Financial Sentiment Analysis
BERT-based Textual Analysis and LSTM-based Equity Return Predictability

Jinge Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Science
of
University College London.

Department of Physics and Astronomy
University College London

September 11, 2022

I, Jinge Wu, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis investigates a textual-based sentiment analysis model by adopting the newly-devised Natural Language Processing (NLP) tools and performs the stock return prediction by integrating sentiment indices from different information sources, through applying the powerful sequential neural networks. During the experiment, we try different neural networks such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) with different methods of word embedding such as Word2Vec and Bidirectional Encoder Representations from Transformers (BERT). Instead of English BERT model, this thesis also considers the implementation of Chinese BERT model, which intends to investigate the co-movement of Chinese financial market and U.S. financial market. Our research shows its power on predictability of equity return by using unstructured textual data which can support financial decisions.

In terms of the BERT model, this thesis applies Google's paper which introduce BERT, a language model based on Transformer, to tackle NLP tasks in financial domain. It has achieved the state-of-the-art on sentiment analysis.

The thesis consists of the following studies:

1. **Data Preparation.** We collect data set from Chinese and U.S. website individually, and finally gain a rich supply of data sets, including 2,000,000 English headlines and 100,000 Chinese headlines. Additionally, we obtain the price of Vanguard Industry Sector ETFs ('SPY'), which contains 11 U.S. sector ETFs and one international ETF. The price data is used for labelling news data in the process of training and testing. After downloading the raw data sets, they are pre-processed with the format needed in the following models.

2. **Model Design.** This part discusses the experiments including the models for Chinese data and English data. It first explains the process of word embedding (using Python libraries to conduct One-hot encoding, Word2Vector and BERT) and then introduce the neural networks used in this thesis. Further, the basic setups for the experiments are introduced.

3. **Fine-tuning Model.** This part describes the training, test and validate data sets used in the model. Moreover, the problem of overfitting is introduced with two solutions which are early stopping and dropout.

4. **Model Testing and Results.** It begins with the introduction of indicators used to evaluate the performance of models such as the confusion matrix, F1-score and cross entropy loss. Following that, we discuss and analyse the results of the experiments and tests.

The thesis presents the following contributions:

1. **Conduct Literature Review on Financial Sentiment Analysis.** The thesis explores and analyses various NLP technique, especially the development of word embedding methods.
2. **Research on the Predicability and Co-movement of Chinese and U.S. Stock Market.**
3. **Provide Algorithms Including API to Extract Data from Websites.** This thesis uses various website for data collection with different methods. The data set used in this thesis cover a wide spread of history in different language. This helps further research for other people.
4. **Assessment of Different Models.** This thesis investigates a variety of NLP models including the state-of-the-art model, BERT. Moreover, we apply neural networks for predicting the status of market. By comparisons with the models, we find that the BERT model has its strength in NLP tasks.
5. **Provide the Current Work Online.** Part of this work will be showcased on the financial-computing.net website, allowing members for further research.

Impact Statement

This thesis applies machine learning algorithms to the stock market, enabling technologically aided financial study and forecasting. The outcomes of this experiment suggest that it is pointing in an exciting direction for future research targeted at improving our understanding of financial markets. We believe that a better understanding and quantification of sentiment correlations will be critical in the future, since this will allow market participants and regulators to make more educated financial decisions.

Market participants, in particular, may benefit from a more comprehensive understanding of news sentiment and market movements that is not limited to individual businesses to make informed investment decisions. Understanding the collective dynamics of news co-occurrence, media sentiment, and corresponding market movements may assist financial regulators in developing a more holistic picture of financial markets, as well as monitoring systemic risks and developing effective intervention strategies.

Acknowledgements

I would like to express my gratitude to Prof. Philip Treleaven and Dr. Denis de Montigny who have supported me during the writing of the thesis. Without their consistent and illuminating instructions, this thesis would have not been written.

I also owe my sincere gratitude to my beloved family and my friends who gave me their help and time in listening to me and helping me work out my problems during difficulties.

I hope everyone can keep healthy in this tough period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis investigates the applications of Natural Language Processing (NLP) techniques in financial sentiment analysis. The work intends to help improve financial technology on the predictability of equity return and find potential volatility in the stock market. This chapter introduces the motivation and experiments conducted in this research. Following that, we discuss the contributions to science and outline the structure of this thesis.

1.1 Motivation

Nowadays, the tools and speed for investors to obtain information have shifted due to the development of technology. Social media and news articles from websites have become essential resources for investors to make decisions. In other words, the stock market is not only affected by structured data, such as historical stock price data, but also affected by some unstructured data, such as the speeches, reviews and blogs [6]. The texts from the Internet can provide information and data that reflect investors' decision psychologically. All these factors combine to make stock market significantly volatile and difficult to predict accurately. Many scholars have used different methods to verify the feasibility of using news sentiment to predict share prices [7] [8] [9] [10].

Furthermore, it is found that since the subprime mortgage crisis, the U.S. market has had a dominant influence on the volatility of other markets [11] [12] [13] [14]. The tremendous volatility of foreign markets gets transferred back to the U.S. market as a result of bad news. According to Zhou et al., there is significant positive impact on the U.S. market from volatility in the Chinese market [14]. When major events take place in China, the U.S. stock market reacts quickly as a consequence. Therefore, we make an assumption that Chinese news can help forecast the U.S. stock market's performance.

However, unstructured data is difficult to predict prices with algorithms because computers cannot read news articles as human beings. Sometimes, outcomes are strongly influenced by subjective experience, resulting in biased results and poor accuracy.

In the past decades, there has been progress in the area of NLP tasks, particularly sentiment analysis. Google researchers have published a series of paper demonstrating the development of sentiment analysis techniques. Bidirectional Encoder Representations from Transformers (BERT) is recently published by Google's Artificial Intelligence Language team [3]. It shows the best performance in dealing with different NLP tasks. At the very beginning, pre-trained models are built on a huge general corpus. After that, the models are well-adjusted so as to deal with downstream tasks like Question Answering (QA), Named Entity Recognition (NER) and Sentiment Analysis (SA).

1.2 Research Objective

In light of the above discussion, this thesis aims to conduct research on NLP methods and to provide forecasts using neural networks. Additionally, we conduct performance study on several models. The outputs from the model can be used to guide investors' judgements and help avoid extra losses.

To be more explicit, this thesis intends to explore two sets of data through the use of several word embedding approaches. The first is an English data set collected from different sources, whereas the second is a Chinese news data set. We validate the stock market's predictability by performing sentiment analysis on the English data. In addition, the Chinese data set is applied to identify the connection between the Chinese and U.S. stock exchanges.

Besides, this thesis makes a contribution by compiling a large data set for future research and making open-source code and files available for usage by other scholars.

1.3 Research Experiments

This thesis considers conducting experiments on financial sentiment analysis using English and Chinese economic news. By doing this, we discuss different methods of word embedding and neural networks for training and evaluating the results of these experiments. The whole process of experiments can be split into the following parts:

1. **Data Preparation.** We begin by obtaining economic information from multiple websites. This necessitates the use of multiple data crawling strategies for various websites. The data set is then cleaned and filtered into the way required in the models.

2. **Model Design.** This thesis addresses numerous approaches for word embedding (i.e. word vectorization), including One-Hot encoding, Word2Vec, and BERT, as well as its advantages and disadvantages. Following the application of word embedding, we investigate the performance of neural network models such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) to predict the state of the stock market.

3. **Fine-tuning Model.** This section focuses on optimizing the model's overall performance. We discuss the problem of overfitting. This thesis considers two approaches to void this which are early stopping and adding dropout layers. Moreover, Adam optimizer is introduced here to improve the performance.

4. **Model Testing and Results.** To begin with, we introduce certain indicators that can be used to assess the performance of models, such as F1-score and loss. Then, we conduct prediction (testing) and compare the results of different models.

1.4 Scientific Contributions

The thesis contributes to science in the following aspects.

1. **Conduct Literature Review of Financial Sentiment Analysis.** The thesis addresses two major approaches in exploring and analysing financial sentiment. The two approaches include lexicon-based method and machine learning-based method. Further, we give details of three widely used approaches to word embedding: One-hot encoding, Word2Vec and BERT and the neural networks used for prediction.

2. **Research on the Predictability and Co-movement of Chinese and U.S. Stock Market.** By doing this, We use economic news to predict the state of the stock market. Further, Chinese news data is applied on forecasting the U.S. market in order to verify the co-movement between two markets.

3. **Provide Algorithms Including API to Extract Data from Websites.** This thesis collects data through a variety of different websites and ways, which will be beneficial for further research.

4. **Evaluate the Efficiency of Different Word Embedding Methods and Neural Networks.** Based on the results of different experiments, this thesis provides analysis on the performance of One-Hot, Word2Vec and BERT as well as CNN, LSTM, GRU, BiLSTM and BiGRU.

1.5 Thesis Structure

The thesis is structured as follows.

- **Chapter 2 Background and Literature Review.** The pertinent literature and important concepts are introduced in relation to this study. We begin by discussing the background of financial sentiment analysis and then describe the methodologies applied in the field of sentiment analysis. Then, we briefly explore the word embedding methods used in the experiments.

- **Chapter 3 Data Preparation.** This chapter discusses the process of data crawling and pre-processing of data set. The original data set obtained from the website is large and has missing values. Due to time constraints for this research, we extract a subset of the raw data and labelling according to the daily stock price.

- **Chapter 4 Model Design.** This chapter introduces the experiments in this project combining word embedding methods including One-hot encoding, Word2Vec and BERT and different neural networks such as CNN, LSTM, GRU, BiLSTM and BiGRU. Besides, detailed process of training neural networks is presented.

- **Chapter 5 Fine-tuning Model.** This chapter discusses how we split data for training and how we avoid over-fitting. We also introduce Adam optimizer to improve training performance.

- **Chapter 6 Model Testing and Results.** This chapter analyses the results of the constructed models. In addition, we provide empirical interpretation of the results.

- **Chapter 7 Conclusions and Future Work.** The chapter summarizes the work done in this thesis and makes suggestions based on the current empirical findings. Further, we discuss the scientific contributions this research has been made. Finally, the thesis ends with an expectation of the future work could be done.

Chapter 2

Background and Literature Review

Basic theory and general background of the project are discussed in the chapter. This section begins by discussing the history of sentiment analysis and then delves into the evolution of numerous sentiment analysis methodologies such as the lexicon-based and machine learning-based approaches. Following that, we focus mostly on the machine learning-based solution, which covers word embedding techniques, including a detailed introduction to One-hot encoding, Word2vce, and BERT.

2.1 Application Domain

2.1.1 Background of Sentiment Analysis

The definition of sentiment analysis is that it is a technique for determining the positivity, negativity, or neutrality of data. Generally speaking, the analysis is conducted based on textual data, whose aim is to help organizations to monitor and understand customers' sentiment, attitude, and feedback to a brand or a product. Due to the rapid development of sentiment analysis in computer science, people can perform complicated and flexible application in this field. Besides consumer reviews, sentiment analysis can be used to analyse a broad variety of topics, including financial markets, disasters, education, software engineering, mental health care and cyberbullying [15]. All of this is essential information for businesses, since it contains useful insights that may assist them in making data-driven decisions [16].

These practical applications and industrial motivations have sparked considerable interest in sentiment analysis research [17]. Liu et al. explored the challenge of analysing textual data from blogs and using it to forecast sales performance [18]. Hong and Skiena investigated several text streams and analysed the relationship between NFL betting lines and public opinion [19]. Tumasjan et al. researched on whether it had impact on German federal election [20]. Yano and Smith modelled the link between political blogs and the overall number of responses in order to find which content would get a lot of attention and learn about readers' interests [21]. Joshi et al. presented analysis of a complete collection of characteristics from blogs with the purpose of predicting film sales [22] [23] [24]. Miller et al. examined the effect of sentiment flow in social networks using mass media and weblog postings [25].

2.1.2 Sentiment Analysis on Financial Market

Financial sentiment analysis derives insights from financial reports, social media and news articles to help make judgements in investing, trading, risk management and financial institution operations. According to Schumaker et al., Financial news and company annual reports have been used as information sources in order to determine how they impact stock market movements [26] [27].

Bollen et al. determined whether collective comments generated from Twitter are associated with the Dow Jones Industrial Average (DJIA) value over time [28]. Bar-Haim et al. presented a framework to identify expert investors in microblogs and utilized it to develop multiple models that forecast stock price increases based on stock microblogging messages (stock tweets) [29]. Feldman et al. proposed The Stock Sonar (TSS), which is a unique hybrid method which enabled investors to quickly digest hundreds of articles each day, aiding them in making fast, educated trading decisions [30]. In 2010, Zhang and Skiena proposed a long-term sentiment-based market-neutral trading approach with minimal volatility by employing quantitative media text analysis system [31].

In the past years, with the fast advance of deep learning algorithms, many researchers intend to develop stock prediction using deep learning models. In the paper written by Sohangir et al., they explored deep learning models to improve sentiment analysis performance on the data set StockTwits [32].

Moreover, it is found that there exists consistent co-movement between the Asia stock markets and that of U.S. in the long run [33]. Chen, et al. noticed out that Chinese, Indian and U.S. stock market are fractionally cointegrated [34]. Nie, Jiang and Yang proposed that China ETF market and the U.S. stock market has a long-run correlation by analysing the results of Granger causality [35].

2.2 Method Development

In terms of sentiment analysis methods, there are two in general: one is lexicon-based method and the other one is the machine learning-based method. The former one belongs to the original research method to conduct and machine learning-based methods has become widely applied these days to improve accuracy.

2.2.1 Lexicon-based Method

In 2002, Turney firstly proposed the method of unsupervised learning algorithm based on lexicon method for classifying reviews [36]. The algorithm can predict different categorizations of the review by computing the average semantic orientation of the sentences which contain adjectives or adverbs [36]. It is known that the system once obtained the accuracy of 74% on average according to the testing reviews in different distinct domains.

The lexicon-based method scores documents by aggregating the sentiment scores of all words in the document. This strategy employs two approaches. One is the method that is based on dictionary. The mechanism of the method is that it identifies the seed words of the opinion and finds synonyms and antonyms of these terms. The other method is based on corpus which contains a seed list of opinion words. An enormous collection of corpuses for other opinion terms are searched to help discover opinion words that are oriented to related and detailed context. Statistical or semantic techniques might be used to achieve this goal.

2.2.1.1 Dictionary-based Approach

The approach includes a dictionary constructed on a number of terms. After that, the dictionary will be labelled different times according to some online dictionary, such as the English thesaurus WordNet or the Chinese dictionary HowNet, aiming to enhance the accuracy of sentiment of each word in the dictionary. During labelling, the sentiment is assessed according to the subjectivity and polarity of the word as well as its synonyms and antonyms. Then the vocabulary is enlarged until no new words can be added to it. Finally, manual examination can help to strengthen the vocabulary.

2.2.1.2 Corpus-based Approach

In this method, the conjunctions between words and statistical characteristics are used to distinguish the sentiment of verbal polarity. The two approaches used in this approach are as follows:

Statistical Approach. Positive polarity is assigned to words that exhibit unpredictable behaviour in positive conduct. The text will present negative polarity if the words display negative recurrence in a negative text. However, when the frequency of the term remains the same in positive and negative texts, the term will be classified as neutral polarity.

Semantic Approach. The sentiment of a text is determined by evaluating the words that are semantically connected to those words. This is accomplished by identifying synonyms and antonyms for a given phrase group.

2.2.2 Machine Learning-based Method

Machine learning methods are based on word vectors. The representation of such vectors is not unique. Word vectors are proposed to make related or similar words closer in terms of distance. The distance of a vector can be measured by applying traditional Euclidean distance or cosine similarity.

The method is based on word vectors. The representation of such vectors is quite common. The purpose of this is to shorten the distance of related or similar words. The distance of a vector could be measured through using traditional Euclidean distance or corsine similarity.

Recent years have witnessed good results achieved by the application of machine learning algorithms to sentiment analysis tasks. The most commonly used deep learning algorithms are Recurrent Neural Networks (RNN), Convolutional Neural Network (CNN) and Support Vector Machine (SVM) [37].

Pang et al. were the first to use machine learning methods for sentiment analysis tasks, using a bag-of-words model and selecting SVM as the classifier. Through studies, they found that the results were significantly better than lexicon-based method [38]. Govindarajan combined genetic algorithms and Naive Bayes (NB) to build a sentiment classification model. The hybrid model was experimentally validated to have better performance than the single model [39].

With the aim of grasping in-depth semantic features of the text and solving the problem of low generalization ability of the model, many researches have proposed applying deep learning algorithms. In deep learning, word vectors are presented as low-dimensional (usually 50 to 100 dimensions) vectors with real numbers using the method of Word Embedding or Distributed Representation. This could be a good solution to the problems of data sparsity and dimensional explosion that may occur in traditional machine learning.

Chen first applied the CNN to NLP tasks. In the paper, he proposed a new algorithm, TextCNN. This algorithm used the word vectorization technique, using a matrix to extract semantic information from texts as the input of the CNN network, and achieved high accuracy in the task of text classification [40]. Ouyang et al. proposed combining CNN networks with Word2Vec to prevent overfitting of the model with a dropout layer layer which is added to the model and perfect the model's performance [41]. Besides, Schmidhuber applied Long Short-Term Memory (LSTM) network to solve problems such as vanishing gradient and exploding gradient in RNN [42]. To simplify the LSTM network, Cho et al. proposed the Gate Recurrent Unit (GRU), which simplifies the three gates of the original LSTM network into two. The GRU network could speed up the training of the model [43].

In 2017, Ashish Vaswani and other scholars implemented the Transformer model. The model abandoned the traditional RNN and LSTM loop structures. Instead, it used the attention mechanism which "pays attention" to the most useful words in predicting the next word in a sentence, to learn the semantics of the document [2]. The attention mechanism processed sentence by mapping significant connections between words and consequently enable to train larger models parallelly and leverage contextual hints to address text ambiguity difficulties.

In November 2018, Google proposed the BERT (Bidirectional Encoder Representations from Transformers) model that pre-trained on 2.5 billion words. The model accomplished unsupervised pre-training and supervised fine-tuning according to specific language-related tasks [3] [44]. Different from other language representation models, BERT applied bi-directional learning to obtain context for words (from left to right and from right to left) simultaneously [45]. It is known that the model has obtained state-of-the-art results through accomplishing many NLP tasks. After completing the pre-training task, an additional output layer is added for fine-tuning [46]. The task is to make substantial structural changes. The model separates representation learning from downstream tasks and has achieved excellent results in text sentimental analysis. In 2019, Araci proposed the FinBERT model, which was trained on the BERT model to adjust a model that was more suitable for financial market sentiment prediction, and achieved a high accuracy rate [47].

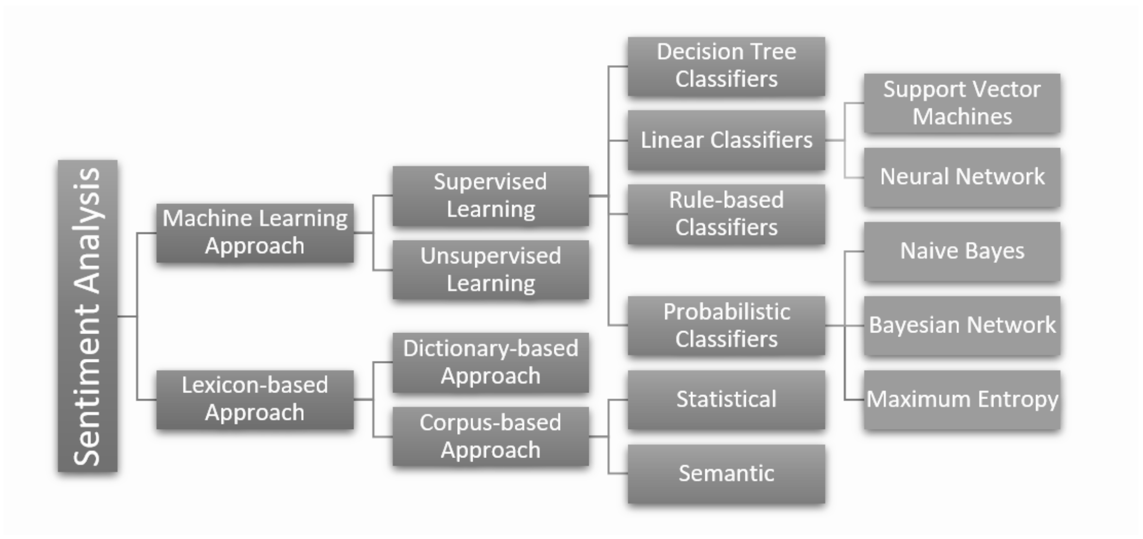


Figure 2.1: Sentiment Analysis Method.

2.2.3 Word Embedding

To create a machine learning or deep learning model, the input data must be in numerical format, as models cannot comprehend text or picture in the same way that people do. We require intelligent methods for converting text input to numerical data, a process known as word vectorization or word embedding. It is a technique utilizing language modelling and feature extraction to map a word to vectors of real numbers. There are some prominent word embedding approaches such as One-hot encoding, Bag-of-words model (BOW), Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, Glove and BERT. In this thesis, we will be covering: One-hot encoding, Word2Vec, Glove, Transformer and BERT.

2.2.3.1 One-hot Encoding

One of the intuitive ways of word embedding is called One-hot Encoding, a mapping approach in which N words are obtained by first aggregating all the words in a corpus and then generating an N-dimensional vector for each individual document in the corpus, with each dimension representing how many specific words are present in that document [48]. This approach is a simpler mapping approach, and the resulting vector representation reflects the information on word frequency.

For example, if we want to encode the list of words ['China','America','England','China'] into word vectors, it is known that there are 3 unique categories which are 'China','America' and

’England’.Then try to use three different binary vectors to present each category, and thus we have the word vectors [[1,0,0],[0,1,0],[0,0,1],[0,1,0]].

2.2.3.2 Word2Vec

In 2013, Google proposed Word2Vec, which is a computationally efficient prediction model that is used for word embeddings from raw text. It displays the words in a multidimensional vector space, where related terms are clustered together [49]. The context for a word is provided by the words that surround it. Through adopting two model architectures (Continuous Bag-of-Words (CBoW) or Continuous Skip-Gram), Word2Vec could generate a distributed representation of input words. The vector representation extracts semantic associations from a collection which is on the basis of the co-occurence of words.

Continuous Skip-Gram

Skip-Gram attempts to predict context words based on the main word. For instance, suppose we wish to extract the embedding for the word ’cat’ from the sentence ’the cat is eating fish’. To begin, we use one-hot encoding to encode all words in the corpus for training. We choose the following word pairings that correspond to the term for which we are looking for the embedding: (cat, the), (cat, is), and (cat, eating) (cat, fish), from each of which, we will build a Neural Network model with a hidden layer, as seen in Figure 2.2.

In the neural network, the input is the word encoded in One-hot encoding, in our case is ’cat’. The input is transformed into the hidden layer using the weight matrix W. The larger the size of W, the more data the embeddings will gather, but the more difficult it will be to learn. Finally, the hidden layer is transformed into the output layer using the weight matrix W'. The model will be performed once for each context word. The model will improve its predictive abilities by attempting to anticipate the context terms.

Once the entire vocabulary has been trained, we will have a weight matrix W. The matrix connects the input to the hidden layer. Now, it is possible that we could acquire the embeddings. As previously stated, if the representation is done correctly, it incorporates semantics, and related words are clustered together in the vectorial universe.

CBoW

CBoW uses a method that is very similar to Skip-Gram, but doing the opposite operation. The model is characterized by inputting a known context and outputting a prediction of the current word.

As is the case with Skip-Gram, we have an input layer (which now consists of the context words in One-hot encoding). We obtain the hidden layer for each context word using the weight matrix W. After that, we average them into a single hidden layer. The layer is then sent to the output layer. By adjusting the weight matrices, the model learns to anticipate the key word. Once training is complete, we utilize the weight matrix W to create word embeddings from the One-hot encoding.

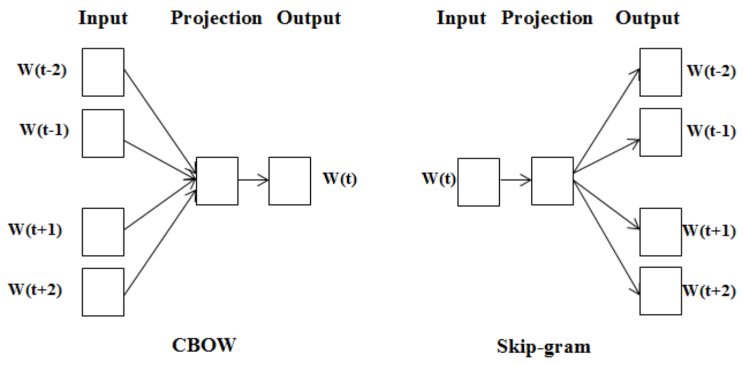


Figure 2.2: Word2Vec Training Model Architecture [1].

According to Mikolov et al., Skip-Gram performs well with small data sets and is better at representing less common words. CBOW, on the other hand, is shown to learn more quickly than Skip-Gram and is capable of representing more frequent words [49].

2.2.3.3 Transformer and BERT

In 2018, Bidirectional Encoder Representations (BERT) was introduced by Google after obtaining a basic understanding of the Transformer encoder architecture. It has obtained state-of-the-art results in many NLP tasks [3] [45] [44] [46].

Here we discuss some key techniques used in BERT model.

1. **Attention**

   In Transformer, its encode-decode architecture relies on self-attention, which applies different weight to predictions.

   The essence of attention is to map a query and a set of key-value pairs to an output. In attention mechanism, query (Q), keys (K), values (V) and outputs are vectors.

   Assume the input comprises queries and keys with dimension d_k and values with dimension d_v. In the paper 'Attention is All You Need', the attention function, i.e. the dot products of queries with all keys, are scaled by dividing $\sqrt{d_k}$ with a softmax function. This is called the 'scaled dot-product attention' (see in Equation. 2.1).

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

   Instead of single attention, the paper applied multi-head attention which enables the model to simultaneously pay attention to input from various representation subspaces at various positions instead of using a single attention function. To be more precise, through adopting learnt linear projections to the dimensions d_k, d_k, and d_v, respectively, the model projects the queries, keys, and values many times in a linear way. Then, it applies the attention function to each of these projected versions of queries, keys, and values in a parallel way, which produces a collection of d_v-dimensional output values. Since each head's dimensions are reduced, the overall computing cost is almost equal to that of the single-head attention with full dimensionality. The paper employed $h = 8$ parallel attention layers and for each of

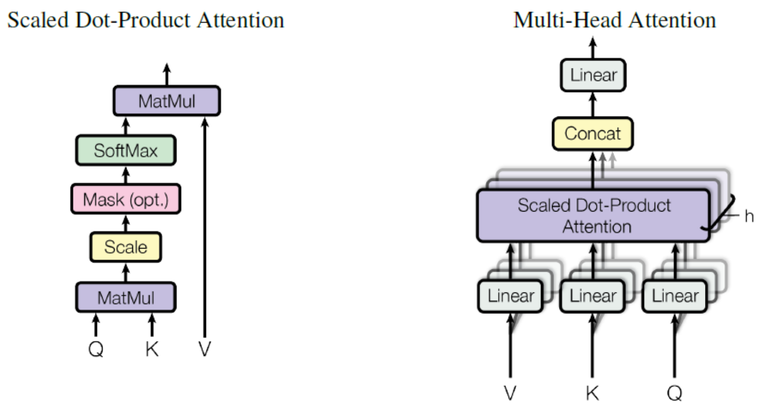


Figure 2.3: Attention [2].

these it uses $d_k = d_v = d_{model}/h = 64$.

2. **Transformer**

An encoder-decoder architecture underpins the Transformer concept. The encoder is the grey rectangle on the left and the decoder is on the right. The Transformer stacks several encoders and decoders (the paper stacks six of them on each other). That is, the output of one encoder is used as the input for the next encoder, and the output of one decoder is used as the input for the next decoder.

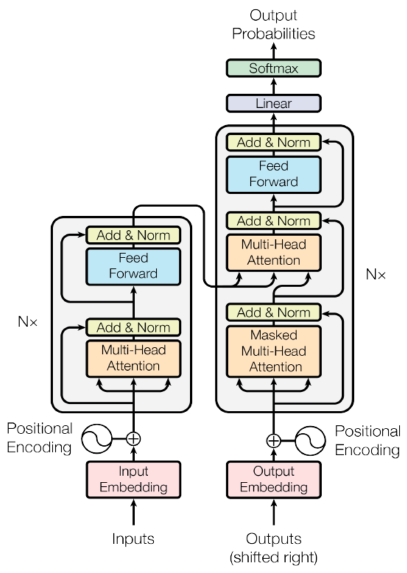


Figure 2.4: Transformer Architecture [2].

3. **Encoder**

   The encoder's goal is to translate each input sequence to a matching abstract continuous representation. The representation consists of the learned information of the sequence. The module is subdivided into two layers: multi-headed attention and a fully connected network. In addition, residual connections are around the two sub-layers. These connections are followed by a layer normalization. According to the paper, all of the sub-layers in the model and the embedding layers present the outputs of size $d_{model} = 512$ [2].

4. **Decoder**

   The decoder is used to produce text sequences. It takes a similar sub-layer as the encoder which contains two multi-headed attention layers, a pointwise feed-forward layer, and residual connections, and layer normalization after each sublayer. It should be mentioned that these sub-layers function in similar ways as the encoder's layers. However, each multi-headed attention layer performs a distinct function. The decoder is ended with a linear layer. The layer plays the role of a classifier and a softmax algorithm for obtaining the word probabilities. The decoder is completed with the help of a linear layer. The layer plays the role of a classifier and a softmax algorithm so as to obtain the word probabilities.

   The decoder is autoregressive. It begins with a start token and accepts a list of previous output as input and the encoder outputs including the input's attention information. When the decoder generates a token as an output, the decoder will stop.

5. **Positional Encodings**

   The Transformer model, as previously mentioned, is capable of processing all words in a sequence in simultaneously. As a result, we must provide information on the input embeddings' positions. This is accomplished through the use of positional encoding. The authors devised an ingenious technique involving the sin and cosine functions.

$$\begin{aligned} PE(pos, 2i) &= sin(pos/10000^{2i/d_{model}}) \\ PE(pos, 2i+1) &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.2}$$

   We construct a vector through using the cosine function for each odd index on the input vector. In terms of the even index, we employ sin function to transform the vector. Then, all of these vectors are added to their corresponding input embeddings, which could send the network information to the position of each vector. Owing to the fact that sin and cosine functions share linear properties which the model could attend to easily, we chose the functions in tandem

6. **Final Linear Layer and Softmax**

   Finally, in order to obtain the output predictions, we need to convert the output vector of the previous decoder into words. Next, we feed the output vector into a linear layer that is completely connected. From the layer, we get a logits vector that represents the size of the vocabulary used in training. A probability score for each word in the lexicon is then obtained by applying the softmax function to the vector obtained before. As a result, we picked the term with the highest likelihood of being correct as our forecast.

7. **BERT**

BERT model is designed to pre-train a deep bidirectional Transformer from textual data in different tasks. There are two steps to implement this. The first is pre-training and the second is fine-tuning.

The model is pre-trained on unlabelled data through Masked Language Model (MLM) task and Next Sentence Prediction (NSP) task. In MLM, part of the words is masked and use the rest word to predict masked words. NSP improves model by understanding sentence relationship. The model is pre-trained on whether the chosen sentence A is followed by sentence B.

Following that, the BERT model is fine-tuned by first initializing using the pre-training parameters, then fine-tuning all parameters using labelled data from the downstream tasks.

BERT uses WordPiece embedding as input or output representations. It selects subwords from the word list and merges them into a new subword by selecting the neighbouring subwords that have the highest probability of improving the language model to be added to the word list. For example, if we used the sentence ”This is useless” and the tokens would be [”this”,”is”,”use”,”less”]. Then the sentence is transformed into ”this is use less”, where ”useless” is cut into ”use” and ”less”. This avoids and extension of the original vocabulary when a new word arises. Moreover, it solves the problem of out of vocabulary [OOV] by using subwords in the existing word list.

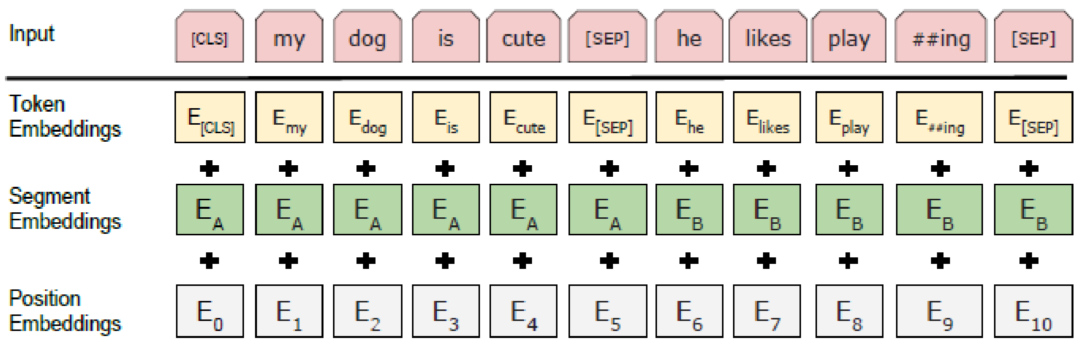

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.5: BERT Word Segmentation [3].

In Figure 2.5, [CLS] is a special classification token added in front of each input texts and [SEP] is the special segmentation symbol between two sentences. As shown in Figure 2.5, the input embedding is denoted as E. The input embeddings are the sum of the token embedding, the segmentation embeddings and the position embeddings.

It works similarly in Chinese BERT. For example, if the sentence ”我爱中国” would be tokenized as [”我”,”爱”,”中”,”国”] even if ”中国” is a word.

Chapter 3

Data Preparation

This chapter describes the data set used in this thesis. This includes the process of data collection, data cleaning, and some transformation of data. Finally, we obtain 20582 samples of Chinese news headlines and 31619 samples of English news headlines. Both data sets contain balanced data in two labels.

3.1 Data Source

This thesis considers using Chinese and English news for sentiment analysis. We obtain several data sets of English news come from the New York Times and Kaggle. The data from Kaggle contains three data sets of news headlines from a variety of websites including the information about headline, URL, publication timestamp and stock ticker symbol. Aside from that, data from the New York Times is gathered using data crawling methods with the information about title, date time and URL. After the initial data cleaning and concatenation, the English news data has 2344025 samples from 2008 to 2020.

The Chinese data is drawn from the Choice Financial Terminal including international and domestic economic news in China. The raw data comprises date time, headline, content, URLs and tag of each website, 92275 pieces from 2016 to 2020 in total.

Additionally, we download the close price of Vanguard Industry Sector ETFs ('SPY') from Yahoo Finance, which including 11 U.S. sector ETFs and one international ETF. The close price is then transferred to daily log return and used for labelling.

3.2 Web Crawling

Web crawling is the process of collecting and parsing raw data from the website. This thesis focuses on web crawling in Python language. The most popular libraries used for web scraping are Selenium, BeautifulSoup and Pandas.

We will be using requests and BeautifulSoup for scraping and parsing the data.

This thesis crawled data from the New York Times, which provides API key for a given month or a given keyword by requiring the URL as the format:

```
# Article Search API to look up articles by keyword
/articlesearch.json?q={query}&fq={filter}
# Archive API to return articles for a given month,
/{year}/{month}.json
```

The general steps of web crawling are:

1. Send a request to the NYT Archive API for a given date.
2. Parse and return response as pandas data frame.
3. Check whether an article has a headline and the content is valid.
4. Get requests and save data in a local folder.

3.3 Data Cleaning

Due to device and time constraints, we propose using smaller data sets in this research. We begin by comparing the time spans of the Chinese and English data sets in order to determine their predictability in the same time period. This results in the erasure of data before to 2016 and after 2020.

Further, we want to average the number of daily headlines. As seen in the Figure 3.1, the greatest number of headlines in a single day is 1599, while the smallest is 0. It is assumed that imbalanced data will have an effect on the model's performance. Thus it is recommended that the original data should be filtered by the maximum number of daily headlines is less than 20. Similarly, for Chinese data, we also ensure that the maximum number of news is less than 20.

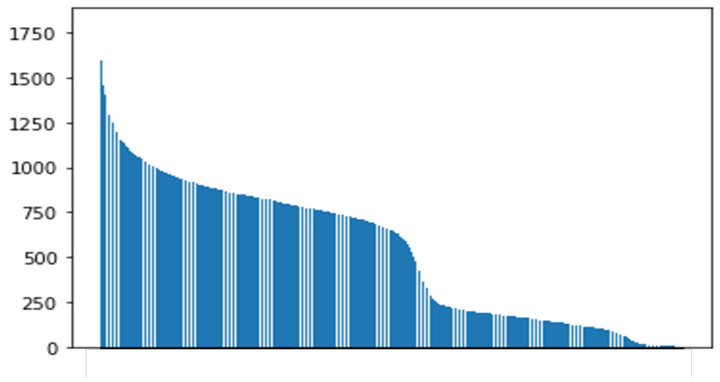


Figure 3.1: English Data Distribution.

Following that, the data sets are cleaned by removing duplicated and null values. Due to the fact that the price data only includes working days, we exclude weekends and holidays in the news data sets to match this.

3.4 Transfer Time Zone

Given that the Chinese data were collected using Beijing Time Zone GMT+8 UTC and the English data were collected using the Eastern U.S. Time Zone UTC-4, it is essential to convert date and time into the same time zone. This thesis considers using the uniform Eastern U.S. Time Zone with all data sets.

3.5 Labelling

After that, we need to label the news headlines using stock price from Yahoo Finance. The 'yfinance' library in Python can help obtain the daily price data for a specific stock. It can provide the historical stock data including the open, high, low and close price.

```python
import yfinance as yf
aapl = yf.Ticker("AAPL")
```

We then calculate the log price of each day:

$$r_t = log(P_{t+1}) - log(P_t) \tag{3.1}$$

where P_t is the close price of each day and P_{t+1} is the close price of the next day. By doing this, we are able to train the model that can predict the future price or status of the market. Following that, we consider labelling data into different groups. At first, we propose using three labels for classification: $\{'\text{neutral}' : 0, '\text{positive}' : 1, '\text{negative}' : 2\}$. However, we have difficulty distinguishing the 'neutral' class, since there is virtually no data with a zero log return unless the market is closed. Thus it is better to have only two labels to reduce ambiguity. We consider using just two labels 'positive' and 'negative'. The rule to cut data into 2 classes is:

- If the log return $r_t < 0$, label = 'negative' or label = 0.
- If the log return $r_t >= 0$, label = 'positive' or label = 1.

In the stock market, a 'positive' label indicates that the price will rise in the future; a 'negative' label indicates that the price will fall in the future. Table 3.2 and Table 3.1 is the distribution of each label after data cleaning. We observe that in both the Chinese and English data sets, the number of samples in each label are generally balanced with minor variance.

Sentiment	Occurrences	% of Total
All	31619	100
'positive'	14784	46.75
'negative'	16835	53.25

Table 3.1: Distribution of Labels in English Data Set.

Sentiment	Occurrences	% of Total
All	20582	100
‘positive’	11721	56.95
‘negative’	8861	43.05

Table 3.2: Distribution of Labels in Chinese Data Set.

3.6 Summary

This chapter goes into detail about the data preparation procedure. It begins with data collection, in which we acquire a significant amount of textual data via data crawling from a number of sources. Following that, we describe how we clean data and transform it to the world’s standard time zone. The data is then labelled and organized in the shape that the models will take.

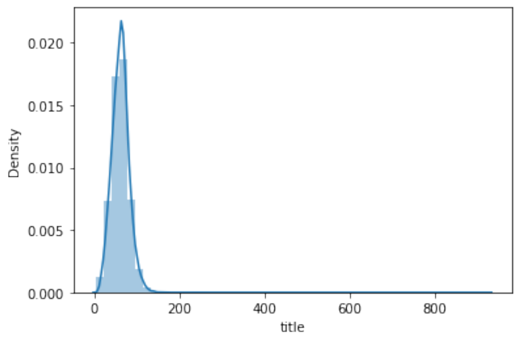
Chapter 4

Model Design

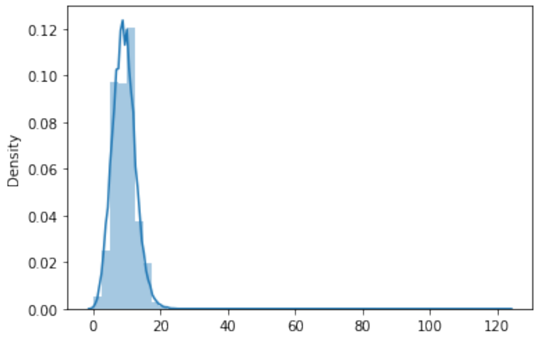
This chapter describes the models that are utilized in the development of this thesis. It begins with the implementation of word embedding techniques, including One-hot encoding, Word2Vec and BERT. Then machine learning algorithms are discussed for prediction, which covers CNN, LSTM, BiLSTM, BRU and BiBRU. Moreover, we demonstrate the baseline model as well as configurations of models.

4.1 Word Embedding

After data cleaning, we need to transform the data to a machine-readable format. To begin with, the sentences are broke up into individual words (tokens), a process known as tokenization. In addition, we apply certain methods to present each word numerically and uniquely. This is known as word embedding. Words with similar meanings will have similar word embeddings. The above procedure can be supported by a great number of Python packages (eg. Keras). Once these tokenized texts have been transferred to embedding matrices, they are ready to be used in machine learning algorithms.



(a) Length of Text



(b) Length of Word Vector

Figure 4.1: Length of Original Text and Word Vector.

Figure 4.1 shows the length of text before and after tokenization. The original text, which is the headline of each news article, has an overall mean of 100. The majority of data has length less than 200. After tokenization, the majority of word vectors has dimensions less than 20. Thus it is reasonable to embed the data into 20 dimensions.

4.1.1 One-hot Encoding

As mentioned before, One-hot encoding is a method for word embedding through building binary columns, which indicates the presence of each possible value from the original data. In Python, Keras provides `Tokenizer` Class for preparation of machine learning models as inputs documents.

The way we use `Tokenizer` is to first learn the dictionary of the text using the `fit_on_texts` method of `Tokenizer`. Then `word_index` is a dictionary that saves the corresponding word and number mapping relationship. Through this dictionary we can convert each word of each string into a number by using `texts_to_sequences`. After that, we pad them to the same length by using the embedding layer to perform a vector presentation and input them into neural networks.

4.1.2 Word2Vec

Word2Vec is a well-known technique for word embeddings through neural networks. It accepts a text corpus as input and returns a vector array. Word embedding through Word2Vec may be used to machine-read language models. Then, mathematical operations on the words are used to recognize their similarity. Generally speaking, a collection of word vectors that is well-trained could gather similar words together in that space [49].

We will use the Gensim package in Python to create word embeddings through applying our own Word2Vec model to a custom corpus by the way of adopting CBOW or Skip-Gram methods [50]. Gensim provides a free Python API to load a pre-trained model from their database:

```python
import gensim.downloader as api
model = api.load("glove-twitter-25")
```

There are various models in this library. For training, Genism Word2Vec demands a 'list of lists' format where documents can be found within a list. It should be mentioned that each list includes lists of tokens from that document. Accordingly, we could train the Genism Word2Vec Model with our own custom corpus. This thesis considers using 'word2vec-google-news-300' model from the Gensim database, which applies the Word2Vec 300-dimensional word vectors trained on 1000 billion words on Google News.

4.1.3 BERT

Finally, we consider using BERT model. There are two ways to use BERT model from Python. This thesis considers using the base BERT model. Denote the number of hidden layers as L, hidden size as H and the amount of attention head as A, it uses L=12, H=768, and A=12 pre-trained for English on the BooksCorpus and Wikipedia. For Chinese BERT model, there are two BERT models commonly used. The first one is published by Google, with L = 12, H = 768 and A = 12. The second one is the version published by HFL by applying the method of Whole Word Masking (WWM) [51]. We will use Google Chinese BERT for experiment.

Moreover, since the BERT model has over 3,000,000 parameters, it requires high storage when training the model. The project applies Google Colab to conduct all experiments. It provides free Jupyter notebooks environment which could run entirely in the cloud for training and testing by using powerful TPUs and GPUs.

4.2 Neural Network

After word embedding, different neural networks can be applied to improve the prediction of sentiment. Generally speaking, a neural network comprises three layers, which are the input layer, the hidden layer and the output layer. In each layer, there are numerous neurons, which receive information from other neurons, process the information and send the result to other neurons. The neural network 'learns' information from the data set by updating its hyperparameter and minimize the gap between the estimation and true value.

4.2.1 Backpropagation Algorithm

Backpropagation is a training algorithm applied in neural networks. It includes 2 steps. The first step is to calculate the output vector from the input vector through the activation function in each layer in a forward way. This is known as the feed forward pass. The second step is to calculate the error and propagate it back to the earlier layers. We can express the algorithm mathematically. Denote:

Symbol	Usage
n_l	The number of neurons in the layer l.
$\sigma(\cdot)$	Activation function.
$W^{(l)} \in \mathbb{R}^{n_l \times n_l - 1}$	The weight matrix from layer $l-1$ to layer l.
w_{ij}^l	The element in matrix $W^{(l)}$, denotes the connected weight from jth neuron in layer $l-1$ to ith neuron in layer l.
$\mathbf{b}^{(l)} = [b_1^{(l)}, b_2^{(l)}, ..., b_n^{(l)}]^{\mathrm{T}} \in \mathbb{R}^{n_l}$	Bias from layer $l-1$ to layer l.
$\mathbf{z}^{(l)} = [z_1^{(l)}, z_2^{(l)}, ..., z_n^{(l)}]^{\mathrm{T}} \in \mathbb{R}^{n_l}$	The net input of neurons in layer l.
$\mathbf{a}^{(l)} = [a_1^{(l)}, a_2^{(l)}, ..., a_n^{(l)}]^{\mathrm{T}} \in \mathbb{R}^{n_l}$	The transferred value through activation function in layer l, i.e. the output value of layer l.
E	Cost function, a measure of evaluating the performance of model. For training set $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, the cost function can be $E_{(i)} = \frac{1}{2}\|\|\mathbf{y}^{(i)} - \mathbf{o}^{(i)}\|\|$, where $\mathbf{o}^{(i)}$ is the output estimate of $\mathbf{y}^{(i)}$.
$\delta_i^{(l)}$	Error of neuron i in the layer l, where $\delta_i^{(l)} = \frac{\partial E}{\partial z_i^l}$.
η	Learning rate which controls the step size in each iteration moving toward a minimum of a loss function.

Table 4.1: Notations.

Backpropagation algorithm is summarized in Algorithm 1.

Figure 4.2 illustrates the process of backpropagation with a 3-layer neural network. Layer 1

Algorithm 1 Backpropagation Algorithm

```
Require: Training set $\mathbf{X}$ with true label $\mathbf{y}$; Activation function $\sigma$; Leaning rate $\eta$.
 1: Randomly initialize the weight and bias $W, \mathbf{b}$, the value should be between 0 and 1.
 2: set $\mathbf{a}^{(1)} = \sigma(W^{(1)}\mathbf{X} + \mathbf{b})$
 3: Feedforward:
 4: for $l = 2, ..., L$ do
 5:     set $\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}$
 6:     set $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$
 7: end for
 8: Output error:
 9: for $i = 1, ..., n_L$ do
10:     $\delta_i^{(L)} = -(y_i - a_i^{(L)})\sigma'(z_i^{(L)})$
11: end for
12: Backpropagate the error:
13: for $l = 2, ..., L-1$ do
14:     for $i = n_2, ..., n_{L-1}$ do
15:         set $\delta_i^{(l)} = (\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^{l+1})\sigma'(z_i^{(l)})$
16:         set $\frac{\partial E}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)}$
17:         set $\frac{\partial E}{\partial b_i^{(l)}} = \delta_i^{(l)}$
18:         update $w_{ij}^{(l)} = w_{ij}^{(l)} - \eta \frac{\partial E}{\partial w_{ij}^{(l)}}$
19:         update $b_i^{(l)} = b_i^{(l)} - \eta \frac{\partial E}{\partial b_i^{(l)}}$
20:     end for
21: end for
22: Use optimizer to update $\mathbf{w}$ and $\mathbf{b}$ until the loss is satisfactorily small.
```

is the input layer with $\mathbf{x} = (x_1, x_2, x_3)$ and bias $\mathbf{b}$, layer 2 is the hidden layer, which has three neurons, and layer 3 is the output layer has 2 nodes.

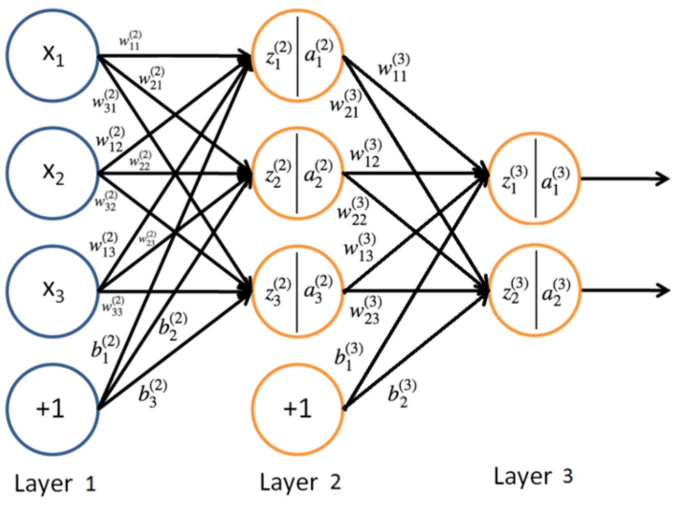


Figure 4.2: Backpropagation Algorithm.

4.2.2 Activation Functions

In most cases, neural network models are used to solve non-linear problems. The activation function can help to transform the summed weight in the last layer non-linearly to the next layer. This thesis covers various frequently used activation functions.

Sigmoid

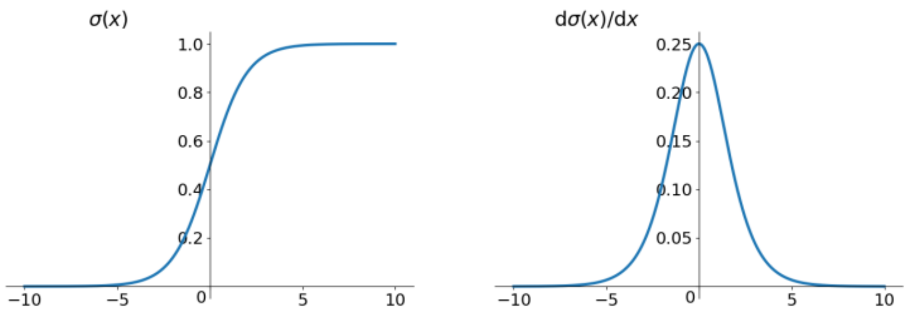


Figure 4.3: Sigmoid Function $\sigma(x) = (1+e^{-x})^{-1}$ and it's Derivative.

The sigmoid function is widely used to calculate the output of the hidden layer. It can map real numbers into the interval (0,1), which is beneficial when predicting the probability as the output. It could also be used for binary classification. The function performs well when the difference in features is or not great. However, the mean value sigmoid function is shifted from zero (i.e. not zero-centered), which may arise in bias during training. Also, it suffers from the vanishing gradient problem, which means the derivative is close to 0 when there is great input values.

Tanh

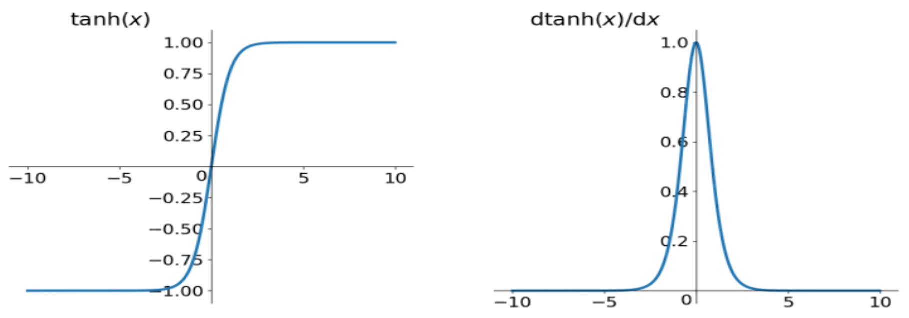


Figure 4.4: Tanh Function $\sigma(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and it's Derivative.

Tanh can map values between -1 and 1. It solves the problem of not zero-centered in sigmoid function. Tanh can also work in binary classification. However, like with the sigmoid function, the problem of vanishing gradient still exists.

ReLU

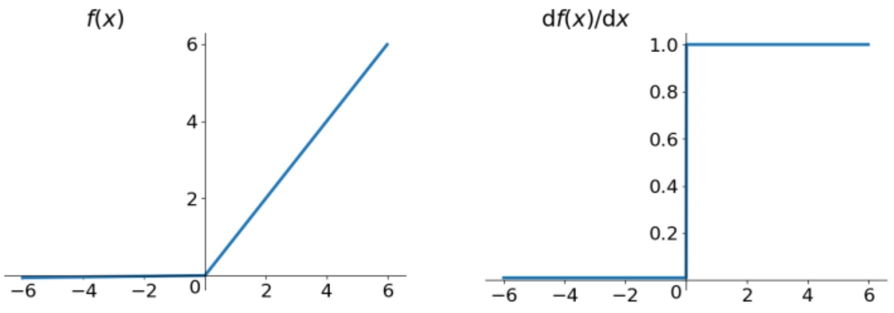


Figure 4.5: ReLU Function $\sigma(x) = max(0,x)$ and it's Derivative.

The Rectified Linear Unit (ReLU) has gained considerable popularity in recent years. ReLU is more trustworthy than sigmoid and tanh as a solution to the vanishing gradient problem and speeds convergence by six times [52]. Regrettably, a disadvantage is that ReLU activation might fail during training. The output passed through the activation function could be negative when there is large weight updates. This is known as the "dying ReLU" problem. We may, however, get around this by establishing a smaller learning rate.

4.2.3 Neural Network Models

Here we discuss different neural networks employed in the experiments. This thesis will not cover some detail in the neural networks. Instead, the following part intends to give a general introduction to these models.

CNN

Convolutional Neural Network (CNN or ConvNet) is a sort of neural network shown to be well-performed in picture recognition and categorization. CNN is composed of three primary operations:

1. **Convolutional Layer**

   The fundamental component of CNN is the convolutional layer. This procedure is that we employ a filter (convolutional kernel) to filter out particular tiny areas of the picture in order to retrieve their feature values. Convolutional kernels produce a rebuilt image, which is called a feature map. With convolution, machine learning algorithms may concentrate on local characteristics while noise in the global data is removed. There are more details on the transformation of dimensions in the convolutional layer in Appendix.

2. **Pooling Layer**

   During the convolution operation, we have been able to reduce and extract features from the input image, but the dimension of the feature image is still very high. The high dimensionality is not only computationally time-consuming, but also leads to overfitting. For this reason, the pooling method, also known as downsamples, is introduced.

The pooling layer is implemented by chunking the feature picture from the convolutional layer. The image is split into discontinuous blocks and the maximum (Max Pooling) or average (Mean Pooling) value inside each block is calculated to get the pooled image. This enables improved picture dimensionality reduction, which reduces the size of the model and increases computing performance, and also decreasing the chance of overfitting and enhancing the resilience of feature extraction.

3. **Fully-connected Layer**

   The Fully-connected layer (FC) also called dense layer plays the role of the classifier in the whole Convolutional Neural Network. The neurons in this layer connect fully to the activations in the previous layer. In pratice, a fully-connected layer could be implemented by a convolutional layer. The only difference between the FC layer and convolutional layer is that the convolutional layer connects part of the previous neurons, and thus have shared parameters. However, both FC layer and convolutional layer implement the dot product of two matrices.

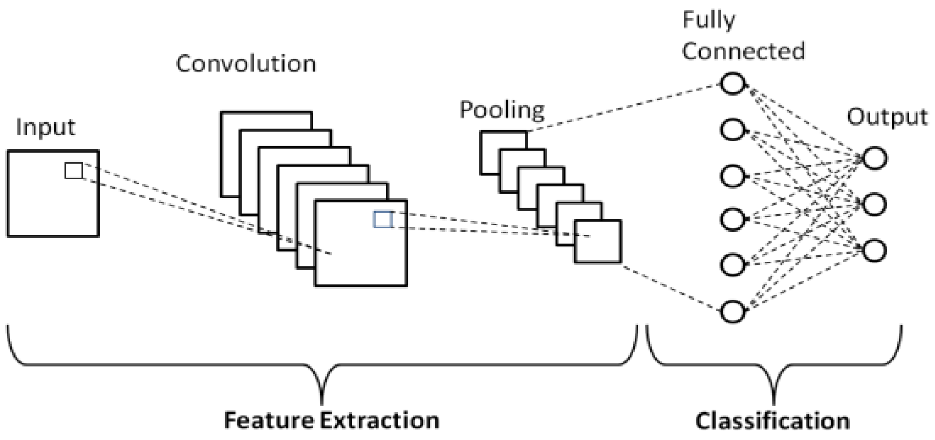


Figure 4.6: Convolutional Neural Network (CNN) [4].

GRU

Recurrent Neural Network (RNN) is a class of artificial neural network that performs well on learning temporal dynamic behaviour. This is implemented by sharing weights between nodes and connected as a directed graph along a temporal sequence. Gated Recurrent Unit (GRU) is a kind of RNN, which solved the vanishing gradient problem. GRU upgrading RNN from controlling information flow via reset gate and update gate.

- **Update gate.** The update gate determines how much past information should be passed to the next state.

- **Reset gate.** The reset gate controls the amount of previous information that should be dropped.

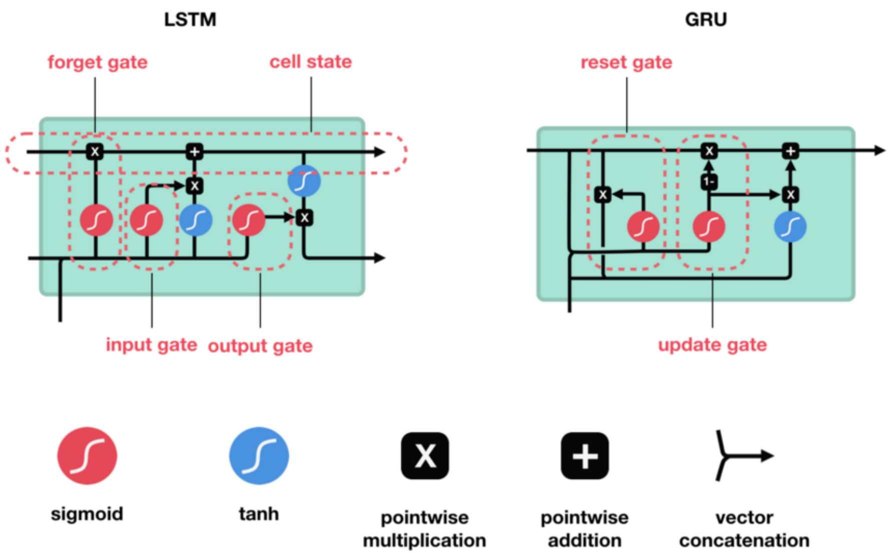


Figure 4.7: LSTM and GRU [5].

LSTM

Long Short-Term Memory (LSTM) is similar to GRU which tackles the problem of vanishing gradient as well. Additionally to GRU, there are two more gates here: 1) forget gate 2) output gate.

- **Forget gate.** It has an effect on what is remembered from a previous cell state and what is forgotten. This affects the retained and deleted quantity of data from the previous state.
- **Output gate.** The output gate controls how much to reveal of a cell. It is in charge of deciding the next concealed state.

Bidirectionality

Bidirectional LSTM and GRU (i.e. BiLSTM and BiGRU) are also considered experiments in this thesis. Bidirectional LSTM is an extension of traditional LSTM that can improve model performance when used to classify sequences. It trains two rather than one LSTM on the input sequence. The first model could learn the sequence of the input and the second one learns the reverse.

Similarly, bidirectional GRU can be thought of as an extension of conventional GRU, as it trains two GRUs on the input sequence rather than one. This could add context to the network and lead to a faster and more complete learning process for the problem.

4.3 Model Setup

Figure 4.8 demonstrates the model setup used in the experiments. First, we apply pre-segmentation on the data sets using One-hot encoding and Word2Vec to get word embeddings. Then the word embeddings are fed into the neural networks. It starts with a 128-unit embedding layer. After that, a deep learning networks chosen from CNN, LSTM, GRU, BiLSTM or BiGRU is considered. Then we consider a pooling layer to reduce the dimension followed by a fully connected layer. Besides, we implement a dropout layer with a rate of 0.4 to avoid overfitting and another fully connected layer is used to output.

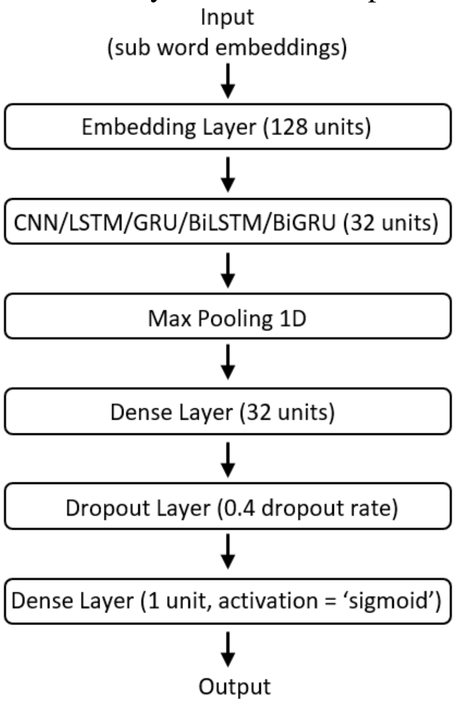

Input
(sub word embeddings)
Embedding Layer (128 units)
CNN/LSTM/GRU/BiLSTM/BiGRU (32 units)
Max Pooling 1D
Dense Layer (32 units)
Dropout Layer (0.4 dropout rate)
Dense Layer (1 unit, activation = 'sigmoid')
Output

Figure 4.8: Baseline Model.

In terms of English BERT model, this thesis implements the $\text{BERT}_{\text{base}}$, which contains L = 12 hidden layers, H = 768 hidden size and A = 12 attention heads with 110 million parameters in total. For the Chinese data, this research only considers BERT model as experiment considering time constraint. The Google Chinese BERT is applied, with similar layers setup as the English one.

4.4 Summary

This chapter details the experiments conducted on the two data sets. We aim to employ three unique word embedding approaches in conjunction with five distinct neural networks in this project. Additionally, this chapter introduces strategies for performing experiments using Python libraries, which are useful in our experiments. While training, the parameters in the models are fine-tuned with more details discussed in chapter 5.

Chapter 5

Fine-tuning Model

This chapter discusses the primary techniques for fine-tuning models. This thesis trains models using the standard 80-20 train-test split and validates using 10% of the training set. To minimize overfitting, we introduce the technique of early stopping and dropout layers throughout training. Additionally, we mention the Adam optimizer, which is used to improve models.

5.1 Training Sample Size

While training, the data set is divided into training set, test set and validation set. The size of training is 20%, and the size of the validation set is 10% of the training set. Here are the distributions of English and Chinese data set after train-test split.

	Label 0	Label 1	Total
Train	12121	10644	22765
Validate	1347	1183	2530
Test	3367	2957	6324
Total	16835	14784	31619

Table 5.1: Distribution of Labels in English Data Set.

	Label 0	Label 1	Total
Train	8439	6380	14819
Validate	938	709	1647
Test	2344	1772	4116
Total	17721	8861	20582

Table 5.2: Distribution of Labels in Chinese Data Set.

5.2 Optimizer

Optimizers can be used to modify the parameters of a neural network such as weight and learning rate, whose aim is to decrease loss to the least. The most popular optimizers include AdaGrad, RMSProp and Adam. AdaGrad preserves a learning rate for each parameter to improve performance on sparse gradients. RMSProp adaptively preserves the learning rate for each parameter based on the mean of the nearest magnitude of the weight gradient. This means that the algorithm has excellent performance on non-stationary and online problems.

Adam gains the advantages of both AdaGrad and RMSProp. It not only calculates the adaptive parameter learning rate based on first-order moment (mean), but also takes advantage of the second-order moment (uncentered variance). To be more specific, the method generates an exponential moving average of the gradient and the squared gradient. The parameters β_1 and β_2 control the decay rate of the moving average.

We compute bias-corrected first and second moment estimates $\hat{m}_t$ and $\hat{v}_t$ using the first moment (the mean) and the second moment (the uncentered variance) of the gradients m_t and v_t respectively as follows

$$\begin{aligned}\hat{m}_t &= \frac{m_t}{1-\beta_1^t} = \frac{\beta_1 m_{t-1} + (1-\beta_1)g_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} = \frac{\beta_2 v_{t-1} + (1-\beta_2)g_t^2}{1-\beta_2^t}\end{aligned} \tag{5.1}$$

and update the weights:

$$w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{5.2}$$

5.3 The Problem of Overfitting

Overfitting is a critical issue in the training of neural networks. This indicates that the model learnt the distribution of the training data too precisely during training in order to generalize its knowledge to previously unknown samples. The generalization ability of a model is usually assessed using the model's performance on the validation set. During model optimization, we expect the model to perform non-increment on the validation error when the training error is reducing. Conversely, when the model performs well in the training set and poorly in the validation set, we consider the model to be overfitting. There are various strategies for avoiding overfitting, this thesis considers dropout and early stopping.

Early Stopping

The concept of early stopping is based on continuously assessing the model's performance on unknown data following training epochs to see whether it finds the potential pattern of overfitting. Specifically, we utilize the training set to update the weights in our model, but test the model's performance on the validation set after each epoch. The training process is terminated if the modifications made during training do not result in improved performance on the validation set. One can employ an early stop by monitoring the validation accuracy and terminates the training process if the value does not increase in 5 epochs:

```
EarlyStopping(monitor="val_accuracy", mode="max", patience=5)
```

Dropout

Early stopping helps avoid overfitting by halting training when the network exhibits indications of the overfitting, but it has no effect on the weight updating itself. Another method to avoid overfitting is to modify the model's underlying weights. Overfitting is frequently caused by weights growing too large or too tiny throughout training. In this case, dropout is a straightforward strategy to avoid this by imposing regularization restrictions on the weight values. Rather of simply applying limitations to weight values, it zeros them out using a pre-defined probability [53]. Thus, a particular probability of the total number of weights is not used at each epoch. With fewer weights in the model during training, the chance of learning very precise patterns in the data is reduced, and therefore avoid overfitting.

```
dropout = tf.keras.layers.Dropout(0.4)
```

5.4 Summary

This chapter addresses the issue of overfitting during the model training process by incorporating the early stopping and dropout method. Moreover, we discuss the Adam optimizer used in this research and the size of training and test sets.

Chapter 6

Model Testing and Results

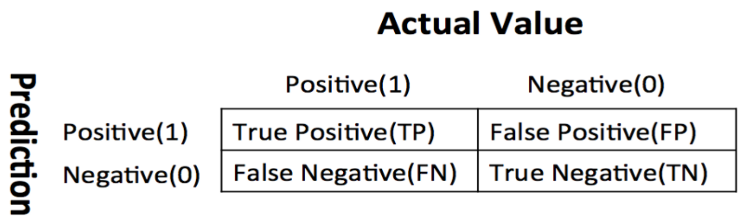
In this part, we will be discussing the results of all experiments in this project and draw conclusions according to the results. On the basis of the test set, we evaluate model performance using the accuracy, loss function, and F1-score.

6.1 Evaluate Model Performance

While training and testing, we need to keep track of the model's performance by employing numerical techniques. This thesis introduces a variety of approaches that can be applied to analyse the performance of model.

6.1.1 Confusion Matrix

The confusion matrix is a 2x2 table that serves as the foundation for a variety of measures. Assuming the classification has just two possible outcomes (1 or 0), a confusion matrix is the product of the prediction and the actual value.



		Actual Value	
		Positive(1)	Negative(0)
Prediction	Positive(1)	True Positive(TP)	False Positive(FP)
	Negative(0)	False Negative(FN)	True Negative(TN)

Figure 6.1: Confusion Matrix.

where

- **True Positive**: cases that positive events are correctly predicted
- **False Positive**: cases that positive events are incorrectly predicted
- **True Negative**: cases that negative events are correctly predicted
- **False Negative**: cases that negative events are incorrectly predicted

It is beneficial to explore the performance of each category in the prediction. In practice, the matrix is not limited to 2x2, and if the classification contains more than two categories, there will be a larger matrix.

6.1.2 Accuracy, Recall, Precision

All of the three values are directly computed according to the result of the confusion matrix. They are expressed as percentages of the absolute value displayed in the confusion matrix.

6.1.2.1 Accuracy

Accuracy is simply the percentage of cases accurately anticipated.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{6.1}$$

Accuracy could measure a model's overall performance. However, it does not reflect performance across all categories. As a result, it is possibly to overlook some crucial information when relying just on it. The use of it is more efficient under the condition that many cases belonging to separate categories are compared or are used in conjunction with other metrics.

6.1.2.2 Recall

Recall can measure our model through identifying true positives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{6.2}$$

When the objective is to catch the greatest possible number of positive cases, recall is utilized. For instance, if the government would like to know how many of the individuals are affected with Covid-19 in the community. Then recall is a critical statistic to track if applying a model to see whether or not a person is affected according to their symptoms.

6.1.2.3 Precision

Precision indicates the proportion of cases predicted to be positive that are actually positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{6.3}$$

Precision and recall are mutually constrained. The reason is that the number of examples we would like to record is negatively correlated to the bar for positive cases we want to catch, which means, the more positive examples we want to record, the lower the bar for positive cases we want to catch. Taking the calculation of the infected cases of Covid-19, we might include all instances with mild symptoms that might be due to regular flu.

6.1.3 F1-Score

The F1-Score is developed to provide a balanced measure between recall and precision.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

If the distribution of positive and negative examples is highly asymmetrical, the average may not be an accurate reflection of model performance.

6.1.4 Cross Entropy Loss

The cross entropy loss is also known as the log loss function, which penalizes instances that are incorrectly categorized and also lacks confidence in those that are successfully classified. More accurate model means smaller loss value.

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \times log(p(y_i)) + (1-y_i) \times log(1-p(y_i)) \tag{6.5}$$

In Equation. 6.5, y denotes the actual result (0 or 1), and the p function denotes the probability that the model correctly predicts that outcome. Therefore, when the true value is 1, we would like to increase the probability to the highest associated with the prediction of 1, which is $p(y)$, but if the real value is 0, we would like to maximize the probability associated with the forecast of 0, which is $1 - p(y)$. Then, we calculate probability's average summation of each sample. Between 0 and 1, the log function is negative.

The value of cross entropy loss might range from 0 to infinity. During model training, we intend to minimize the validation loss in each epoch.

6.2 Discussion of Results

Table 6.1 and 6.2 demonstrate the results of experiments mentioned in the chapter 4. During the experiments, data are split into training set, test set and validation set (see Table 5.1 and 5.2). The batch size while training is 256, which is the number of training examples in one iteration. The smaller the batch size, the less storage it needs during training, but may have less accuracy on estimating the gradient. Moreover, we use the validation loss for early stopping with a patience of 5 epochs. Table 6.1 and is the general results of all experiment using both English and Chinese news data set.

For English models, we make comparisons by using different neural networks with different word embedding methods. The value of precision, recall, F1-score, test accuracy and cross entropy loss for all models are shown in Figure 6.1. In general, we conclude that BERT model gives the best performance, which achieves a high accuracy and F1-score of 90%. Moreover, we find that Keras BiGRU performs the best in Keras embedding (One-hot encoding) with highest F1-score and least loss, and GRU performs the best in Word2Vec embedding. All these models

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Loss**
Keras_CNN	0.8291	0.8288	0.8289	0.8301	0.5805
Keras_LSTM	0.8118	0.8119	0.8118	0.8130	0.6455
Keras_GRU	**0.8352**	0.8272	0.8294	0.8322	0.5794
Keras_BiLSTM	0.8216	0.8189	0.8199	0.8217	0.6155
Keras_BiGRU	0.8330	**0.8336**	**0.8333**	**0.8342**	**0.5782**
Word2Vec_CNN	0.8171	0.8176	0.8173	0.8184	0.6269
Word2Vec_LSTM	0.8168	0.8180	0.8172	0.8181	0.6280
Word2Vec_GRU	**0.8231**	0.8216	**0.8222**	**0.8238**	**0.6084**
Word2Vec_BiLSTM	0.8220	**0.8219**	0.8219	0.8232	0.6106
Word2Vec_BiGRU	0.8189	0.8164	0.8173	0.8192	0.6242
BERT	**0.9025**	**0.9084**	**0.9052**	**0.9069**	**0.3560**
CH_BERT	0.7834	0.7832	0.7876	0.7810	0.5629

Table 6.1: Classification Results.

can achieve a relatively high accuracy of over 80% on test data, and there is no significant difference on the test result among these 5 neural networks.

For Chinese data set, BERT also performs well which approaches an accuracy of 78% and F1-score of 79%. This verifies the co-movement of Chinese market and U.S. market and the predictability of stock market using economic news.

Model	**0**	**1**
Keras_CNN	0.8433	0.8145
Keras_LSTM	0.8270	0.7966
Keras_GRU	**0.8512**	0.8076
Keras_BiLSTM	0.8383	0.8014
Keras_BiGRU	0.8460	**0.8206**
Word2Vec_CNN	0.8314	0.8032
Word2Vec_LSTM	0.8298	0.8047
Word2Vec_GRU	**0.8389**	0.8055
Word2Vec_BiLSTM	0.8367	**0.8071**
Word2Vec_BiGRU	0.8358	0.7989
BERT	**0.9184**	**0.8919**
CH_BERT	0.6908	0.8524

Table 6.2: F1-score with Labels.

Table 6.2 lists the F1-score of each label for all models. It is noticeable that the F1-score is slightly different for each label. Generally, the F1-score is greater in the label with more data since it learns better. Specifically, the Chinese data set has more ’negative’ data (i.e. label = 1), thus the F1-score of label 1 is generally higher than F1-score of label 0. And the relationship is the same in English data set.

6.3 Training Time

During training, we employ early stopping for English data and the training will terminate within a 5-epoch non-increasing validation accuracy. Table 6.3 is the number of training epochs and running times for each model. We find that the BERT model spends the longest time on training although it performs the best on test accuracy. This reasonable since it has around 110 million parameters in each iteration. Therefore, there is a trade-off between the test accuracy and the computational complexity.

Model	**No. epochs**	**time(s)**
Keras+CNN	8	9
Keras+LSTM	7	9
Keras+GRU	6	6
Keras+BiLSTM	6	8
Keras+BiGRU	7	11
Word2Vec+CNN	19	8
Word2Vec+LSTM	11	9
Word2Vec+GRU	16	16
Word2Vec+BiLSTM	13	16
Word2Vec+BiGRU	9	11
BERT	6	1635
CH_BERT	5	2732

Table 6.3: Running Times and Epochs of Each Model.

Additionally, we observe that the overall number of training epochs used by Word2Vec embedding before early stopping is larger than Keras (One-hot encoding) embedding. As a result, the training time of Word2Vec embedding is longer than Keras (One-hot encoding) embedding. This is because we use greater dimension in Word2Vec embedding. This thesis loads the 'word2vec-google-news-300' model for training, which results in a 300-dimensional word vectors in the embedding layer. While in Keras (One-hot encoding) model, the dimension of embedding layer is 20. As a result, this needs a larger batch size to learn in each step. However, if we retain the same batch size as Keras (One-hot embedding), this may result in greater the number of epochs needed training and thus the training time is extended.

6.4 Summary

To sum up, this chapter discusses the results of all experiments. Among them, we observe that BERT performs well in both data sets by comparing F1-score, accuracy and loss of different models even though it costs more time for training. According to the results, we can draw the conclusion that there is predictability in the stock market using economic news. In addition, the results from the Chinese model demonstrates that Chinese financial market can be used to forecast the U.S. market.

Chapter 7

Conclusions and Future Work

In this chapter, a summary of the thesis will be made and the contributions of this project be presented. Finally, we provide recommendations for future work to be carried on.

7.1 Summary of Work

This thesis has introduced some powerful techniques of financial sentiment analysis and developed several experiments. To be more specific, the experiments and results are:

1. **Data Preparation.** This thesis uses two different data set for model training, which are the English news crawling from various websites and Chinese new from the Choice Financial Terminal. After that, data cleaning, data pre-processing, and several transformations of data are applied to prepare data into the format needed in the following models.

2. **Model Design.** This thesis employs three different methods for word embeddings. Following that, we implement distinct neural networks to predict the status of market. This thesis considers two statuses in general, one is the upper movement and the other is the lower movement of the market.

3. **Fine-tuning Model.** While training, we adopt early stopping and add the dropout layer so as to prevent overfitting and fine-tuning the parameters to perfect each model's performance. Moreover, the Adam optimizer is introduced, which is a powerful optimization tool in NLP model training.

4. **Model Testing and Results.** We introduce different indicators to evaluate the performance of experiments, such as F1-score, accuracy and cross entropy loss. From the results, we conclude that there is predictability on U.S. stock market using economic news. Moreover, Chinese economic news has significant effects on the U.S. stock market. Besides, by comparisons of different model, we have shown that BERT model performs best in both English and Chinese data set.

7.2 Contributions

This thesis considers the following contribution to science:

1. **Conduct Literature of Financial Sentiment Analysis.** This thesis discusses two primary techniques for sentiment analysis in finance: lexicon-based analysis and machine learning-based analysis. In the experiments, we primarily focus on the machine learning-based method.

2. **Conduct Literature of NLP Techniques in Sentiment Analysis.** This thesis introduces three widely used approaches of word embedding: One-hot encoding, Word2Vec and BERT. We discuss these strategies in great detail from a mathematical and computational standpoint.

3. **Research on the Predictability and Co-movement of Chinese and the U.S. Stock Market.** While numerous studies have been conducted on the correlation between the Chinese and U.S. financial markets, this thesis focuses on sentiment analysis of macroeconomic news, which is a pioneering study in this sector. Additionally, we conclude that it is possible to forecast the market using economic news and that there is a relationship between the Chinese and U.S. financial markets. This enables investors to receive real-time information from news websites and make more informed decisions.

4. **Provide Algorithms Including API to Extract Data From Websites.** This thesis collects data through a variety of different websites and ways. The data sets used in this thesis span a broad range of time periods in English and Chinese. Due to time constraints, this thesis conducts experiments using a smaller size of data. However, this contributes to the future research for other individuals.

5. **Evaluate the Efficiency of Different Word Embedding Methods and Neural Networks.** Based on the results of different experiments, this thesis provides analysis on the performance of One-hot encoding, Word2Vec and BERT as well as CNN, LSTM, GRU, BiLSTM and BiGRU. The results show that BERT has its strength on sentiment analysis with high accuracy.

6. **Provide the Current Work Online**. Part of this work will be showcased on the financial-computing.net website, allowing members for further research.

7.3 Future Work

There are improvement could be done in the future.

1. **Use Large data for Model Generalization.** Due to limitations of devices and time, this research considers using smaller data set, which is a subset of the data set we obtain. It is suggested that using the full data set for training will give better performance.

2. **Improvement on Fine-tuning Neural Networks.** When larger data is used for training, it is better to construct networks with more layer and units to avoid underfitting. Moreover, it is suggested to consider more approaches to avoid overfitting.

3. **Using Intro-day Data (Higher Frequency).** This thesis conducts experiment using daily price. We provide an option for future work to use price data with higher frequency. By using intro-day data, we can make better predictions within a day and make decisions more frequent and flexible.

4. **Regression Instead of Classification to Obtain More Precise Prediction.** This thesis considers a binary classification which does not give precise prediction on stock market. We suggest improving the prediction by increasing the number of labels (i.e. multi-classification) or even regression. This can help investors with better decision when reacting to the market.

5. **Improvement on BERT Model.** There can be improvement on fine-tuning BERT model. For example, we suggest using cross-validation to train more data. Moreover, according to Xu et al., the performance of BERT can be improved with two effective mechanisms: self-ensemble and self-distillation [54].

Appendix A

GitHub: https://github.com/JINGEWU/Master_Thesis

Appendix B

Calculation of Convolutional Layer [55]

1. Input: a volume of size $W_1 \times H_1 \times D_1$
2. Requires four hyperparameters:
   - Number of filters K,
   - Their spatial extent F,
   - The stride S,
   - The amount of zero padding P.
3. $W_2 = (W_1 - F + 2P)/S + 1$
4. $H_2 = (H_1 - F + 2P)/S + 1$
5. $D_2 = K$
6. Output: a volume of size $W_2 \times H_2 \times D_2$

Equations in LSTM

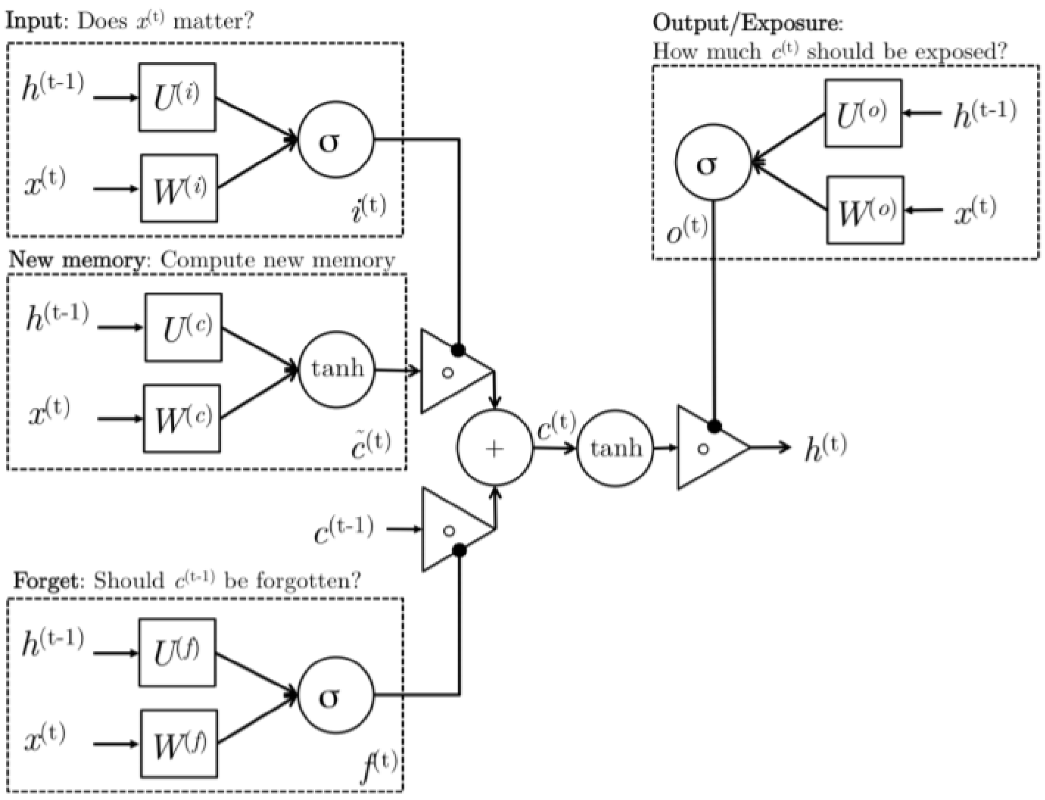


Figure 1: LSTM Architecture [56].

$$
\begin{aligned}
i^{(t)} &= \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) && \text{(Input gate)}\\
f^{(t)} &= \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) && \text{(Forget gate)}\\
o^{(t)} &= \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) && \text{(Output/Exposure gate)}\\
\tilde{c}^{(t)} &= \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) && \text{(New memory cell)}\\
c^{(t)} &= f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{(Final memory cell)}\\
h^{(t)} &= o^{(t)} \circ \tanh(c^{(t)})
\end{aligned}
$$

Equations in GRU

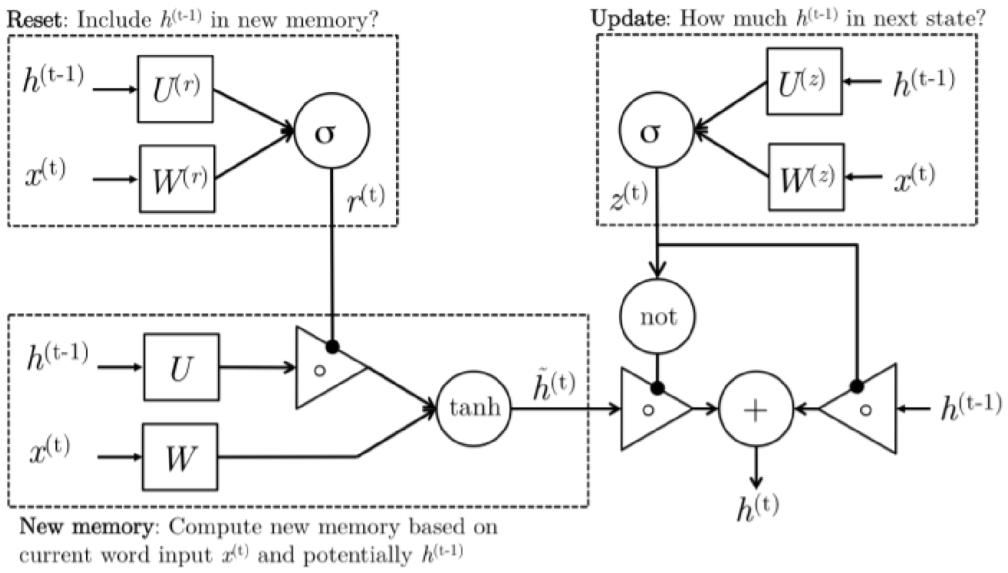

Reset: Include $h^{(t-1)}$ in new memory?
Update: How much $h^{(t-1)}$ in next state?
$h^{(t-1)}$
$U^{(r)}$
σ
$x^{(t)}$
$W^{(r)}$
$r^{(t)}$
$U^{(z)}$
$W^{(z)}$
$z^{(t)}$
not
U
W
tanh
$\tilde{h}^{(t)}$
+
$h^{(t)}$
New memory: Compute new memory based on current word input $x^{(t)}$ and potentially $h^{(t-1)}$

Figure 2: GRU Architecture [56].

$$
\begin{aligned}
z^{(t)} &= \sigma(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)}) && \text{(Update gate)} \\
r^{(t)} &= \sigma(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)}) && \text{(Reset gate)} \\
\tilde{h}^{(t)} &= \tanh(r^{(t)} \circ Uh^{(t-1)} + Wx^{(t)}) && \text{(New memory)} \\
h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{(Hidden state)}
\end{aligned}
$$

Appendix C

	CNN	**GRU**	**LSTM**	**BiLSTM**	**BiGRU**
Input Layer	20	20	20	20	20
Embedding	20x128	20x128	20x128	20x128	20x128
CNN/GRU/LSTM	1x32	20x32	20x32	20x64	20x64
Pooling	128	32	32	64	64
Dense	32	32	32	32	32
Dropout	32	32	32	32	32
Dropout Layer	1	1	1	1	1

Table 1: Neurons in Keras Networks.

	CNN	**GRU**	**LSTM**	**BiLSTM**	**BiGRU**
Input Layer	20	20	20	20	20
Embedding	20x300	20x300	20x300	20x300	20x300
CNN/GRU/LSTM	1x32	20x32	20x32	20x64	20x64
Pooling	128	32	32	64	64
Dense	32	32	32	32	32
Dropout	32	32	32	32	32
Output Layer	1	1	1	1	1

Table 2: Neurons in Word2Vec Networks.

	BERT
Input Layer	100
Input Mask	100
Segmentation Layer	100
Keras Layer	768
Dense	32
Dropout	32
Output Layer	1

Table 3: Neurons in BERT.

Bibliography

[1] Dima Suleiman, Arafat Awajan, and Nailah Al-Madi. Deep learning based technique for plagiarism detection in arabic texts. In *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pages 216–222. IEEE, 2017.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Van Hiep Phung, Eun Joo Rhee, et al. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9(21):4500, 2019.

[5] Michael Nguyen. Illustrated guide to lstm's and gru's: A step by step explanation. *Online] Towards Data Science*, 2018.

[6] Nuno Oliveira, Paulo Cortez, and Nelson Areal. Stock market sentiment lexicon acquisition using microblogging data and statistical measures. *Decision Support Systems*, 85:62–73, 2016.

[7] Spandan Ghose Chowdhury, Soham Routh, and Satyajit Chakrabarti. News analytics and sentiment analysis to predict stock price trends. *International Journal of Computer Science and Information Technologies*, 5(3):3595–3604, 2014.

[8] Wesley S Chan. Stock price reaction to news and no-news: drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260, 2003.

[9] Yang Liu, Qingguo Zeng, Huanrui Yang, and Adrian Carrio. Stock price movement prediction from financial news with deep learning and knowledge graph embedding. In *Pacific rim Knowledge acquisition workshop*, pages 102–113. Springer, 2018.

[10] Pisut Oncharoen and Peerapon Vateekul. Deep learning for stock market prediction using event embedding and technical indicators. In *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, pages 19–24. IEEE, 2018.

[11] Xiangyi Zhou, Weijin Zhang, and Jie Zhang. Volatility spillovers between the chinese and world equity markets. *Pacific-Basin Finance Journal*, 20(2):247–270, 2012.

[12] Nikolaos Antonakakis, Ioannis Chatziantoniou, and George Filis. Dynamic co-movements of stock market returns, implied volatility and policy uncertainty. *Economics Letters*, 120(1):87–92, 2013.

[13] Nikolaos Antonakakis. Exchange return co-movements and volatility spillovers before and after the introduction of euro. *Journal of International Financial Markets, Institutions and Money*, 22(5):1091–1109, 2012.

[14] Kim Hiang Liow. The dynamics of return co-movements and volatility spillover effects in greater china public property markets and international linkages. *Journal of Property Investment & Finance*, 2014.

[15] Mika V Mäntylä, Daniel Graziotin, and Miikka Kuutila. The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32, 2018.

[16] Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

[17] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[18] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Arsa: a sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 607–614, 2007.

[19] Yancheng Hong and Steven Skiena. The wisdom of bookies? sentiment analysis versus. the nfl point spread. In *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[20] Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4, 2010.

[21] Tae Yano and Noah A Smith. What's worthy of comment? content and comment volume in political blogs. In *Fourth international AAAI conference on weblogs and social media*, 2010.

[22] Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology*, volume 1, pages 492–499. IEEE, 2010.

[23] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296, 2010.

[24] Eldar Sadikov, Aditya Parameswaran, and Petros Venetis. Blogs as predictors of movie success. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 3, 2009.

[25] Mahalia Miller, Conal Sathi, Daniel Wiesenthal, Jure Leskovec, and Christopher Potts. Sentiment flow through hyperlink networks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, 2011.

[26] Robert P Schumaker, Yulei Zhang, Chun-Neng Huang, and Hsinchun Chen. Evaluating sentiment in financial news articles. *Decision Support Systems*, 53(3):458–464, 2012.

[27] Jia-Lang Seng and Hsiao-Fang Yang. The association between stock price volatility and financial news–a sentiment analysis approach. *Kybernetes*, 2017.

[28] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[29] Roy Bar-Haim, Elad Dinur, Ronen Feldman, Moshe Fresko, and Guy Goldstein. Identifying and following expert investors in stock microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1310–1319, 2011.

[30] Ronen Feldman, Benjamin Rosenfeld, Roy Bar-Haim, and Moshe Fresko. The stock sonar—sentiment analysis of stocks based on a hybrid approach. In *Twenty-third IAAI conference*, 2011.

[31] Wenbin Zhang and Steven Skiena. Trading strategies to exploit blog and news sentiment. In *Fourth international aAAI conference on weblogs and social media*, 2010.

[32] Sahar Sohangir, Dingding Wang, Anna Pomeranets, and Taghi M Khoshgoftaar. Big data: Deep learning for financial sentiment analysis. *Journal of Big Data*, 5(1):1–25, 2018.

[33] Lixia Loh. Co-movement of asia-pacific with european and us stock market returns: A cross-time-frequency analysis. *Research in International Business and Finance*, 29:1–13, 2013.

[34] Heng Chen, Bento J Lobo, Wing-Keung Wong, et al. Links between the indian, us and chinese stock markets. *National University of Singapore, Department of Economics, Working Paper*, 602, 2006.

[35] He Nie, Yonghong Jiang, and Baoqing Yang. Do different time horizons in the volatility of the us stock market significantly affect the china etf market? *Applied Economics Letters*, 25(11):747–751, 2018.

[36] Peter D Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032*, 2002.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[38] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*, 2002.

[39] M Govindarajan. Sentiment analysis of movie reviews using hybrid method of naive bayes and genetic algorithm. *International Journal of Advanced Computer Research*, 3(4):139, 2013.

[40] Yahui Chen. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

[41] Xi Ouyang, Pan Zhou, Cheng Hua Li, and Lijun Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*, pages 2359–2364. IEEE, 2015.

[42] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[43] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[44] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[45] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[46] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

[47] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

[48] Jason Brownlee. *Long short-term memory networks with python: develop sequence prediction models with deep learning*. Machine Learning Mastery, 2017.

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

[50] Downloader api for gensim. `https://radimrehurek.com/gensim/downloader.html`.

[51] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*, 2019.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[54] Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. Improving bert fine-tuning via self-ensemble and self-distillation. *arXiv preprint arXiv:2002.10345*, 2020.

[55] Karpathy Andrej. Cs231n convolutional neural networks for visual recognition. In *CS231n Convolutional Neural Networks for Visual Recognition*. Stanford University, 2016.

[56] Richard Socher and Richard Socher Mundra. Cs 224d: Deep learning for nlp1. 2016.