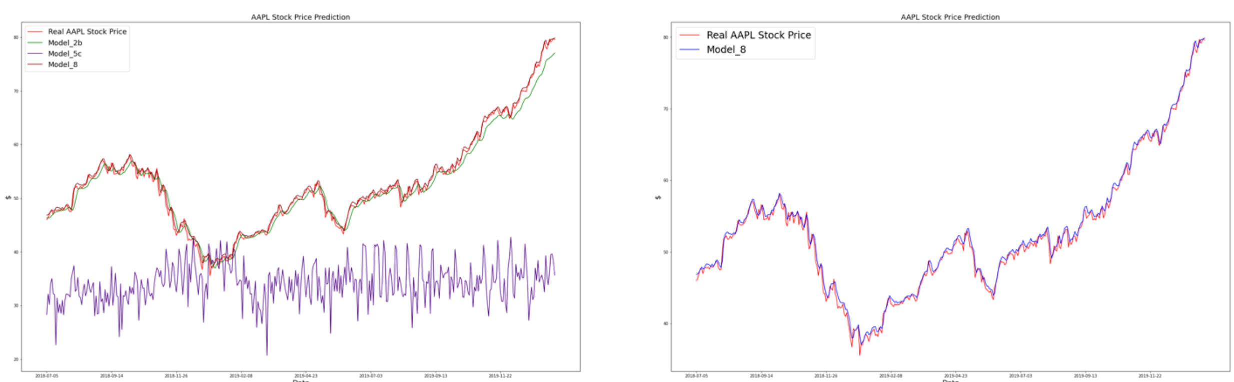

Figure 7.12 Left: Closer Perspective of Multiple Models. Right: Our best Model.

Bidirectional LSTM outperforms other time series models for every stock. Quality of news headlines might play an important role in analysis because there are only 158 days without news out of 1894 days and errors for embedding models are larger than 30 which means there is average 30% difference between predictions and real prices. Comparison between Model_7 and Model_9, sentiment scores are not informative with headline vector by BERT. Model_8 again gives smallest MAPE, MAE and RMSE.

Chapter Eight:

Conclusion and Discussion

Table 8.1 The Best Model in Time Series, Embedding, and Combined Models.

Ticker	Model	MAPE	MAE	RMSE
BA	Model_2b	1.7766	6.2972	8.4879
	Model_5a	40.2427	145.4781	160.0848
	Model_8	0.9521	3.3952	4.5568
BAC	Model_2b	1.1405	0.3282	0.4383
	Model_5a	29.8403	8.9151	10.2604
	Model_8	0.5406	0.1556	0.1955
XOM	Model_2b	1.0858	0.8098	1.0289
	Model_5a	11.1683	7.9629	9.8078
	Model_8	0.6887	0.5212	0.6407
UBER	Model_2b	2.3639	0.7551	0.9751
	Model_5b	12.9310	3.9197	4.6411
	Model8	3.0765	0.9387	1.0554
JNJ	Model_2b	0.9566	1.2965	1.8897
	Model_5c	18.9405	26.0472	28.2778
	Model_8	0.7580	1.0371	1.3188
AAPL	Model_2b	1.6852	0.8788	1.1741
	Model_5c	33.1868	18.4446	20.8859
	Model_8	1.0405	0.5222	0.6774

This thesis focuses on stock price prediction by combination of different models including sentiment analysis, word embedding model, and time series models to better analyze time series data in financial field. Time series models are able to grab the periodic status but its regression line always resembles moving average line and always shows lag. Sentiment analysis is able to extract polarity of given text and can be used for sudden and short-term influence.

In Chapter seven, we predict stock prices using time series models, word embedding models, and combined models. The results show the advantages and disadvantages of each model. Time series models are able to performance prediction with small error but it is less helpful to time series data in finance field because predictions by time series models are likely either higher than actual prices or lower than actual prices. On the graphs, regression line by time series data resembles a smooth move average line, which is lagging. Besides lexicon-based scaling system and word embedding models are able to extract information of given text, which is reaction to textual data such as news. Our proposed approach is complemented by these models, and it is likely to keep low error like time series models and sensitive to news like sentiment analysis approaches. The results in Chapter seven suggests potential combination of time series models and sentiment analysis approaches.

Despite stochastic characteristic of neural networks, bidirectional LSTM (Model_2b) outperforms the other time series models in 6 out of 6 stocks, but on the figures, it does not look like models with low error. Prediction based on word embedding models is unreliable because of high error so embedding models is not helpful to stock prediction but word embedding is able to extract information from words. Our proposed model Model_8 show the potential not only to keep as lower error as time series models and but also to extract information from words. In 5 out of 6 chosen stocks, Model_8 shows beneficial performance, and also show sensitivity on graph. Regression line by Model_8 shows less lag than other models.

References

[1]. Bharathi, S., & Geetha, Angelina. (2017). Sentiment Analysis for Effective Stock Market Prediction. International Journal of Intelligent Engineering and System. (Vol. 10, No. 3).

[2]. Pai, P.-F., & Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. The International Journal of Management Science. 497-505.

[3]. Adebiyi, A. A., & Adewumi, O. A. (2014). Stock Price Prediction Using the ARIMA Model. UKSim-AMSS 16^{th} International Conference on Computer Modeling and Simulation.

[4]. Huynh, H.D., Dang, L.M., & Duong, D. (2017). A New Model for Stock Price Movements Prediction Using Deep Neural Network. SoICT 2017: Proceedings of the Eighth International Symposium on Information and Communication Technology. 57-62.

[5]. Abdi, A., Shamsuddin, S. M., Hasan, S., & Piran, J. (2019). Deep learning-based sentiment classification of evaluative text based on Multi-feature fusion. Information Processing & Management, 56(4), 1245-1259. doi:10.1016/j.ipm.2019.02.018

[6]. Fox, C. (1989). A stop list for general text. ACM SIGIR Forum, 24(1-2), 19–21.

[7]. Ignatow, G., & Mihalcea, R. (2018). An Introduction to Text Ming Research Design, Data Collection, and Analysis. SAGE Publications India Pvt. Ltd.

[8]. Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3), 130–137.

[9]. Fellbaum, C. (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[10]. Turney, P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, (2002), Philadelphia, Pennsylvania, 417-424

[11]. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 **[cs.CL]**

[12]. Géron, A. (2019). Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo: O'Reilly.

[13]. Khan, W., Ghazanfar, M.A., Azam, M.A., Karami, A., Alyoubi, K.H., & Alfakeeh, A.S. (2020). Stock Market Prediction using Machine Learning Classifiers and Social Media, News. Journal of Ambient Intelligence and Humanized Computing.

[14]. Kaur, G., & Bajaj, K. (2016). News Classification and Its Techniques: A Review. IOSR Journal of Computer Engineering. 18(1), 22-26. DOI: 10.9790/0661-18132226.

[15]. Fredriksen, V., Jahren, B., & Gambäck, B. (2018). Utilizing Large Twitter Corpora to Create Sentiment Lexica. In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 2829–2836, Miyazaki, Japan. ELRA.

[16]. Goldberg, Y., & Levy, O. (2014). Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. arXiv:1402.3722 **[cs.CL]**

[17]. Stock market quotes & financial news. (n.d.). Retrieved February 06, 2021, from https://www.investing.com/

[18]. Yahoo finance - stock Market Live, Quotes, business & finance news. (n.d.). Retrieved February 06, 2021, from https://finance.yahoo.com/

[19]. Understanding lstm networks. (n.d.). Retrieved February 07, 2021, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[20]. NLP text preprocessing: A practical guild and template. (Weng, Jiahao). Retrieved February 07, 2021, from https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79

[21]. Dr. S. Vijayarani et al , International Journal of Computer Science & Communication Networks, Vol 5(1),7-16

[22]. Camacho-Collados, J., & Pilehvar, M. T. (2018). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46

[23]. Ghag, K.V., & Shah, K. (2015). Comparative Analysis of Effect of Stopwords Removal on Sentiment Classification. IEEE international Conference on Computer, communication and Control (IC4-2015).

[24]. Sentiment analysis. (2021, February 03). Retrieved February 07, 2021, from https://en.wikipedia.org/wiki/Sentiment_analysis

[25]. Abdi, A., Shamsuddin, S. M., Hasan, S., & Piran, J. (2019). Deep Learning-based Sentiment Classification oof Evaluation text based on Multi-feature Fusion. Information Processing and Management. 56, 1245-1259.

[26]. Taboada, M., Brooke, J., Tofiloski, M., Voll, K.,, & Stede, M. (2011). Lexicon-based Methods for Sentiment Analysis. Computational Linguistics. 37, 268-307.

[27]. Mohammad, S. M., Kiritchenko, S.,, & Zhu, X. (2013). NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In Proceedings of the seventh international workshop on Semantic Evaluation Exercises.
arXiv:1308.6242 **[cs.CL]**

[28]. Word embedding. (2021, January 24). Retrieved February 15, 2021, from https://en.wikipedia.org/wiki/Word_embedding

[29]. Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representation. Proceedings of NAACL-HLT. 746-751.

[30]. Rong, X. (2016). Wword2vec Parameter Learning Explained. arXiv:1411.2738 **[cs.CL]**

[31]. Google code archive - long-term storage for Google code project hosting. (n.d.). Retrieved February 13, 2021, from https://code.google.com/archive/p/word2vec/

[32]. Ruder, S. (2020, March 20). An overview of gradient descent optimization algorithms. Retrieved May 03, 2020, from https://ruder.io/optimizing-gradient-descent/index.html

[33]. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks.

[34]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 15, pp. 1929-1958.

[35]. Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. Published as a conference paper at ICLR. arXiv:1412.6980 **[cs.LG]**

[36]. Sarle, W.S. (1994). Neural Networks and Statistical Models. Proceedings of the Nineteenth Annual SAS Users Group International Conference.

[37]. Malkiel, B.G., & Fama, E.F. (1970) Efficient Capital Markets: A review of theory and empirical work. The Journal of Finance. Vol. 25, 383-417. https://doi.org/10.1111/j.1540-6261.1970.tb00518.x

[38]. Gidofalviand, G., Elkan, C. (2001). Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*.

[39]. Schumaker, R.P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS-09)*. vol. 27, no. 2, pp. 12:1–12:19.

[40]. Liang, X., Zhang, H. , Xiao, J., & Chen, Y. (2009). Improving option price forecasts with neural networks and support vector regressions.' *Neurocomputing*. 72, 13–15, 3055–3065.

[41]. Lu, C.-J., Lee, T.-S., & Chiu, C.-C. (2009). 'Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Syst.* vol. 47, no. 2, pp. 115–125.

[42]. Emioma, C.C., & Edeki, S.O. (2021). Stock price prediction using machine learning on least-squares linear regression basis. Journal of Physics: Conference Series. doi:10.1088/1742-6596/1734/1/012058

[43]. Hochreiter,, S.,& Schmidhuber, J. (1997). Long Short-term Memory. Neural Computation. 9, 1735-1780.

[44]. Kumar S., & Ningombam, D. (2018). Short-Term Forecasting of Stock prices using Long Short Term Memory. International Conference on Information Technology (ICIT). 10.1109/ICIT.2018.00046

[45]. Yadav, A., Jha, C. K., & Sharan. (2019). Optimizing LSTM for time series prediction in Indian stock market. International Conference on Computational Intelligence and Data Science (ICCIDS) 10.1016/j.procs.2020.03.257

[46]. Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep Learning for Stock Prediction Using Numerical and Textual Information. ICIS.

[47]. Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P.,, & Anastasiu, D. C. (2019). Stock Price Prediction Using News Sentiment Analysis. 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications. 10.1109/BigDataService.2019.00035

[48]. Morck, R., Shleifer, A., Vishny, R., Shapiro, M., & Poterba, J. (1990). The Stock Market and Investment: Is the Market a Sideshow? *Brookings Papers on Economic Activity, 1990*(2), 157-215. doi:10.2307/2534506

[49]. Abarbanell, J. S., & Bushee, B. J. (1997). Fundamental Analysis, Future earnings, and Stock Price. Journal of Accounting Research. Vol. 55 no. 1.

[50]. Park, C.-H. (2007). What do we know about the profitability of technical analysis? Journal of Economic Surveys. Vol. 21, No. 4, pp 786-826.

[51]. Yao, J. T., & Tan, C. L. (2001). Guidelines for financial forecasting with neural networks. International Conference on Neural Information Processing. pp. 757–761.

[52]. Remus, W., & O'connor, M. (2001). Neural Networks For Time Series Forecasting. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer Academic Publishers.

[53]. Wang, Y., & Guo, Y. (2020). Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. China Communications, vol. 17, no. 3, pp. 205-221. doi: 10.23919/JCC.2020.03.017.

[54]. Musa,, Y., & Joshua, S. (2020). Analysis of ARIMA-Artificial Neural Network Hybrid model in Forecasting of Stock Market Returns. Asian Journal of Probability and Statistics. 6(2), 42-53.

[55]. Fix, E., & Hodges, J. L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine,, Randolph Field, Texas.

[56]. Altman, N S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistics. 46(3): 175-185.10.1080/00031305.1992.10475879

[57]. Puspitasari, D. A., & Rustam, Z. (2018). Application of SVM-KNN Using SVR as Feature Selection on Stock Analysis for Indonesia Stock Exchange. AIP Conference Proceedings 2023.

[58]. Cortes, C., &Vapnik, V. (1995). Support vector networks. Machine Learning 20: 273–297.

[59]. Drucker, H., Burges, C.J.C., Kaufman,, L., Smola, A., & Vapnik V. (1997). Support vector regression machines. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.), Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 155–161.

[60]. Nava, N., Di Matteo, T., & Aste, T. (2018). Financial time series forecasting using empirical mode decomposition and support vector regression. Risks, 6(1):7.

[61]. Xiao, C., Xia, W., & Jiang, J. (2019). Stock price forecast based on combined model of ARI-MA-LS-SVM. Neural Comput & Applic 32, 5379–5388 (2020). https://doi.org/10.1007/s00521-019-04698-5

[62]. Meesad, P., & Rasel, R. I. (2013). Predicting stock market price using support vector regression. 2013 International Conference on Informatics, Electronics and Vision (ICIEV). pp. 1-6, doi: 10.1109/ICIEV.2013.6572570.

[63]. Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), 1–8.

[64]. Renault, T. (2019). Sentiment analysis and machine learning in finance: a comparison of methods and models on one million messages. Digital Finance. https://doi.org/10.1007/s42521-019-00014-x

[65]. Liveris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN-LSTM model for gold price time-series forecasting. Neural Computing and Applications. https://doi.org/10.1007/s00521-020-04867-x

[66]. Maqsood, H., Mehmood, I., Maqsood, M., Yasir, M., Afzal, S., Aadil, F., Selim, M. M., & Muhammad, K. (2019). A local and global event sentiment based efficient stock exchange forecasting using deep learning. International Journal of Information Management. 50, 432-451.

[67]. Sagala,, T. W., Saputri, M. S., Mahendra, R., & Budi, I. (2020). Stock Price Movement Prediction Using Technical Analysis and Sentiment Analysis. Association for Computing Machinery. https://doi.org/10.1145/3379310.3381045

[68]. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[69]. Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. *arXiv preprint arXiv:1911.09512*.

[70]. Severyn, A., & Moschitti, A. (2015, June). Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)* (pp. 464-469).

[71]. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE. vol. 86, no. 11, pp. 2278–2324, 1998.

[72]. Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-Based Model to Forecast Stock Prices. *Complexity*, *2020*.
[73]. O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
[74]. Lu, C. J., Lee, T. S., & Chiu, C. C. (2009). Financial time series forecasting using independent component analysis and support vector regression. *Decision support systems*, *47*(2), 115-125.
[75]. Liang, X., Zhang, H., Xiao, J., & Chen, Y. (2009). Improving option price forecasts with neural networks and support vector regressions. *Neurocomputing*, *72*(13-15), 3055-3065.
[76]. Vapnik, V.N. (1995) The Nature of Statistical Learning Theory. Springer Science + Business Media, New York. https://doi.org/10.1007/978-1-4757-2440-0
[77]. Yang, H., Chan, L., & King, I. (2002, August). Support vector machine regression for volatile stock market prediction. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 391-396). Springer, Berlin, Heidelberg.
[78]. Chen, J., Chen, H., Huo, Y., & Gao, W. (2017). Application of SVR models in stock index forecast based on different parameter search methods. *Open Journal of Statistics*, *7*(02), 194.
[79]. Ito, K., & Nakano, R. (2003, July). Optimizing support vector regression hyperparameters based on cross-validation. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* (Vol. 3, pp. 2077-2082). IEEE.
[80]. Cao, J., & Wang, J. (2019). Stock price forecasting model based on modified convolution neural network and financial time series analysis. *International Journal of Communication Systems*, *32*(12), e3987.
[81]. Stone, P. J., Dunphy, D. C., & Smith, M. S. (1966). The general inquirer: A computer approach to content analysis.
[82]. Esuli, A., & Sebastiani, F. (2006, May). Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC* (Vol. 6, pp. 417-422).
[83]. Svmlight.joachims.org. (n.d.). Retrieved Feb 15, 2021, from http://svmlight.joachims.org/)
[84]. McCallum, A. K. (1996). "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering." http://www.cs.cmu.edu/~mccallum/bow.
[85]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.
[86]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
[87]. Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.
[88]. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[89]. Malo, P., Sinha, A., Korhonen, P., Wallenius, J., & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, *65*(4), 782-796.

[90]. Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014, March). Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation* (pp. 106-112). IEEE.

[91]. Google code archive - long-term storage for Google code project hosting. (n.d.). Retrieved April 05, 2021, from https://code.google.com/archive/p/word2vec/