Tampere University

Yi Zhou

DEEP NEURAL NETWORK BASED ALGORITHMS PERFORMANCE COMPARISON FOR SENTIMENT CLASSIFICATION

Faculty of Information Technology and Communication Sciences
M.Sc. thesis
April 2019

ABSTRACT

Yi Zhou: Deep neural network based algorithms Performance Comparison for Sentiment Classification
M.Sc. thesis
Tampere University
Programme in Software Development
April 2019

Text sentiment classification is one important task in machine learning area. Deep neural networks based algorithms have the best performance on the text sentiment classification. In this master thesis, three deep neural network models were tested on the TripAdvisor hotel reviews dataset and Stanford sentiment tree dataset. Experiments showed that the bidirectional long short term memory network achieved the highest accuracy while the very deep convolutional neural network had the worst predicted results on the TripAdvisor dataset. With the misclassified hotel reviews analyzed, it found that the hotel reviews with more contradictory sentimental words were easier to be predicted wrongly.

There are 4 chapters in this thesis. Chapter 1 introduces the sentiment classification task and its use cases. Chapter 2 illustrates the background of deep neural network based algorithms on text classification task. Chapter 3 is the experiment work and discussion for the classification result of the experiment. Chapter 4 completes the thesis work with the conclusions.

Keywords: deep neural networks, convolutional neural network, recurrent neural network, sentiment classification, hotel reviews

PREFACE

Tampereella, 26th April 2019

Yi Zhou

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROGRAMS AND ALGORITHMS

LIST OF SYMBOLS AND ABBREVIATIONS

BiLSTM	Bidirectional Long Short Term Memory
BPTT	Backpropagation through time
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GRU	Gate recurrent units
ILSVRC	ImageNet Large Scale Visual Recognition Competition
JSON	JavaScript Object Notation
LR	Linear Regression
LSTM	Long short-term Memory
NB	Naive Bayes
NLP	Natural Language Processing
PCA	Principle Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic gradient descent
SVM	Support Vector Machine
TAU	Tampere University
Text-CNN	Text Convolutional Neural Network
TUNI	Tampere Universities
URL	Uniform Resource Locator
VD-CNN	Very Deep Convolutional Neural Network

1 INTRODUCTION

1.1 Motivation

The rise of online social media and other websites has led to an explosion of text data. Users can express their opinion conveniently and an increasing number of users are willing to share their opinions online. For example, users share product reviews on the Amazon website after purchasing a product and leave their comments on IMDB [1] or other similar websites after watching movies. Users also leave comments on friends' posts on social media websites, such as Facebook [2], and Twitter [3]. One of the typical scenarios of online reviews is hotel reviews. After staying in a hotel, people share their opinion about accommodation on websites, such as Booking [4], Airbnb and TripAdvisor [5]. These hotel reviews are important and useful for both customers and hotel owners. Through analyzing these reviews, hotel owners can know where their problems are and solve these mentioned issues to improve their service quality. Companies can find the defect of their products and address these problems in the next version of products. However, review data has increased dramatically during these years and human-labor method is not practical to handle and analyze the massive data. This situation produces an urgent problem in industry. For example, in sociology, economics and other areas, sentiment analysis has shown its significant meaning and the wide applied prospect. All of these urgent needs have been a big challenge for researchers, how to effectively analyze and utilize these review data.

One method to analyze these reviews is the text sentiment analysis (also named opinion mining). Sentiment analysis requires machine learning related knowledge and natural language processing technique. Many machine learning algorithms were researched and developed to apply in the sentiment analysis task. Sentiment analysis has been an important subtopics of natural language processing (NLP). There are many specific small research directions in text sentiment analysis. One main sub research direction is the text sentiment classification. In text sentiment classification, many different directions have been researched to solve this task. One method is publishing the new text review dataset as the common benchmark, researcher use the specific dataset to compare the

[1] website link: `https://www.imdb.com/`
[2] website link: `https://www.facebook.com/`
[3] website link: `https://twitter.com//`
[4] website link: `https://www.booking.com/`
[5] website link: `https://www.TripAdvisor.com/`

performance of their algorithms. Another method is introducing a new classification algorithm to achieve more accurate prediction result. Regarding introducing new classification algorithms, recent research attention is focusing on deep neural network based method since the big success of deep learning technique in computer vision in 2012. Deep neural network based algorithms have achieved big improvement than previous algorithms on the text sentiment classification.

1.2 Objective

The goal of this thesis is to illustrate the whole processing step of using deep neural network based algorithms for the text sentiment classification task. These steps include retrieving the new text data, pre-processing these reviews, constructing deep neural networks and comparing the performance of these algorithms on two text review dataset. The topic area is identified as sentiment classification on hotel reviews. The aim of this thesis is to explore the performance of three deep neural networks for the sentiment classification task on text review. The research questions in this thesis can be summarized as below.

The first research question is to compare the performance of three deep neural network models on the TripAdvisor dataset and Stanford sentiment tree dataset for the text classification task. The metrics includes accuracy and F score. After completing the experiment, it can know which deep neural network has the highest accuracy and F score and which model has the worst accuracy and F score on these two datasets.

The second research question is to compare the effect of the GloVe and FastText word embedding technique for word vector initialization on deep neural network models for the text sentiment classification task. Through the experiment, it can know which word embedding initialization technique has the better performance on the text review sentiment classification task.

The third research question is to analyze the misclassified hotel reviews on the sentiment predication task and find out the difference between misclassified reviews and correct predicted reviews.

2 LITERATURE REVIEW

The goal of the literature review chapter in this thesis is to provide comprehensive background information about the sentiment classification algorithms. In this chapter, the general research progress of deep neural network methods on the sentiment classification task is firstly described. Then the detailed concept and related theoretical background about a deep neural network are introduced, including the basic module of a deep neural network, the common regularization technique and the common strategy for training a deep neural network. The third subsection introduces one basic concept, word embedding, in natural language processing technique. In the fourth subsection, common assessment metrics are introduced to evaluate the performance of different algorithms on the text sentiment classification task.

2.1 Sentiment classification task introduction

The objective of text sentiment classification is to classify texts into different categories accurately according to the sentimental polarity expressed from the text. Generally, the sentiment of the text is labeled into 3 classes or 5 classes. For the type of 3 classes, the sentiment of one review is classified as negative, neutral or positive. For the 5 classes type, sentiment for one text is labeled as one of the following class: very negative, negative, neutral, positive or very positive. Table 2.1 shows examples of sentiment classification on text reviews.

Use cases of sentiment analysis

Sentiment text classification is an important task in Natural Language Processing (NLP). Sentiment analysis has been widely applied in many areas with many applications, such as search engine, text information retrieval, machine reading and understanding etc. There are many use cases for applying text classification methods.

Table 2.1. *examples of text review sentiment classification*

example	**sentiment class**
“The strange thing is that it works”	positive
“Shattered image isn’t complex, it’s just stupid and boring”	negative
“Far from bewitching, the crucible tests the patience”	neutral

One example is the news category classification task. Text news related to different topics needs to be classified to their correct corresponding topics. There are many topics including sports, finance, technology, health etc. Each topic is considered as one class. Sports news should be classified as the sport category instead of finances. With the help of the text classification technique, users can avoid wasting time on reading unrelated articles. The second example is that sentiment analysis can be used to predict market trends through analyzing customers' reviews about some specific products. A real example of this application is about Samsung Note 7 battery crisis. The number of negative sentiment tweets of Samsung in Twitter increased dramatically after the Samsung note 7 battery crisis happened. Through observing the change of the number of positive and negative sentimental tweets, Samsung company can evaluate the effect of this crisis ahead.

Another common usage of sentiment analysis is collecting and analyzing software application reviews [22] [21]. After each update of an application in iOS or Android operating system, software developers want to know what users think of the newly developed version. Through utilizing the sentiment analysis technique on these application reviews, developers can conclude clear feedback from users' comments.

Sentiment analysis is also helpful in the politics area. When a new policy is proposed, the government can gather people's opinion and discussion content from multiple sources to get feedback on the new policy. Then the government can improve or adjust policy in the future.

2.2 Deep neural network

In this section, details of one single neuron and activation functions are first introduced. Secondly, the model structure of DNN is described. Thirdly, two main types of deep neural networks, recurrent neural network and convolutional neural network, are elaborated.

2.2.1 Neural network

A single neuron is composed of each layer in the artificial neural network. Activation functions are the basic component of an artificial neuron.

Single neuron cell

The single neuron model is shown in Figure 2.1. The inputs are the vector (x_1, x_2, x_3 ... x_t), the matching weight for the input is the matrix (W_{1j}, W_{2j}, W_{3j} ... W_{nj}). All inputs are first multiplied by the W variable, the temporary result is generated and inputted into the activation function. The final result for the single neuron is outputted. This value will be treated as the input value for the next layer. The mathematical form of forward computation in a single neuron is given in Equation 2.1. In the equation, b is the bias, x

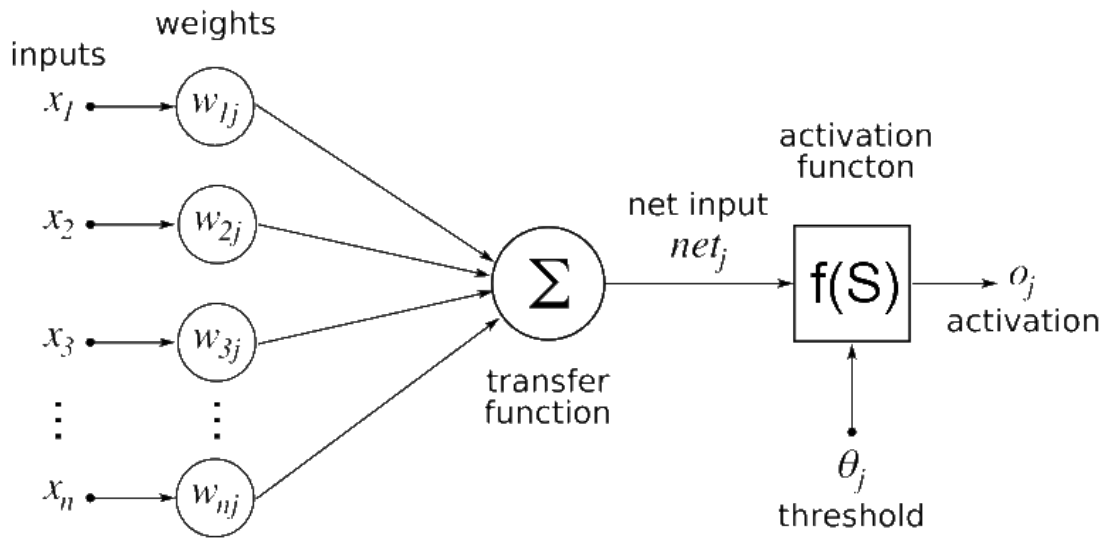

Figure 2.1. *Single neuron model*

is the input to neuron, w is the weight in one single neuron, n represents the number of inputs from the previous layer, i counters from 0 to n, f denotes the activation function.

$$o_j = f\left(b + \sum_{i=1}^{n} x_i w_i\right) \tag{2.1}$$

Activation function

For the activation function, one numerical value is accepted as the input. The output is calculated after applying the mathematical calculation. There are a few common used activation functions for single neuron such as sigmoid, rectified linear unit (ReLU)[43].

The sigmoid activation function is one common used activation function for calculating probability. The mathematical form of the sigmoid function is given in Equation 2.2. The plotting of this activation function is shown in Figure 2.2. It is seen that the sigmoid function takes a real-valued number and maps the output into a range between 0 and 1. In particular, when a large negative number is inputted into the sigmoid function, the output value is near 0. If a large positive number is inputted into the sigmoid function, one value near 1 will be generated. For the sigmoid function, the sum of probability for all the output neurons is 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

The ReLU activation function is one popular activation function. The mathematical form of the ReLU activation function is given in Equation 2.3 and is plotted in Figure 2.3. It shows that ReLU activation function is a non-linear function. The output value range of the ReLU is in the positive area or zero for any numerical input. The ReLU activation function has two advantages. The first one is that the ReLU function is simple to com-

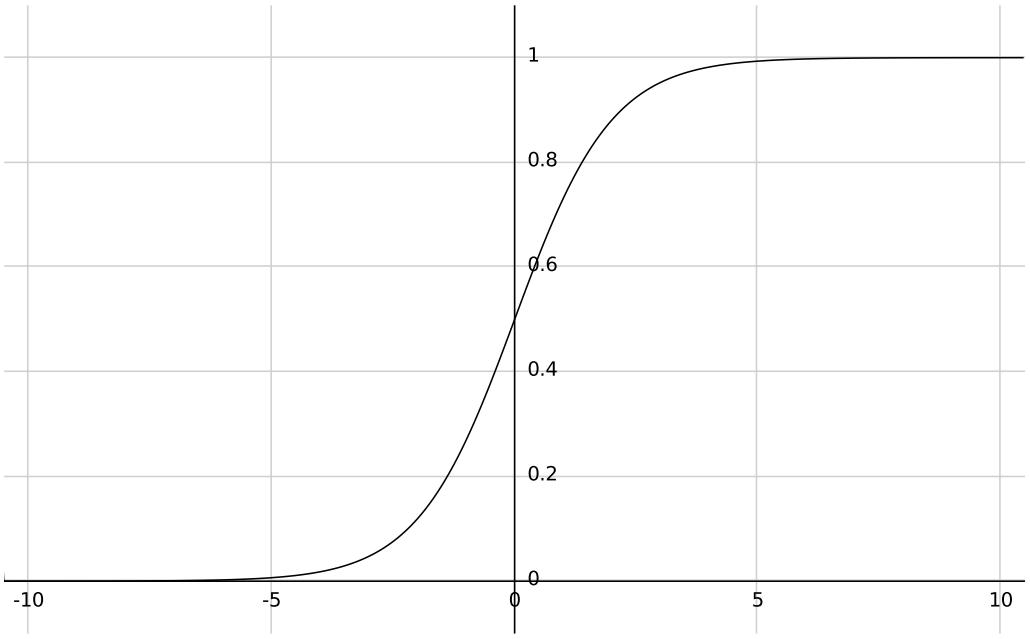

Figure 2.2. *sigmoid activation function*

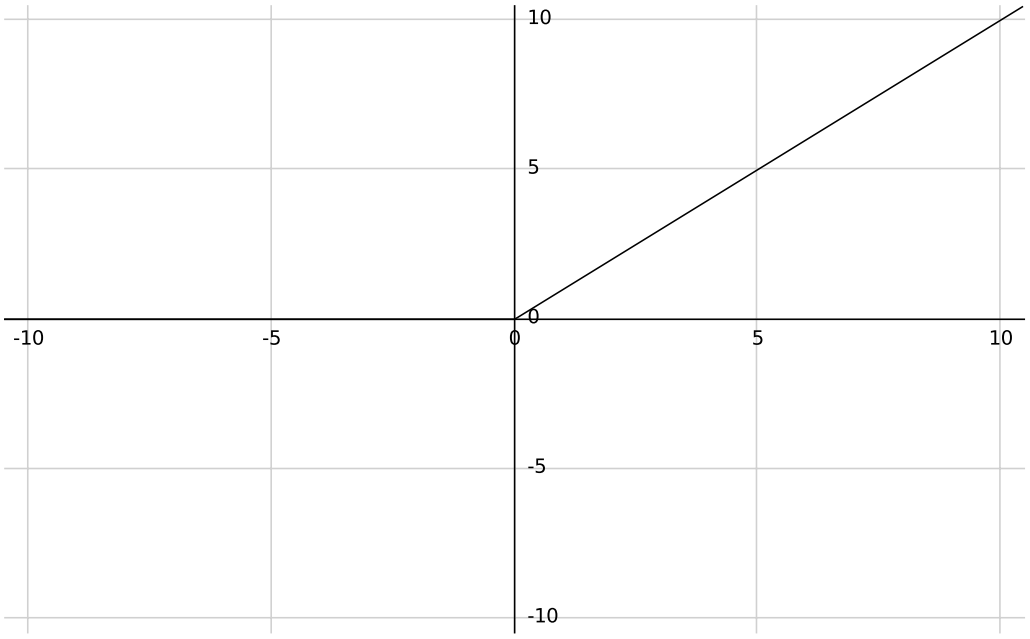

Figure 2.3. *ReLU activation function*

pute, this feature gives the benefit on computing speed. The second advantage is that the gradient of ReLU is constant. Therefore, ReLU does not saturate like the sigmoid activation functions. These two advantages lead to fast converge speed when a DNN model with ReLU activation function is trained.

$$f(x) = \max(0, x) \tag{2.3}$$

The Tanh activation function is a non-linearity function. The mathematical form of this function is given in Equation 2.4. The output range of the Tanh function is $[-1, 1]$. The

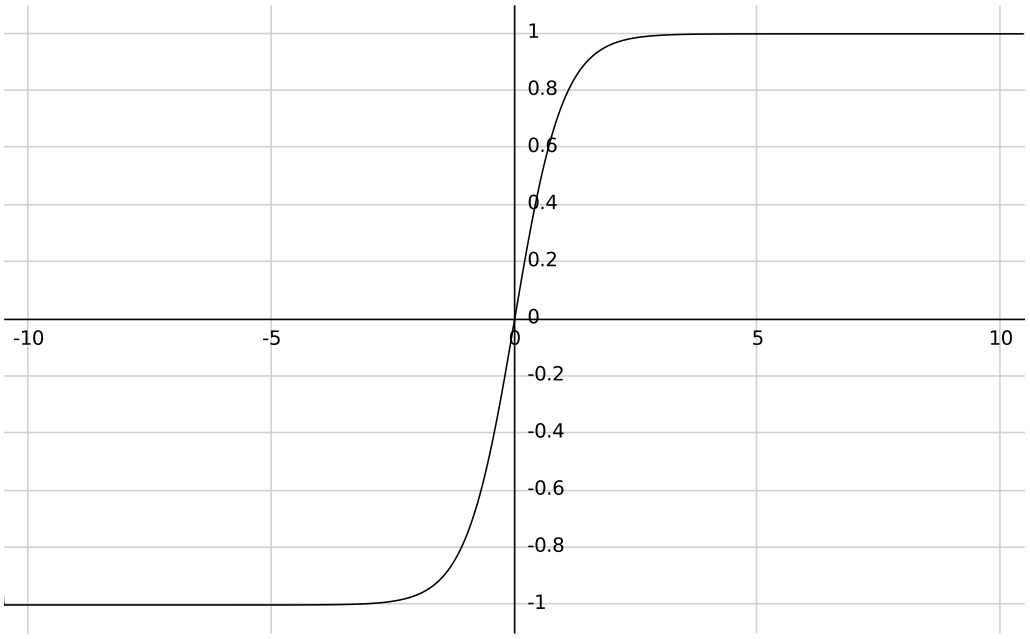

Figure 2.4. *Tanh activation function*

Tanh activation function is like the sigmoid neuron, which the activation function saturates. However, the output of the Tanh function is zero-centered. The plotting of this activation function is shown in Figure 2.4. Therefore, the Tanh activation function is preferred than the sigmoid function in practice.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.4}$$

Neural network

The typical architecture of a neural network consists of input layer, hidden layer and output layer. One classical demonstration of a neural network is the multi-layer perceptron. The architecture of multi-layer perception is shown in Figure 2.5. From this figure, it shows that there are one input layer, one hidden layer and one output layer. Regarding the architecture of deep neural network, deep neural networks consist of more that one hidden layers. Deep neural networks is also named deep learning [20].

2.2.2 Convolutional neural network

Convolutional neural network (CNN) is an important category of deep neural network. CNN models are commonly applied for solving computer visual imagery related tasks. CNN models usually consist of convolutional layers, hidden layers, pooling layers and fully connected layers.

Convolutional layer

Convolutional layer is the key element for building a CNN model. Through the convolution

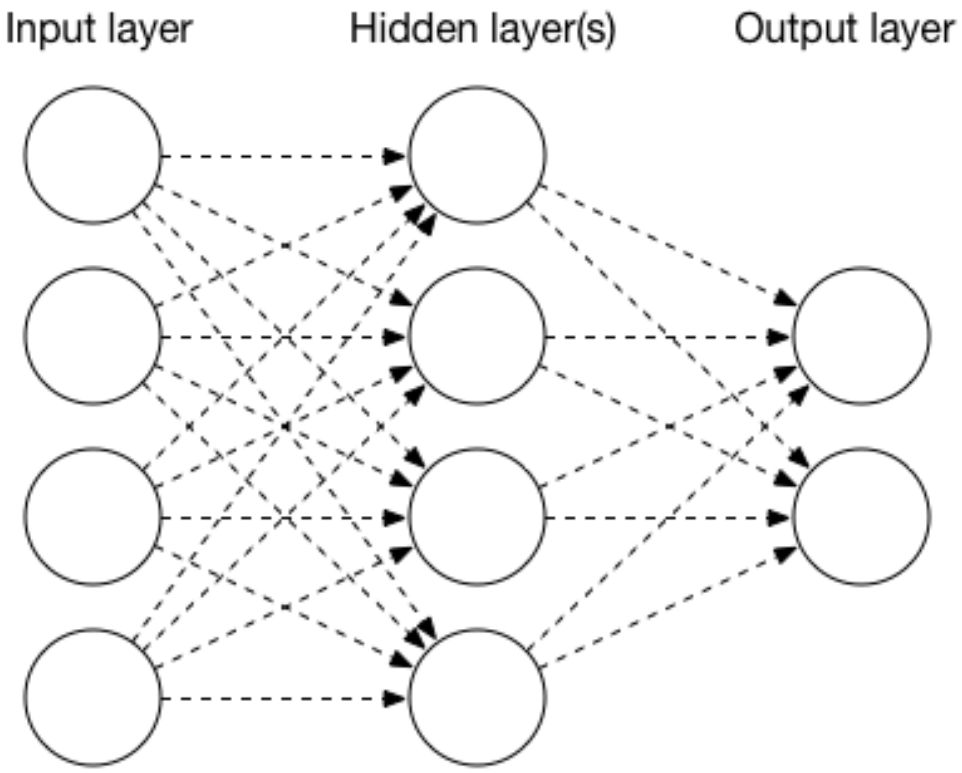

Figure 2.5. *Typical model of the multi-layer perceptron*

operation, parameter sharing scheme is applied to reduce the number of parameters. One demonstration of the convolution operation in CNN model is shown in Figure 2.6. It shows that the volume size of the input feature map is 3×4, the convolution kernel size is 2×2 and the stride step is 2. The dot product is applied to each element. The output volume is generated after the convolution operation. The volume size of the output is 2×3.

Pooling layer

Pooling layer is another important component in CNN models. The pooling operation in CNN is to aggregate the spatial feature. There are two main pooling operations. The first one is the max pooling operation and the second one is the average pooling. For the max pooling operation, it extracts the most obvious feature from the original feature map. For the image data, the max pooling operation can detect edges for the image data. The average pooling operation extracts features in a smooth manner. The average pooling operation takes all value inside the convolutional windows and calculate the average value out of the window. This indicates that the average pooling operation takes into all values into account. The figure 2.7 shows one demonstration of comparing the max pooling operation and the average pooling operation. The feature map is 4×4 sizes. The convolutional windows size is 2×2. After the max pooling operation, the biggest value in each window is selected. For the average pooling operation, the mean value in each convolutional window is calculated.

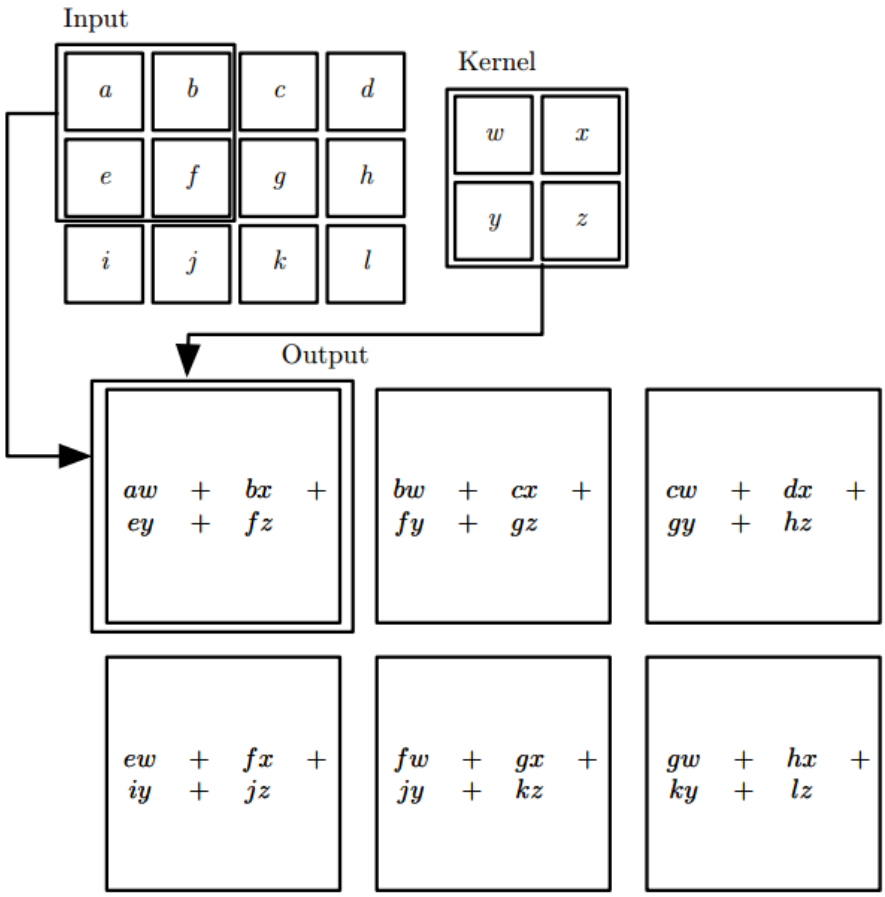

Figure 2.6. *Convolution operation illustration*

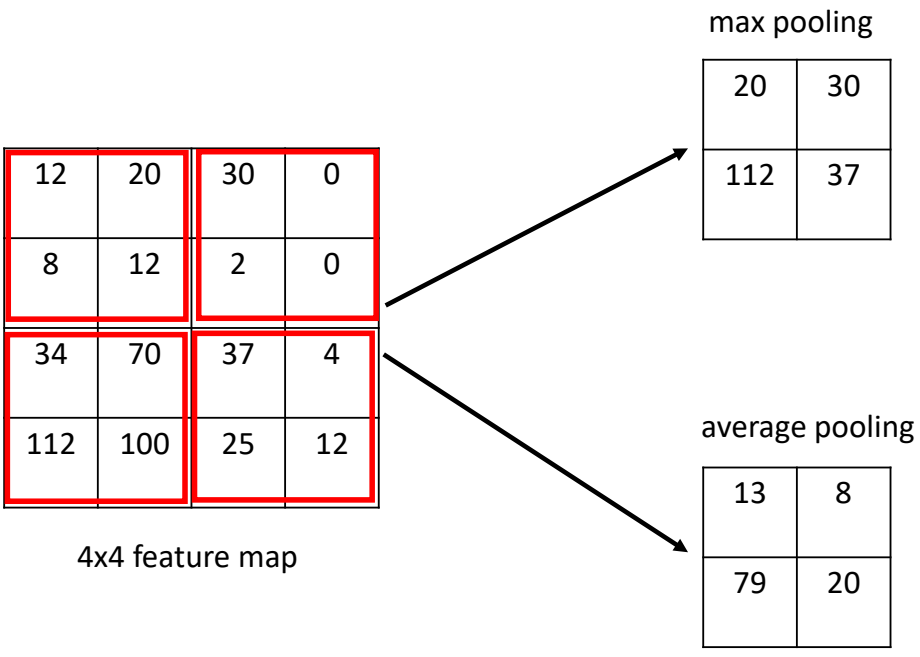

Figure 2.7. *Max pooling and average pooling operation comparison*

Common CNN models

During the development of deep neural networks, many important and powerful variant CNN models were proposed. AlexNet [18] won the champion in the ILSVRC [32] 2012 competition. The task for this competition is to classify 1000 images classes. These classes are dog, cat, flowers, plane etc. AlexNet model achieved the highest 83.6%accuracy, and it decreased 9.4% error rate when compared to the model in the previous year. Regarding the AlexNet architecture, it is a typical modern CNN model, which consists of five convolutional layers with ReLU activation function, three max-pooling layers, two normalization layers, two dropout layers and one fully connected layer with softmax activation function. One feature of AlexNet model is that two identical branches were followed after the input because of the limitation of the computing resources. The output of the two final fully connected layers were concatenated together.

In 2013, VGG-16 CNN model [3] won the ILSVRC champion. One key contribution of the VGG-16 model was the model depth. The VGG-16 model increased the depth of 8 layers in AlexNet to 16 layers of VGG model. It was the deepest CNN model in 2013 and the VGG-16 model decreased the error rate to 7.3%. VGG-16 model showed the powerful learning ability. For the model architecture, the VGG-16 model comprised 13 convolutional layers and 3 fully connected layers. The reception filters size for convolution operation was 3×3. Another contribution was the concept of repeating module to construct deep neural networks. It was the first time to use repeated sub-module to construct CNN model. This idea of using repeating block was inherited by many other models.

In 2015, ResNet [6] was proposed by Kaiming et al. ResNet model with 50 convolutional layers won the ILSVRC competition in the year. One key innovation of ResNet model was proposing shortcut operation to mitigate the gradient vanishing problem in deep neural networks. Deep neural networks become harder to train and suffer gradient vanishing problem when the depth of the model increases. Through the shortcut operation, the gradient can pass to the next layers more easily. Shortcut operation becomes the one useful technique for building CNN model.

2.2.3 Recurrent neural network

Recurrent neural network (RNN) [25] is another important type of deep neural network. RNN model is good at processing sequential data. There are many different types of sequence data. Text is one important category of sequence data. RNN model can be categorized into four different types. There are one-to-one RNN model, one-to-many RNN model, many-to-one RNN model, and many-to-many RNN model. The detailed structure for these four models is illustrated in Figure 2.8. These four models have corresponding specified tasks. The tag prediction task for one sentence in the NLP area is the typical scenario for the many-to-many RNN model. Regarding the many-to-many RNN model,

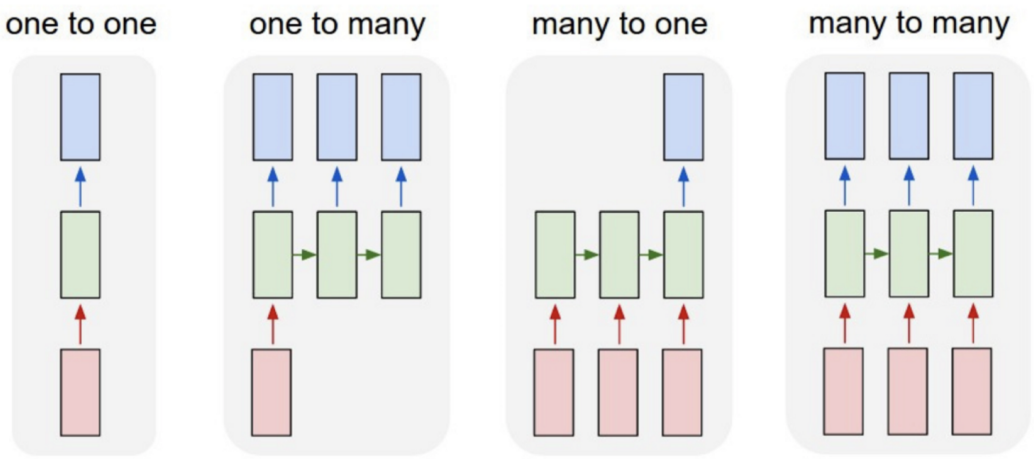

Figure 2.8. *Four modes to run RNN*

one typical application for this type of RNN model is the machine translation task. For the many-to-one RNN model, the sentiment classification task is one representative task. In the sentiment classification task, each word in the sentence is considered as one input, the final predicted result of the sentiment class for this sentence is the only one output.

When comparing to CNN models, one difference between RNN and CNN is the depth of layers. The depth of layers in the CNN model can be deep, in some case, the depth of CNN is over 100. However, the most common depth of layer in RNN model is one layer, it is rare for RNN model to use many layers. The main reason for shallow layers in RNN is that RNN model is unfolding and calculated along with the time steps while CNN model does calculate in the space instead.

RNN uses the hidden state to store information which is generated in the previous time step. When RNN model unfolds along with the time steps, the typical structure of unfolding computational graphs is shown in Figure 2.9. In this figure, the input is x, the predicted result is o, the loss function is L, the ground truth result is y, the weight matrix between input and hidden state is presented by U. The weight matrix for the connection between the hidden state and the output is V. W denotes the weight matrix between the hidden state in previous time step and the hidden state at current time step. The variables $x^{(t-1)}$, $x^{(t)}$, $x^{(t+1)}$ are the three inputs at three different time step $t-1$, t, $t+1$ respectively. The variable $x^{(t-1)}$ connects the hidden state $h^{(t-1)}$. $o^{(t-1)}$ denotes the output in the $t-1$ time step. The hidden output $h^{(t-1)}$ is calculated using Equation 2.5.

For the calculation in the next t time step, variable $x^{(t)}$ denotes the input in t time step. The equation for calculating the predicted value o is given in Equation 2.5. The total loss for the given sequence x with y as the targeted values is the sum of the losses over all the time steps.

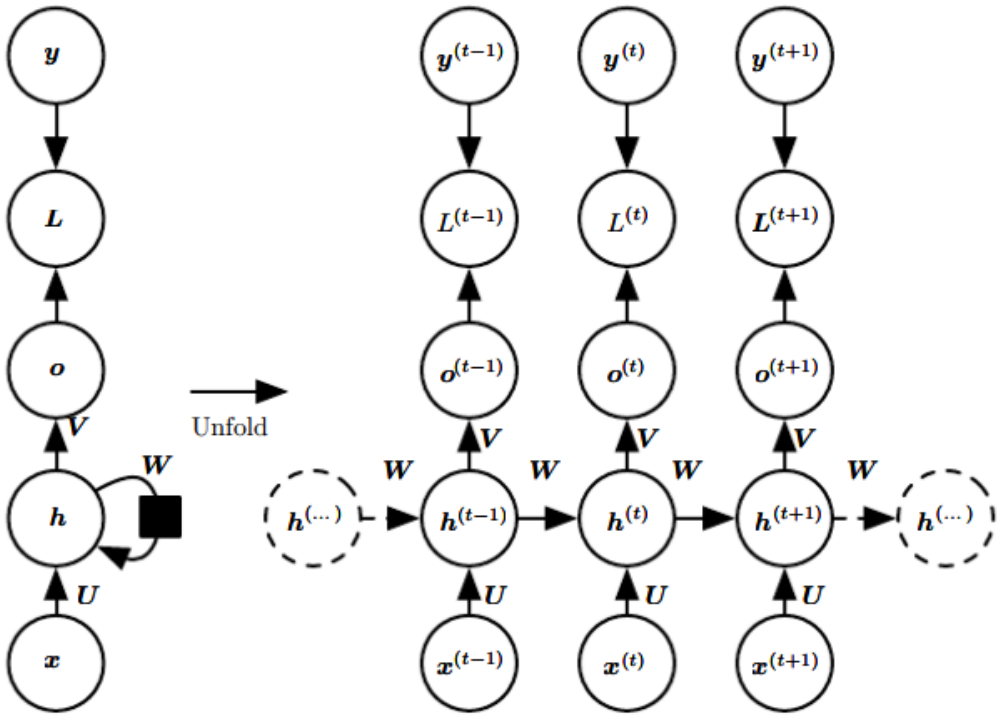

Figure 2.9. *The unfolding the recurrent neural network with time steps [4]*

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{2.5}$$

$$h^{(t)} = Tanh(a^{(t)}) \tag{2.6}$$

$$o^{(t)} = c + Vh^{(t)} \tag{2.7}$$

Long short term memory neural network

One drawback of traditional RNN is that it is hard to capture the semantic and syntactic relation between two words with long distance. Many methods were proposed to solve this problem. one method is applying the long short-term memory neural network. This variant of RNN can mitigate gradient vanishing problem by using gate control mechanism.

Long short term memory network (LSTM) [7] is an artificial recurrent neural network and was proposed by Jürgen Schmidhuber etc in 1997. LSTM has the feedback connection when comparing to other recurrent neural networks. One main feature of LSTM model is the ability to learn long-term dependency. For the text sequence data, one big challenge for current deep neural network models is how to keep semantic and syntactic relation when the length of text is growing. This problem is known as the short memory issues for text modeling. The second advantage of the LSTM model is that it can mitigate the gradient exploding problem. During the training phrase of RNN model, gradients are usually exploded when the propagation of the gradients feeds forward. LSTM mitigates the gradient exploding problem through the gate mechanism. The design of the LSTM

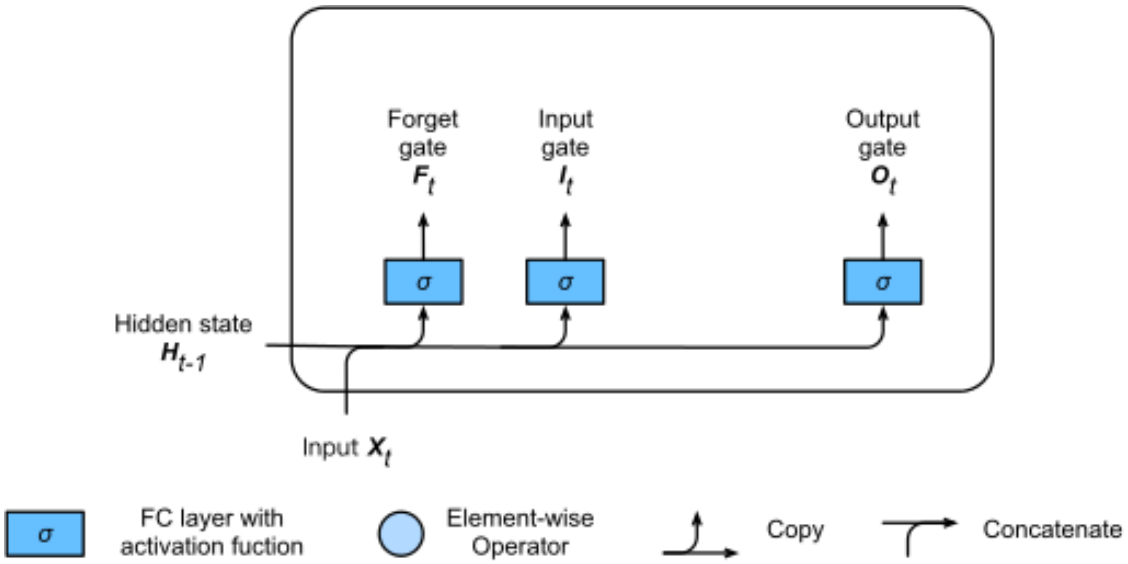

Figure 2.10. *LSTMs memory cell structure [44]*

model overcomes the technical challenges of RNN to deliver on the promise of sequence prediction with neural networks.

Regarding the structure of the LSTM cell, it is shown in Figure 2.10. Three gates are designed to control the information flow in the neural network, they are the input gate, the output gate and the forget gate respectively.

In the t time step, the variable X_t is the input of the LSTM cell. In general, X_t is represented by word vector embedding. I_t is the input gate, F_t is the forget gate, O_t is the output gate, H_{t-1} is the hidden state in the $t-1$ time step.

Moreover, the variable W_{hi} is used to represent the weight matrix between the hidden state and the input gate, the variable W_{hf} is the weight matrix between the hidden states and the forget gate, the variable W_{ho} represents the weight matrix between the hidden state and the output gate. The output values of I_t, F_t and O_t in a LSTM cell are calculated in the fully connected layer with sigmoid function as the activation function. The value range of the output belongs to $[0, 1]$. The three gate values and the hidden states variable use Equation 2.8 to update their values in each time step.

$$I_t = sigmoid(X_t W_{xi} + H_{t-1} W_{hi} + b_i), \tag{2.8}$$

$$F_t = sigmoid(X_t W_{xf} + H_{t-1} W_{hf} + b_f), \tag{2.9}$$

$$O_t = sigmoid(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{2.10}$$

Bidirectional recurrent neural network

In feed-forward recurrent neural network models, it is assumed that the semantic and syntactic information of the current word is determined by the previous appearing words.

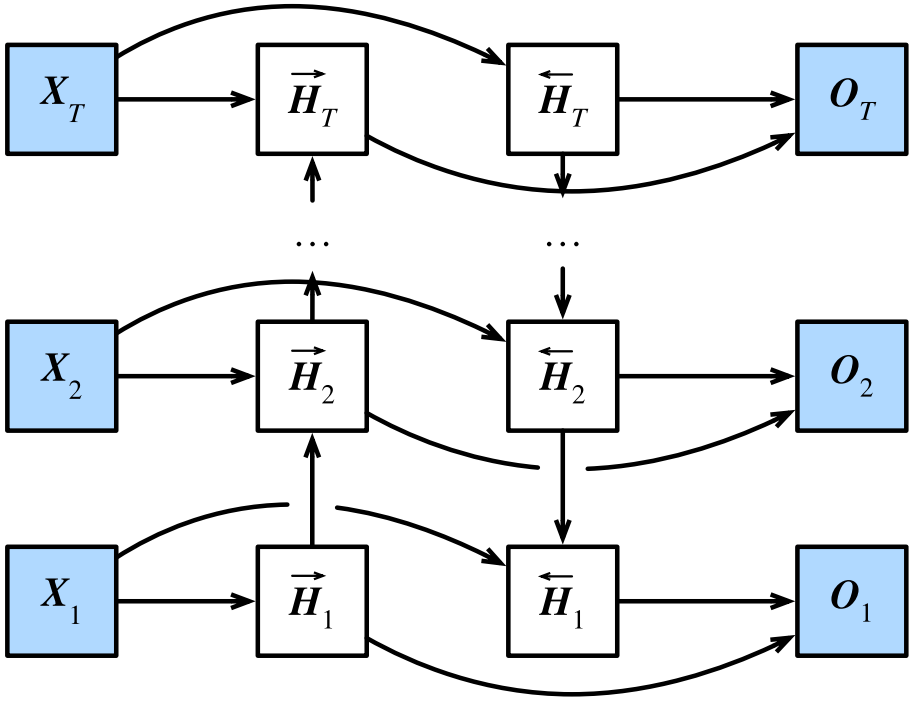

Figure 2.11. *The structure of the bidirectional recurrent neural network [44]*

When the computational graph of the recurrent neural network is unfolding along with the time steps, the value of variables in the current time step is determined by the variables in the earlier time steps. However, the meaning of a word is determined by both the earlier words and the later words together in many situations. The variable in the current time step should consider the values from the previous and later time steps. Context information has significant improvement for the text classification task, especially in the long text case. For the semantic information, the one-way recurrent neural network only considers the information from the previous appearing texts, not using the information from the latter text. This results in the problem of losing semantic information.

The bidirectional recurrent neural network [33] was proposed to tackle this type of problem. Figure 2.11 demonstrates the structure of a bidirectional recurrent neural network. In the bidirectional recurrent neural network, it is assumed that the forward hidden state for this time step is $\overrightarrow{H}_t \in \mathbb{R}^{n \times h}$, the backward hidden state is $\overleftarrow{H}_t \in \mathbb{R}^{n \times h}$, n is the number of samples which are inputted to the model, h is the number of forward hidden units. The forward and backward hidden states are computed using Equation 2.11.

$$\overrightarrow{H}_t = \phi(X_t W_{xh}^{(f)} + \overrightarrow{H}_{t-1} W_{hh}^{(f)} + bh^{(f)}) \tag{2.11}$$

$$\overleftarrow{H}_t = \phi(X_t W_{xh}^{(b)} + \overleftarrow{H}_{t+1} W_{hh}^{(b)} + b_h^{(b)}) \tag{2.12}$$

The forward and backward hidden states $\overrightarrow{H}_t$, $\overleftarrow{H}_t$ are concatenated to obtain the hidden state $H_t \in \mathbb{R}^{n \times 2h}$. The hidden state $H_t \in \mathbb{R}^{n \times 2h}$ is inputted to the output layer. The output layer computes the output $O_t \in \mathbb{R}^{n \times q}$ (q is the number of outputs). Equation 2.13 is used

to calculate the outputs.

$$O_t = H_t W_{hq} + b_q \tag{2.13}$$

2.2.4 Optimization algorithm

When deep neural networks are trained, the aim of training a DNN model is to reduce the value of loss function. The optimization algorithms are used to update the value of model parameters to reduce the value of the model objective function. When the training phase ends, the parameters of the model at the time are the parameters that the model learned from the training phase.

Objective function

Objective function is also named loss function in deep neural networks. The objective function is the average value of the losses when the training data set is loaded to train a DNN model. The common objective function of a neural network is defined in Equation 2.14. In this equation, $f(\boldsymbol{x}_i)$ is the loss function for the x_i sample in the training data set, n is the number of training samples, $f(\boldsymbol{x})$ is the objective function for all n samples.

When the feed-forward process for training deep neural networks is completed, the average value of the losses is calculated. Then the back-propagation algorithm is applied to calculate the gradient value of x in each layer of DNN models with using Equation 2.15.

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) \tag{2.14}$$

$$\nabla f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}). \tag{2.15}$$

Stochastic gradient descent

During training processing, the optimization algorithms are critical for training deep neural networks. For running deep neural networks experiments, it usually takes a few days or longer time to complete the training process. Therefore, the optimization algorithm has an obvious effect on the cost time for the training process. There are many optimization algorithms such as stochastic gradient descent (SGD) [1], Adam [15]. In this thesis, SGD is used as the optimization algorithm.

One demonstration of applying SGD is shown in Figure 2.12. The input is a two-dimensional vector, $x = [x_1, x_2]$, the objective function is $fx) = x_1^2 + 2x_2^2$. SGD optimization algorithm

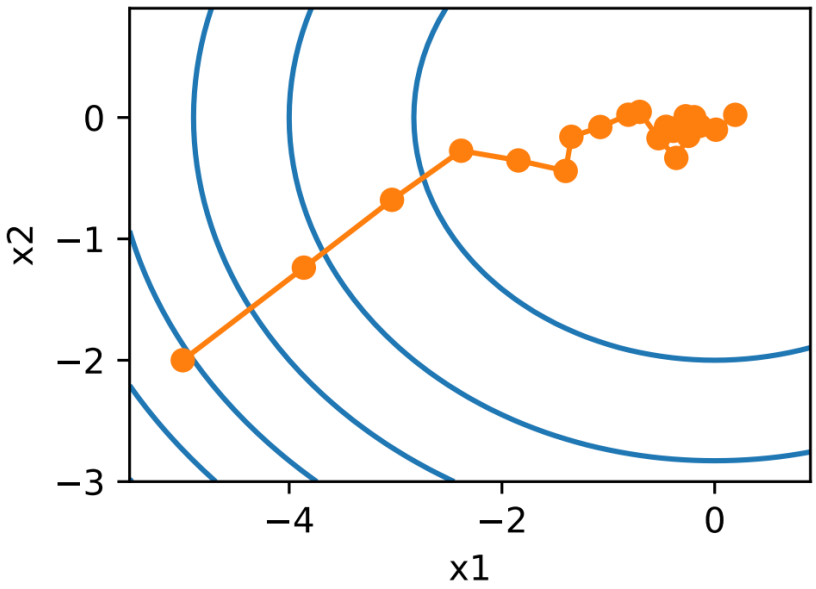

Figure 2.12. *Stochastic gradient descent optimization algorithm for the function* $f(x) = x_1^2 + 2x_2^2$ *[44]*

is used to find the minimal value. The initial position for x is $(-5, -2)$, the iteration for updating x is 20 times. After the 20 times iteration, the function $f(x)$ finds the minimum point $(0, 0)$.

2.2.5 Back-propagation

Back-propagation [31] is used to compute the gradient in different layers in deep neural networks from the cost function. Through back-propagation, the information from the loss function is flowed backwards in the deep neural network. It uses the derivative chain rule to compute derivatives. When data is inputted to a model, the deep neural network model will first perform feed forward propagation to update the value in different layers in the model. Then the loss value will be calculated. Back-propagation makes information from loss function to flow backward to the model.

Back-propagation through time

When back-propagation is applied in recurrent neural network, back-propagation through time (BPTT) [42] algorithm is the variant of back-propagation to compute the gradient. The normal back-propagation cannot be applied directly since RNN is time sequence model. Therefore, back-propagation through time was introduced for the sequence data. The recurrent neural network is required to expand along with the time steps in BPTT to obtain the dependencies between model variables and parameters in different time steps.

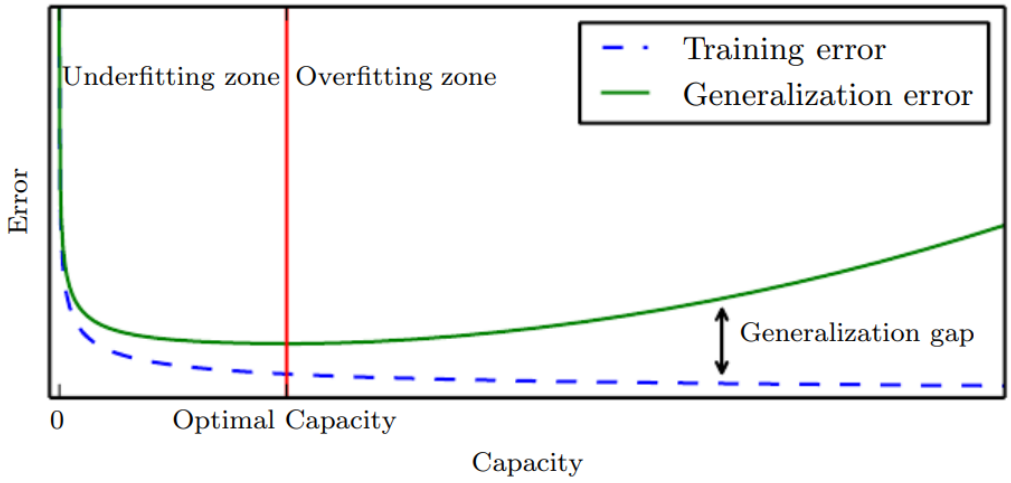

Figure 2.13. *Overfitting in deep neural network [4]*

2.2.6 Regularization technique

Deep neural networks have strong representation learning ability. However, the lack of control over the learning process in deep neural networks can lead to the overfitting problem. The overfitting indicates that models have low generalization ability. This results in the bad prediction ability for test data even though the model achieves high performance in training data or validation data.

Overfitting problem

Overfitting occurs when the gap between the training error and testing error is large. The illustration of the overfitting problem is shown in Figure 2.13. This situation indicates that the deep neural network model is trained to have a good fitting ability on the train data instead of learning the pattern of data.

There are many regularization techniques to improve the generalization ability for deep neural networks. The common regularization methods include dropout, batch normalization, parameter norm penalties, early stopping mechanism, multitask learning technique etc.

Dropout

Dropout is one regularization technique to deal with the overfitting problem for DNN models. Dropout [35] technique can be considered as one way of approximately combining exponentially many different neural network architectures efficiently. The term “dropout” refers to dropping out neurons in a hidden and normal layer in a neural network. Dropping neural neurons out indicates temporarily removing the neurons from the neural network along with all their incoming and outgoing connections. The concept of dropout is shown in Figure 2.14. When the dropout technique is applied to the hidden layers in the multilayer perceptron, there is a probability for neurons in the hidden layer to be skipped.

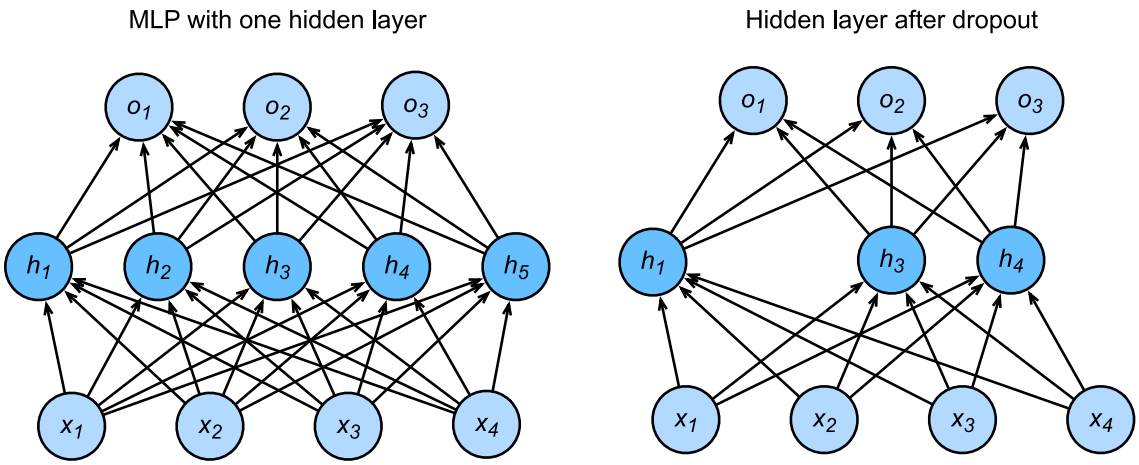

Figure 2.14. *Comparison of the architecture of multi-layer perceptron with and without dropout technique [44]*

In experiments, the rate of the dropout is usually set as 0.5. In [35], the authors experimented that convolutional neural network with dropout technique in fully connected layers had 14.32 % error rate on the CIFAR-10 dataset [17]. However, the original convolutional neural network without applying dropout contained 14.98 % error rate on the CIFAR-10 dataset. In the CIFAR-100 dataset experiments, the dropout technique reduced the error rate from 43.48 % to 37.20 % when compared to the mode without using dropout. The authors also tested the dropout technique in the ImageNet dataset. The best method based on standard vision features achieved a top-5 error rate of 26 % at the time. However, the convolutional neural network with applying dropout achieved a test error of about 16 %. These experiments indicated that the dropout technique can improve the generalization ability of deep neural networks.

Batch normalization

Batch normalization [10] is another wide applied technique for improving the generalization ability of DNN. Batch normalization is usually inserted after fully connected layers or convolutional layers and before the activation functions. The normalized value is inputted to the activation functions. Through the batch normalization operation, the weight values of different layers in a neural network model are normalized with following 4 steps. The first step of applying batch normalization is to calculate the mini-batch mean with using Equation 2.16. Secondly, the mini-batch variance is computed using Equation 2.17. The third step is to apply the normalization for the mini-batch with using Equation 2.18. The fourth step is using the Equation 2.19 to scale and shift the variables.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.16}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2.17}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2.18}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BatchNormalization}_{\gamma,\beta}(x_i) \tag{2.19}$$

In [10], the authors experimented that the effect of batch normalization in the Inception deep neural network [37]. Their experiments showed that the Inception variant model with batch normalization technique outperformed the basic version of Inception model with 0.5% higher accuracy in the ImageNet dataset. The Inception variant with batch normalization achieved 72.7% accuracy while the basic Inception model had 72.2% accuracy. The experiments result indicated that the performance of the Inception V3 model [38] was improved by 0.5% accuracy through the batch normalization technique. What's more, the cost time for training the DNN model was reduced when the batch normalization was applied according to their experiments. The reason for consuming less time for training DNN with batch normalization technique was that batch normalization can boost the speed of converging for training deep neural networks.

2.3 Word embedding

Word embedding is one type of algorithms which maps words or phrases from vocabulary to vectors of continuous representations in numerical values form. In word embedding, each word is represented by a vector. The similarity of different words can be measured by the distance metrics such as calculating cosine similarity. When compared to traditional word embedding technique such as one-hot encoding word vectors, the distributed representation of words considers the semantic context information. What's more, the curse of high dimensionality and vector sparsity problem can be mitigated by utilizing the word embedding technique.

There are many famous algorithms to generate word embedding vectors for words in corpus. Three different type of algorithms are introduced in this thesis. The first one is the word2vec, the second type is the FastText and the third word embedding algorithm is the GloVe.

Word2vec

The word2vec [26] word embedding algorithm was firstly introduced in 2013. The skip-

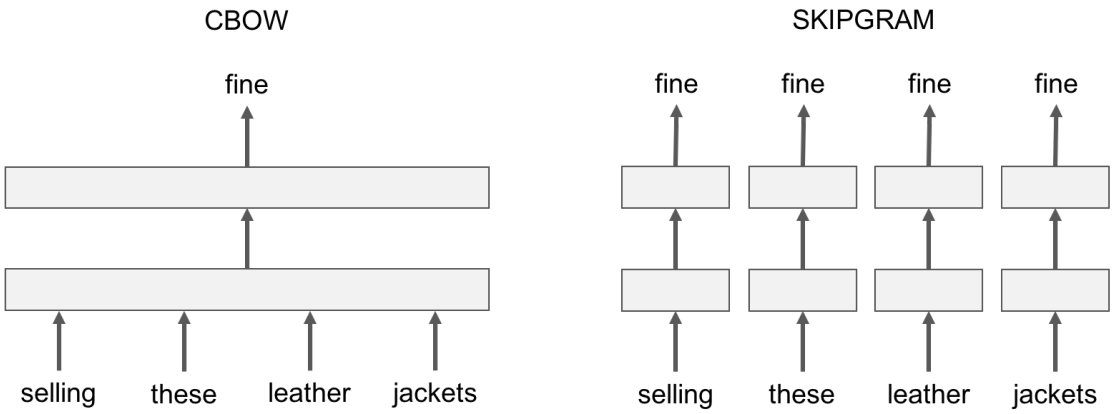

Figure 2.15. *The continuous-bag-of-words versus the skipgram method in FastText model [8]*

gram and the continuous bag of words (CBOW) are two variant models for computing word vector representations.

The skipgram model learns to predict a target word with using the nearby appearing word. However, the CBOW model predicts the target word according to its surrounding context. The context is represented as a bag of the words contained in a fixed size window around the target word. An example for illustrating the difference between the skipgram and the CBOW model is shown in Figure 2.15. The sentence "I am selling these fine leather jackets" is given and the target word is "fine". The CBOW model uses all the surrounding words inside the window size, including "selling", "these", "leather", "jackets". The sum of these word vectors is used to predict the target word. The skipgram model takes one current word to predict the target using a random near word, like "these", "leather". According to the result of the experiments, the skipgram model had better performance than the CBOW model in general.

One demonstration of wording embedding is illustrated in Figure 2.16. In this figure, two categories of English words are listed. The first category of words is the countries, including "France", "Germany", "Spain" etc. The second type of words is the corresponding capital cities, including "Paris", "Berlin", "Madrid" etc. The word embedding for these English words was trained using the skipgram model. After finishing the training process, the 300 dimensional word vectors for the English words were generated. The principal component analysis (PCA) [11] technique was applied to reduce the dimension size to 2. After that, these two values were used as the coordinates for words to visualize on the figure. It is seen that the English words of these two different concepts were clustered clearly on each side. The relationship between these two categories words was learned. This indicates that the syntactic and semantic relationship of these words was kept in the word embedding presentation form.

GloVe

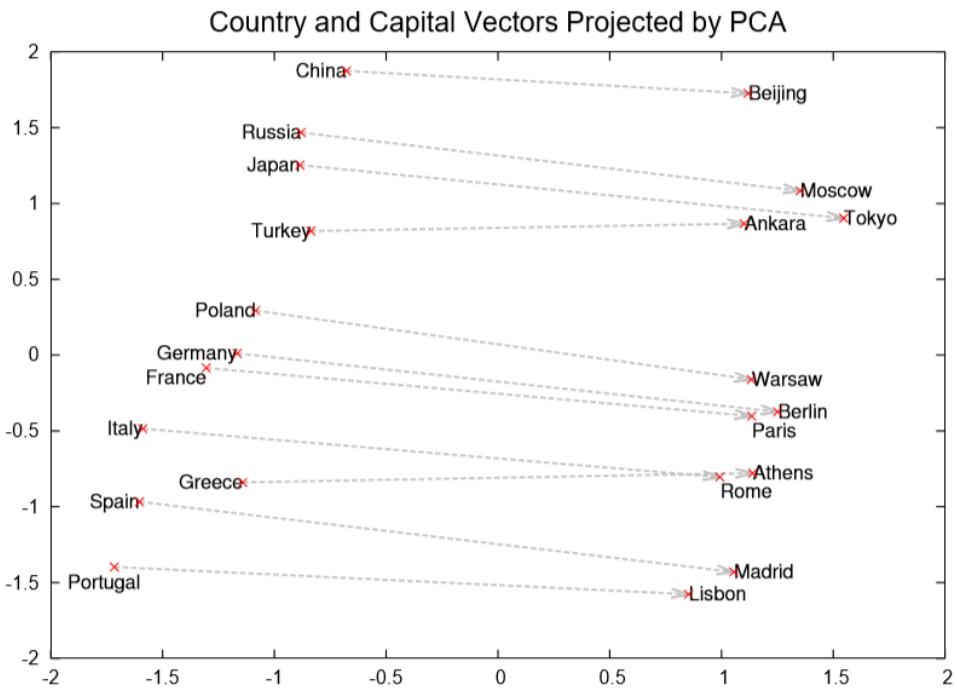

Figure 2.16. *Two-dimensional PCA projection of the vectors of countries and their capital cities [26]*

GloVe [28] is one popular deep neural network based word embedding algorithm. It is a unsupervised learning word embedding algorithm for learning word vectors. GloVe was developed after the word2vec and has a few changes compared to the skip-gram model. The first change is that the GloVe adopted square loss instead of cross-entropy loss as the objective function to train the word embedding model. Secondly, the GloVe considers the global statistics information of the English words based on the whole dataset.

FastText

The FastText [12] is another popular neural network to generate accurate word vectors. One feature of the FastText model is that the training speed is fast. In the author's experiments, FastText model needed 4 seconds to complete one epoch to perform the sentiment classification task on the Yelp Full review dataset. However, other models needed more than 30 minutes to finish one epoch. The architecture of the FastText model is shown in Figure 2.17. It shows that the model structure of the FastText was simple. The architecture of FastText contains one layer for the word embedding, one hidden layer and one layer for the result prediction.

GloVe and FastText are two popular word embedding technique. These two algorithms provide the pretrained word embedding to download. The detail for these two algorithms is described in Table 2.2. The training corpus for GloVe are wikipedia and gigaword, and FastText used wikipedia as the training corpus. In the aspect of tokens size, GloVe has 6 billion tokens and FastText has 16 billion tokens.

One hyper-parameter for word vectors is the dimensional size. Generally, three word

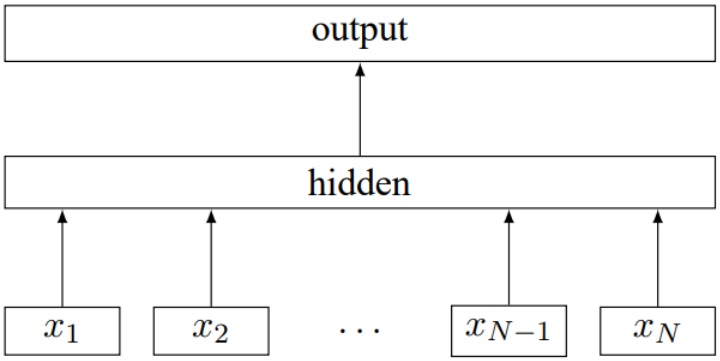

Figure 2.17. *FastText model architecture for generating the vector representation for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable [12]*

Table 2.2. *GloVe and FastText word embedding model comparison*

model	dimensions	vocabulary	corpus	tokens
GloVe	300	400k	Wikipedia, Gigaword	6 Billion
FastText	300	2520k	Wikipedia	16 Billion

embedding dimension sizes are common, there are 50, 100 and 300 dimensional sizes. One factor of the syntactic and semantic information of one word is the dimensional size. In [28], the experiment showed that the accuracy on the analogy task was higher when the dimensional size of one word is larger. In this thesis, 300 dimensional size for the word embedding was experimented for the sentiment classification task.

2.4 Sentiment classification algorithm overview

Zhang et al [45] proposed a survey on the evolution of sentiment analysis research area, including the introduction of fundamental algorithms from the perspective of linguistics, statistics and computer science. The authors mentioned the challenge in this research area and analyzed why sentiment classification is a difficult task. In [24], authors introduced probability-based classification techniques to classify reviews which were collected from mobile phone applications on Google play store and Apple app store. These reviews were classified into four different categories: bug reports, feature requests, user experiences, and text rating.

Many efforts have been devoted to exploring the sentiment classification area from different perspectives, including introducing new text review dataset, proposing new text classification algorithms etc.

One research perspective is considering the length and form of the text review. In this perspective, sentiment classification algorithms are divided into three small research direction: the sentence-level sentiment classification algorithm, the paragraph-level sentiment classification algorithm, and document-level sentiment classification algorithm. In

the sentence-level sentiment classification task, one whole sentence is treated as input, the target is to predict the sentimental polarity of the sentence. For the paragraph-level sentiment classification, the vector representation of one paragraph is generated through integrating the sentence vectors, then the sentimental class for the paragraph is predicted by algorithms. In the task of the document-level sentiment classification, its aim is to classify the whole document text as expressing a positive, neutral or negative opinion. A paragraph is first to do the feature extraction to obtain the vector representation. After that, the paragraph vector is generated by integrating all paragraph vectors.

From the perspective of algorithms, the evolution of text classification algorithm can be generally divided into three main types, linguistic rule-based method, statistical machine learning method, and deep neural network-based approach.

Linguistic rule-based algorithms

The first type is the linguistic rule-based algorithm. It is the earliest applied method to perform sentiment classification task. The linguistic rule-based method uses lexicon dictionary to classify the sentiment of text. In this method, the sentiment word dictionary is first built. The frequencies of different words are calculated. The different weights for words in the sentence are dispatched. Then the final sentimental score for the text is computed. One representative algorithm of this method is the VADER [9].

Statistical machine learning based algorithms

The second type of algorithm is based on statistical machine learning knowledge. The general process of applying statistical machine learning method contains two steps, the first step is to extract features from the text. The handcrafted features require lexical and semantic information such as bag-of-words, n-grams, and part-of-speech tags. The second step is to use statistical machine learning classifiers to predict the sentiment class for the specific text. There are many representative works. The common algorithms include linear regression (LR), naive bayes (NB) [30], support vector machine, hidden markov models [29], conditional random fields [36] etc.

One significant work is the support vector machine. In [27], the authors applied the SVM algorithm to classify text reviews. The SVM algorithm achieved the state-of-art performance at the time. In the authors' experiments, the bi-grams features were used to extract text. The SVM algorithm contained 82.9% test accuracy.

Deep neural network based algorithms

The third method uses deep neural networks to classify the sentiment of text, which has achieved state-of-the-art results in the sentiment classification task nowadays. The DNN-based methods have improved the performance of sentiment classification algorithm greatly when compared with the other two type algorithms. Moreover, DNN-based methods also simplify the process of the sentiment classification task, DNN-based method integrates feature extraction and classification steps into one end-to-end processing.

There are many famous deep neural networks on the text classification task. For example,

Kim et al proposed the Text-CNN model [14]. This model applied convolutional neural network on the text. In 2013, Socher et al [34] proposed a recursive deep neural model for the text classification task. The recursive deep neural network obtained the state of the art accuracy on the many text review datasets at the time. The model achieved 80.7% accuracy on the fine-grained movie sentiment classification. In 2015, Zhang [46] proposed the character-level convolutional network to perform the text classification task. The character-level convolutional network was the first model to use character instead of word to classify text. Previously, deep neural network based models use word-level text input. This paper showed that the recurrent neural network had a strong learning representation ability for both characters and words.

2.5 Assessment criteria

To evaluate the performance of algorithms on the text classification task, four critical metrics are calculated in this thesis, namely accuracy, precision, recall and F1-score. These metrics can measure how well the algorithm performs. What's more, to obtain better understanding of the prediction result produced by algorithms, the confusion matrix is needed as well. The short description of these metrics is introduced below.

2.5.1 Accuracy and F1-score

One common evaluation method to compare multi-class classification algorithms is to use precision and recall metrics.

Accuracy

The mathematical function for the accuracy metrics is given in Equation 2.20.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.20}$$

- True positive (TP), the classifier predicts 1 where the actual class is 1.
- False positive (FP), the classifier predicts 1 where the actual class is 0.
- True negative (TN), the classifier predicts 0 where the actual class is 0.
- False negative (FN), the classifier predicts 0 where the actual class is 1.

The mathematical function of precision is given in Equation 2.21. The definition of recall is given in Equation 2.22.

$$precision = \frac{TP}{TP + FP} \tag{2.21}$$

Table 2.3. *The confusion matrix*

		predicted Class	
		Positive	Negative
actual class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

$$recall = \frac{TP}{TP + TN} \tag{2.22}$$

Precision is also named positive predictive value. Precision is the fraction of relevant instances among the retrieved instances. Recall is also named sensitivity and is the fraction of relevant instances that have been retrieved over the total amount of relevant instance. High precision means that the classifier predicts almost no inputs as positive unless they are positive. A high recall is explained as the classifier misses almost no positive values.

F1-score

The F1-score (also known as F-score or F-measure) is the harmonic mean of precision and recall. The equation of the F1-score is formulated in Equation 2.23.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{2.23}$$

The F-score is described as a weighted average of the precision and recall. The best value of the F-score is 1 and the worse value is 0. In the multi-class classification task, the average of F1 score for each category with weighting depends on the average parameter. In the experiments, each data sample is treated as the same weight when the F1-score is calculated.

2.5.2 Confusion matrix

The confusion matrix is used as one method to visualize the classification result. For the binary classification case, true and false are the only two possible outcomes. The table 2.3 shows the confusion matrix for the binary classification. The prediction result of the algorithm will calculate four outcomes (TP, FN, FP, TN) for each class.

3 EXPERIMENTS

In this chapter, the TripAdvisor dataset is first introduced. The main processes for making this dataset include collecting the raw data, pre-processing the hotel reviews. The statistics overview of the TripAdvisor dataset and the Stanford sentiment treebank dataset (SST) is presented as well. In the second section, the three deep neural network models for the text sentiment classification task is elaborated. The content for the third section is to describe details for the experiments. Then the performance of three deep neural network models is evaluated on the TripAdvisor dataset and SST dataset. The comparison of these three algorithms on different evaluation metrics is described in details in the last section.

3.1 Datasets

3.1.1 TripAdvisor dataset

Data collection

In this thesis, the own-made dataset, the TripAdvisor dataset, is made for this thesis. For collecting the hotel reviews data, a Python web crawling framework, Scrapy [16], was used to crawl hotel review texts data from the TripAdvisor website. There are two main reasons for choosing the TripAdvisor website as the raw data source. The first reason is that this website is popular and has a large number of user reviews, which indicates that the data is big enough and the hotel reviews are representative. The second reason is this website does not have a strong anti-crawling mechanism. Therefore, it has a high probability to crawl the hotel review data without encountering big obstacles. The second step for making the TripAdvisor dataset is to clean the raw dataset, including discarding incomplete hotel reviews. For example, some hotel reviews miss the date of the review in the raw data format. The third step is to select the review column and the sentiment class column. 100000 hotel reviews were picked as the whole dataset in the experiment. Then 70% of the whole dataset was split into the training set, 10% of the dataset was treated as the validation set, and the remaining 20% is the test set.

One original hotel review in the TripAdvisor website is shown in Figure 3.1. It demonstrates that this hotel review was given 5 scores by a user named "000Glenn" in November 2012. This hotel review record has three separated paragraphs. The title of this

Figure 3.1. *Raw hotel review data on the TripAdvisor website [39]*

review is "Outstanding". The hotel is "Sofitel Legend The Grand Amsterdam". The text format of this hotel review was crawled from the website and saved into the JavaScript Object Notation (JSON) format. The stored JSON data is listed in Table 3.4.

Table 3.1. *Raw data sample of the TripAdvisor hotel review*

JSON data	**value**
title	Outstanding
hotelStars	5.0
userId	000Glenn
hotelLocation	Oudezijds Voorburgwal 197, 1012 EX Amsterdam, The Netherlands
hotelName	Sofitel Legend The Grand Amsterdam
score	5
date	2012.11.10
url	`https://www.TripAdvisor.com/ShowUserReviews-g188590-d189389-r145091651-Sofitel_Legend_The_Grand_Amsterdam-Amsterdam_North_Holland_Province.html`
hotelUrl	`https://www.TripAdvisor.com/Hotel_Review-g188590-d189389-Reviews-Sofitel_Legend_The_Grand_Amsterdam-Amsterdam_North_Holland_Province.html`
review	I rate this hotel as the best I've stayed in. It occupies a beautiful, historic building sandwiched between two canals in the heart of old 'Dam.

From the Table 3.1, it shows that one record of the raw hotel review includes the title of the review, the quality level of the hotel, the id of the user, the hotel location, the score rating

Table 3.2. *The number of stars matching with score or class*

score	sentiment	corresponding class
1 star	very negative	0
2 star	negative	1
3 star	neutral	2
4 star	positive	3
5 star	very positive	4

Table 3.3. *The number of stars matching with score or class*

statistics	very negative	negative	neutral	positive	very positive
number of reviews	5313	6012	15462	36902	46311
number of sentences	8.88	8.65	7.67	6.88	6.63
number of words	176.85	171.11	147.29	124.77	114.02

of the review, review date, the URL of this review and the review content. The columns of the review and the score were extracted as the input and its corresponding sentimental class. To make the text classification dataset, only text reviews and their corresponding classes are needed to keep, other information is not necessary to keep. Therefore, in the experiment, two column values (the "review" column and the "score" column) were selected from the original JSON data.

The TripAdvisor dataset overview

In order to classify the sentiment of hotel reviews, the label for each hotel review was set with the reviewers' score rating. In this experiment, the matching relation between the scoring and the sentimental class is described in Table 3.2.

For calculating the number of sentences from the whole text review, The NLTK [23] library was used as the library. Figure 3.2 shows the distribution of the number of reviews for each class in the TripAdvisor dataset. The *very positive* sentimental class has the largest number of hotel reviews data among all other four sentiment classes, which is more than 40000 data samples. The *positive* class has the second largest number of hotel reviews in the TripAdvisor dataset and it has over 30000 data samples. However, for the *very negative* class, it contains less than 10000 data. This indicates that the data distribution of different sentimental classes hotel review is unbalanced. In this thesis, the problem of unbalanced training data is not considered.

The average number of sentences in one hotel review for one sentimental class in the TripAdvisor dataset is shown in Figure 3.3. It shows that the *very negative* class has the largest number of the sentences in each hotel reviews, which contains 8.65 sentences on average. However, the *very positive* sentimental class has the least number of the sentences, which has 6.63 sentences while the number of the words in the *very negative* class is 176.85 on average.

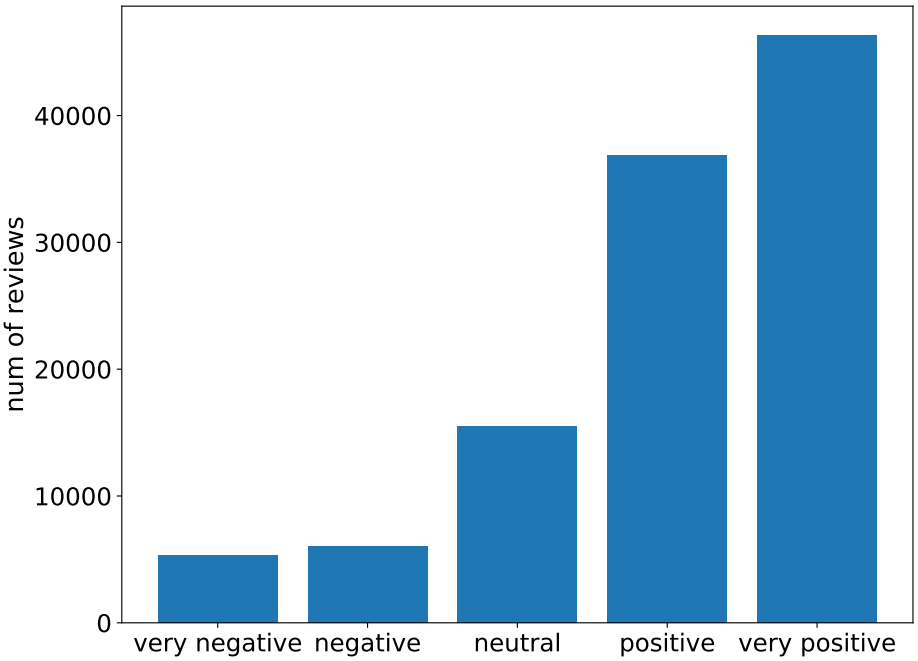

Figure 3.2. *Distribution of the number of hotel reviews for each class in the TripAdvisor dataset*

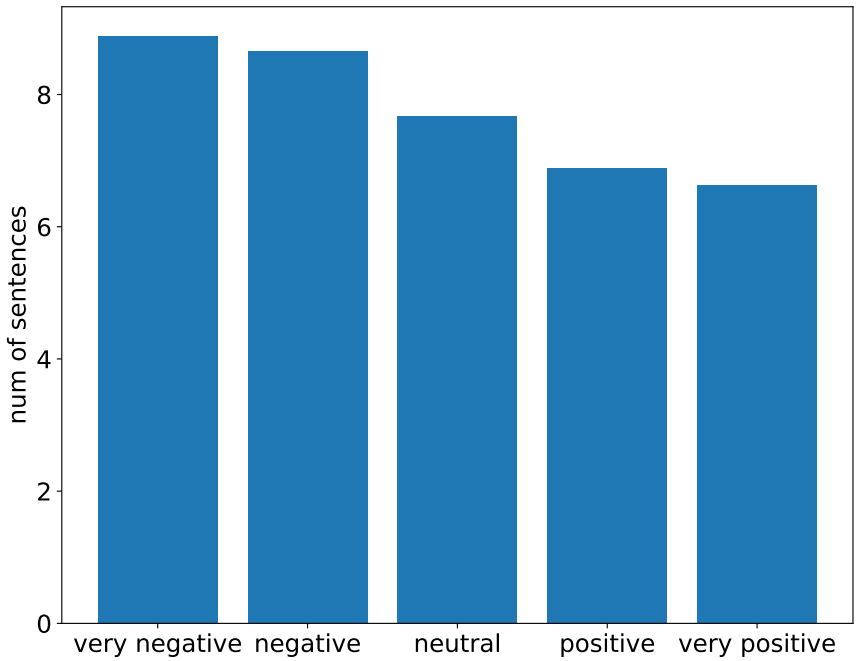

Figure 3.3. *Average number of sentences in one hotel review per sentimental class in the TripAdvisor dataset*

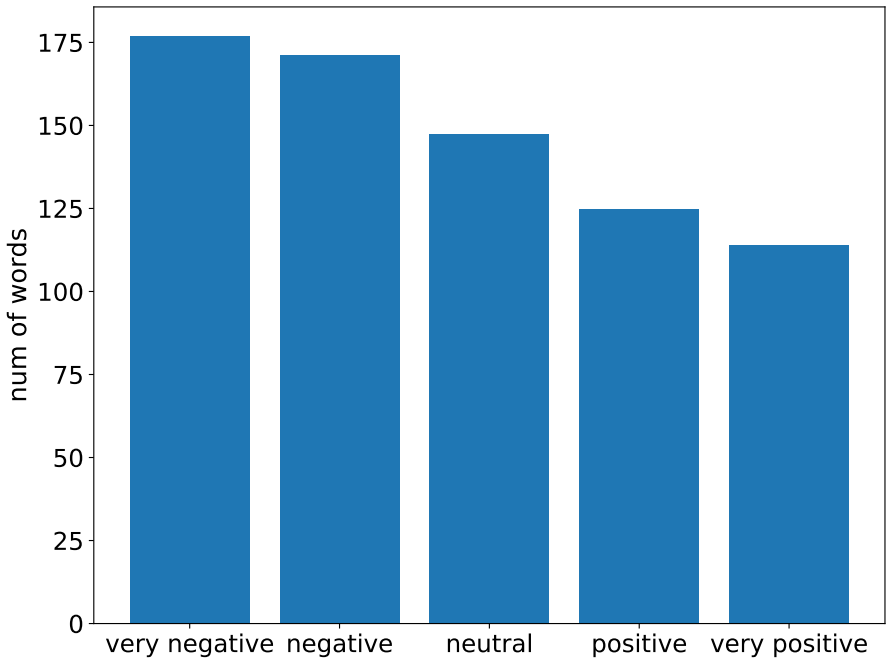

Figure 3.4. *Average number of words per class in the TripAdvisor dataset*

Table 3.4. *Overview of the TripAdvisor dataset, $|C|$ denotes the number of classes, $|V|$ denotes the vocabulary size*

data	**value**
dataset	TripAdvisor
$\|C\|$	5
$\|V\|$	99686
mean number of sentence	7.08
mean number of words	128.46
training set	70000
validation set	10000
testing set	20000

Table 3.5. *Data sample from the TripAdvisor hotel review dataset*

hotel reviews	sentiment class
"just fantastic the staff are amazing every single one of them. The rooms are spacious and very clean. I cant wait to go back!!If you go here book your room on the 4th floor for the added extras. Oh and my 8 year old got free breakfasts on both days bonus.Thankyou for a great stay."	5
"Have stayed at this hotel numerous times, modern hotel, good amenities and rooms are to a good standard. A special mention must go to the staff in the restaurant and bar area both for breakfast and evenings who are friendly and make you feel very welcome. Hotel is positioned 2 minutes form tube station which will take you to Oxford Circus in central London with no changes. Overall I would recommend this hotel both for business or leisure."	4
"The Hotel seems a good distance out form the city centre but the Central Line takes you into Oxford Street in 20 mins and the Hotel is literally 1 minute from Hanger Lane Tube station. The hotel itself is fine. One disappointment was the breakfast in the Club Lounge. Really seemed like minimum effort and probably the poorest I've ever had at a Crowne Plaza.Would stay there again because the location is very convenient"	3
"Just poor. I've not posted on TA for many years as I forgot my log in details, however this place is so crap I decided to hunt out my details to warn others. The building and facilities are as you'd expect but the staff are just not interested. The check in experience left me wanting to go and play with traffic. They are incompetent and arrogant to the end. Sadly I stay here from time to time with work and will be posting about every bad experience I have, I'm sure there will be more."	2
"So after a very long day I arrived to check in for 4 nights and was informed that although I had a requested a Double room there was only twins available. Great! First time I have slept in a single bed in 20 years. So I head to the room to find stained bedding, dirty curtains, dirty cups and hey that's life they will have someone look at it the next day.Next day 1 item resolved. And the cups I cleaned and used had been left dirty and the coffee not even replenished (or the chocolates). Glad I only have another 2 nights to endure."	1

3.1.2 Stanford sentiment treebank dataset

The Stanford sentiment treebank (SST) dataset is one popular and common dataset in the text classification task. The SST dataset contains 215154 data samples in the training set and 11855 data samples for testing. All reviews in the SST are related to the movie content. There are two classification form of the sentimental labels in the SST dataset. One is the five-way fine-grained classification and the other one is the binary classification. For the fine-grained classification type, the five classes are: very positive, positive, neutral, negative, very negative. For the binary classification version, it only contains positive and negative sentimental classes. In this experiment, the five-way fine-grained classification data type was selected. The overall statistics of the SST dataset is listed in Table 3.7. From the table, the number of data samples in the training set is 8544, the validation set has 1101 data records and the testing set contains 2210 data records. Five review samples are demonstrated in Table 3.6. There are three columns in the table. The first column is the movie review, the second column is the given sentimental score and the third column is the corresponding class label for the given sentimental score.

Table 3.6. *Stanford Sentiment Treebank*

SST dataset review example	sentimental score	class
"What really surprises about Wisegirls is its low-key quality and genuine tenderness ."	0.931	5
"Not for everyone , but for those with whom it will connect , it 's a nice departure from standard moviegoing fare ."	0.778	4
"The gorgeously elaborate continuation of " The Lord of the Rings " trilogy is so huge that a column of words can not adequately describe co-writerdirector Peter Jackson 's expanded vision of J.R.R. Tolkien 's Middle-earth ."	0.500	3
"Emerges as something rare , an issue movie that 's so honest and keenly observed that it does n't feel like one ."	0.375	2
"Though it is by no means his best work , Laissez-Passer is a distinguished and distinctive effort by a bona-fide master , a fascinating film replete with rewards to be had by all willing to make the effort to reap them ."	0.181	1

Table 3.7. *Statistics of the SST dataset*

data	value
dataset	Stanford sentiment treebank
number of classes	5
vocabulary size	19500
training set	8544
validation set	1101
testing set	2210

3.2 Deep neural network models

For the text sentiment classification task, there are many well-designed deep neural network models [19] [5] [40]. Three different deep neural networks are introduced for the sentiment classification task in this thesis. The first model is the Text Convolutional Neural Network (Text-CNN), the second model is the Very Deep Convolutional Neural Network (VD-CNN) and the third model is the Bidirectional Long Short Term Memory model (BiLSTM).

There are three main steps for applying deep neural networks on sentiment classification task - word representation, feature extraction and classification. In the step of word representation, the pre-trained word embedding mechanism is adopted. The FastText and GloVe pre-trained word embedding techniques are tested. Word embedding technique mitigates the problem of the curse of dimensionality in deep neural networks. In the step of feature extraction, deep neural networks extract feature from the word embedding through the neural network, the vector representation for the text review is generated. In the step of classification, fully connected layers are used to classify the vector.

One example of applying deep neural networks on sentiment classification task is illus-

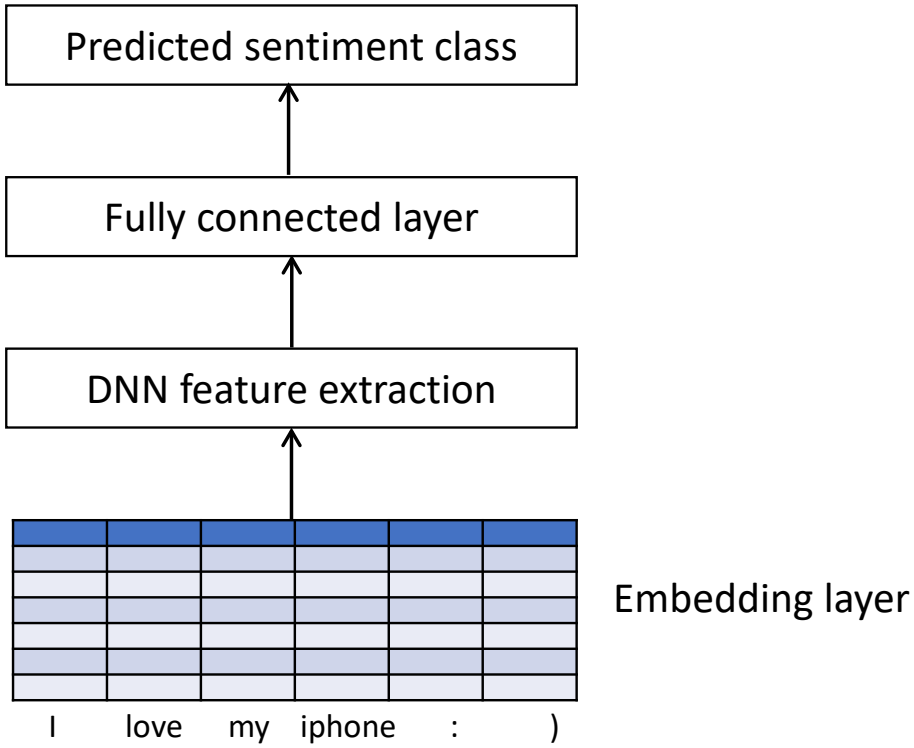

Figure 3.5. *Sentiment classification example of applying DNN model*

trated in Figure 3.5. From the figure, the review, “I love my new iPhone”, was first passed into the embedding layer to convert words into the corresponding word embedding vectors. In the second step, the deep neural network was used to extract the feature from the word vectors. In the final step, the fully connected layer was added and the probabilities on different sentimental classes were calculated, the sentimental class with the highest probability is the predicted class.

3.2.1 Text convolutional neural network

Text convolutional neural network was first proposed in [14]. The authors demonstrated how to apply convolutional neural network on the sentiment classification task. The key contribution of this model is that the Text-CNN model was the first work for using CNN on the text classification task.

The model architecture is shown in Figure 3.6. It illustrates how the Text-CNN model classifies one text review into the negative or positive sentimental class. First of all, the review, “I like this movie very much!”, was mapped into the word vectors in the embedding layer. The dimension size of the word embedding was 5 in this figure. The whole sentence was converted into a 7×5 matrix. After that, the convolution operation was operated on the matrix, the three kernel sizes of the convolution were 2, 3 and 4. For

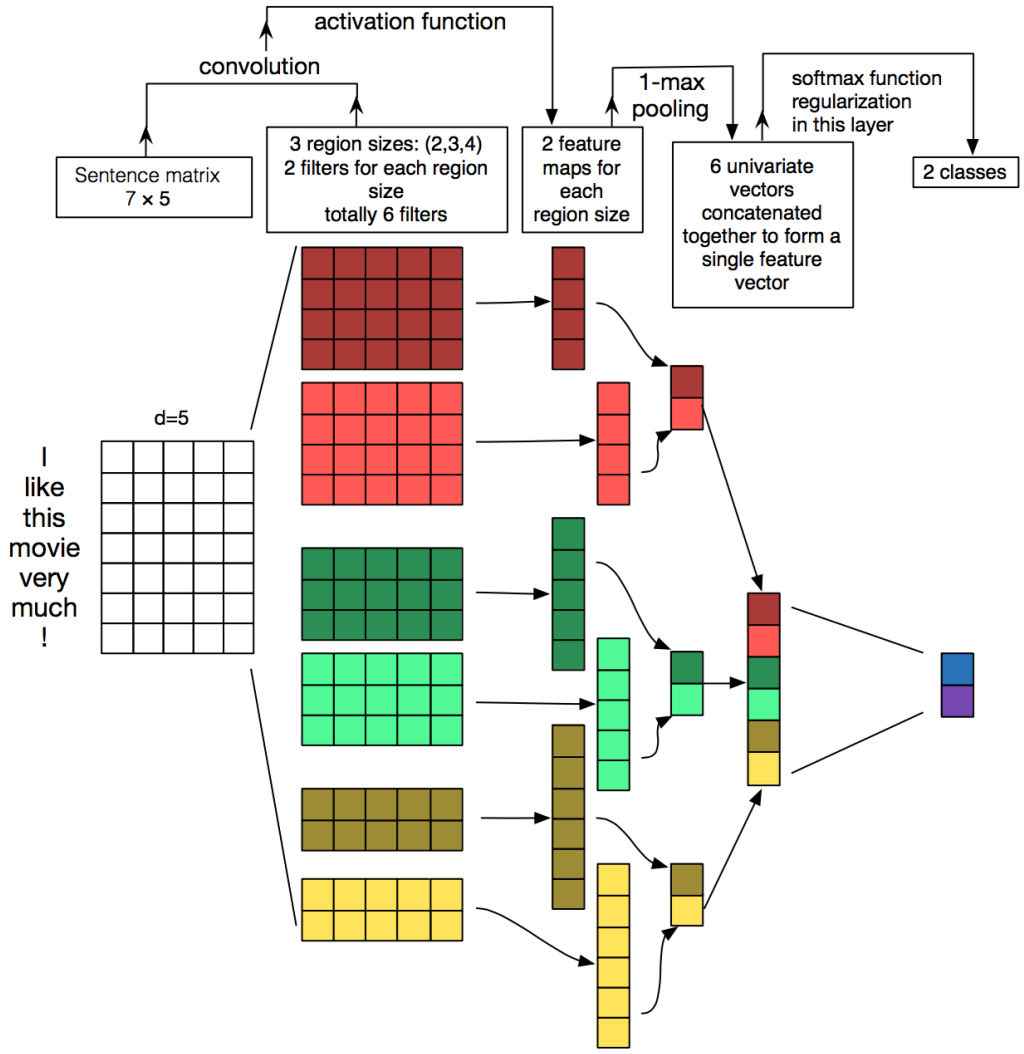

Figure 3.6. *The Text-CNN model architecture [14]*

each convolutional layer, there were 2 filters. Each filter was treated as one channel. In the next step, 2 feature maps were generated from the previous two channels. The max-pooling layer was followed. Then 6 uni-variate vectors were concatenated together to form a single feature vector. The softmax function was used as the activation function in the final layer.

In the authors' experiments, the Text-CNN model achieved the highest accuracy with 88.1% on the SST-2 dataset at the time. In this thesis, the Text-CNN model was used to test on the TripAdvisor dataset and the SST dataset.

3.2.2 Very deep convolutional neural network

The very deep convolutional neural network (VD-CNN) was proposed in [3]. The VD-CNN model operated the input text at character-level instead of words. What's more, small convolutions and pooling operations were applied in the VD-CNN model. Regarding the architecture of the VD-CNN model, four convolutional layers were first stacked. Then three max-pooling layers were added. In previous CNN models for the text classification task, neural networks were shallow, which were up to 6 convolutional layers. However, the VD-CNN was deeper than previous deep neural networks in text classification area at the time. What's more, the VD-CNN model utilized convolutions operation with different kernel sizes. ReLU activation function was applied in the fully connected layers in the VD-CNN model.

The architecture of the VD-CNN model with 29 convolutional layers is shown in Figure 3.7. In this demonstration, the text review was first converted into the word vectors according to the lookup table. One temporal convolution with kernel size as 3 was followed. Then 8 convolutional blocks were added after the temporal layer. Each convolutional block consisted of two convolutional layers and one pooling layer. The function of the pooling layer was to change the dimension of the output through the max pooling or the average pooling operation. The kernel size of each temporal convolution was 3. Two fully connected layers with ReLU activation function were appended. The number of neurons in the last fully connected layer was the same of the number of outputs. The number of neurons depended on the number of different sentimental labels on different text classification task.

One contribution of the VD-CNN model was applying the optional shortcut on the text classification task. The optional shortcut was added between each convolutional block. The idea of the shortcut was from ResNet [6]. In ResNet, shortcut operation was added to mitigate the gradient vanishing problem. In the VD-CNN model, the shortcut had the same function in this model since the model had many layers.

The VD-CNN model was tested on several classification tasks, including text sentiment classification, topic classification, and news categorization. The authors' experiments showed that the VD-CNN model obtained the state-of-art results when compared to other models without data augmentation at the time. What's more, the VD-CNN model with 29 convolutional layers achieved 35.28% error rate and the VD-CNN model with 9 convolutional layers got 37.63% test error on the Yelp full dataset. In the Amazon full review dataset, the error rate of the VD-CNN model was reduced by 1% when the depth of the model increased from 9 to 29.

Convolution operation extracted n-gram features over the different size of tokens from text through using different kernel sizes of convolution. Different n-gram lengths (2-gram or 3-gram short phrases) were needed to model short and long span relations among words in one sentence. Max pooling operation extracted the obvious feature from the windows-

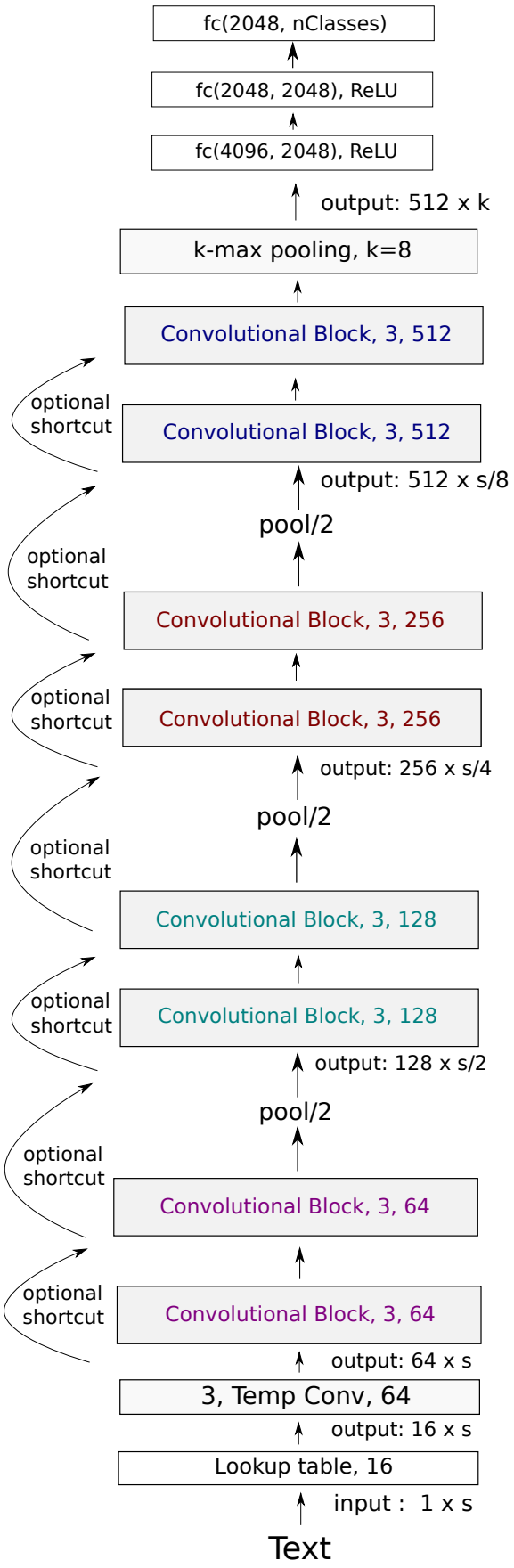

Figure 3.7. *Veep deep convolutional neural network architecture for text classification task [3]*

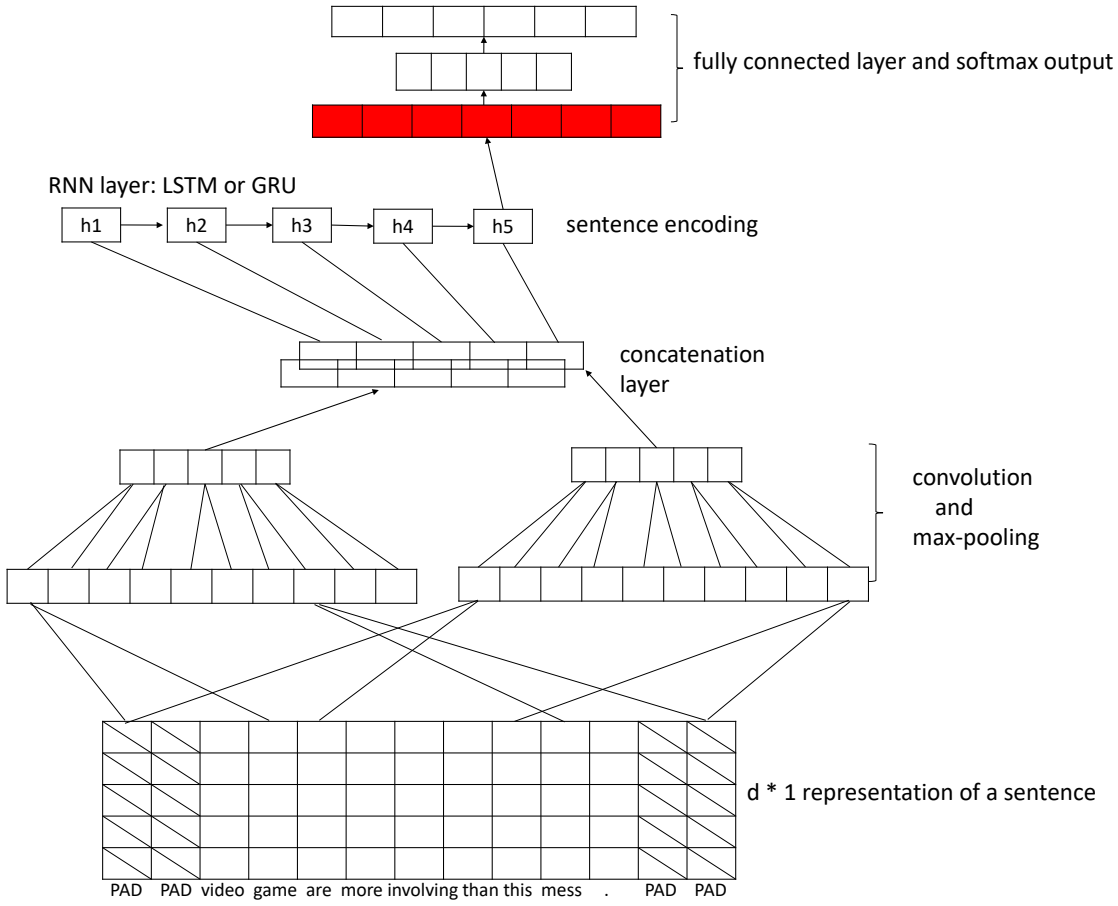

Figure 3.8. *BiLSTM model architecture for an example sentence*

size length text while average pooling operation contained the average information from the sentence. In terms of the effect of pooling operation, the max pooling layer performed better than the average pooling layer in the authors' experiment.

3.2.3 Bidirectional long short term memory neural network

Bidirectional long short term memory neural network (BiLSTM) [41] combined bidirectional recurrent neural networks and convolutional layers into one model for the text classification task. In previous studies, there were some work which combined RNN and CNN in computer vision area. However, there was no study to apply the type of model on text classification task. [41] was the first work to apply the combination of RNN and CNN for the text classification task.

The architecture of the BiLSTM model is shown in Figure 3.8. In this example, the sentence, "video games are more involving than this mess", was selected to give the demonstration. First of all, the English words were passed to the embedding layer and then these words were converted to the word vectors. Two paddings were added on each side. Four paddings were added to the sentence in total. Two convolution and one max-pooling operations were applied in this model respectively. After that, the concatenation

operation was followed. The recurrent neural network layers were added after convolutional layer and the vector representation for the sentence was generated.

There were many different variants for the BiLSTM model. For example, the CNN-LSTM-GloVe is one variants. It was a model with GloVe pre-trained word vectors, max pooling layer and LSTM. Another variant is the CNN-GRU-FastText model. The model used FastText pre-trained word vectors, max pooling, and gated recurrent unit [2] together.

The authors experimented the BiLSTM model in the SST1 dataset. The CNN-LSTM-word2vec model achieved the best performance with 51.50 % accuracy. However, the BiLSTM model got 49.1% accuracy and this model only used recurrent neural network layer without a combination of the convolutional layer.

3.2.4 Loss function on the sentiment classification task

The loss function in this text sentiment classification task uses cross-entropy loss. The total loss of the objective function in the text sentiment classification is given in Equation 3.1. The predicted result was denoted by $\hat{y}_c$, where C is the number of sentimental classes, $\mathbf{y}$ is the one hot ground truth label vector of the final output. In the experiments, the C value is 5, namely *very negative*, *negative*, *neutral*, *positive*, *very positive*. The predicted result is yielded after the softmax activation function. Then the index of the biggest value is selected as the final predicted result. The equation for calculating the probability is given in Equation 3.2, where z is the output vector of the last fully connected, z is input to the softmax activation function.

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{C} \sum_{c \in \{1,\ldots,C\}} y_c \log \hat{y}_c \tag{3.1}$$

$$\hat{y}_c = \text{softmax}(\mathbf{z}) = \frac{\exp\left(z^{(c)}\right)}{\sum_{c \in \{1,\ldots,C\}} \exp\left(z^{(c)}\right)} \tag{3.2}$$

Distance ranking metric

The formulation of the distance ranking metric is given in Equation 3.3. Here, M is the collection of the three models, [Text-CNN, BiLSTM, VD-CNN], $i \in \{1, 2, 3\}$. The higher *distance* value indicates the more inaccurate sentiment prediction reviews are by these three deep neural networks . The predicted sentimental class was denoted by $\hat{y}_i$, The ground truth sentimental class was denoted by y_i.

$$distance = \sum_{i \in M}^{n} \left| \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)} \right| \tag{3.3}$$

3.3 Experiments

3.3.1 Data preprocessing

For the data preprocessing, the input length of the text review is set as 1024. If the length of the review is less than 1024, then the padding is added to the text review to increase its length. If the length of one text review is over 1024, the review is truncated into 1024. For the preprocessing of unknown characters, the English word is deleted if the character is not included in the following dictionary. The dictionary is a total of 69 tokens and shown below.

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:’’’/\|_@#$%ˆ&*˜‘+-=<>()[]{}
```

3.3.2 Experiment environment requirements

The experiment requirements include many factors such as the operating system, deep learning framework etc. The experiment results depend on the environment in some sense. For example, the running time for deep neural networks can have different values on different computer machines.

In the experiments, the detailed environment for training and testing deep neural networks is listed in Table 3.8. In the experiments, the operating system is the Ubuntu 18.04, Python was selected as the programming language. The code implementation was completed on PyTorch [13] framework. All experiments were performed on a single Nvidia TITAN Xp GPU.

PyTorch is one popular open source deep neural network framework. It is developed and maintained by Facebook. One advantage of PyTorch framework is that PyTorch provides both eager mode and graph mode. In the eager mode, developers can easily debug the neural network source code to check the computational graph of a deep neural network. In the graph mode, deep neural network models is optimized for deploying in the industry. Through using PyTorch framework, the difficulty of implementing a neural network is decreased. PyTorch contains common deep learning modules. Many sub-libraries were developed for different tasks. For the natural language processing application, there is the torchtext library. The torchtext library also provides pre-trained GloVe and FastText word embedding vectors.

3.3.3 Experiments result

Model training

Table 3.8. *Experiment environment configuration*

experiment development environment	**tools**
programming language	Python 3.5
operating system	Ubuntu 18.04
PyTorch	1.0.0
torchtext	0.4.0
other libraries	torchtext
GPU	Nvidia TITAN xp
compute unified device architecture (CUDA) library	9.0
memory	32 GB

The three deep neural network models were trained using stochastic gradient descent as the optimization algorithm to update model parameters. Regarding the regularization technique applied in these three DNN models, the dropout was used in the last fully connected layer and the dropout rate was set to 0.5 to prevent the over-fitting problem in the experiments. The initial learning rate for training models was 0.01 and the learning rate was divided by 10 after the decaying loss on the validation set staying on a plateau for 10 epochs. The batch size for training three models was 32. The momentum hyper-parameter was 0.9 for all experiments. All deep neural networks were trained to minimize the negative log-likelihood loss function. The three DNN models were trained for 50 epochs. The early stop mechanism was applied for training models, the training processing was terminated after the accuracy was not improved for over 20 epochs.

Table 3.9. *Experiment hyperparameters setting configuration*

model hyperparameters setting	**value**
batch size	32
embedding dimension	300
dropout rate	0.5
initial learning rate	0.01
epochs	50

The forward and backward propagation were executed in a loop during the training process. In the forward propagation step, the word vectors were initialized with using GloVe and FastText pretrained 300-dimensional word vectors for reviews. The word vectors were passed to the deep neural network. In the backward propagation, the loss was calculated and the loss information was feed backward to the different layers in the model to update the value of parameters.

The training detail of three DNN models on the TripAdvisor dataset is described in Table 3.10. All values were calculated under the same experimental requirements setting. From the table, it shows that the VD-CNN model has 17 million parameters, which is the biggest number of parameters. The BiLSTM model has the least number of the parameters, 3.5

million. The Text-CNN model has almost as same amount of parameters as the BiLSTM. Regarding the training and testing time for one epoch, the VD-CNN model needed to take 619.5 seconds. The BiLSTM model cost the second longest time, 249.0 seconds for one epoch. The Text-CNN model used only 69.6 seconds, which is the least time among all three models. One reason is that the VD-CNN model is the large model and contains the largest number of parameters. Therefore, the VD-CNN model costs a longer time than other models for training. What's more, the BiLSTM model cost around 4 times longer time than the Text-CNN model to complete one epoch. The reason is that the RNN models can not utilize parallel computing because of the time dependency of text sequence data. This feature results in that the RNN models have low efficiency on the usage of GPU when compared to CNN models.

Table 3.10. *Training phrase for these three models*

model	number of parameters	one epoch time(second)
BiLSTM	3,490,781	249.0
VD-CNN	17,318,629	616.5
Text-CNN	3,362,405	69.6

Testing result

The testing result for the three DNN models is described in Table 3.11. In the experiments, the performance of these three DNN models was compared on the TripAdvisor dataset and the Standford Sentiment Tree dataset. The effect of word embedding technique (FastText and GloVe) for initializing word vectors was also investigated among the three DNN models in these two datasets.

Testing result on the TripAdvisor dataset

From the table 3.11, it shows that the BiLSTM model outperformed the other two methods on the TripAdvisor dataset in both FastText and GloVe word embedding method. The BiLSTM model had 1.41% higher accuracy than the VD-CNN model and 1.54% higher accuracy than the Text-CNN model when the GloVe pre-trained word vector was applied. The BiLSTM also outperformed the VD-CNN model 1.3% higher accuracy and the Text-CNN 0.73% higher accuracy when FastText pre-trained word vectors were used.

Regarding the comparison of the GloVe and FastText word embedding technique, the experimental result confirmed that word vector initialization has an impact on the prediction result for the text sentimental classification task, which indicates the word vectors initialization affects the learning ability of the three deep neural networks. For the BiLSTM and the VD-CNN models, GloVe word embedding achieved a better result than FastText word embedding. The BiLSTM with GloVe word embedding had 73.73% accuracy while the BiLSTM model with FastText had 73.25% accuracy.

Regarding the F1 score metrics, the Text-CNN model and BiLSTM model had the 0.68 F1-score on the TripAdvisor test dataset, the VD-CNN model archived the 0.67 F1 score on the TripAdvisor dataset, which was the lowest F1 score among all three models.

Testing result on the SST dataset

The three DNN models were tested on the Stanford Sentiment Treebank (SST) dataset as well. The experiments showed that the VD-CNN model had the worst performance when compared to the Text-CNN and the BiLSTM model. The VD-CNN model had 36.60% accuracy with using FastText word embedding and 37.36% accuracy with using GloVe word embedding. The BiLSTM model contained the highest accuracy, 41.79% on the SST.

Table 3.11. *Three models performance evaluated on the TripAdvisor and SST dataset*

model	**embedding**	**dimension**	**accuracy(%)**		**F1 score**	
			TripAdvisor	SST	TripAdvisor	SST
BiLSTM	GloVe	300	73.73	36.35	0.68	0.35
BiLSTM	FastText	300	73.25	32.69	0.68	0.34
VD-CNN	GloVe	300	72.31	25.58	0.67	0.25
VD-CNN	FastText	300	71.95	26.97	0.67	0.26
Text-CNN	GloVe	300	72.19	26.32	0.68	0.30
Text-CNN	FastText	300	72.52	31.15	0.68	0.33

Result of the confusion matrices

The prediction result of the BiLSTM, Text-CNN and VD-CNN models were also investigated. The confusion matrices of these three DNN models were calculated and shown below. All these confusion matrices were calculated with using GloVe instead of FastText word embedding.

For the Text-CNN model, the experimental result on the TripAdviosr test set is shown in Figure 3.9. The experiment result of the VD-CNN model is shown in Figure 3.10. For the BiLSTM neural network, the confusion matrix of the experiments results on the TripAdvisor test dataset is shown in Figure 3.11. It shows that the *positive* class contains the largest number of misclassified hotel reviews. In detail, the *positive* sentimental class in the TripAdvisor dataset had 33% of the whole reviews were wrongly predicted to the *very positive* class. What's more, 30% out of the whole *neutral* class hotel reviews in the TripAdvisor test set were misclassified to the *positive* class while only 15% out of the *neutral* sentimental category was misclassified as the *negative* class. It indicates the BiLSTM model is more prompted to misclassify the hotel review toward the positive sentiment than the negative sentiment.

The second observation from the three confusion matrix figures is that the *very positive* sentiment class achieved the highest classification accuracy when compared to the other four sentimental classes. Moreover, the three models achieved the best result on the precision, recall and F1 score evaluation metrics on the *very positive* sentiment class than other four sentimental classes.

The third observation is that these three models had the worst result on the *negative* sentiment reviews while achieved the best predicted result on the *very positive* class.

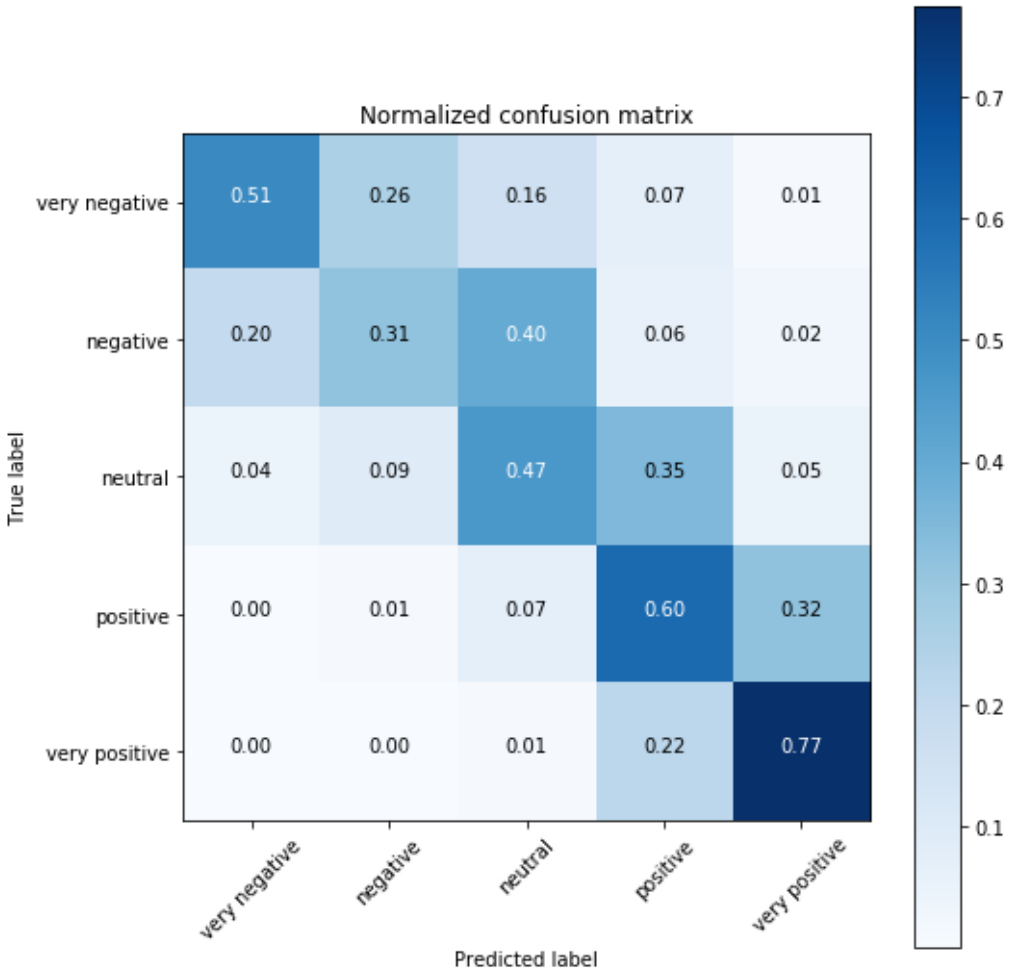

Figure 3.9. *Confusion matrix of the Text-CNN model prediction result on the TripAdvisor dataset*

3.4 Discussion

The reasons for misclassifying hotel reviews in the TripAdvisor dataset by the three DNN models are analyzed. Table 3.12 presents the top 5 misclassified reviews in the TripAdvisor dataset by the three models according to the descending order of the $distance$ metrics. The ranking of these hotel reviews was based on the $distance$ value metrics. The distance value on the most misclassified hotel review in the table is 14.

The analysis of the misclassified sentimental predication result on table 3.12 indicates that long hotel reviews with more contradictory sentimental words were more prompted to be classified wrongly. For example, one hotel review in the TripAdvisor dataset is shown below.

"I stayed at the Putney Bridge Premier Inn hotel for one night on the 31.01.2012. After Booking in, my wife and I went to London to watch a shoe and for some dinner. A great night was enjoyed and we headed back to the hotel for a good nights sleep. After falling asleep, we were both woken by a noisey couple of people going past our door. We fell

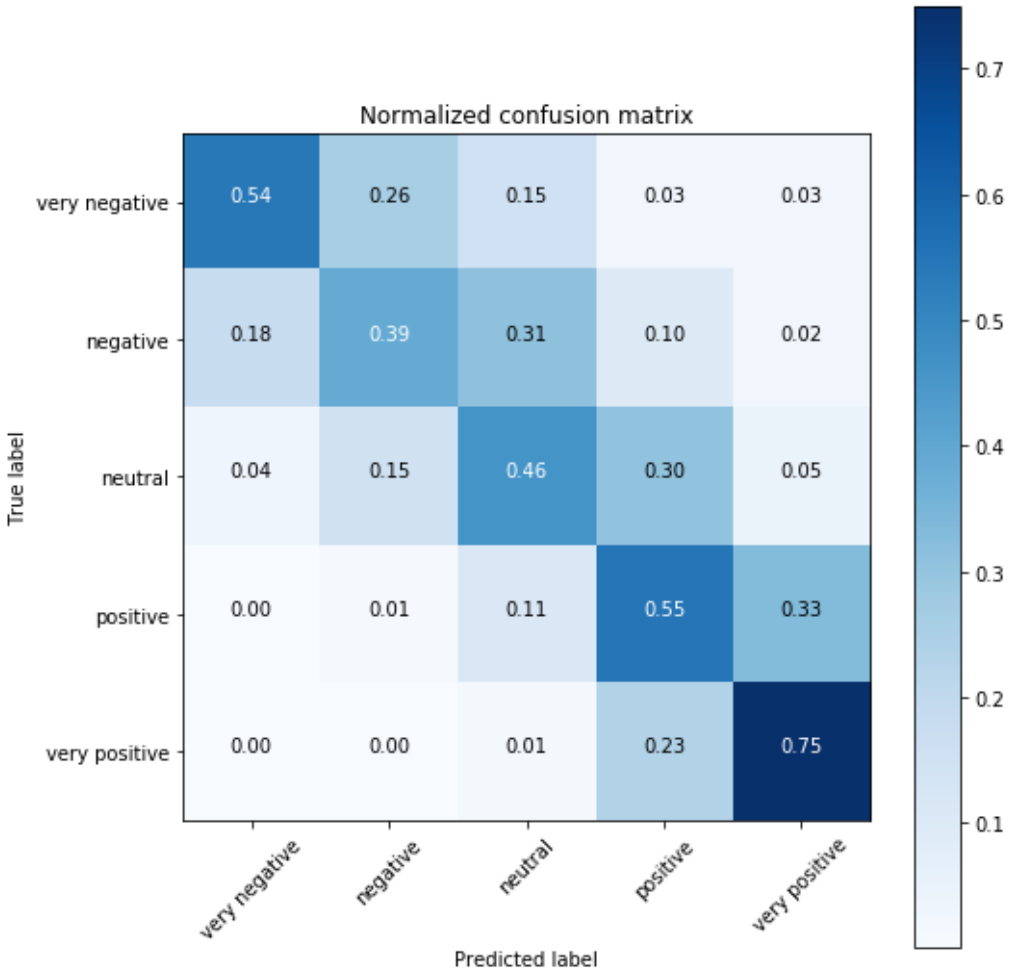

Figure 3.10. *Confusion matrix of the VD-CNN model prediction result on the TripAdvisor dataset*

back to sleep and was again woken by the same female voices making as much noise as before. Not happy.Then again fell back to sleep and woke for our breakfast at 6am. We were about to be seated when asked about our nights sleep by Izabella Veres. I explained about the loud voices that woke us. She appoligised on behalf of the company and showed us to our table. After finishing our food Izablla approached us to say she had reported the inccedent and a member of staff would be talking to us. Within 10 mins of first asking us about the night, the stiuation had been resolved and a full refund given by Mr Marcello. I also run a business with staff, and left so impressed with the way I was handled that I have left this report as a way of saying THANK YOU. I stay with Premier Inn hotels as much as I can allready as do my members of staff on business. With this high standard of respect shown by the staff, we as a company will continue to do so for many years to come."

The ground truth class is the *very positive* class. The Text-CNN model and the VD-CNN model wrongly predicted this review as the "*very negative*" class and the BiLSTM model misclassified this review as the *negative* class. This review contains both negative and positive words. The negative words include "noisey", "Not happy" and "appoligised".

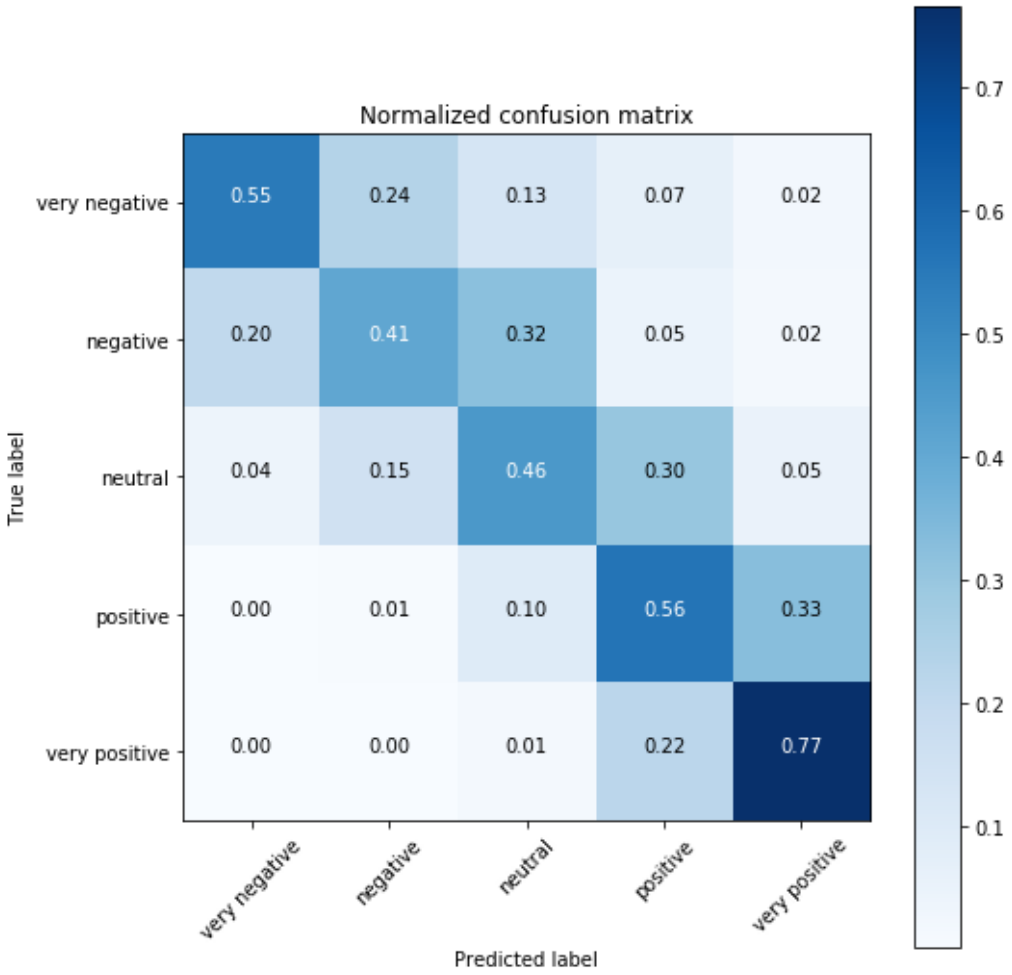

Figure 3.11. *Confusion matrix of the BiLSTM model prediction result on the TripAdvisor dataset*

Regarding the positive words, the review contains "great", "enjoyed" and "good" words. These positive sentimental words expressed the guest was satisfied with staying in the hotel while the negative words expressed the "negative" feeling at the same time. This contradiction increases the difficulty for correctly classifying the review for deep neural network models. Even though the overall sentiment of this long hotel review is "positive", all three DNN models can not capture the semantic relationship of these words in such long text. And these three models misclassified the hotel reviews as the negative sentimental class. For the other three misclassified hotel reviews in the table, they are long text review and they contain contradictory words as well.

Another example is the last hotel review shown in the table. The ground truth sentimental class for this review is the "very negative" sentimental class. However, this review does not contain any sentimental words. It is more prompted to narrative one story than expressing the reviewer's opinion. All three DNN models wrongly predicted it as the "very positive" class.

Table 3.12. *Top 5 misclassified hotel reviews in the TripAdvisor dataset*

hotel review	sentiment class	distance
" I stayed at the Putney Bridge Premier Inn hotel for one night on the 31.01.2012. After Booking in, my wife and I went to London to watch a shoe and for some dinner. A **great** night was **enjoyed** and we headed back to the hotel for a **good** nights sleep. After falling asleep, we were both woken by a **noisey** couple of people going past our door [...]"	4	14
"I did not actually stay at this hotel as they were fully booked but I'd like to give credit where credit is due. The hotel I had booked turned out to be **incredibly unprofessional** and seemed very **dodgy** and I decided that it was not worth my time staying there. So at 1am I found myself wandering the streets of London trying to find another hotel, which seemed impossible as everywhere was fully booked. I explained my situation to the night managers Jorge and Bogdan who very **kindly** let me sit in the hotel bar area use the WIFI and to find a hotel nearby [...]"	4	14
"The hotel is located at the corner of Strand and Aldwych. It is in a very central location for Covent Garden, the Thames, Trafalgar square etc. The staff is very **professional** and the room was very clean. One thing that really **bothered** us is that they cleanign staff leaves the windows wide open when they clean the room. This is **good** if you want fresh air, but one [...]"	4	12
"Booked to stay here for the opening weekend of Secret Cinema (which was cancelled at the last minute).Despite cancelling very late, the hotel was extremely understanding and **gracious** about it and didn't charge us at all.It's a **shame** we didn't get to stay, but thought that their **good** grace should be noted."	4	12
"Pop in to the bar for a couple of drinks, and a pint of Diet Coke is £6! I have to go to the bar to order, yet you still have the audacity to include a service charge. The service was pretty substandard too."	1	9

4 CONCLUSION

In this thesis, three different deep neural networks (BiLSTM, Text-CNN, and VD-CNN) were experimented to perform the sentiment classification task on the TripAdvisor dataset and SST dataset. Moreover, two different word embedding initialization technique were compared in the experiments.

In the TripAdvisor dataset, the BiLSTM model with GloVe word embedding technique achieved the best performance with reaching 73.73% test accuracy in the 5 classes task. The VD-CNN model with FastText word embedding had 71.95% test accuracy on the TripAdvisor dataset, which was the worst prediction result. The BiLSTM model also maintained 0.68 F1 score in the TripAdvisor test dataset while the VD-CNN model had 0.67 F1 score with both GloVe and FastText word embedding in the experiment.

In the SST dataset, the BiLSTM model with GloVe word embedding reached the highest test accuracy with 36.35%. The BiLSTM model had the best performance in terms of accuracy and F1 score when compared to the VD-CNN and the Text-CNN models in the TripAdvisor dataset and the SST dataset.

The GloVe word embedding outperformed the FastText method in the Text-CNN model and the BiLSTM model in both the TripAdvisor dataset and the SST dataset. In the VD-CNN model, the GloVe method had slightly higher accuracy than the FastText on the TripAdvisor dataset but performed a little worse in the SST dataset.

REFERENCES

[1] L. Bottou. Stochastic Gradient Descent Tricks. *Neural Networks: Tricks of the Trade*. 2012.

[2] J. Chung, C. Gulcehre, K. Cho and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[3] A. Conneau, H. Schwenk, L. Barrault and Y. Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781* (2016).

[4] I. Goodfellow, Y. Bengio and A. Courville. *Deep learning*. MIT press, 2016.

[5] A. Hassan and A. Mahmood. Convolutional recurrent deep learning model for sentence classification. *Ieee Access* 6 (2018), 13949–13957.

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

[7] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation* 9 (1997), 1735–1780.

[8] https://fasttext.cc. *unsupervised-tutorial*. `https://fasttext.cc/docs/en/unsupervised-tutorial.html`. Accessed February 25, 2019.

[9] C. J. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Eighth international AAAI conference on weblogs and social media*. 2014.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[11] I. Jolliffe. *Principal component analysis*. Springer, 2011.

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

[13] N. Ketkar. Introduction to pytorch. *Deep learning with python*. Springer, 2017, 195–208.

[14] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] D. Kouzis-Loukas. *Learning Scrapy*. Packt Publishing Ltd, 2016.

[17] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[18] A. Krizhevsky, I. Sutskever and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, 1097–1105.

[19] S. Lai, L. Xu, K. Liu and J. Zhao. Recurrent convolutional neural networks for text classification. *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[20] Y. LeCun, Y. Bengio and G. Hinton. Deep learning. *nature* 521.7553 (2015), 436.

[21] X. Li, Z. Zhang and K. Stefanidis. Mobile App Evolution Analysis Based on User Reviews. *SoMeT*. 2018.

[22] X. Li, Z. Zhang and K. Stefanidis. Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates. *ICSEA 2018* (2018), 109.

[23] E. Loper and S. Bird. NLTK: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[24] W. Maalej, Z. Kurtanović, H. Nabil and C. Stanik. On the automatic classification of app reviews. *Requirements Engineering* 21.3 (2016), 311–331.

[25] T. Mikolov, M. Karafiát, L. Burget, J. Černock and S. Khudanpur. Recurrent neural network based language model. *Eleventh annual conference of the international speech communication association*. 2010.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*. 2013, 3111–3119.

[27] B. Pang, L. Lee and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics. 2002, 79–86.

[28] J. Pennington, R. Socher and C. Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, 1532–1543.

[29] L. R. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *ieee assp magazine* 3.1 (1986), 4–16.

[30] I. Rish et al. An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. 2001, 41–46.

[31] D. E. Rumelhart, G. E. Hinton, R. J. Williams et al. Learning representations by back-propagating errors. *Cognitive modeling* 5.3 (1988), 1.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115.3 (2015), 211–252.

[33] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45.11 (1997), 2673–2681.

[34] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, 1631–1642.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15.1 (2014), 1929–1958.

[36] C. Sutton, A. McCallum et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning* 4.4 (2012), 267–373.

[37] C. Szegedy, S. Ioffe and V. Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *AAAI*. 2016.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 2818–2826.

[39] TripAdvisor. *Hotel Review*. `https://www.tripadvisor.com/ShowUserReviews-g188590-d189389-r145147482-Sofitel_Legend_The_Grand_Amsterdam-Amsterdam_North_Holland_Province.html`. Accessed February 20, 2019.

[40] S. Wang, S. Mazumder, B. Liu, M. Zhou and Y. Chang. Target-sensitive memory networks for aspect sentiment classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, 957–967.

[41] X. Wang, W. Jiang and Z. Luo. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, 2428–2437.

[42] P. J. Werbos et al. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78.10 (1990), 1550–1560.

[43] B. Xu, N. Wang, T. Chen and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[44] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola. *Dive into Deep Learning*. `http://www.d2l.ai`. 2019.

[45] L. Zhang and B. Liu. Sentiment Analysis and Opinion Mining. *Encyclopedia of Machine Learning and Data Mining*. 2017.

[46] X. Zhang, J. Zhao and Y. LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*. 2015, 649–657.

A APPENDIX

- The TripAdvisor dataset is provided in this link, https://drive.google.com/open?id=19TdLOrqjAQ11lpYRx1r2GcYo79fZXloE.
- The source code is provided in this link, https://github.com/yipersevere/Deep-neural-netw