Sentiment Analysis Using Deep Learning: A Comparison Between Chinese And English

MIAO WEI
Supervisor: Ronan Reilly
Department of Computer Science
National University of Ireland, Maynooth

Dissertation submitted for the degree of
Master of Science
at
Maynooth University

October 2017

Contents

List of Figures

Abstract

With the increasing popularity of opinion-rich resources, opinion mining and sentiment analysis has received increasing attention. Sentiment analysis is one of the most effective ways to find the opinion of authors. By mining what people think, sentiment analysis can provide the basis for decision making. Most of the objects of analysis are text data, such as Facebook status and movie reviews. Despite many sentiment classification models having good performance on English corpora, they are not good at Chinese or other languages. Traditional sentiment approaches impose many restrictions on the raw data, and they don't have enough capacity to deal with long-distance sequential dependencies.

So, we propose a model based on recurrent neural network model using a context vector space model. Chinese information entropy is typically higher than English, we therefore hypothesise that context vector space model can be used to improve the accuracy of sentiment analysis. Our algorithm represents each complex input by a dense vector trained to translate sequence data to another sequence, like the translation of English and French. Then we build a recurrent neural network with the Long-Short-Term Memory model to deal the long-distance dependencies in input data, such as movie review. The results show that our approach has promise but still has a lot of room for improvement.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work, except where specifically indicated in the text.

Miao Wei

Oct, 2017

Acknowledgements

I would like to express my sincere gratitude to my supervisor Ronan Reilly for his constant support, advice and patience. I appreciate his vast knowledge and skill in many areas.
I am also grateful to Joseph Timoney and Barak Pearlmutter for many helpful advices in writing reports.
Finally, I want to thank my families and friends for always have faith in me.

Chapter 1

Introduction

Since web technologies and e-business development, sentiment analysis as a critical area in Natural Language Processing (NLP) is widely used in a lot of places, such as customer analysis and product review systems, recommender system and Question & Answer system etc. Generally explain, sentiment analysis is a method that mining the author's or others' opinion through analysing the review text or comments (Yi, Yi, Nasukawa, Bunescu, & Niblack, 2003). Thus sentiment analysis also can be called as 'opinion mining' (Pang & Lee, 2008). The purpose of sentiment analysis is studying people's sentiment or rating contents towards specific objects, then extracting the useful information which is held in the text contents, such as attitudes of the writer, emotional reaction about a document and evaluation levels. According to the sentiment analysis result, the decision maker can be more accurate to improve their products (Dave, Lawrence, & Pennock, 2003) or give the appropriate response in the recommendation systems (Terveen, Hill, Amento, McDonald, & Creter, 1997) .

1.1 Background

The traditional interpretation of Sentiment Analysis is usually to use some means of investigation to obtain product feedback or opinions about particular objects, like prediction in political polls (Tumasjan, Sprenger, Sandner, & Welpe, 2010), or the evaluation of movies (Tatemura & Junichi, 2000). However, after the data has been collected, the decision maker then still needs to spend a significant amount of time to analyse the raw data which resulted from the investigation. There has now been an explosion in the availability of data through the internet as it is a significant resource as millions of users can put their variety of opinions online.

In our daily life it is possible to find information on so many things either on social media or some professional website, such as Twitter, IMDB, eBay, or Amazon. This kind of useful information is mostly available in text form. At the same

time though, users not only write a text comment for the object but also need to give a tag star level for the object, like Amazon.com or tabao.com star rating system. In addition to this, the website will process the collected data which is written by all the customers and compute an average rating score, and will furnish the average score to customers by way of a decision support. It is reasonable to assume that people's shopping behaviour patterns have been changed by the proliferation of reviews online.

According to a report analysis, the availability of approved online reviews have had a significant impact on offline purchases (David Kirkpatrick, 2016). In the report, it says that almost 82 percent of customers prefer to use their smartphones as shopping assistants rather than to consult a shopping guide. They would like to check the target products' pricing and reviews. Figure 1.1 illustrates the impact of online research reviews across different product categories. Notably for technology items of appliances and electronics the influence of reviews is the greatest.

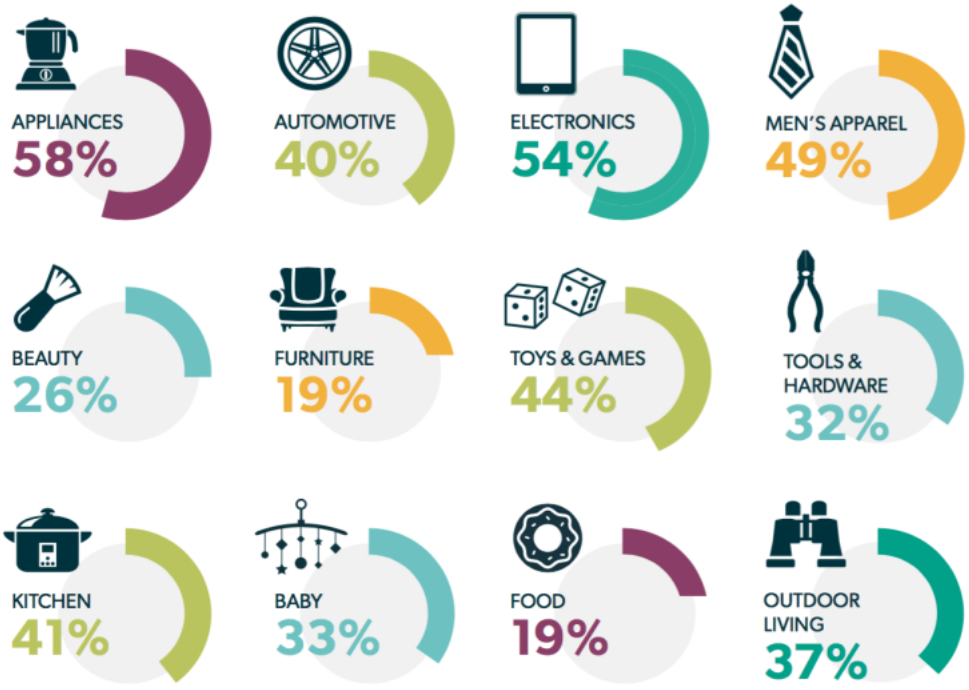

Figure 1 Level of influence that reviews have on different product purchases("Report: For every $1 online influenced by reviews, offline impact at least $4," 2016)

Thus, it can be taken from this point that sentiment analysis of online reviews can generate significant value for users. At the same time, according to the characteristics of different industries, sentiment analysis of the text on the social

media or another platform would have a positive indicator effect to a marketer or investment banker attempting to predict the trend of the market for a particular product. The ‘Twitter mood predicts the stock market’ paper (Bollen, Mao, & Zeng, 2010) explains how there is a correlation between the stock market and the public mood which is sourced from Twitter. This analysis demonstrated how they used 10 million tweets from 2008 to predict stock market trend.

Sentiment analysis is important but the huge level of data mean that it cannot be done manually and needs computer algorithms to implement it. These have to be efficient to be able to deal with the large amounts of information. Also, they need to be accurate because understanding off natural language is not easy.

There are a large variety of models and approaches powering NLP application, and these models achieved great performances, such as machine learning models (Pang, Lee, & Vaithyanathan, 2002). In traditional NLP approaches, sentiment classification is treated as a topic-based text categorization (Lin & He, 2009) or lexicon-based classification problem (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011). However, these approaches normally require developers have rich prior-knowledge and manually set the corresponding features. Recently, deep learning approaches have obtained the state-of-the-art performance across many different tasks, most notably visual classification problems. Deep neural networks (DNNs) not only achieved the brilliant performance on the image processing area (Nguyen, Yosinski, & Clune, 2014;Dosovitskiy & Brox, 2015), but also enhanced the accuracy of NLP tasks, such as machine translation and language understanding problems (Sutskever, Vinyals, & Le, 2014) (Cho, van Merrienboer, Gulcehre, et al., 2014). Compared to traditional NLP models that require developers manually specify or extract the features from data, deep learning can automatically learn the features (dos Santos & Gatti, 2014; Socher et al., 2013). However, most of these NLP models are trained in English corpus, and there is some difference between languages, such as Chinese and English. In our work, we will cover word vector representations, neural networks, recurrent neural networks, long-short –term memory as well as context vector model and implement the above models try to explore different corpus whether produces a different result for the above models.

1.2 Motivation

Sentiment analysis can be described as having three levels: the primary task in sentiment analysis is the polarity classification of a document or a given input text, sentence. Polarity classification is based on a hypothesis that the overall opinion in an opinionated text is about one single issue or item, classify the opinion as falling under one of two opposing sentiment polarities (positive and negative) (Pang & Lee, 2008). The second stage is the fine-grained analysis that ranking the attitude of an object into 5 levels. Fine-grained analysis also can be considered as how positive is the particular document. The advanced level of sentiment analysis is concerned with the analysis of the components of a text, the holder of attitude, target of attitude, type of attitude. In our work, we mainly focus on the primary task.

The traditional approach which has been applied to process text and to predict the sentiment typically focuses on the topic model, for instance, Latent Dirichlet Allocation Category Language Model (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). Most previous research on sentiment classification more or less used knowledge-based ways. Subsequently, there are many efforts that try to implement the model which relies on the non-topic-based text categorisation. The work of (Hatzivassiloglou, McKeown, Hatzivassiloglou, & McKeown, 1997; Turney, 2002; Turney & Littman, 2002), They performed work which focuses on classifying the semantic polarities of words or phrases through using linguistic heuristics or pre-selected emotional attribute dictionary. However, these models normally are limited to many factors, the accuracy of traditional approaches depend on the prior-knowledge and the semantic orientation dictionary, For example, dictionary-based approach requires developers to collected manually with know orientation words from corpora WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1990) or thesaurus (Mohammad, Dunne, & Dorr, 2009). However, it is difficult to collected the similar resource for other languages, such as Chinese.

Now, machine learning and deep neural networks have become accepted tools and have been shown to be effective once they are trained to do classification an regression tasks, many efforts on have been made to apply the machine learning models to tasks associated with sentiment analysis. Bo pang and Lillian Lee (Pang

et al., 2002) considered the issues of using overall sentiment to classify a document through predicting by applying basic machine learning models such as Naive Bayes and Support Vector Machine. However, their models did not obtain the better performance than the traditional topic-based categorisation (Zhou, Li, & Liu, 2008).

Because machine learning cannot directly process the input text, many works prefer to build a vocabulary and use the word index to instead of the text in the sentence. However, machine learning models require a fixed-length input. To better fit the problem to the machine learning model and deep neural network models, the word index approach should be replaced by another efficient representation model. As usual, NLP systems treat words as discrete atomic symbols, such as “one-hot” representation and bag-of-words representation.

Unfortunately, there are three weaknesses in “one-hot” representation and bag-of-words representation (Turian, Ratinov, & Bengio, 2010). Firstly, for one-hot representation, because of randomly give value to individual word, thus this model can leverage very little useful information to systems, like the similarity of words and context relationship between sentences. Then, discrete data leads to data sparsity, means we need extra processing to solve the sparsity or reduce the dimensionalities or collect more data to obtain a statistical work model . Finally, and also is the most critical point, they lose the ordering of the words and ignore the semantics of the words (Le & Mikolov, 2014).

Whether in Chinese or English, context is a rarely used information resources in current computing tasks, even though context data can naturally improve the understanding of overall text sentiment, particular in Chinese. Previous research consider a document or input text is exist by the form of sequence data, with a word appears one after another, the whole document sentiment will be changed at any time if the new word appears. This situation could be considered as the “Garden Path” problem (Strzalkowski, 1999), because the overall main sentiment of a document is unknown until reading it to the very end. Based on the these reasons, a model is required that not only can efficiently process the sequence data and analysis the input text, but also can judge which history information should be removed when the model is learning the representation of context. For instance:

“Chris Craft is better looking than Limestone, but Limestone projects seaworthiness and reliability.”(Wikipedia)

Through analysing the above sentence, two brand names and two attitudes apparent. In the first half, the sentence showed an intense negative emotion for ‘Limestone’, but in the second half, the expression of emotion turned to positive and changed the semantics polarity of the whole sentence. Thus it is important to consider finding an approach that can forget the first half sentence information and correct the state information when it captures new data, from which it can automatically generate the overall sentiment which relies on the viewpoint in the second half as the core.

With the help of Long-Short-Term Memory (LSTM) RNNs, the model can determine whether needs to abolish the past information in the memory block structure (Hochreiter & Schmidhuber, 1997). LSTM and RNNs achieved excellent performance on some problematic sequential issues such as speech recognition and machine translation (Cho, van Merrienboer, Gulcehre, et al., 2014). Based on the hypothesis of the work (Cho, van Merrienboer, Gulcehre, et al., 2014) that they consider RNN encoder structure can map an input sequence to a fixed length state vector. This state vector (or context vector) is a summary of the whole input sequence. Hence, we proposed a RNNs as the encoder to represent context vector from sentences or documents (Cho, van Merrienboer, Gulcehre, et al., 2014). Then, we used the context vector of paragraphs as the input of machine learning classifier to classify the sentiment of documents.

Subsequent the intention is to examine the difference between Chinese and English based on the same vector representation model or the corresponding classifier, such as Word2vec with support vector machine (D. Zhang, Xu, Su, & Xu, 2015), Bag of words with the neural network. The main interest is to study that if the language is different, whether it will result being assigned a different affection. Additionally, a significant difference between Chinese and English during raw data processing stage is that Chinese corpora input data should be firstly segmented but English not. The reason is explained in section 4.1.3. After segmentation, we used the same vector representation model to train the machine learning algorithms for

the Chinese corpus and the English corpus respectively. The primary vector learning models investigated is the one-hot representation approach, word2vec. For the classifier model, The most classical machine learning model, the SVM is selected as the baseline classifier. It is compared with the currently most popular multilayer neural network and deep neural networks (DNNs), the LSTM (Sutskever et al., 2014) / Gated Recurrent Unit (GRU) model (Chung, Gulcehre, Cho, & Bengio, 2014), which was trained in advance for the classification job.

At the final stage, we would try to implement the RNNs encoder structure from Sequence-to-Sequence model as an independent context-to-vector model (Context2vec), context2vec which we expect can learning the vector representation with high efficiency. According to the theory of previous work, 'After reading the end of the sequence, the hidden state of the RNN is a summary c of the whole input sequence.' (Cho, van Merrienboer, Gulcehre, et al., 2014), We propose a hypothesis that the hidden state vector is a context vector, and this context vector should have the ability to express the overall sentiment of a document or a piece of input sequence. The context2vec structure is a part of the sequence-to-sequence model which was used to finish the machine translation and got a brilliant result (Cho, van Merrienboer, Gulcehre, et al., 2014; Sutskever et al., 2014; Luong, Sutskever, Le, Vinyals, & Zaremba, 2014; Bahdanau, Cho, & Bengio, 2014). The sequence-to-sequence model consists of two blocks: RNNs encoder, RNNs decoder. The Encoder's main function is transit the input data one by one to a fixed length state vector. While the decoder is used to train the target output and think of the condition of using the state vector which passed from the encoder. This Sequence-to-Sequence model achieved the state of art performance on the task of Machine Translation (Sutskever et al., 2014).

Chapter 2

Related Work

The objective of this chapter is to give an overview of the basic concepts used in this thesis. It describes the two main components in the sentiment analysis process and some related models. The first component is the embedding model which used to map text words to vectors. It presents the theories underlying commonly used approaches TF-IDF (Ko & Youngjoong, 2012), N-gram, Bag-of-words and Word2vec model (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). In addition, the word2vec model has been shown the state-of-art performance in sentiment analysis. The section 2.2 introduces various classifier techniques for classifying the target label and the sentiment polarities from the input data.

2.1 Word and Phrase Embedding

Word and phrase embedding is a feature learning methods in NLP (Mikolov, Sutskever, et al., 2013). In the traditional NLP tasks, people prefer to treat words as discrete atomic symbols, like in the “one-hot” representation, for example, “King” could be represented as ‘0001’ and ‘Queen’ can be represented as ‘1000’. However, discrete atomic symbols cannot show the relationship and similarity between words. At the same time, the large discrete data maybe leads to data sparsity (Friedman, 1997). This is an issue because to overcome the problem and to successfully train statistical models usually lots of data is required or if it is too large specific dimension reduction methods can be applied. Besides, data sparsity can result in what is known as “ the curse of dimensionality” (Friedman, 1997). It is important that using discrete atomic symbols cannot represent the semantic similarities between linguistic items. Thus the distributional hypothesis (Harris, 1981) was proposed by Harris.

Due to the distributional hypothesis theory, which states that words that appear in the same contexts share semantic meaning (Harris, 1981), two categories models were constructed: count-based methods and predictive methods. The difference is count-based methods just computes the statistics of the frequency of

words co-occurs with its neighbour words in a large corpus, and then map the collected count statistics into a small, dense vector in the vector space for each word (Baroni, Dinu, & Kruszewski, 2014). In contrast, the predictive models directly learns a small, dense set of embedding vectors through predicting a word from its neighbour words. The typical representative of count-based methods is Latent Semantic Analysis approach (LSA) (Deerwester et al., 1990). The classical model of predictive methods is neural probabilistic language model (Yoshua Bengio et al., 2003).

2.1.1 Latent Semantic Analysis approach: Bag of Words

The Latent Semantic Analysis is a technique in count-based methods. It is used to analyse the relationships between the document and words that it contains. The distributional hypothesis is the fundamental principle of LSA. LSA can use a term-document matrix which describes the occurrence of terms in documents. To extract features of co-occurrence matrix, some mathematical methods always can be used in the process, such as Singular Value Decomposition (SVD) (Golub & Van Loan, 1996) and Non-negative Matrix Factorisation (NMF) (D. D. Lee & Seung, 2001).

In order to represent the co-occurrence relationships, LSA chose a way that computes the frequency of occurrence words in a set of documents and then produces a term documents matrix to describe frequencies. The method was called term-document (TD) matrix. However, the term-document matrix is limited by unbalance-size documents. For example, a word appears 3 times in a document which only has ten words, and appears 100 times in another document which contains 1000. This condition is called as unbalance-size document set. Finally, the TD matrix cannot accurately measure the relationship between words and documents. For example, the big part of documents usually contain lots of stop words, like ‘the’, ‘a’, ‘and’, this kind of words always is the highest frequency in a text. To address this problem, TD matrix was replaced by another more accurate measure way, TF-IDF. The term frequency-inverse document frequency (TF-IDF) is a popular normalisation way which gives weight to per term by the inverse of document frequency. Generally speaking, either TF or TF-IDF is measure methods of bag-of-words (Martineau et al., 2008). In the bag-of-words model, a text or a

sentence is represented as the bag of its words, it ignores the word order and word grammar.

The BOW is a very popular model which simplifies the process of representing the input text as fixed-length feature vectors. It exhibits good performance for topic classification. However, it has two significant weaknesses: 1. It loses the ordering of the words. Due to lack of consideration to the word order, if different sentences with the same words but in a different order are used they will have the same representation; 2. It ignores semantic relationships between words (Le & Mikolov, 2014). Thus, BOW has no ability to reveal any information about context data. As an alternative, people have used the N-gram model instead to reveal the context data (Cavnar, Cavnar, & Trenkle, 1994).

2.1.2 N-gram approach

Even though the bag-of-words get a good performance on the topic classification, it still has some weakness that it ignores the order of words, in other words, it cannot reveal more information about context data (Le & Mikolov, 2014). As an alternative, people have used the N-gram model to capture the context data as more as possible (Cavnar et al., 1994).

The N-gram model is widely used in the machine translation, words correction system and speech recognition field (Kukich & Karen, 1992). The N-gram algorithm computes the probability of a sequence data and use the maximum likelihood estimation technique to predict the next word. When we are using a language model to predict the next word, the core theory is derived from Markov algorithm which assumes that the current word depends on the previous $n-1$ words. This is a crucial assumption to simplify the estimation problem. Normally, the different size of N-gram has different term as following table 1:

Term of N-gram	Size (n)
Unigram	1
Bigram	2
Trigram	3

Four-gram	4

Table 1 Different size of N-gram has different name

Due to one algorithm in word2vec model, Skip-grams, is partially similar with N-gram (Mikolov, Sutskever, et al., 2013), and to better understanding the word2vec algorithm, now we will give an example to show how the n-gram represents the words relationship in the text. Suppose we have a corpus of three sentences. We utilize a bigram to analysis the corpus.

<s> I am Sam </s>

<s> Sam I am</s>

<s> I do not like green eggs and ham </s>

Because the n is 2, so we calculate the probabilities of couple words occurrence as following:

$$P(\ I \mid < s >) = \frac{2}{3}$$

Equation 1

Word 'I' appears twice after start symbol '<s>'. Other probabilities are the same.

$$P(Sam| < s >) = \frac{1}{3};\ P(am|I) = \frac{2}{3};\ P(do|I) = \frac{1}{3};\ P(</s > |Sam) = \frac{1}{2};$$

Equation 2

So from the above we can see that the N-gram model to a certain extent take into account the order of words.

2.1.3 Neural Probabilistic Language Models

The neural probabilistic language is a common type of predictive method in word embedding, and it is based on neural networks to learn distributed representations model (Yoshua Bengio et al., 2003). The basic idea of the neural language model is to learn to associate each word in a dictionary with a continuous-valued vector representation. Neural probabilistic language models have the ability to effectively

solve the curse of dimensionality which occurs in the traditional feature extraction method (Yoshua Bengio et al., 2003).

Due to the data sparsity issues, learning algorithms usually need a large amount of training data when attempting to solve some complicated problem. This requirement is called as the curse of dimensionality (Yoshua Bengio, Courville, & Vincent, 2012). The most common case is when the number of training examples increases exponentially as the number of features which need to be learned grows. So it always happens with natural language models that the input is large tuples of sequences of text data. For example, if we want to use our model to joint distribution of 10 words from a vocabulary V which size is 100000, it will generate $10^{50} - 1$ free parameters (Yoshua Bengio et al., 2003).

2.1.4 Word2vec Model

Word2vec is one of useful neural probabilistic language models (Mikolov, Sutskever, et al., 2013). It achieved the state-of-arts performance in lots of NLP tasks which produce fixed-length vectors used to represent words(Mikolov, Sutskever, et al., 2013). The basic idea of word2vec is that according to the input training corpus, it should produce a vector space. It then assigns a corresponding vector to each unique word and map it into the vector space. The word2vec model includes two opposing algorithms: CBOW and Skip-Gram. The distinction between CBOW and Skip-gram is elaborated in (Mikolov, Corrado, Chen, & Dean, 2013).

2.1.4.1 Continuous Bag-of-Words (CBOW)

The CBOW can be illustrated using a simple example. Suppose we have a sentence ‘The cat sits on the mat’, we would like to treat the ['The', 'cat', 'on', 'the', 'mat'] as the context words, and then use these context words to predict the missing word ‘sits’. From the above sentence, we can find the CBOW will rely on the context words to output the current word.

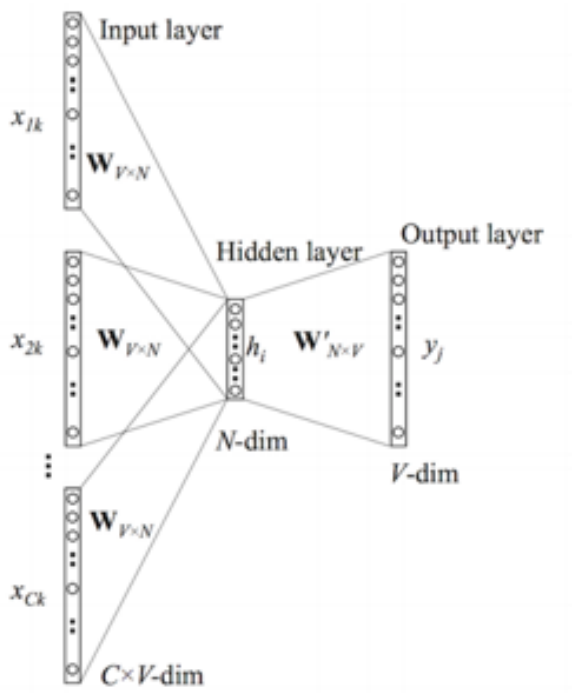

Figure 2 CBOW Structure ("CBOW of Word2Vec," 2017)

Figure 2 shows the CBOW structure, the CBOW structure consists of three layers: input layer, hidden layer and output layer. We will explain the parameters in the section 2.1.4.2.

2.1.4.2 How to using the Word2vec model

Firstly, we should define our model. We set the window size value to be m and the vocabulary size to be $|V|$. For the input word or context words X_c is used to define it. The output word is represented by Yc .Then a three layer neural network can be created to learn the embedding vectors. The input layer will captures C words that the word window size is C, and pass the encoded word which has weighted by W into the hidden layer which constructed by a fixed length N-dimensional vector. Finally, the operated data will transit to the output layer and generate the output word, Y_c. The weight matrix which connects between the input layer and hidden layer should be a $V \times N$ matrix, W. On the other hand, the weight matrix between the hidden layer and output layer should be a $N \times V$ matrix which denoted as W'.

Assuming we have already learned the weight matrix W and W'. The whole process could be divided into several steps:

1. The input raw text should be encoded into one-hot representation vectors which use a binary code to replace unique word in the vocabulary. The one hot representation vector length equals to the vocabulary size $|V|$.
2. According to the window size C capture the context words which surround current word Y_c. Then words captured by the window should be replaced by encoded word vectors.

$$(x_{(c-m)}, \ldots\ldots, x_{(c-1)}, x_{(c+1)}, \ldots\ldots, x_{(c+m)}) \text{ where } x_i \in X.$$

Equation 3

3. Using the model generated embedding word vectors $v \in R^{n\times|V|}$ to replace of words in the context window.

$$(v_{(c-m)} = Vx_{(c-m)}, v_{(c-m+1)} = Vx_{(c-m+1)}, \ldots\ldots, v_{(c+m)} = Vx_{(c+m)})$$

Equation 4

4. Average the collected vectors to get

$$\hat{v} = \frac{v_{(c-m)} + v_{(c-m+1)} + \ldots\ldots + v_{(c+m)}}{2m}$$

Equation 5

5. Generate the score vector

$$\boldsymbol{z} = W'\hat{v}$$

Equation 6

6. Using the softmax function to measure the score which obtained from the former step, and choose the max probability word as the target word.

The above description briefly introduced how to get the embedding vector. Next, it will be illustrated how the output is computed from the raw input data. Firstly, it is necessary to evaluate the hidden layer H. The evaluation equation is:

$$h = \frac{1}{C}W \cdot \left(\sum_{i=1}^{C} x_i\right)$$

Equation 7

the hidden layer output is the average of the input xi weighted by the matrix W. Then, to compute the out layer result.

$$W_j' = v_{w_j}'^{T} \cdot h$$

Equation 8

where v_{w_j}' is the j th column of the output matrix W'.

Finally, the softmax function computes the output y_j from

$$y_j = p(w_1, \ldots, w_c) = \frac{exp(v_j)}{\sum_{j=1}^{V} exp(v_j')}$$

Equation 9

2.1.4.3 Updating the hidden-output layer weights

We defined the object function by cross entropy as:

$$E = -log\, p(\omega_o \mid \omega_I)$$
$$= u_{j*} - log \sum_{j=1}^{V} exp(u_j)$$

Equation 10

According to the above loss function, the derivative of the loss function E regards with the j^{th} node in the output layer u_j is given by

$$\frac{\partial E}{\partial v_j} = y_j - t_j$$

Equation 11

where $t_j = 1$ if j = j*, otherwise $t_j = 0$.

Then, the chain rule is applied to compute the above equation.

$$\frac{\partial E}{\partial \omega_{ij}'} = \frac{\partial E}{\partial \mu_j} \cdot \frac{\partial \mu_j}{\partial \omega_{ij}'}$$
$$= (y_j - t_j) \cdot h_i$$

Equation 12

At the end, the update principle of the gradient descent approach is followed and the eqns. is obtained

$$\omega_{ij}'^{(new)} = \omega_{ij}'^{(old)} - \lambda \cdot (y_j - t_j) \cdot h_i$$

Equation 13

Where λ is the learning rate.

2.1.4.4 Skip gram architecture of word2vec

The skip gram architecture is within word2vec. Because of the highly efficient learning capabilities of the Skip gram model, it usually used to train a large corpus. The CBOW, mentioned in Section 2.1.4.1, is based on the surrounding words to predict the target. However, the Skip Gram model is just the opposite in that it uses a specific word in the middle of the window (that is, the input word x_c) to infer the context words. Consider the simple example to obtain an insight into the Skip gram structure:

"The cat sits on the mat."

We used a window whose size is 1 for this example. So the input will follow the format (context, target) pairs given below. Specifically, the Skip-gram model just inverts the order of context and target words.

Source Text	Input
"The cat sits on the mat."	([#, cat], The)
"The cat sits on the mat."	([The, sits], cat)
"The cat sits on the mat."	([cat, on], sits)
"The cat sits on the mat."	([sits, the], on)
"The cat sits on the mat."	([on, mat], the)
"The cat sits on the mat."	([' . ', mat], mat)

Table 2 Skip-gram input with size 1

The Skip-gram architecture can be simply inverted as figure 3.

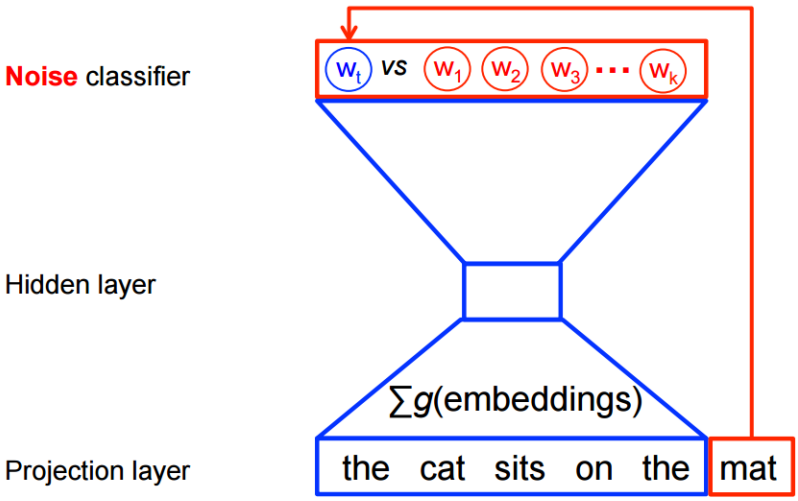

Figure 3 Skip-gram structure("Vector Representations of Words | TensorFlow," 2017)

The structure of Skip-gram is similar with CBOW, it also has three layers. The input layer reads the target words, and the output layer predicts context words.

Then the process is similar to CBOW process. The goal is to decrease the cost function through updating the embedding parameters by applying gradient descent algorithm. Finally, it can be used to represent the semantic relationship between words, and it has been shown to achieve an excellent performance on lots of NLP tasks ("Chinese comments sentiment classification based on word2vec and SVMperf," 2015; Mikolov, Sutskever, et al., 2013).

Although word2vec has successfully shown that it has the ability to represent the semantic mean, but it still has the weakness that it does not consider the order of words and sentences (Le & Mikolov, 2014; Socher, 2014). To solve the order-lost issue, Mikolov proposed a new algorithm, Doc2vec (Le & Mikolov, 2014). This algorithm can learn a paragraph vector which can contain more context information than word2vec model. Empirical results show that paragraph vectors outperform count-based methods on several text classification tasks and sentiment analysis tasks (Le & Mikolov, 2014).

Word embedding technologies have a prominent position for NLP tasks. However, they are just the first stage. In the following section, the aim is to give the background into the machine learning classifiers and provide insights into how the

machine learning helps the implementation of the classification tasks based on the word embedding vectors.

2.2 Machine learning & Neural networks for classification

Three standard machine learning and neural network algorithms are Support Vector Machine (Pang et al., 2002), Multilayer Neural Network (Aggarwal & Zhai, 2012) and Recurrent Neural Networks (LeCun, Bengio, & Hinton, 2015). Machine learning tasks can be divided into three broad categories: 1. Supervised learning, 2. Unsupervised learning and 3. Reinforcement learning. In the thesis, more attention will be given to the supervised learning approaches, particularly the classical supervised learning model: the Support Vector Machine.

2.2.1 Support Vector Machine (SVM)

Given a set In machine learning, the Support Vector Machine is the classical learning algorithm which widely used in many tasks, such as classification (Suykens & Vandewalle, 1999). It can be illustrated as follows: Given a set of labelled training examples, each example belongs to one of two classes, 0 and 1.

The principle of the SVM's principle is to map each sample as a point in an *n*-dimensional vector space. Then, it will build a high dimensional hyperplane to separate the clustered data points. SVM has two kinds' classifier functions, one is linear SVM and another is non-linear SVM. Intuitively, we need to find the best separation hyperplane which has the largest margin between two classes, where margin is defined as the distance to the nearest any-class training data point. In other words, we need to find the maximum-margin hyperplane.

2.2.1.1 Linear SVM

To explain the Linear SVM classifier's theory the first step is to clarify the format of the training dataset of n points. The i-th sample in the training data set is denoted by x_i, and the i-th target label is denoted y_i. These are as Cartesian coordinates

$$(x_1, y_1), (x_2, y_2) \cdots (x_{n-1}, y_{n-1})$$

Equation 14

Where y_i is either 1 or 0.

In binary classification problem, 1 and 0 mean two classes that the case x_i belongs to either one of two classes. Additionally, the x_i is a vector with a fixed length features. In the next step the data is mapped into an n-dimensional feature space. The SVM try to find the best separating line by searching the support vectors that closest data points to the decision hyperplane. For linear SVM, it is treats the data as being linearly separable. This means that linear SVM can be considered in the way that both data points in space can be separated by a straight line. In Figure 4 an example is provided with a set of 2-D data points. The circles and squares representing similar data values and appear as being clustered together. It is easy to visualize a line to distinguish the clusters.

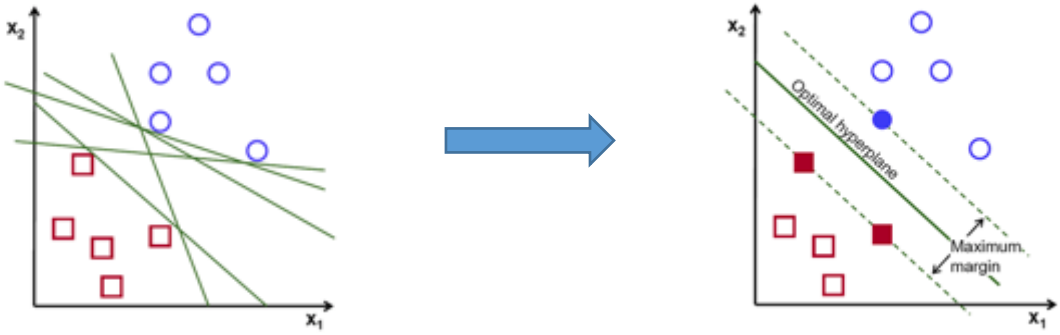

Figure 4 A linear separable set of 2D-points (opencv dev team, 2017)

Figure 4 describes the learning process that the SVM learn and determine the optimal hyperplane from many solutions which denoted by multiple lines to the problems. The optimal hyperplane has the largest minimum distance to the training examples. At the same time, the optimal decision hyperplane maximizes the margin of the training data that the distance between the support vectors.

The linear SVM could only solve the linear separable issues, but in some cases this is not sufficient because the data is not linearly separable. In this case a non-linear classifier is required.

2.2.1.2 Nonlinear SVM

The linear SVM as a linear classifier could process the linearly separable data very well, but it has the weakness that it can not correctly divide the non-linear data.

Hence, (Boser, Guyon, & Vapnik, 1992) proposed a way to classify the non-linear data by applying the kernel function. The core point of nonlinear SVM is the SVM will implement the kernel function to map the input vector from a low dimensional to a high dimensional space. Since the data points are linearly indistinguishable and cannot be linearly classified in the low dimensional space, the kernel function leads to the data point becomes linear separable in a high dimensional space. Thus the kernel function allows the nonlinear SVM to fit the optimal hyperplane in a transformed feature space. This is illustrated in Figure 5.

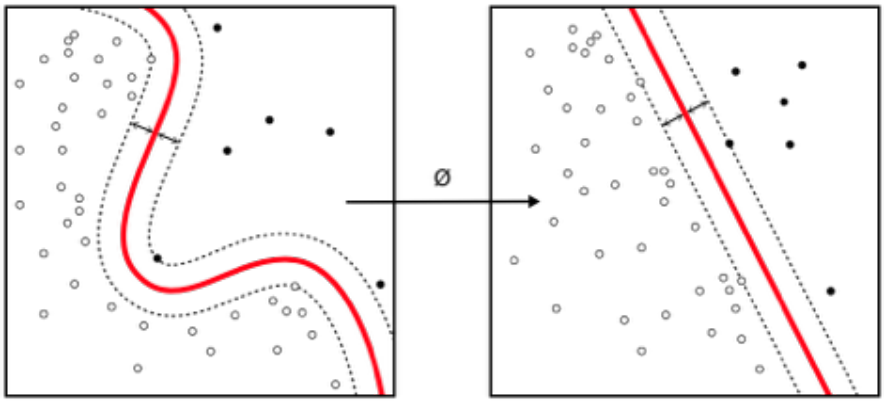

Figure 5 Kernel machines are used to compute a non-linearly separable function into a higher dimension linearly separable function (Alisneaky, 2011)

2.2.2 Multilayer Neural Network

Consider an alternative supervised learning method to SVM, multilayer neural networks have also been shown to achieved a good performance on the classification tasks (“Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” 1998). Neural networks are inspired by the biological neural networks. Currently, neural systems are widely used in the pattern recognition (Bishop, 1995), image processing (“Image processing with neural networks—a review,” 2002) and machine learning area.

In the human brain, the transmission of information is relayed on neurons. These neurons are connected to each other and eventually form an efficient and sophisticated network. The same thing also happens in the neural network model, and the neural network model is usually composed of one or more layers, each layer contains a certain number of nerve nodes (Axon). Each neural node will be input with

a fixed length features vector (the input value also could be the other neurons output), then output a single real value (that can possibly be the input to other units subsequently).

2.2.2.1 Perceptron

Next, the basic unit structure of neural network model will be examined. The neural network consists of perceptrons. The simplest way to understand the perceptron algorithm is that the perceptron unit outputs a linear combination of its inputs by using a fixed-length vector as input, and then outputs 1 if the calculated result is greater than the threshold, otherwise, it equals -1. The input data is denoted as $X = (x_1, x_2, \cdots, x_n)$. The mathematical expression to represent this is as follows:

$$O(x_1, x_2, \dots, x_n) = \begin{cases} 1, & if\ (w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0) \\ -1, & if\ (w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n < 0) \end{cases}$$

Equation 15

Where w_i is the weight for every feature of input data and it determines the importance of features in the input data. Besides, w_0 is the bias value, and also can be called as the threshold value. Figure 6 shows the structure of the perceptron and how it works.

Normally, we will rewrite the above math equation as

$$O(\vec{X}) = sgn(\vec{W} \times \vec{X}) = \begin{cases} 1 \\ 0 \end{cases}$$

Equation 16

Where $sgn()$ means the activation function.

In practice, it is normal to choose among activation function, such as sigmoid function and $tanh$ function.

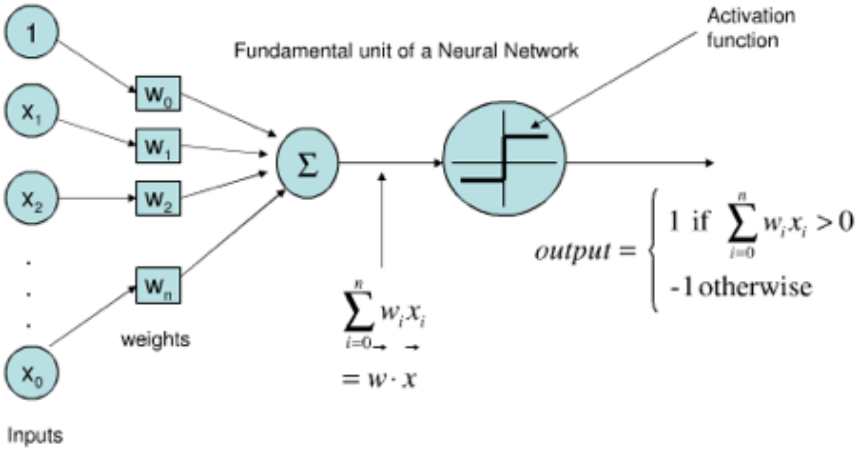

Figure 6 Procedures of a single layer perceptron network (source: Wikipedia)

So how to obtain the weight W is the primary task. To correctly predict the label of the target data, there are update approaches to update the weight values known as (a) the Perceptron rule (Freund & Schapire, 1999) and (b) the Delta rule (Russell, 2012). Either Perceptron Rule or Delta Rule can guarantee that the cost function will converge to a reasonable range, and finally captures the optimal weight parameter values.

In order to learn the optimal weight parameter, the first step is that it is necessary to initialize the weight matrix using random values. During learning progress, if the algorithm detects an incorrect prediction, it can automatically modify the corresponding weights and update them until a perfect fit for the training cases, and thus find the minimum cost value. The update rule as follows:

$$W_i = W_i + \Delta W_i$$
$$\Delta W_i = \alpha(t - o)x_i$$

Equation 17

Where α is learning rate, t is the target label, o is the perceptron output.

It should be noted that we usually set the learning rate to a small value. Otherwise, it will cause the loss function cannot be convergence and leads to the underfitting situation if we set the learning rate to a large value.

A premise of using the perceptron update rule is that training examples are linearly separable. If the training dataset is not linearly separable, another alternative

method can be applied to update the weight, the Delta rule, to prevent the occurrence of non-convergence situation. The Delta rule uses the gradient descent method to find the best approximation set of weights. We use the following mathematical expression to write the linear operation output of the perceptron.

$$o(X) = W \times X$$

Equation 18

The distinction between Delta rule and Perceptron rule is that the above equation does not include the threshold W_0. To compute the difference between the target function and our hypothesis function, a cost function must be defined to measure the training error of the hypothesis function.

$$E = \frac{1}{2}\sum_{j}(t_j - y_j)^2$$

Equation 19

Where E means the training error, t_j is j-th target output, y_j is the output label of training case j.
With the error of the cost function decreasing, we can get the fitness weight parameter.

2.2.2.2 Multilayer Neural Network

A single perceptron unit can be used to classify linearly separable data, but it is not suited for solve some complicated problems like speech recognition (Hinton et al., 2012) and machine translation. Thus people prefer to use a multilayer neural network to solve these for non-linear high dimensional training datasets ("Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," 1998). For example, we can analyse and judge the letter which is correct by identifying the sound spectrum. Multilayer neural network has the ability to solve the more complex issues.

A multilayer neural network consists of a tuple of neural nodes, and each node usually is a perceptron unit, so that the neural network which is made of multiple perceptron units also can be callas a multilayer perceptron (MLP). At the

same time, the output of each nerve node will be the input of the other neural nodes, except for the final output layer which only has one node. From figure 7, it can be observed that this multilayer neural network has three layers, and each layer has four nodes, represented as circles. Those circles which wrote +1 symbol indicate the threshold (which also can be called as bias).

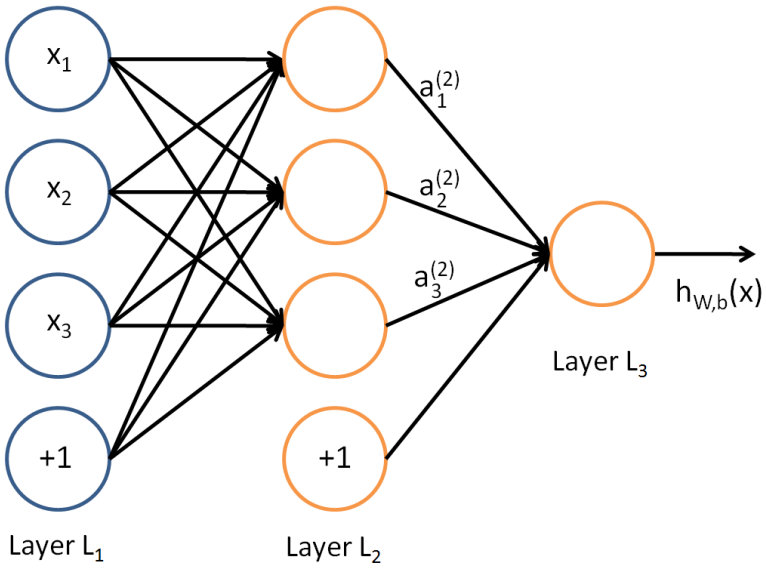

Figure 7 A neural network structure("Unsupervised Feature Learning and Deep Learning Tutorial," 2017)

In figure 7, the first layer L_1 is the input layer, and the second layer L_2 is the hidden layer, the third layer L_3 is the output layer. In figure 7, it can be seen that there are four nodes in the input layer and the hidden layer. However, considering the input layer and the hidden layer separately, they have three neural nodes each because of the last node in each layer is the bias node, it won't get any value from the previous layer. In the hidden layer, each neural node will collect data which is weighted by the parameters from the previous layer, then output to the next layer after a linear operation and activation processing. The mathematical expressions for each output in L_2 would be as follows:

$$a_1^2 = f(W_{11}^1 x_1 + W_{12}^1 x_2 + W_{13}^1 x_3 + b_1^1)$$
$$a_2^2 = f(W_{21}^1 x_1 + W_{22}^1 x_2 + W_{23}^1 x_3 + b_2^1)$$

$$a_3^2 = f(W_{31}^1 x_1 + W_{32}^1 x_2 + W_{33}^1 x_3 + b_3^1)$$
$$h_{W,b}(x) = a_1^3 = f(W_{11}^2 a_1^2 + W_{12}^2 a_2^2 + W_{13}^2 a_3^2 + b_1^2)$$

Equation 20

Where a_i^l denotes the activation function output value of unit i in layer l. W_{ij}^l means the weight parameter associated with the connection between unit i in layer l and unit j in layer $l+1$. b_i^l denotes the bias node value of unit i in the l layer. Eventually, $h_{W,b}(x)$ will output a real value whose computation depends on the previous layer output (a_1^2, a_2^2, a_3^2 and b_1^2).

The above algorithm is called feedforward neural network. It is operates on data from the input and processes it layer by layer, until finally output the result. The above MLP model will be used in our experiments to verify whether there is any difference between English corpus and Chinese corpus.

2.3 Assessing The Previous Work In Sentiment Classification

This section briefly reviews the previous work on sentiment classification. In many traditional text categorization methods' the performance depends on the knowledge and linguistic heuristics or the quality of the manual construction of discriminant-word lexicons (Pang et al., 2002). Supervised machine learning approaches have been widely used to do text categorization, especially through the application of the Support vector machine (SVM) (D. Zhang et al., 2015). Through finding a hyperplane for word vectors, SVM can achieve a good performance (Pang et al., 2002). However, SVM has some weaknesses. The major one is that it uses a kernel function to map the nonlinear data to a high dimensionalities space, so then it can find a hyperplane to do classification(Moreno & Ho Hewlett-Packard, 2003). This weakness increases the complexity of computation. Hence the SVM can't be used to process large data sets. Another weak point regarding SVM is that it doesn't consider the semantic relationship between words.

Approach	Advantage	Disadvantage
Support Vector Machine (SVM) (Pang et al., 2002)(Vanzo, Croce, & Basili, 2014)	1. SVM's generally outperforms the Naïve Bayes classifier; 2. It is a large-margin technique that uses a hyperplane unlike the probabilistic classifier (Pang et al., 2002).	1. One major weakness of SVM having to determine the kernel function to compute the distances among data points (Moreno & Ho Hewlett-Packard, 2003). 2. Semantic features have been rarely considered in SVM sentiment classification (D. Zhang et al., 2015). 3. The accuracy of the word vectors is positively correlated with the performance;
Word2vec + SVM (D. Zhang et al., 2015)	1. This considers the semantic relationship between words; 2. Because it has a low computational complexity, it can compute very accurate high-dimensional word vectors from large data sets (Mikolov, Corrado, et al., 2013);	1. It ignores the context information; 2. It does not take account of word order; 3. it lacks the ability to process long range text.

LSTM (Huang, Cao, & Dong, 2016)	1. It can process long-range context, solving the independence issue; 2. It considers the word order and has the ability to extract the overall sentiment from the context; 3. It solves the general sequence to sequence problem (Sutskever et al., 2014).	1. The size of the training data set determines the performance of the models; 2. From the literature it appears that it is better to apply it in conjunction with another technique (Wang, Yu, Lai, & Zhang, 2016).

Table 3 The Comparison of Different approaches

To observe the semantic relationship and improve the performance of processing of high dimensionality data, the word2vec model was proposed and was shown to achieve excellent results (Tang et al., 2014). The Word2vec model has a high efficiency for learning word vectors from the raw text by using the Skip-gram or CBOW model. More recently, the word2vec model has also been shown to offer significant improvements over other techniques for Chinese corpus (D. Zhang et al., 2015).

Although word2vec can directly map words into high-precision vectors and represent semantic relationships between words, the extraction of contextual semantic information and word order are ignored (Le & Mikolov, 2014). In order to solve these problems, researchers have tried to better understand the overall sentiment of text passages by increasing the processing granularity from the level of a word to that of a paragraph or a document. They then proposed various of algorithms, such as Paragraph vector (or Doc2vec) (Le & Mikolov, 2014), Recursive neural network (Socher, 2014), or RNN and LSTM (Chung et al., 2014; Huang et al., 2016). Other works verified that RNN and LSTM have greater potential to produce better predictions of sentiment polarity than other models (Le & Mikolov, 2014)(Tai, Socher, & Manning, 2015). In particular, (Tang, Qin, & Liu, 2015)'s work utilized LSTM and RNN with many GRU for document level sentiment classification of the IMDB dataset.

Therefore, for the problem of the thesis it appears that the best approach is to use a form of LTSM with RNN. However, it has to be noted that this approach has not been verified on a Chinese corpus so it may not produce results that are as good as report for English corpora.

Chapter 3

Methodology

This chapter we begin by describing context2vec methodology and the algorithm. Context encoder is a main component of Sequence-to-sequence model. RNN encoder-decoder structure is the basic idea. Recurrent Neural Network (RNN) has shown outstanding performance in many problems that require sequence processing, such as speech recognition (LeCun et al., 2015; Graves, Mohamed, & Hinton, 2013), machine translation (Cho, van Merrienboer, Gulcehre, et al., 2014).

The main advantage with the RNN is that it takes the word order into account, and thus is suitable to tackle a sequence problem. In traditional Natural Language Processing (NLP), for simplicity the viewpoint was often held that words are independent from each other, , and the word order does not affect the final result (Taboada et al., 2011).

However, when facing issues associated with analyzing the whole semantics, including sentiment, of a sentence or whole documents, such a hypothesis that words order is not considered in the prediction will result in a low accuracy of sentiment inference. For example, in Chinese, because of the existence of particular word segmentation rules and the fact that a combination of different orders between words can cause a totally different meaning, the algorithm based on the phrase independence theory will always leads to a low accuracy that measuring the expressiveness of the overall sentence. To overcome such difficulties Mikolov proposed the implementation of RNN for language in his work (Le & Mikolov, 2014). This work illustrates that the consideration of word order with previous input information can promote the accuracy of prediction.

3.1 Model Overview

The main goal of our model is to learn a generic context embedding vector for variable-length sentence. To do this, we propose a novel neural network architecture, which is based on the Recurrent Neural Network Encoder-Decoder model (Chung et al., 2014; Cho, van Merrienboer, Gulcehre, et al., 2014; Cho, van

Merrienboer, Bahdanau, & Bengio, 2014). We use this to extract a summary vector of the whole input sequence with using a most useful sequence processing neural model Long-Short-Term Memory (Hochreiter & Schmidhuber, 1997).

This model consists of two parts: one is the RNN Encoder-Decoder structure and the other is a Classifier. A core element of the RNN Encoder-Decoder structure is the Long-short-term-memory unit (Cho, van Merrienboer, Gulcehre, et al., 2014).

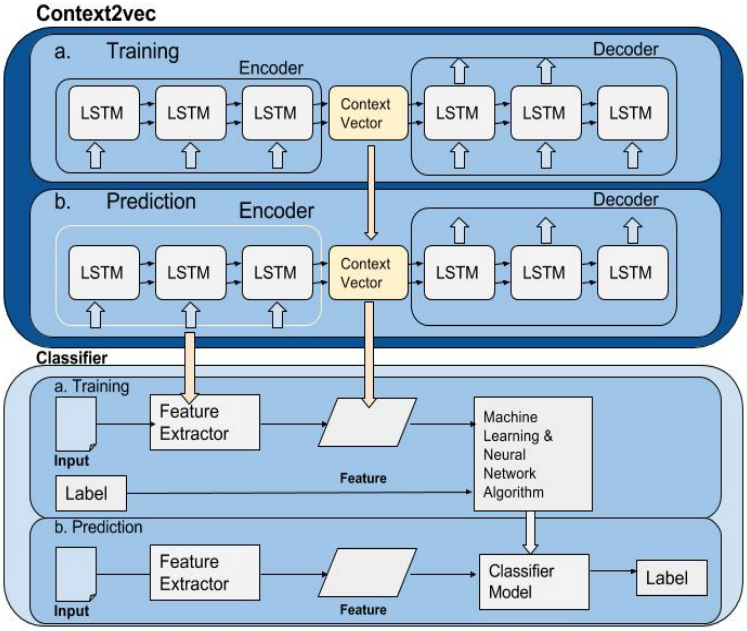

Figure 8 Context vector encoder structure

Figure 8 illustrates the process of the context vector and the classifier. The process can be divided into several steps:

1.According to sequence to sequence experimental protocol, the RNN encoder-decoder is constructed. The encoder reads the sequence until the stop symbol. It learns to generate a state vector (Sutskever et al., 2014). This state vector is what we want to extract.

2.In the completion of the encoder processed, the state vector is transmitted to the decoder. While reading the state vector, the decoder also reads the target sequence. According to the principle of sequence to sequence, the model should correctly predict the sequence same with input sequence of encoder. If decoder detects an incorrect prediction, it will change the parameters value. The learning

process is lasting until the decoder can output the target sequence with high accuracy.

3.After the training phase, the encoder can be used to convert the movie review to the context vector. The context vector length should equal to LSTM number in a layer, and it will be saved for the classification.

4.In this step, the classifier is built to learn how to classify context vector which has been converted from the movie reviews by the encoder.

5.Finally, the evaluation system evaluates the model accuracy by using the test data.

The whole model adopts the strategy of combining a deep neural network with different classifiers. The intention is that with this system to then try to construct a distribution space in the RNN encoder-decoder which has been trained with a Chinese corpus. It will map the sequence input into the distribution space and represent the input sentence as a fixed-length vector representation of the overall semantic meaning of the input (Cho, van Merrienboer, Gulcehre, et al., 2014). In addition, because of Long-Short-Term memory can solve the long distance dependency problem, the historical information in the sequence input will be fully considered whether it is forgotten or keep (Hochreiter & Schmidhuber, 1997; Y. Bengio, Simard, & Frasconi, 1994).

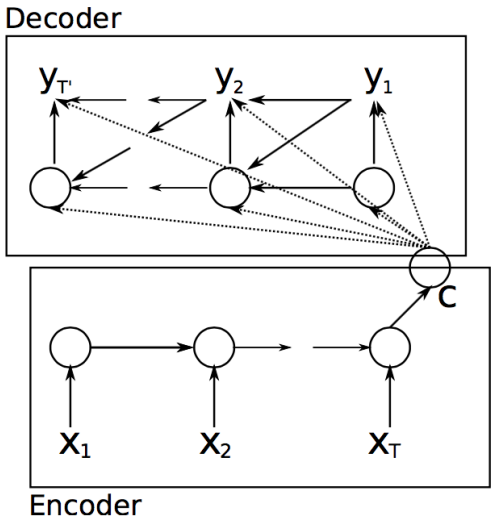

Figure 9 An illustration of the RNN Encoder-Decode(Cho, van Merrienboer, Gulcehre, et al., 2014)

Figure 9 illustrates the structure of RNN encoder-decoder. The encoder consists of RNN, it reads the input sequence X_i until the stop symbol, and generates the state vector (also can be called as context vector) C that a summary of the whole input sequence. The decoder also consists by RNN, and is trained to output the target sequence y_i based on the the maximum conditional probability principle for the target sequence (Cho, van Merrienboer, Gulcehre, et al., 2014).

3.2 Sequence-to-sequence Basics

A basic Sequence-to-Sequence model (Seq2seq) mainly consists of three parts, which are: an encoder, a decoder and a state vector (Sutskever et al., 2014). The essence of the algorithm is that the model will fetch the input data from the source sequence by the encoder, then process it through the primary element, such as LSTM units or the Gated Recurrent Units (GRU). Both these units are inspired by the recurrent neural networks, and have the ability to learning whether to keep the past information (Hochreiter & Schmidhuber, 1997; Chung et al., 2014). This will be seen to be very important to the Chinese language learning tasks(C. Zhang, Zeng, Li, Wang, & Zuo, 2009).

After the processing of LSTM or GRU, it encodes the input data into a fixed-size state vector. Thus we can assume the state vector is the summary of the input sequence (Cho, van Merrienboer, Gulcehre, et al., 2014). Then the decoder will collect the state vector transmitted from the encoder. Accordingly, the decoder will use the state vector and the target input sequence to learn how to produce the same target sequence. The whole process can be represented by the figure 10.

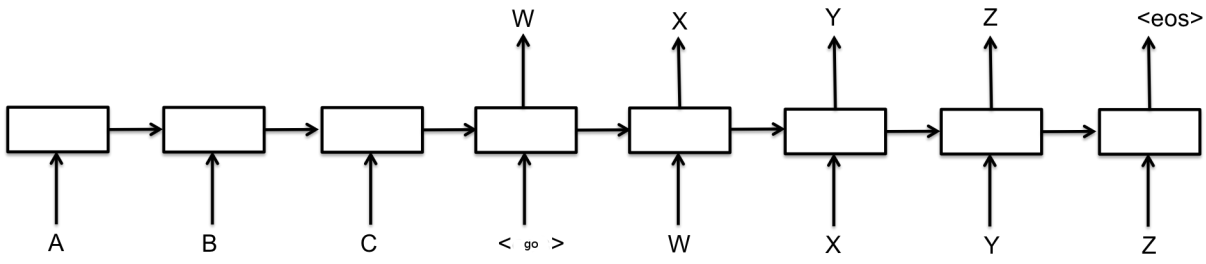

Figure 10 An example that sequence-to-sequence model reads an input 'ABC' and produces 'WXYZ' as the output sentence(Sutskever et al., 2014)

Where ABC is an input sentence and the model generates the output sentence WXYZ. The model begins to predict with the symbol "<go>" and it working until the model output the symbol "<eos>"(Sutskever et al., 2014).

Each box in figure 10 represents a cell unit, and in our model, we chose to use the LSTM unit as the primary unit. From the structure of the model, it can be observed that information moves from left to right through it. The target sequence data is inputted into the right portion of boxes as the target result which used to correct the error, and the right part of boxes begins to produce the output sequence when the cell unit detected the input data is 'stop' symbol.

In our hypothesis, the state vector is a summary of the input sequence data and represents the maximum conditional probability of target sequence (Cho, van Merrienboer, Gulcehre, et al., 2014). Next, it is important to examine how the recurrent network helps to obtain the features from the input sequence.

3.2.1 Recurrent Neural Network

It is important to examine how the Recurrent Neural Network helps to obtain the features from its input sequences. When the RNN tries to infer the next word, it not only processes the current input data, but also treats the previous calculation results as part of the solution(LeCun et al., 2015). To illustrate better how it works it is useful to unfold an RNN model by time as the figure 11.

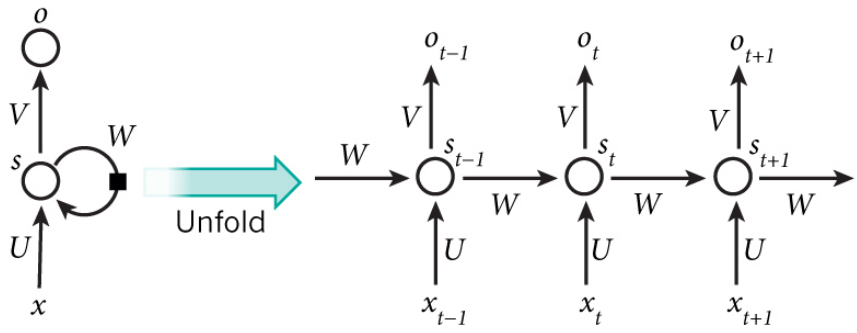

Figure 11 A recurrent neural work and the unfolding in time of the computation(LeCun et al., 2015)

Figure 11 shows that an RNN being unrolled into a full state network with respect to time. X_t is the input at time step t, and it usually will input a word symbol. S_t is the hidden state at time step t. S_t stored the previous information which will be called in next step (Chung et al., 2014). The S_t is computed based on the hidden historical state and the current input at the current time step t:

$$S_t = \sigma(Ux_t + WS_{t-1})$$

Equation 21

Where U is the weight parameter that connects the first and second layers, V is the parameter matrix of output layer, and W is the weight matrix or vector associated with the current hidden state with the hidden state data of next step.

Moreover, the same function and the same set of parameters U, W, V are used at every time step. In addition, there are many options for the activation function σ , such tanh or Relu (Getoor, Scheffer, & International Machine Learning Society., 2011). The activation function will determine whether to use the previous information during the prediction. o_t represents the output at step t.

$$S_t = tanh(Ux_t + WS_{t-1})$$

Equation 22

$$o_t = softmax\,(VS_t)$$

Equation 23

Different function are appropriate for different tasks, and for seq2seq model, a reasonable choice is to use softmax to predict the next word through detecting the probabilities across the vocabulary.

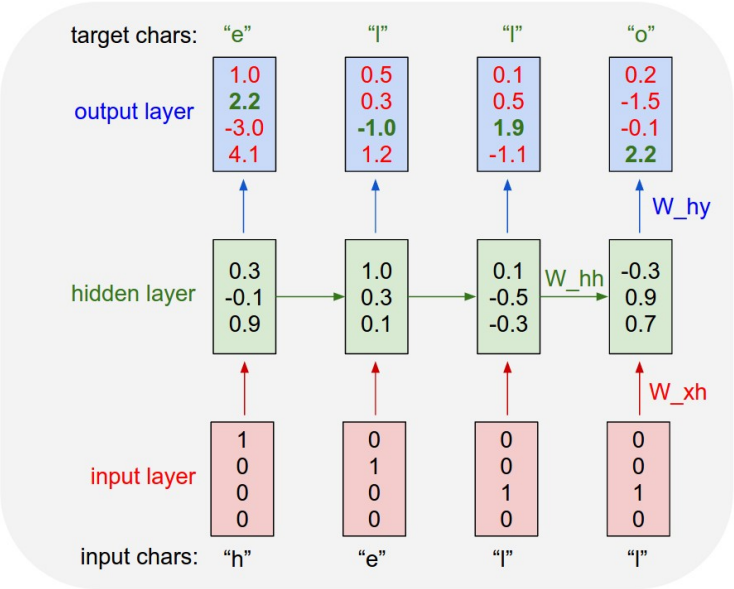

Figure 12 An example RNN with 4-dimensional input and output layers (Karpathy, 2015)

Now an example from 'Character-Level Language Models' (J. Lee, Cho, & Hofmann, 2016) to illustrate how to train the RNN to predict the next word. Suppose a dictionary exists that includes 'h','e','l' and 'o' four characters. In the training case for a sequence data 'hello'. It is desirable to correctly infer that the last character is 'o' when 'h','e','l','l' is entered. The first step is to use the "one-hot" representation to encode these four characters as independent codes as shown in figure 12, ['h',1000],['e',0100],['i',0010] and ['o',0001] respectively. On entering the first character 'h', the hidden layer computes the formula $h_{'h'} = tanh(Ux_{'h'} + Wh_{t-1})$. Because of 'h' is the first input character, the term h_{t-1} can be initialized to 0. The hidden state vector $h_{'h'}$ is [0.3, -0.1, 0.9] is obtained. The final output is $o_{'h'} = [\,1.0, 2.2, -3.0, 4.1\,]$ which means rnn has assigned 1.0 confidence to 'h', 2.2 to 'e', -3.0 to 'l' and 4.1 to 'o' (Karpathy, 2015).

In the output layer in figure 3.5, it can be seen that two sets of values are distinguished by red and green colours in the nodes for each character. The green colour is associated with the value that correctly match target, while the red colour is used for are those that do not match the target. However, the target output label should be 'e' in the output layer. Thus, compared with the target label, the score of correction characters will be increased if the output is same as the target label, and the score of incorrect characters should decrease. This process is repeated many times until the

error of the entire neural network converges to a constant and the result is correctly predicted (Karpathy, 2015).

3.2.2 Long-Short-Term Memory (LSTM)

Although the RNN model can theoretically solve the long-term dependencies problem, in fact, it did not achieve the desired results. This issue was approved by Hochreiter (Hochreiter & Schmidhuber, 1997) and Bengio (Y. Bengio et al., 1994). Thus a more complex model is proposed to solve this problem, the model is called as Long-Short-Term memory units (LSTM).

LSTM is a variation of RNN, it not only can address help language model to promote the accuracy of prediction for long-distance terms, but also can be used to prevent the vanishing gradient issues during the back-propagation through time (BPTT) which used descent the error (Y. Bengio et al., 1994). LSTM helps RNN model dramatically increases the length of learning, which also reflects the ability to store information significantly enhanced.

3.2.2.1 LSTM Structure And Workflow

From the structural point of view, a LSTM model mainly has the following components: a memory cell body, a forget gate, an input gate and an output gate (Hochreiter & Schmidhuber, 1997). The function of the cell body is fundamentally similar to RNN, which is used to store the current input data and the historical state vector.

The main function of the input gate is it will read the data from the training set and decide whether to update the current state. On the other hand, the output gate is mainly used to control the output of cell body. The core of the forget gate is that it will manage the LSTM self-recurrent connection, deciding whether to remember or forget the previous state vector. Figure 13 shows a diagram of the structure of the LSTM (Hochreiter & Schmidhuber, 1997; Gers & Schmidhuber, 2001; Sutskever et al., 2014; Chen, Qiu, Zhu, Liu, & Huang, 2015).

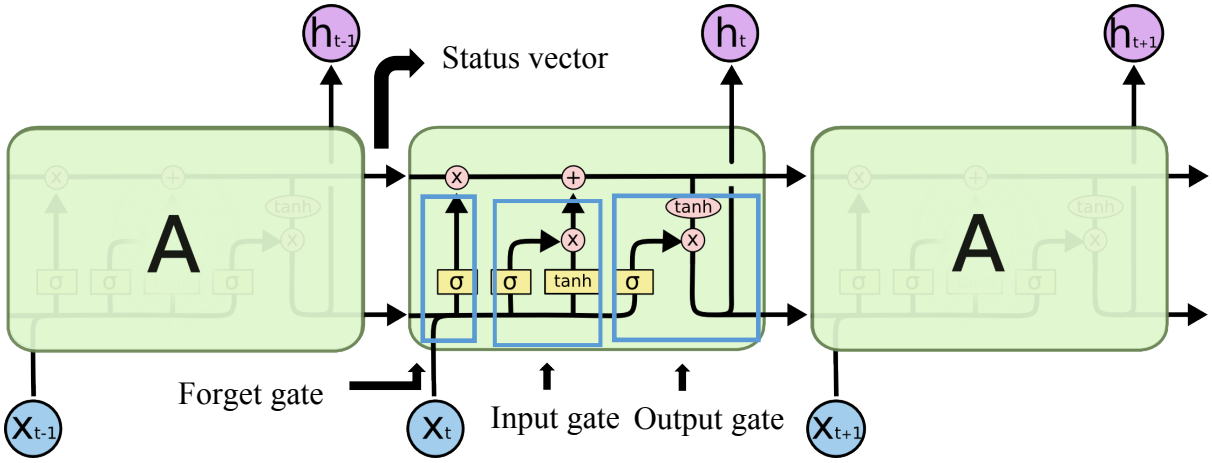

Figure 13 An LSTM contains four layers("Understanding LSTM Networks -- colah's blog," 2016)

In figure 13, it can be seen that each cell body has two input data streams. The first is the current input X_t and the second data stream is the previous cell status. Both of these data will affect the control of the three gates. The specific steps are as follows

1. The forget gate will be based on the current input data to control what information should be ignored. For example, assuming previous sentences have been read which include gender information. However, in the present sentence there appears a new object with a new gender attribute. In order to accurately use the corresponding words, it is necessary to abandon the historical gender information.

2. According to the data read by the input gate, it is necessary to consider which information should be updated and stored in the state vector. With reference to the same example, in this step, the body cell will read the gender attribute to update the state vector and replace of the old state vector.

3. Finally, once an updated cell status has been obtained, then the algorithm needs to learn which part of the state vector should be output and whether to only output the part data which has been modified. Referring again to the example, having the new gender data, the output gate will determine which part should be output and whether to output the changed portion of cell status. Now we will examine the math equation insight of LSTM.

3.2.2.2 The math equations for gate controller

We would like to use the variable Ct to represent the cell state vector of the LSTM, the cell states will go through the entire training process. All of the gates used the sigmoid activation function to control the flow of information, and the gates are affected by three elements: the output from the previous time step, the current input and optionally the cell state vector (Chen et al., 2015).

The value of the forget gate which is represented by f_t is computed using.

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f)$$

Equation 24

Where W_f denotes the weight matrix for forget gate.

The input gate is represented by i_t.

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_f)$$

Equation 25

Where W_i represents the weight matrix for input gate

And the input gate i_t decides which value that will be sent to be updated. This update function will be implemented by a tanh function that creates a new candidate values $\underline{C_t}$.

$$\underline{C_t} = tanh(W_c \cdot [h_{t-1}, X_t] \ + \ b_c)$$

Equation 26

Where W_c denotes the weight matrix for $\underline{C_t}$.
After obtaining the new candidate values $\underline{C_t}$, the model will update the old cell state $\underline{C_{t-1}}$ to the new cell C_t. The old state is multiplied by the forget gate f_t, and combined with an operation involving the product of i_t and $\underline{C_t}$.

$$C_t \ = f_t \ * C_{t-1} + i_t * \underline{C_t}$$

Equation 27

Finally, the current generation value h_t is found from the current cell value and the output gate result o_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

Equation 28

3.2.3 Context Encoder Based on Sequence-to-Sequence Model

In our model, we propose an encoder model that uses the encoder model structure of the sequence-to-sequence model as a context vector encoder. This is used to convert a variable-length sequence data into a fixed-length vector representation (Sutskever et al., 2014).

The first stage is the training process, and the primary purpose is to train the Seq2seq model. The collected Wikipedia Chinese corpus is used as the training source dataset. During the training process, the encoder will gradually read every symbol in the input sequence until the end. Simultaneously, to get the conditional probability distribution for the Chinese language, we also use the Chinese corpus provides target sequence data which will be used in the decoder (Cho, van Merrienboer, Gulcehre, et al., 2014). The decoder will read the target sequence data. The model will learn the probability distribution and the parameter matrix from the source input data to the target sequence data. Thus the whole task could be represented by the following mathematical expression:

$$\boldsymbol{P(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_t)}$$

Equation 29

When the encoder reads a sentence, it will output a context state vector c, and this vector will be transmitted to the decoder. The decoder reads the target sequence data and will determine the output of current step by predicting the next word in the case of a given context vector c. The following mathematical equations can express how to compute the hidden state h_t in the decoder:

$$h_t = f(h_{t-1}, y_{t-1}, c)$$

Equation 30

$$P(y_t|y_{t-1}, y_{t-2}, y_{t-3}, \ldots, y_1, c) = g(h_t, y_{t-1}, c)$$

Equation 31

Where $y_t$is target word, and $h_t$is the state vector of the current body cell and c is the context vector which come from the encoder.

The aim is to learn the conditional probability distribution for Chinese corpus, thus it is desirable to set the target sequence of the decoder to be the same sequence as source sequence data. According to the previous work (Cho, van Merrienboer, Gulcehre, et al., 2014) that they consider the context vector c to be a summary of the input sequence data, The trained encoder of sequence-to-sequence model will be employed as a feature extractor, and will provide the context vector for the classifier(Tang et al., 2015).

Chapter 4

Data preparation and tool introduction

This chapter focuses on the details of the data, including the data sources, the data format, the positive and negative case number, and some necessary information regarding the dataset. In the experiments, classification of the training dataset into two categories can be carried out based on the model function. The first type of training dataset is used in the training of seq2seq model (Sutskever et al., 2014) which is used to generate context vector. The second type of data is the target sentiment classification data which is used in the machine learning or neural network classifier.

4.1 Data Analysis

This chapter focus on our data details, including data sources, data format, positive and negative case number and some necessary information of dataset. In our experiments, we can classify the training dataset into two categories based on our model function. The first type of training dataset is used in the training of seq2seq model which used to generate context vector. The second type of data is the target sentiment classification data which used in the machine learning or neural network classifier.

4.1.1 The Chinese Wikipedia Corpus

The Chinese Wikipedia corpus is a text corpus created from the Chinese internet encyclopaedia Wikipedia in 2012. For the building corpus was used Wikipedia dump. The corpus was segmented by Stanford Chinese segmenter. Figure 14 shows an unprocessed example of the Chinese Wikipedia corpus. The unprocessed corpus contains a large number of useless data, such as HTML code.

Figure 14 A unprocessed Chinese Wikipedia corpus

In order to obtain a useful Chinese corpus, it is necessary to do several steps:

1.Extract useful Chinese articles from the original corpus. Therefore, the HTML code and English words must be removed;

2.Chinese has two word styles: Chinese traditional Chinese and simplified Chinese. All of words in traditional Chinese should be convert to simplified Chinese;

3.The Stanford segmenter should separate all the words in the sentence using the delimiter ‘space’;

Finally, it is possible to obtain a cleaned Chinese corpus as shown in figure 15 which can then be used to train the sequence-to-sequence model.

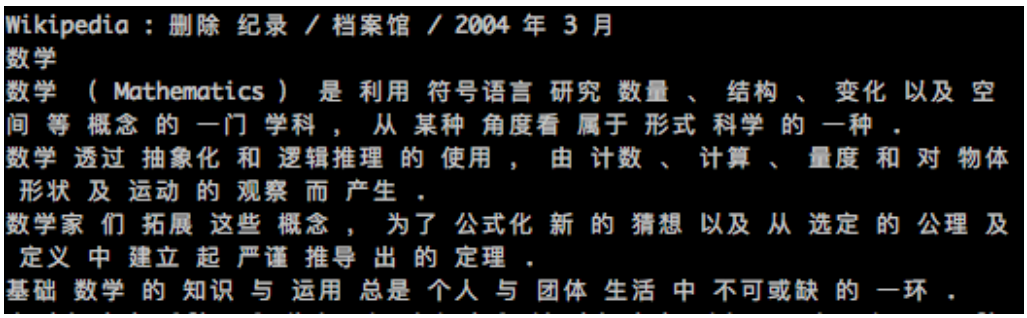

Figure 15 A sample of the cleaned Chinese Wikipedia corpus

4.1.2 Sentiment Corpus: Movie Review Analysis

To obtain suitable training data of for sentiment classification, this can be mainly sourced from several categories: the first category is review data of products; the second type is the short articles posted on the social networks; the third category is the comment and evaluation of film and music (Pang & Lee, 2008).

The reviews of products are primarily posted on e-commerce sites, such as Amazon.cn, taobao.com. After consumers have purchased products online, they will give a comprehensive evaluation of the goods based on their satisfaction, in terms such as the performance of the commodity, its appearance, and any other aspects. At the same time, each consumer will manually evaluate the product from one star to five stars.

The second category data mainly comes from twitter.com and microblogging content. There is a previous work released a large scale Chinese short text summarisation dataset which consists of over two datasets constructed from the Chinese microblogging website Sina Weibo (Hu, Chen, & Zhu, 2015).

The third category of evaluation data is the most popular corpora in the sentiment analysis task, IMDB. However, it is challenging to obtain an extensive text summarisation dataset about film and music in Chinese. Thus eleven different types of film comment data which has been labelled was collected by using a web crawler tool.

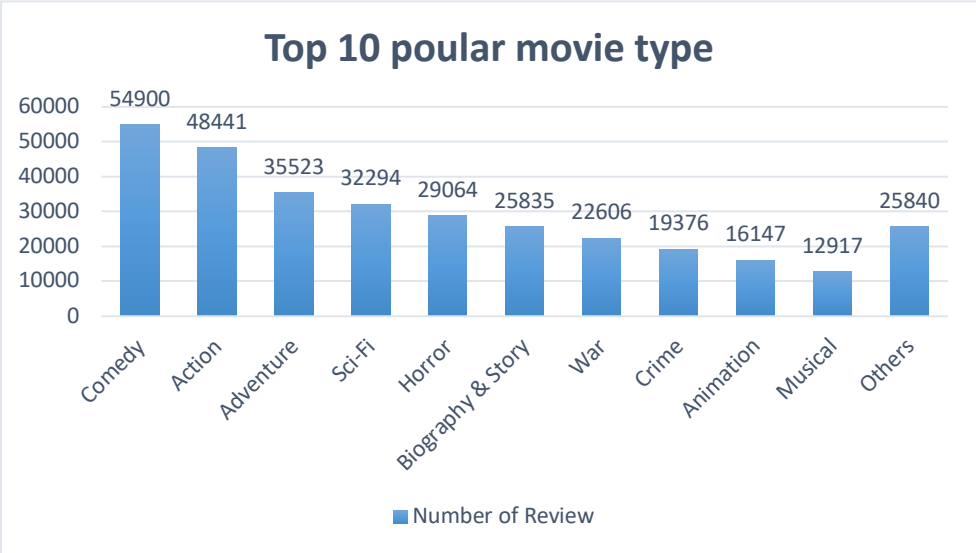

Figure 16 Top 10 popular movie type and numbers of different types in the corpus

Only reviews were selected that have rating scores or rating stars, leading to a collection of 322943 valid reviews in total. The distribution of reviews about the movie genre type is given in figure 16, with comedy films having the greatest number of reviews and action being close behind it. The category other collates the remaining reviews. The average length of each comment was 20.53 words. In addition, it is necessary to convert our reviews' label from rating scores or rating stars into one of three categories: positive, neutral and negative.

In the classification task, the sentiment prediction is usually treated as a binary classification and 0 and 1 are used to represent the positive and negative cases respectively. As the figure 17 shows, the collected sentiment corpus has five levels which is same with Amazon and taobao.com rating system. A single star corresponds to the worst level, and five stars is the best level. The sentiment degree of reviews is positively correlated with the number of stars. However, it is evident that the number of reviews which are tagged as four stars is much bigger than the others. The differing number of reviews for each star level does create an imbalanced data problem.

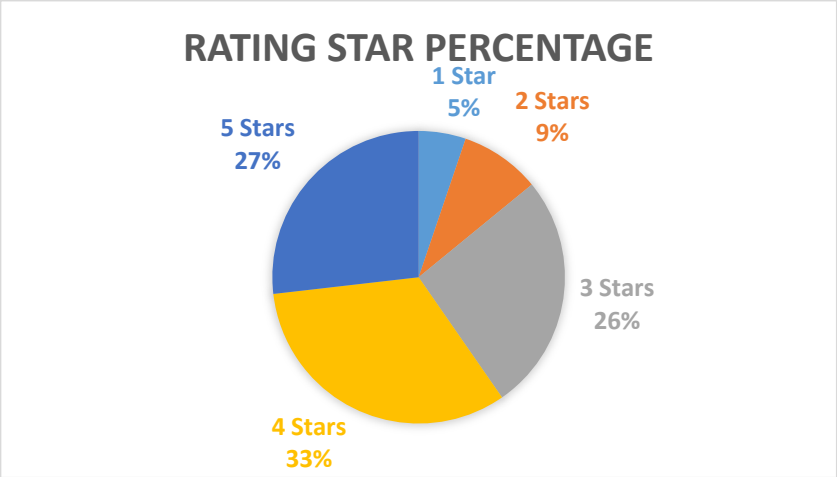

Figure 17 The percentage of different rating stars present in the dataset

Imbalanced data refers to the problem that the number of different classes is not equal in the training datasets of classification (Batista, Prati, & Monard, 2004). This frequently occurs in classification problems. For example, if there was an attempt to train a classifier to predict the gender of student, the training set would be judged to be an imbalanced data in the situation where the training dataset contains attributes for one hundred males and ten females. It will most likely result in the majority of

testing data trials predicting the gender to be male rather than female. To avoid the imbalanced data problem, an equal number of training cases and test cases are randomly chosen from the different classes available

At the same time, according to the sentiment categorization of different star classes and the binary classification requirement, the 3-star level reviews are divided into neutral, 1-star and 2-star' reviews to become the negative training dataset, 4-star and 5-star reviews were used to form the positive training data.

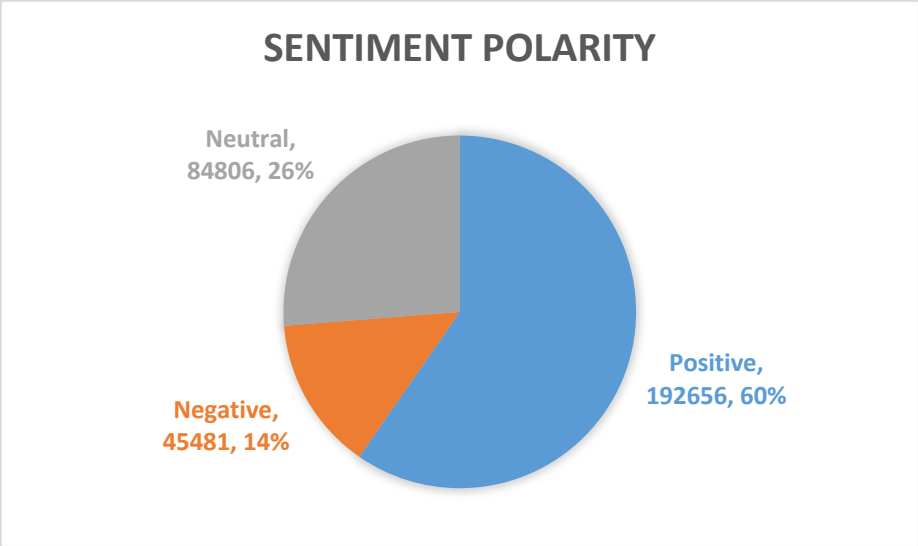

Figure 18 Numbers of different sentiment polarities

In our experiments, we just focused on discriminating between positive and negative. Thus we removed all of the reviews which belong to neutral level or 3 stars level. Finally, it resulted in a training dataset which included 25000 positive cases and 25000 negative cases. In addition, the test dataset consisted of 7500 positive cases and 7500 negative cases.

Through the analysis of the training dataset, minority reviews includes emoji expressions, such as "╥﹏╥" (meaning to cry) and "(=^･^=)" (meaning to happy). The majority of emoticons are made up of punctuations, and within the training dataset it is possible to find many emoticons of the form of 😀. Particularly on microblogging platforms, due to the input length restrictions, emoticons are commonly used to express the authors' mind(Pak & Paroubek, 2010). However, emoticons are not of interest in relation to the text corpus and were removed.

4.1.3 Chinese Segmentation

Within NLP, the treatment for Chinese and English is quite different. In English, people usually use space as the delimiter for the separation between words. However, in Chinese, there is no space or other delimiters to separate words. Furthermore, the primary unit in English is 26 letters, while the underlying unit in Chinese is the character, of which there are thousands. For example, in the sequence data:

“Semantic analysis is the key area in NLP.”

The basic unit is letters, and every word in this sentence consists of some English letters. Translating this sentence into Chinese, it becomes

“语义分析是 NLP 中的关键领域。”

It is obvious that there is no space between words, and the basic unit is Chinese characters. A word with meaning usually has two or more characters in Chinese. For instance, it is meaningless if we just capture a Chinese character from the above sentence “语”. However, if “语” and “义” are combined together, then they mean a word “semantic”. Therefore, segmentation is acting an essential role in NLP tasks for Chinese.

Segmentation is used to divide a string into different level of units, such as words (Lafferty, McCallum, & Pereira, 2001), or topics (Reynar, 1998). Due to the primary unit of Chinese being characters, and the fact that there is no delimiter between the Chinese words, so words are segment based on the understanding of the entire sentence or on the availability of specific context information. For example, considering the character sequence, ‘这个苹果不大好吃。’ this sentence is easily misinterpreted if the segmentation is incorrect. This sentence has two different word

segmentation strategies that would produce opposite meanings. The first segmentation strategy is “这个 / 苹果 / 不大好吃。” (This apple is not delicious.). On the contrary, if we combine ‘不大’ with ‘苹果’, then we will get an opposite meaning sentence, that ‘这个 / 苹果不大 / 好吃。’ (This apple is not big, and it is delicious.). Generally speaking, the different combination of Chinese characters has different meanings, so incorrect word segmentation will have a significant impact on our model’s performance.

Currently, the favoured segmentation tool for Chinese are Stanford segmenter (Chang, Galley, & Manning, 2008) (Tseng et al., 2005), Jieba segmenter and so on. According to different approaches, the segmentation algorithms can be divided into two categories. The first class is the segmenter based on the lexicon-based Max-Match approach, such as jieba segmenter. The basic idea behind this tool is it constructs a prefix dictionary structure, and then it searches all possible word combinations. The second type used the conditional random field sequence model (CRF) (Lafferty et al., 2001) to do the segmentation, like Stanford segmenter for Chinese (Tseng et al., 2005) (Chang et al., 2008). The CRF model is widely used in machine learning and is applied for sequence data prediction that it predicts sequences of labels for input sequences data.

Some experiments proved the Stanford segmenter has a stronger ability to deal with Chinese out-of-vocabulary words and has a higher accuracy than the lexicon-based approach (Chang et al., 2008). According to the comparison and the feature of Chinese that the different words segmentation strategies lead to ambiguous, we used Stanford segmenter to assist us to do the segmentation task.

4.2 Deep learning tool

In this section, the tools, including the libraries and platform used in our experiments are introduced. In the corpora collection phase a web-crawler was constructed using the python language to get the latest corpus data. Then, a regular expression approach was employed to collect and filter the information so that it retained all the relevant Chinese content, such as movie names, movie reviews and evaluation levels, that is, the number of stars. In the training phase for the RNN-encoder model, the Seq2seq model was implemented using the Tensorflow library (“TensorFlow,”).

Then we used the Scikit-learn toolkit ("scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation," 2017) to help us build and train our classifier.

4.2.1 Web crawler

A web crawler is a tool which can be used to obtain the webpage content from the World Wide Web. It generally used in conjunction with a search engine to grab web pages. The web crawler starts with one or more initial web pages, then it can automatically detect other URLs (also known as seeds) which appear on the initial webpage and push these into a URL queue (also known as the HTML frontier) so that the web crawler can visit each URL in the queue one by one. The web crawler will repeat the above process until it meets the collection requirements (Shestakov, 2013). In these experiments, a topical topical crawler was build which focuses on a specific topic in the movie domain. The working principle of Web crawler can be seen in figure 19.

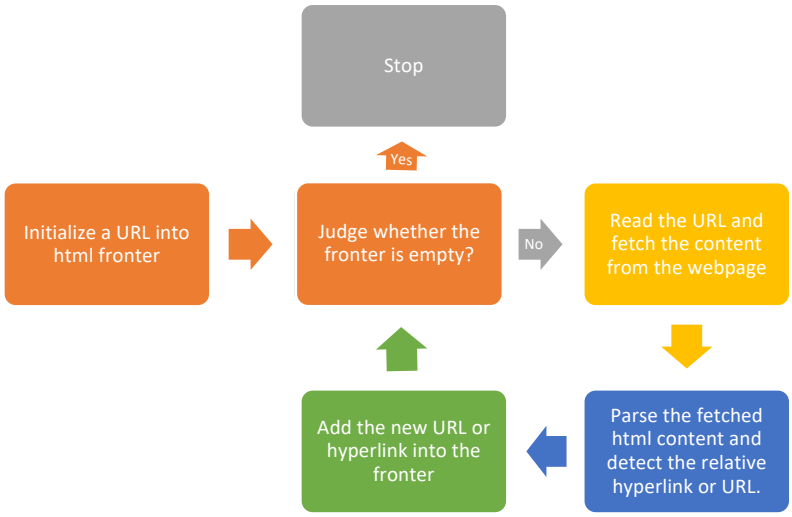

Figure 19 A web-crawler workflow

During our collecting progress, we used the Urllib2 library was used to help fetch the target page content, which returned the raw HTML contents which included the movie detail and the users' review information. Urllib2 offers many convenient functions, such as 'request' and 'urlopen'. If it is required to grab some data from a webpage, then a query to the server must be constructed. Additionally, to avoid blocking by anti-web-crawler systems, it is possible to include headers that act to fake the request to make it seem as an ordinary visitor. Then the 'urlopen' function

can be opened to receive the response of the HTML content of the target page from the server. Finally, a regular expression –based approach is required to extract and clean the HTML raw content.

4.2.2 Machine learning and deep learning libraries

The machine learning tool which used to build the SVM classifier was discussed already in Section 2. Next, some essential functions of the Scikit-learn toolkit ("scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation," n.d.) will be briefly. Then we will briefly introduce the Tensorflow. Moreover, an example describing how to build a recurrent neural network using Keras ("Keras Documentation,").

4.2.2.1 Scikit-learn toolkit, Tensorflow and Keras

Chapter 2 introduced the SVM model. Following on from this an overview of the Scikit-learn toolkit is given now. Due to the fact that python is open source, possesses extensive libraries and is relatively straightforward to use, the Scikit-learn library was chosen as the primary tool to make a Support Vector Machine model. It features many machine learning algorithms, such as regression, clustering and classification, including SVM.

When developers try to build a classifier to predict the target label of data based on SVM, Scikit-learn provides three different SVM classes for various tasks, Support Vector Classification (SVC), Nu-Support Vector Classification (Nu-SVC) and Linear Support Vector Classification (LinearSVC). The most common type of SVM class in the Scikit-learn should be the SVC, it is based on the standard SVM algorithm and various parameters can be set to optimize the use of the algorithm. The Nu-SVC is similar to the SVC class, except for a slight difference which is that the Nu-SVC uses a parameter to control the number of support vectors. In fact, the parameter 'nu' represents an upper bound on the fraction of training errors and a lower bound on the fraction of supports("scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation,").

The most obvious difference between LinearSVC and SVC is that the LinearSVC's kernel function has been fixed as 'linear'. This means that you can not change its kernel to another.

The Scikit-learn library not only contains many classical machine learning algorithms, but also provides lots of components which usually appear in machine learning tasks, such as the estimator, the pipeline, preprocessing, and feature extraction. A quick overview of the functions of some core components is:

1.Estimator: the estimator class is the fundamental class for all classifiers. It mainly contains two functions: fit(), and predict(). The fit() function is used to train the algorithm, and predict() is responsible for predicting the test data based on the trained algorithm.

2.Pipeline: the purpose of the pipeline is to record the progress and to compare performances which were made by using different parameters.

3.Preprocessing package: this package offers some useful classes, like one-hot encoder, label encoder (which is used to convert the label to a numerical value), normalizer and so on.

4.Feature_extraction package: This package is a collection of methods used to extract the features from data. For example, the function text.TfidfVectorizer is used to generate a TF-IDF representation as an alternative to the original count feature matrix.

Some deep learning tools can also be mentioned, Tensorflow and Keras are the popular deep learning libraries used for the construction of deep neural networks. The Tensorflow library allows developers to create various deep neural network structures by using data flow graphs. Data flow graph is the core of tensorflow, every node in the data flow graph is a computation operation, and the responsibility of the line which connects two nodes is describing the relationship between two nodes.

The Tensorflow construction process can be divided into two steps: firstly users need to build the framework for objects of a deep neural network model. Then users start with creating a session and pushing the data into the framework. The model visualization that is a feature of TensorBoard allows users to check the framework of their deep neural network. Furthermore, TensorBoard not only offers a model visualization function, but also can record every step during the training progress. In fact, if developers want to determine the DNN's parameters, they

usually need to observe the updating of parameters throughout the training progress. TensorBoard can generate many diagrams to show the data changes, an example is provided in figure 20.

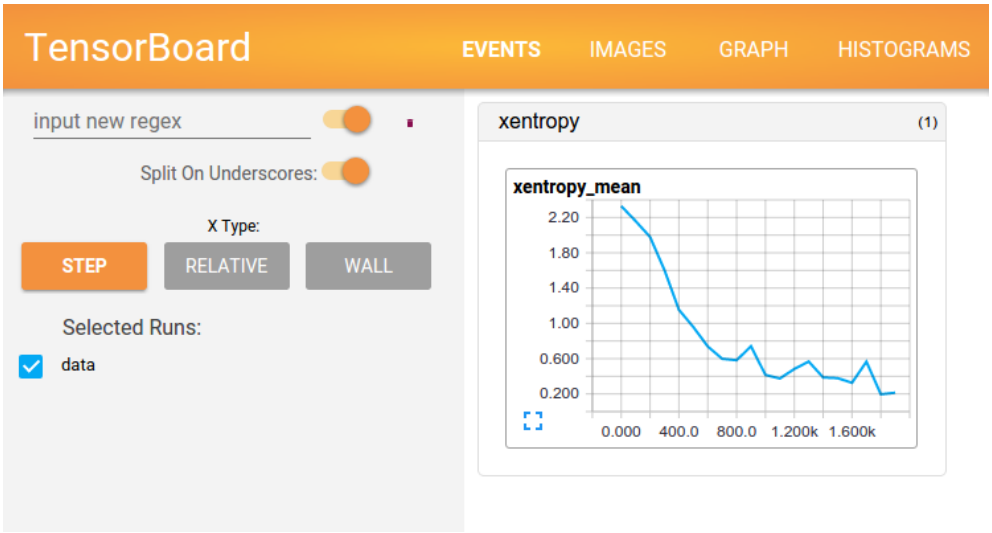

Figure 20 TensorBoard(“TensorFlow,” 2017)

When developers are training a model on Tensorflow, some summary data can be generated. The summary data is stored in an event file which is read by Tensorboard. Here, model visualization can effectively help the developers to check their model and correct the parameters. It is worthwhile doing this as it can dramatically improve the efficiency and accuracy of the development. In next chapter, it will be shown how Tensorflow is used as the primary tool for the context encoder development for the RNNs, and also how it is used to convert the input sequence of testing data into a context vector.

The act that model visualization and training data can be recorded are two of the advantages of Tensorflow. At the same time, another significant advantage of Tensorflow is that it allows developers to use GPUs to speed up the algorithm training and additionally, it supports distributed training. However, even though Tensorflow has many advantages, it also has some weakness. Tensorflow is based on the python language, so it is easy to learn. However, the various module calls in Tensorflow are very complicated, especially as regards the parameter setting for the

connections between different modules. Due to Tensorflow version updating, experiment code always need to be rewritten as a result of API adjustment.

To improve the speed of our experiments, Keras is a helpful library to establish our deep neural network and MLP. Keras is a python-based deep neural network library that uses Tensorflow as the backend. It also allows users to change the backend to other DNN libraries, such as Theano. User-friendliness is the fundamental principle for Keras framework design. Keras offers a very simple and convenient APIs. Users just need to set the parameters for the modules when they try to call some functions. Moreover, it can directly give feedback about the algorithm, and show both the loss value and accuracy data to users. Tensorflow requires users to treat every computation step as an operation node in the data flow structure. Sometimes, it will be very complicated for users to confirm the correction of every step with lots of parameters.

On the contrary, Keras's modularity solves this problem for users, and users can consider every model as a separate block. All of the modules can be freely combined, and users can create some new models based on the foundational modules. For example, if users try to create a hidden layer with LSTM, they just need to declare a 'sequential model' and then use the 'add' function to put the LSTM into the model structure, and finally just need to specify the 'number' of LSTM units. This modularity dramatically reduces the user's workload.

By way of an illustration, a classical classification problem is shown that MNIST classification also uses introduce the convenience of Keras. MNIST is a handwritten digital dataset, usually used to test the image processing system, and also widely used to train and to test in the field of machine learning. The MNIST dataset can be directly imported from the Keras dataset, which has divided into a training dataset and a testing dataset. Every dataset contains two types data: one is the handwritten image as figure 21.

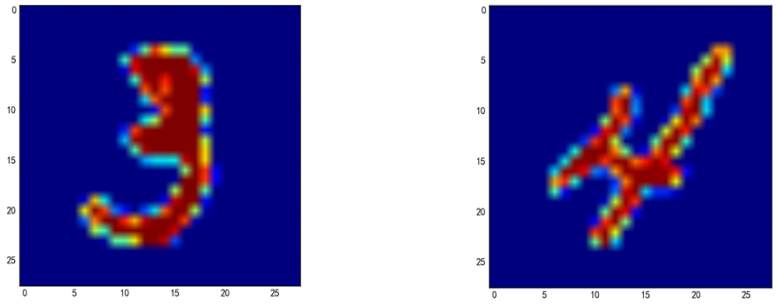

Figure 21 MNIST handwritten images

Each handwritten image has 28 x 28 pixel. Another one is the label corresponding to the handwritten image from 0 to 9.

The entire process can be divided into four steps:

1. In the first step, the dataset needs to be preprocessed.
2. The next step is defining and declaring the layers of the model, and setting the various parameters of the algorithm.
3. In the third step, the model will be compiled and trained.
4. The final step is the model evaluation.

For this example, this means that firstly, the handwritten images need to be reshaped to a 784x1 vector and use the ‘to_categorical’ function in the ‘np_utils’ class which is provided by Keras to encode data labels. In practise, the label is encoded to a binary vector that it is suited for the prediction. This encoding method is similar to the “one-hot” representation which was described in the Chapter 2. Note that because the labels have ten values from 0 to 9, these will be encoded to a vector whose length is ten, and the element position in the vector is used to represent the label value. So ‘6’ will be encoded as [0000001000] for example.

Secondly, the model should be defined, and then declared to be a sequential model because it is required to construct a feedforward neural network with a hidden layer which contains 128 cells. According to the structure of the MLP, this means adding an input layer, a hidden layer and the softmax function which is used to do the multi-class classification. The detail of function and parameters as are shown in figure 22.

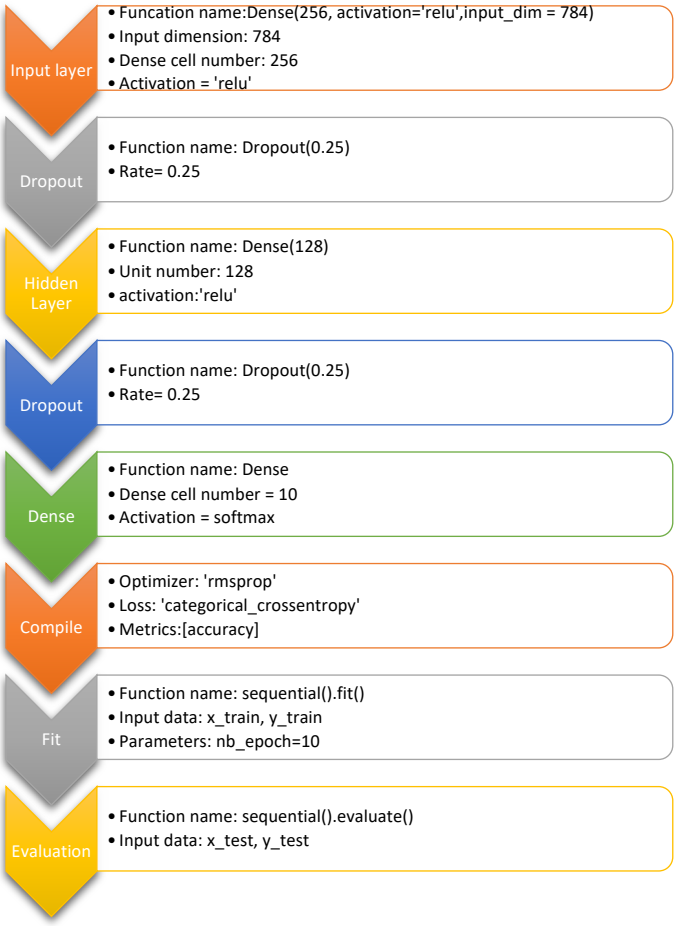

Figure 22 An example of Keras

From figure 22, the input layer is first declared, and the parameters with the values as shown. The 'Input dimension' parameter corresponds to the converted image vector which is 784 x 1 in size. A dropout layer is included to avoid overfitting problems (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). To promote the model accuracy, a hidden layer with 128 cells is added before the output layer. Moreover, because there are ten category labels in the corpus, the output layer also needs to emit a vector of length ten, where each element in the vector corresponds to the labels from 0 to 9 respectively. For the binary classification, the sigmoid function is widely used by developers. In the multiclass classification problem, softmax always is the first choice for developers (In output layer). Finally, the performance of the multiclass classification achieved 98.15% accuracy.

Through the above description, we review the tools and libraries used in the experiment. Then we will introduce the experimental content.

Chapter 5

Experiment

This Chapter explains the experiments that were performed to better understand the performance of the word vector and the context vector representations on different models and different languages for sentiment analysis. The experimental design is first explained followed by their implementations. The results obtained are then analysed followed by a discussion.

5.1 Experimental Setting

In this section, we give the details of the experimental setting, including experimental purposes and evaluation standards.

5.1.1 Purpose of Experiment

The experiments were divided into three subtasks:

Subtask 1: For sentiment analysis in this stage, the IMDB dataset (Maas et al., 2011) was used along with Chinese IMDB (CIMDB) dataset. The IMDB datasets consist of 25000 movies reviews, and they had been labelled by the users. Each word in the reviews was encoded and replaced by word indexes in vocabulary. A LTSM recurrent neural networks model was constructed by using the Keras library for sequence classification. The behavior of the model was compared for the two different language corpora.

Subtask 2: As mentioned earlier previous work used the word embedding with SVM classifier to classify the sentiment polarity of Chinese comments (D. Zhang et al., 2015). A similar experiment was conducted to evaluate the performance of the word2vec model with SVM using the corpus CIMDB as input. In addition, it used the mapped word vectors which were converted by the word2vec model so that they could be input to the MLP classifier.

Subtask 3: In the first subtasks, the investigation was directed towards the difference between Chinese corpus and English corpus for the same type reviews. In this subtask, the proposed context vector encoder is used to map every sentence in

reviews to a fixed length vector. Different types of classifiers are compared to determine which type of classifier suits for use with context vectors. Then, among those classifies it seeks to find which of them achieves the best performance.

A summary of experiments in terms of the models examined and the input corpora is shown in table 4:

Models	Corpus
LSTM	CIMDB & IMDB
Word2vec + SVM	CIMDB
Word2vec + MLP	CIMDB
LSTM-RNN Encoder + SVM	CIMDB
LSTM-RNN Encoder + MLP	CIMDB
LSTM-RNN Encoder + LSTM	CIMDB

Table 4 Summary of the experiments carried out

5.1.2 Evaluation Metrics

To better evaluate the performance of the models, the Recall, Precision and other measurement standards are used in the evaluation process. In order to understand the advantages and disadvantages of each of the models, the evaluation metrics which were provided by Seki (Seki et al., 2007) in NTCIR-6 are used.

The NTCIR is the National Institute of Informatics Test Collection for Information Retrieval Systems. The opinion analysis task was first featured in papers given at the NTCIR-5 workshop in 2005. It was served as a pilot-task at NTCIR-6 and NTCIR-7 (Pang & Lee, 2008). Through this system, the investigation into the different opinion extraction sub-tasks resulted in a set of useful evaluation metrics. Thus, the Recall, Precision, and the F-measure were found to be the best metrics for the polarity classification (Seki et al., 2007). The error rate is another popular metric (Bespalov, Bai, Qi, & Shokoufandeh, 2011).

On the other hand, sentiment classification or polarity classification also can be considered as a text classification task, so the Accuracy metric should be the most intuitive evaluation criterion. Further, Our evaluation system used these five metrics and implements them using formulas as given (Sebastiani & Fabrizio, 2002)’s work.

As introduced in the work of (Sebastiani & Fabrizio, 2002), a binary classification produces four prediction results: true positive, false positive, true negative and false negative. They are usually denoted as TP, FP, TN and FN:
TP means true positive that the number of correct positive prediction;
FP means false positive that the number of incorrect positive prediction;
TN indicates true negative that the number of correct negative prediction;
FN means false negative that incorrect negative prediction.
Table 5 illustrates the logical relationships between these four measures as a confusion matrix (Sebastiani & Fabrizio, 2002).

Actual Label	Prediction	
	True	False
True	TP	FN
False	FP	TN

Table 5 Confusion matrix for binary classifier

Like previous work, we use Accuracy, Precision, Recall, Error rate and F-measure as the experimental measures (D. Zhang et al., 2015).

1. Accuracy：the accuracy is the most common evaluation standard, which directly reflects the model prediction results. In the test dataset, the accuracy is equal to the number of correct prediction divided by the number of entire test data.

$$Accuracy = \frac{TP + \square N}{P + N}$$
$$= \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 32

Where P indicates the number of positive cases in a test set, N means the number of negative cases in a test set.

2. Precision: precision is the proportion of the correct positive predictions in all of the positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

Equation 33

3. Recall: Recall is computed as the number of correct positive predictions divided by the all of the positive cases numbers in test data

$$Recall = \frac{TP}{TP + FN}$$

Equation 34

4. Error rate: Error rate is computed as the number of all incorrect predictions divided by the total number of the dataset.

$$Error\ rate = \frac{FP + FN}{P + N}$$

Equation 35

5. F-Measure: Sometimes, the previous criteria are conflicting, result in the inability to measure the performance of a model accurately. Thus the f-measure is a balanced measure which performs a weighted average of precision and recall values:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Equation 36

5.2 Experimental protocols

The detail of experimental protocols is discussed in this section, including parameter values of models, model structures and the data workflow.

5.2.1 Task 1

In this experiment, the LSTM model is implemented using the Keras library to classify the sentiment class on both the English and Chinese IMDB corpora and compare the results. The parameters of the model first need to be set according to the corpus data. Considering the more efficient extraction of valid feature words, the words whose frequency was less than 3 were ignored and the term ‘UNK’ was used

to represent them as unknown instead. During the investigation, there were 60034 words which appeared with a frequency of two or greater in the corpus. Thus a vocabulary was created with size 60034 which contained 60034 words with the highest number of occurrences. Figure 23 shows an overview of the LSTM model structure implemented by Keras.

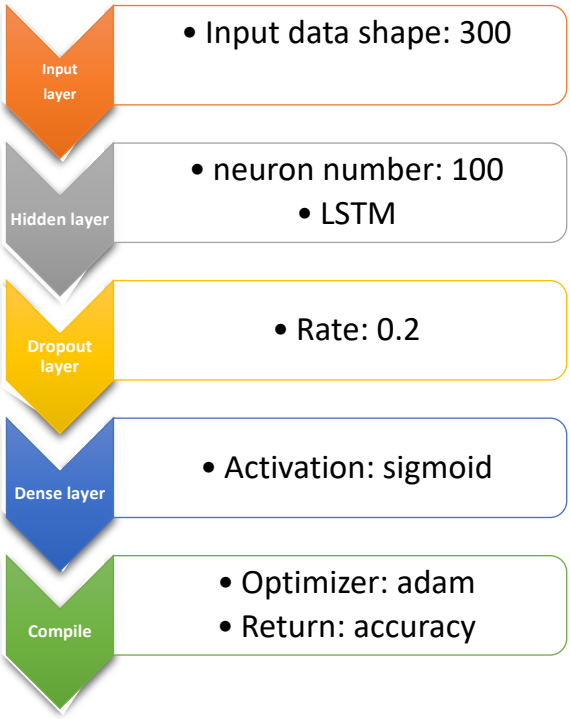

Figure 23 A LSTM neural network work structure

It was necessary to take into account the issue that the length of each sentence in the corpus is not the same. The pad function was used to extend all sentences to a fixed length, and so the input data dimensionality was set to be 300 as can be seen in Figure 23. The input layer converts the input data to be a fixed length feature vector by lexicon. Then, the processed input vector is transmitted to a dropout layer which prevents the model from overfitting as mentioned in the previous chapter (Srivastava et al., 2014). The hidden layer in the model consists of 100 LSTM neurons. After the hidden layer here, the data enters into a sigmoid function. After completing the definition of all the modules, the compile function is passed the optimizer type being used and also the accuracy metrics which will be collected during the model training are set.

5.2.2 Task 2

In this subtask, experiments are conducted to investigate the performance of the word2vec model that is used to maps the words from the reviews to a vector space. Then we input the word vector data to the SVM or the MLP to classify the sentiment polarity of the comment texts, such as if it belongs to a positive class or negative class. Figure 24 shows the framework of this experiment set.

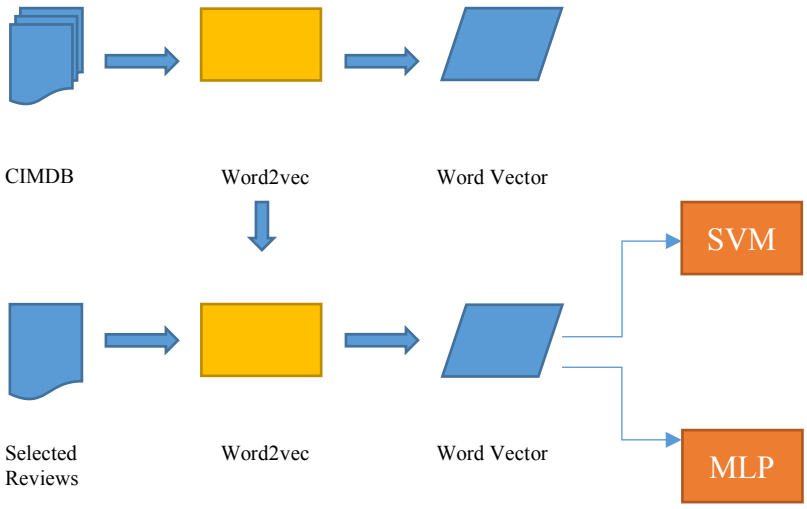

Figure 24 The structure of word2vec and SVM

In the previous chapter it was explained that an extra step of word segmentation needs to be carried out when processing Chinese data. Thus, the Stanford segmenter was applied to convert the review strings in the training corpus into words. Then, the experiment protocol of (Mikolov, Sutskever, et al., 2013) and (D. Zhang et al., 2015) were followed to learn the word vector and map the words into the word vector space. The Tensorflow Deep learning library was then used to analyse this data and create a display of the similarity among the words contained in the vector space. The parameters used in the word2vec model are given in table 6.

Parameters	Value
Training strategy	Skip gram
Training batch size	128
Embedding size	200
Skip window	2
Optimizer	Stochastic gradient descent

Table 6 the detail of the word2vec model

The word2vec model learns to map the word to a word vector space and generates a vector with a fixed length of 200 for each word. These word vectors are stored in a memory space that will be used in the next step. Next step, the trained word2vec model uses the stored vector to convert each of word in reviews. Since the average words number of reviews is 20.53, and each word is represented by a vector belongs to $\mathcal{R}^{200\times1}$. Thus we compute all the words in the reviews with the mean approach. The i-th word's vector is denote by v_i, and each review's vector is represented by S_j where j is the j-th review in the corpus. The expression as follows:

$$S_j = \sum_{\square=0}^{n} (v_i)$$

Equation 37

Where n is the number of words in the review.

After completing the vector transformation of the reviews in the CIMDB, it is necessary to construct the classifier models. Here, different tools, as described in Chapter 4, are used to build the classifier model. The Scikit-learn library is utilized to build the SVM model, and the Keras library is used to build the framework of MLP.

The review vector which has just been converted is ready for input to the SVM model. However, before starting the classification it needs to be determined whether

the data is linearly separable, as this impacts the choice of SVM model. By plotting the output of Word2Vec as shown in Figure 25 it can be surmised that the type of the data is non-linear separable. Thus the linear-SVC model of Scikit-learn can be excluded and the kernel function of SVC is set to be the ‘RBF’ kernel function.

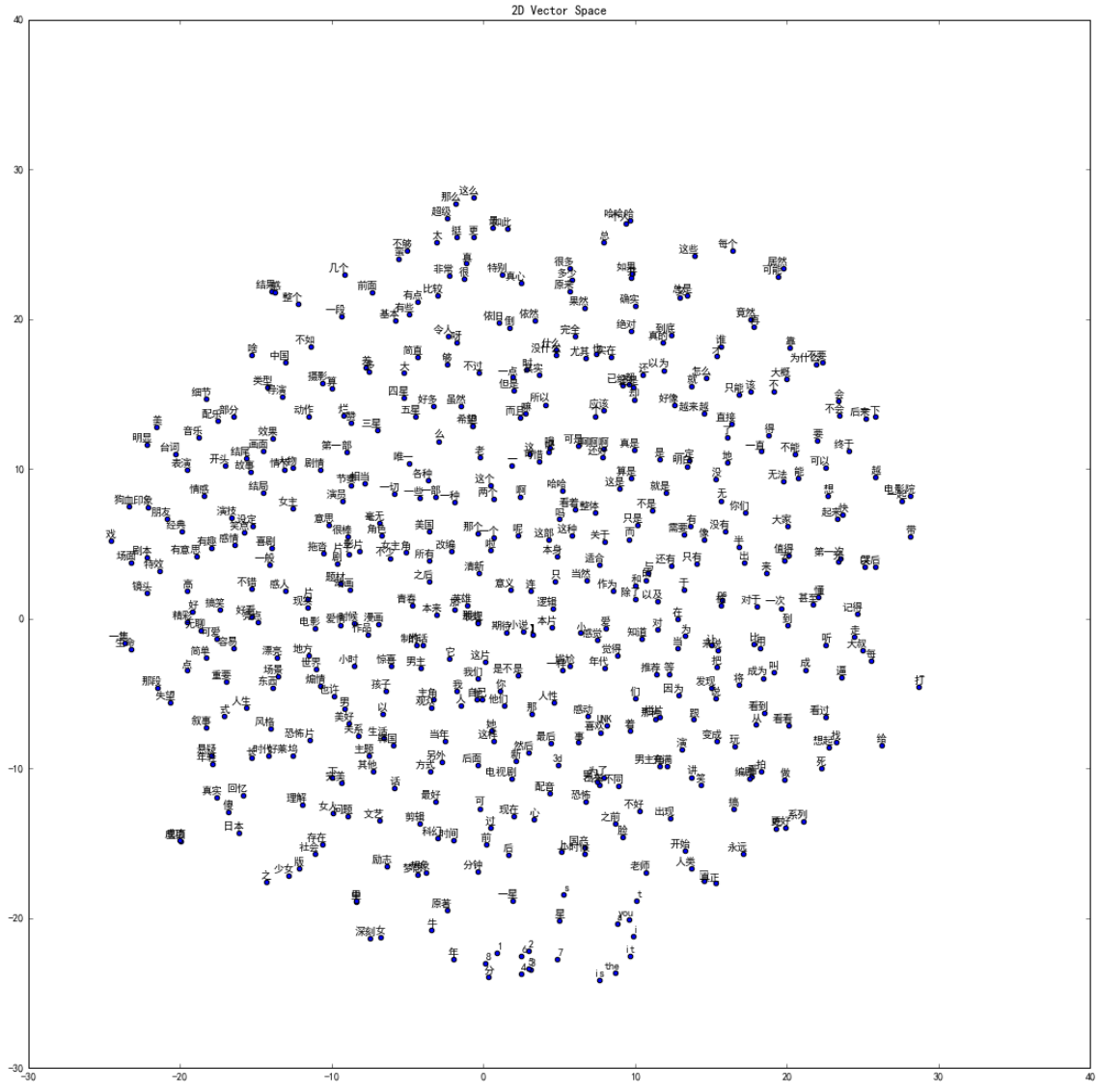

Figure 25 Word2vec illustrated by 2D image shows the distribution of Chinese words. For example, the digitals cluster together, and the highest frequency English words cluster together at the bottom of the figure.

5.2.3 Task 3

For the third subtask, the experiment is divided into two parts:

1. A context encoder is created and is trained using the CIMDB based on the sequence to sequence model (Sutskever et al., 2014). Then, the trained context encoder is used to convert the review comments in the CIMDB into vectors of length 300. If a review includes more than one sentence, it is possible to just use the average of the whole review using the mean approach to build the vector.

2. Then three types of classifiers are built and are utilized to classify the reviews in the CIMDB. Finally, the performance of different classifiers is measured by the measures of Accuracy, Precision, Recall, and the F-Measure detailed in Section 5.1.2

5.3 Experiment results and analysis.

In table 6, the result are shown for a feed-forward neural network with LSTM applied to the different language corpuses. Note that the CIMDB is we collected movie review corpus in Chinese, and the IMDB is the movie review in English (Maas et al., 2011).

In the experiment, 25000 reviews were randomly extracted from the IMDB and CIMDB respectively as the training set, and another 25000 reviews were randomly extracted as the test set. The experimental parameter values setting were described in section 5.2.1, and the training epoch was set to 3. For this experiment, the accuracy measure is used to evaluated the performance.

From the experimental results, the accuracy of Chinese corpus CIMDB is 69.99%, and the accuracy of English corpus IMDB is 85.99%. From the comparison it can be seen that the same model using different language corpus will return a significantly different value for the accuracy. It could be suggested that the performance difference is attributable to the greater demands required for the correct segmentation of the Chinese language.

Model	Corpus	Accuracy
Long-short-term memory	IMDB	85.99%

Long-short-term memory	CIMDB	69.99%

Table 7 Accuracy of different corpus for binary classification using LSTM

Following this, the word2vec model was used to implement the word embedding for Chinese corpus CIMDB.

Model	Accuracy	Precision	Error	Recall	F1
Word2vec + SVM	72.64	68.05	27.36	83.34	74.92
Word2vec + MLP	75.61	72.07	25.12	81.22	76.39

Table 8 Evaluation of different models using CIMDB corpus

From the table 8, it can be observed that Word2vec with SVM classifier and Word2vec with MLP classifier have very similar results across all measures for the task of binary classification. For example, Word2vec with SVM classifier achieved an accuracy of 71.74% and Word2vec with MLP classifier achieved an accuracy of 75.61%. From the evaluation of F1, the performance of SVM and MLP are not much different, only disagreeing by less than 1.5 units.

Comparing the LSTM and SVM classifiers, their performance in the task of binary classification for Chinese corpus is improved from 69.99% shown in Table 6 to 75.61% from Table 8. The results from both Tables illustrate that the Word2vec model can help improve the performance of CIMDB classification without having to borrowing any sentiment lexicons or manual rules.

Model	Accuracy	Precision	Error	Recall	F1
Context-encoder + SVM	65.89	59.37	34.11	75.34	66.40
Context-encoder + MLP	64.42	58.41	35.60	72.32	64.62
Context-encoder + LSTM	54.87	53.48	45.12	74.82	62.37

Table 9 Evaluation of different classifier based on context2vec

Table 9 shows the results of context2vec with different classifiers. In general, context2vec did not achieve what was hoped for. The average accuracy of three classifiers is 61.72%. Thus, their performance is lower than word2vec with SVM and MLP by a factor of 12%. The results clearly show that worst of all is LSTM so it can be concluded that it is not best to combine it with the context2vec embedding model. However, using it with SVM and MLP shows an better performance by approximately 10% in both cases. Overall, the measures for Context2Vec with SMV and MLP show very similar values.

Chapter 6

Conclusion and Future Work

In this chapter, we highlight the contributions that have been made in this dissertation and discuss topics that merit exploration in future work.

To summarise the outcome of the experiments, first of all it can be said that the three goals were successfully achieved. In the first experiment, the LSTM neural network was applied to classify different corpora. It was demonstrated that there is a 16% difference in the sentiment classification accuracy between Chinese corpora and English corpora. We consider the main reason is that the Chinese corpus classification introduces difficulty because of the additional task of word segmentation. In the second experiment, we employed the word2vec model with the different classifiers, SVM and MLP. The result showed that in this problem domain different types of classifier don't impact on the performance of word2vec model. Moreover, compared the experiment 1 that using LSTM classifier without word embedding and the experiment 2 that using word2vec model, the accuracy of models for sentiment classification has been significantly improved by use of word2vec model. The final experiment is used to verify whether the combination of the context2vec with different classifiers can produce good results. Unfortunately, the result showed that our proposed context vector representation for the sequence text does not achieve a good performance when used with any of the three classifiers investigated.

Finally, we have proved three conclusions through experiments:

1. The type of language in the case of the Chinese or the English corpora will exhibit a significant impact on the performance for the task of sentiment classification. According on the comparison of experimental process between two corpus in experiment 1, the main factor affecting the classification accuracy could be the word segmentation required for the Chinese word data.

2. The accuracy comparison between experiment 1 and experiment 2, it can be seen that using the word2vec model can significantly improve the performance for the Chinese language corpus.

3. Our proposed hypothesis that context vector can be used to improve the accuracy of prediction for classification did not achieve the desired result. However, it still has room for improvement. In the future work, the character-level model (J. Lee et al., 2016) will be applied to deal the Chinese corpus, thus we can ignores the affection of segmentation. At the same time, we will try to use the convolutional neural network (CNN) to classify the CIMDB. In the previous work, CNN shows it achieved a good performance based on the word vector (Kim, 2014). Thus, it has the potential to classify the context vector.

In the future work, we would like apply our model to fine-grained opinion mining tasks. We would also like to explore the performance of our model works on other language corpus. Although context2vec model did not achieve the expectation, we believe that context vector still worth studying.

Bibliography

Aggarwal, C. C., & Zhai, C. (2012). *Mining text data*. Springer. Retrieved from https://books.google.ie/books?hl=en&lr=&id=vFHOx8wfSU0C&oi=fnd&pg=PR3&dq=neural+network+sentiment+analysis&ots=oc1iVEgIUq&sig=M5IBNYzfkg9Cp7pg-RfJyq0ggYY&redir_esc=y#v=onepage&q=neural network sentiment analysis&f=false

Alisneaky. (2011). File:Kernel Machine.png - Wikimedia Commons. Retrieved October 29, 2017, from https://commons.wikimedia.org/wiki/File:Kernel_Machine.png

Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. (1998). *Atmospheric Environment*, *32*(14–15), 2627–2636. http://doi.org/10.1016/S1352-2310(97)00447-0

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. Retrieved from http://arxiv.org/abs/1409.0473

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count , predict ! A systematic comparison of context-counting vs . context-predicting semantic vectors. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.*, 238–247. http://doi.org/10.3115/v1/P14-1023

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20. http://doi.org/10.1145/1007730.1007735

Bengio, Y., Courville, A., & Vincent, P. (2012). Representation Learning: A Review and New Perspectives. Retrieved from http://arxiv.org/abs/1206.5538

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., Kandola, J., Hofmann, T., … Shawe-Taylor, J. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. http://doi.org/10.1109/72.279181

Bespalov, D., Bai, B., Qi, Y., & Shokoufandeh, A. (2011). Sentiment Classification Based on Supervised Latent n-gram Analysis. Retrieved from http://delivery.acm.org/10.1145/2070000/2063635/p375-bespalov.pdf?ip=149.157.116.88&id=2063635&acc=ACTIVE

SERVICE&key=846C3111CE4A4710.AB4E84BDC4F162A6.4D4702B0C3E38B35.4D4702B0C3E38B35&__acm__=1527177906_b16f2a7124a91c0f66ddc8f5b782454f

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Clarendon Press. Retrieved from https://books.google.ie/books?hl=en&lr=&id=T0S0BgAAQBAJ&oi=fnd&pg=PP1&dq=multilayer+Neural+Networks+pattern+recognition&ots=jM20sGbzod&sig=YycuHjeP8IGJnYIV5jPrFykxcoY&redir_esc=y#v=onepage&q=multilayer Neural Networks pattern recognition&f=false

Bollen, J., Mao, H., & Zeng, X.-J. (2010). Twitter mood predicts the stock market. *Journal of Computational Science*, Pages 1-8. http://doi.org/10.1016/j.jocs.2010.12.007

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92* (pp. 144–152). New York, New York, USA: ACM Press. http://doi.org/10.1145/130385.130401

Cavnar, W. B., Cavnar, W. B., & Trenkle, J. M. (1994). N-Gram-Based Text Categorization. *IN PROCEEDINGS OF SDAIR-94, 3RD ANNUAL SYMPOSIUM ON DOCUMENT ANALYSIS AND INFORMATION RETRIEVAL*, 161--175. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9367

CBOW of Word2Vec. (2017). Retrieved October 29, 2017, from http://www.cs.nthu.edu.tw/~shwu/courses/ml/labs/10_Keras_Word2Vec/10_Keras_Word2Vec.html

Chang, P.-C., Galley, M., & Manning, C. D. (2008). Optimizing Chinese Word Segmentation for Machine Translation Performance, 224–232. Retrieved from https://nlp.stanford.edu/manning/papers/acl08-cws-final.pdf

Chen, X., Qiu, X., Zhu, C., Liu, P., & Huang, X. (2015). Long Short-Term Memory Neural Networks for Chinese Word Segmentation. *Emnlp*, (September), 1197–1206.

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. Retrieved from http://arxiv.org/abs/1409.1259

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk,

H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Retrieved from http://arxiv.org/abs/1406.1078

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Retrieved from http://arxiv.org/abs/1412.3555

Dave, K., Lawrence, S., & Pennock, D. M. (2003). Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *Proceedings of the twelfth international conference on World Wide Web - WWW '03* (p. 519). New York, New York, USA: ACM Press. http://doi.org/10.1145/775152.775226

David Kirkpatrick. (2016). Study: Online reviews have big impact on offline purchases | Marketing Dive. Retrieved October 31, 2017, from https://www.marketingdive.com/news/study-online-reviews-have-big-impact-on-offline-purchases/415355/

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. http://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

dos Santos, C. N., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. *Proceedings of the 25th International {…}*, (August), 69–78. Retrieved from http://www.anthology.aclweb.org/C/C14/C14-1008.pdf

Dosovitskiy, A., & Brox, T. (2015). Inverting Visual Representations with Convolutional Networks. Retrieved from http://arxiv.org/abs/1506.02753

Freund, Y., & Schapire, R. E. (1999). Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, *37*(3), 277–296. http://doi.org/10.1023/A:1007662407062

Friedman, J. H. (1997). On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77. http://doi.org/10.1023/A:1009778005914

Gers, F. A., & Schmidhuber, J. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, *12*(6), 1333–1340. http://doi.org/10.1109/72.963769

Getoor, L., Scheffer, T., & International Machine Learning Society., Y. (2011). *Domain adaptation for large-scale sentiment classification: a deep learning approach*. *Proceedings of the 28th International Conference on International Conference on Machine Learning*. [International Machine Learning Society]. Retrieved from https://dl.acm.org/citation.cfm?id=3104547

Golub, G. H. (Gene H., & Van Loan, C. F. (1996). *Matrix computations*. Johns Hopkins University Press. Retrieved from https://dl.acm.org/citation.cfm?id=248979

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). IEEE. http://doi.org/10.1109/ICASSP.2013.6638947

Harris, Z. S. (1981). Distributional Structure. In H. Hiż (Ed.), *Papers on Syntax* (pp. 3–22). Dordrecht: Springer Netherlands. http://doi.org/10.1007/978-94-009-8467-7_1

Hatzivassiloglou, V., McKeown, K. R., Hatzivassiloglou, V., & McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics* - (pp. 174–181). Morristown, NJ, USA: Association for Computational Linguistics. http://doi.org/10.3115/976909.979640

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., … Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. http://doi.org/10.1109/MSP.2012.2205597

Hochreiter, S., & Schmidhuber, J. (1997). LONG SHORT-TERM MEMORY. *Neural Computation*, *9*(8), 1735–1780. Retrieved from http://www7.informatik.tu-muenchen.de/~hochreit

Hu, B., Chen, Q., & Zhu, F. (2015). LCSTS: A Large Scale Chinese Short Text Summarization Dataset. Retrieved from http://arxiv.org/abs/1506.05865

Huang, M., Cao, Y., & Dong, C. (2016). Modeling Rich Contexts for Sentiment Classification with LSTM. Retrieved from http://arxiv.org/abs/1605.01478

Image processing with neural networks—a review. (2002). *Pattern Recognition*, *35*(10), 2279–2301. http://doi.org/10.1016/S0031-3203(01)00178-9

Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks.

Retrieved May 22, 2018, from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Keras Documentation. (n.d.). Retrieved October 25, 2017, from https://keras.io/

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. http://doi.org/10.1109/LSP.2014.2325781

Ko, Y., & Youngjoong. (2012). A study of term weighting schemes using class information for text classification. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12* (p. 1029). New York, New York, USA: ACM Press. http://doi.org/10.1145/2348283.2348453

Kukich, K., & Karen. (1992). Technique for automatically correcting words in text. *ACM Computing Surveys*, *24*(4), 377–439. http://doi.org/10.1145/146370.146380

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282–289). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645530.655813

Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. Retrieved from http://arxiv.org/abs/1405.4053

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. http://doi.org/10.1038/nature14539

Lee, D. D., & Seung, H. S. (2001). Algorithms for Non-negative Matrix Factorization. Retrieved from http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization

Lee, J., Cho, K., & Hofmann, T. (2016). Fully Character-Level Neural Machine Translation without Explicit Segmentation. Retrieved from http://arxiv.org/abs/1610.03017

Lin, C., & He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09* (p. 375). New York, New York, USA: ACM Press. http://doi.org/10.1145/1645953.1646003

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., & Zaremba, W. (2014).

Addressing the Rare Word Problem in Neural Machine Translation. Retrieved from http://arxiv.org/abs/1410.8206

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (pp. 142–150). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2002472.2002491

Martineau, J., Martineau, J., Finin, T., Finin, T., Fink, C., Fink, C., … Others. (2008). Delta TFIDF: An Improved Feature Space for Sentiment Analysis. *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM*, *29*(May), 490–497. Retrieved from http://ebiquity.umbc.edu/papers/select/person/Tim/Finin/

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 1–12. http://doi.org/10.1162/153244303322533223

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Retrieved from http://arxiv.org/abs/1310.4546

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, *3*(4), 235–244. http://doi.org/10.1093/ijl/3.4.235

Mohammad, S., Dunne, C., & Dorr, B. (2009). Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 2 - EMNLP '09* (Vol. 2, p. 599). Morristown, NJ, USA: Association for Computational Linguistics. http://doi.org/10.3115/1699571.1699591

Moreno, P. J., & Ho Hewlett-Packard, P. P. (2003). A New SVM Approach to Speaker Identification and Verification Using Probabilistic Distance Kernels. *EUROSPEECH*. Retrieved from https://www.isca-speech.org/archive/archive_papers/eurospeech_2003/e03_2965.pdf

Nguyen, A., Yosinski, J., & Clune, J. (2014). Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. Retrieved from

http://arxiv.org/abs/1412.1897

opencv dev team. (2017). Welcome to opencv documentation! — OpenCV 2.4.13.4 documentation. Retrieved October 31, 2017, from https://docs.opencv.org/2.4/index.html

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *LREc* (Vol. 10).

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends R in Information Retrieval*, *2*, 1–2. http://doi.org/10.1561/1500000001

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02* (Vol. 10, pp. 79–86). Morristown, NJ, USA: Association for Computational Linguistics. http://doi.org/10.3115/1118693.1118704

Reynar, J. C. (1998). Topic Segmentation: Algorithms And Applications. Retrieved from http://repository.upenn.edu/ircs_reports

Russell, I. (2012). The Delta Rule. Retrieved October 29, 2017, from http://uhavax.hartford.edu/compsci/neural-networks-delta-rule.html

scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation. (n.d.). Retrieved October 30, 2017, from http://scikit-learn.org/stable/

scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation. (2017). Retrieved October 31, 2017, from http://scikit-learn.org/stable/

Sebastiani, F., & Fabrizio. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. http://doi.org/10.1145/505282.505283

Seki, Y., Evans, D. K., Ku, L.-W., Chen, H.-H., Kando, N., & Lin, C.-Y. (2007). Overview of Opinion Analysis Pilot Task at NTCIR-6. *Proceedings of NTCIR-6 Workshop Meeting*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.364.9258&rep=rep1&type=pdf

Shestakov, D. (2013). Current Challenges in Web Crawling. In F. Daniel, P. Dolog, & Q. Li (Eds.), *Web Engineering: 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings* (pp. 518–521). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-39200-9_49

Socher, R. (2014). RECURSIVE DEEP LEARNING FOR NATURAL LANGUAGE

PROCESSING.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, 1631–1642. Retrieved from http://www.aclweb.org/anthology/D13-1170

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. Retrieved from http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Strzalkowski, T. (1999). *Natural language information retrieval*. MIT Press.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Retrieved from http://arxiv.org/abs/1409.3215

Suykens, J. A. K., & Vandewalle, J. (1999). Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, *9*(3), 293–300. http://doi.org/10.1023/A:1018628609742

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, *37*(2), 267–307. http://doi.org/10.1162/COLI_a_00049

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (pp. 1556–1566). Retrieved from https://www.aclweb.org/anthology/P15-1150

Tang, D., Qin, B., & Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1422–1432). Retrieved from http://aclweb.org/anthology/D15-1167

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification *. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (pp. 1555–1565). Retrieved from http://www.aclweb.org/anthology/P14-1146

Tatemura, J., & Junichi. (2000). Virtual reviewers for collaborative exploration of movie reviews. In *Proceedings of the 5th international conference on Intelligent user interfaces - IUI '00* (pp. 272–275). New York, New York, USA: ACM Press.

http://doi.org/10.1145/325737.325870

TensorFlow. (n.d.). Retrieved October 29, 2017, from https://www.tensorflow.org/

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997). PHOAKS: a system for sharing recommendations. *Communications of the ACM*, *40*(3), 59–62. http://doi.org/10.1145/245108.245122

Tseng, H., Tseng, H., Chang, P., Andrew, G., Jurafsky, D., & Manning, C. (2005). A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. *PROCEEDINGS OF THE FOURTH SIGHAN WORKSHOP ON CHINESE LANGUAGE PROCESSING*, 168--171. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.329.6123

Tumasjan, A., Sprenger, T. O., Sandner, P. G., & Welpe, I. M. (2010). Predicting elections with Twitter: What 140 characters reveal about political sentiment. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. http://doi.org/{{{doi}}}

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning, 384–394. Retrieved from http://metaoptimize.

Turney, P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, (2002), Philadelphia, Pennsylvania*, 417–424. Retrieved from http://arxiv.org/abs/cs/0212032

Turney, P. D., & Littman, M. L. (2002). Unsupervised Learning of Semantic Orientation from a Hundred-Billion-Word Corpus. Retrieved from http://arxiv.org/abs/cs/0212012

Understanding LSTM Networks -- colah's blog. (n.d.). Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unsupervised Feature Learning and Deep Learning Tutorial. (n.d.). Retrieved October 29, 2017, from http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

Vanzo, A., Croce, D., & Basili, R. (2014). A context-based model for Sentiment Analysis in Twitter. In *the 25th International Conference on Computational Linguistics* (pp. 2345–2354). Retrieved from http://www.aclweb.org/anthology/C14-1221

Vector Representations of Words | TensorFlow. (n.d.). Retrieved October 29, 2017,

from https://www.tensorflow.org/tutorials/word2vec

Wang, J., Yu, L.-C., Lai, K. R., & Zhang, X. (2016). Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 225–230). Retrieved from http://www.aclweb.org/anthology/P16-2037

Yi, J., Yi, J., Nasukawa, T., Bunescu, R., & Niblack, W. (2003). Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. *IN IEEE INTL. CONF. ON DATA MINING (ICDM*, 427--434. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.102.7125

Zhang, C., Zeng, D., Li, J., Wang, F.-Y., & Zuo, W. (2009). Sentiment analysis of Chinese documents: From sentence to document level. *Journal of the American Society for Information Science and Technology*, *60*(12), 2474–2487. http://doi.org/10.1002/asi.21206

Zhang, D., Xu, H., Su, Z., & Xu, Y. (2015). Chinese comments sentiment classification based on word2vec and SVMperf. *Expert Systems with Applications*, *42*(4), 1857–1863. http://doi.org/10.1016/J.ESWA.2014.09.011

Zhou, S., Li, K., & Liu, Y. (2008). Text Categorization Based on Topic Model. In *Rough Sets and Knowledge Technology* (pp. 572–579). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-540-79721-0_77