Tampere University

Yi Zhou

SENTIMENT CLASSIFICATION WITH DEEP NEURAL NETWORKS

Faculty of Information Technology and Communication Sciences
Master of Science thesis
May 2019

ABSTRACT

Yi Zhou: Sentiment classification with deep neural networks
Master of Science thesis
Tampere University
Master's Degree Programme in Software Development
May 2019

Sentiment classification is an important task in Natural Language Processing (NLP) area. Deep neural networks become the mainstream method to perform the text sentiment classification nowadays. In this thesis two datasets are used. The first dataset is a hotel review dataset (TripAdvisor dataset) that collects the hotel reviews from the TripAdvisor website using Python Scrapy framework. The pre-processing steps are then applied to clean the dataset. A record in the TripAdvisor dataset consists of the text review and corresponding sentiment score. There are 5 sentimental labels: very negative, negative, neutral, positive, and very positive. The second dataset is the Stanford Sentiment Treebank (SST) dataset. It is a public and common dataset for sentiment classification.

Text Convolutional Neural Network (Text-CNN), Very Deep Convolutional Neural Network (VD-CNN), and Bidirectional Long Short Term Memory neural network (BiLSTM) were chosen as different methods for the evaluation in the experiments. The Text-CNN was the first work to apply convolutional neural network architecture for the text classification. The VD-CNN applied deep convolutional layers, with up to 29 layers, to perform the text classification. The BiLSTM exploited the bidirectional recurrent neural network with long short term memory cell mechanism. On the other hand, word embedding techniques are also considered as an important factor in sentiment classification. Thus, in this thesis, GloVe and FastText techniques were used to investigate the effect of word embedding initialization on the dataset. GloVe is a unsupervised word embedding learning algorithm. FastText uses shallow neural network to generate word vectors and it has fast convergence speed for training and high speed for inference.

The experiment was implemented using PyTorch framework. It shows that the BiLSTM with GloVe as the word vector initialization achieved the highest accuracy 73.73% while the VD-CNN with FastText had the lowest accuracy 71.95% on the TripAdvisor dataset. The BiLSTM model achieved 0.68 F1-score while the VD-CNN model obtained 0.67 F1-score on the TripAdvisor dataset. On the SST dataset, BiLSTM with GloVe again achieved the highest accuracy 36.35% and 0.35 F1-score. The VD-CNN model with GloVe had the worst evaluation result in terms of accuracy and F1-score. The Text-CNN model performed better than the VD-CNN model even thought the VD-CNN model has more layers in most cases.

By analyzing the misclassified reviews in the TripAdvisor dataset from the three deep neural networks, it is shown that the hotel reviews with more contradictory sentimental words were more prone to misclassification than other hotel reviews.

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

PREFACE

I want to express my deep gratitude to my supervisors, Prof. Jyrki Nummenmaa, Associate Prof. Kostas Stefanidis and Associate Prof. Heikki Huttunen. They introduced me into this research area. Without their guide, this master thesis is not possible to complete. Apart from the supervision, I would like to appreciate the kindness and patience of my supervisors since the thesis took a long time to complete.

I would like to express my thanks to my family. They help me a lot for my study in this lovely country, Finland. I also express my thanks for my colleges, they provided lots of help during the process of writing the thesis.

Tampere, Finland, 20th May 2019

Yi Zhou

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

BiLSTM	Bidirectional Long Short Term Memory
BPTT	Backpropagation Through Time
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DNN	Deep Neural Network
ELMo	Embeddings from Language Models
FN	False Negative
FP	False Positive
GRU	Gate Recurrent Units
ILSVRC	ImageNet Large Scale Visual Recognition Competition
JSON	JavaScript Object Notation
LR	Linear Regression
LSTM	Long Short Term Memory
MLP	Multi-Layer Perceptron
NB	Naive Bayes
NLP	Natural Language Processing
NNLM	Nerual Network Language Model
PCA	Principle Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SST	Stanford Sentiment Treebank
SVD	Singular Value Decomposition
SVM	Support Vector Machine
Text-CNN	Text Convolutional Neural Network
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive

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 opinions 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 [5] and TripAdvisor [6]. These hotel reviews are important and useful for both customers and hotel owners. Through analyzing these reviews, hotel owners can know where the hotel 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.

A 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 subtopic 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 methods have been researched to solve the task. For example, a method is publishing the new

[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.airbnb.com`
[6] website link: `https://www.TripAdvisor.com`

text review dataset as the common benchmark, thus, researchers can use the dataset to compare the performance of 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 methods since the huge success of deep learning technique in computer vision in 2012. Deep neural networks have achieved the state of the art performance for sentiment classification.

1.2 Objective

The goal of this thesis is to illustrate the whole processing steps of applying deep neural networks for sentiment classification. These steps include retrieving the new text data, cleaning the data, constructing deep neural networks and comparing the performance of the algorithms on the data. The topic area is identified as sentiment classification on hotel reviews. The performance of three different deep neural networks is evaluated 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 networks on the TripAdvisor dataset and SST dataset for sentiment classification. The metrics include accuracy and F1-score. After completing the experiments, the model with the highest accuracy and F1-score and the model with the lowest accuracy and F1-score on these two datasets are shown.

The second research question is to compare the effectiveness of the GloVe and FastText word embedding techniques for word vector initialization on deep neural networks for the text sentiment classification task. Through the experiments, the word embedding initialization technique with the better performance on the text review sentiment classification task is shown.

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.

The structure of the thesis is described as follow. Chapter 2 introduces the background knowledge of neural networks and natural language processing. Chapter 3 deals with the experiments. Chapter 4 contains the classification results and evaluation. Finally, chapter 5 draws the conclusion of the thesis and the future work.

2 THEORY

This chapter provides the comprehensive background knowledge about the sentiment classification algorithms. The detailed concept and related theoretical background about neural network are first introduced in the first section, including basic module of a neural network, common regularization technique, and strategies for training DNN models. In the second section, concepts in NLP are described including word embedding technique and sentiment classification algorithms. In the third section, common assessment metrics are introduced to evaluate different algorithms for sentiment classification.

2.1 Neural network theory

In this section, details of a single neuron and activation functions are first introduced. Secondly, two types of deep neural networks (recurrent neural network and convolutional neural network) are elaborated. Thirdly, we describe the optimization algorithms, back-propagation and regularization technique.

2.1.1 Neural network

Single neuron cell

A single neuron is composed of layer in artificial neural networks. The structure of a single neuron is shown in Figure 2.1. The inputs are the vector $\boldsymbol{x}$ = [$\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ ... $\boldsymbol{x}_n$], n denotes the number of inputs, the matching weight for the input is the vector $\boldsymbol{W}_j$ = [$\boldsymbol{W}_{1j}$, $\boldsymbol{W}_{2j}$, $\boldsymbol{W}_{3j}$ ... $\boldsymbol{W}_{nj}$], j is the j_{th} neuron in the layer. The inputs are first multiplied by $\boldsymbol{W}_j$, the temporary result is generated and the bias value is added to the temporary result, b is the bias. Then this result is inputted into the activation function f. The corresponding mathematical form of the forward computation in a single neuron is given in Equation 2.1, where i counters from 1 to n, y_j is the output value of the j_{th} neuron for the input $\boldsymbol{x}$.

$$y_j = f\left(b + \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{W}_{ij}\right) \tag{2.1}$$

Activation functions

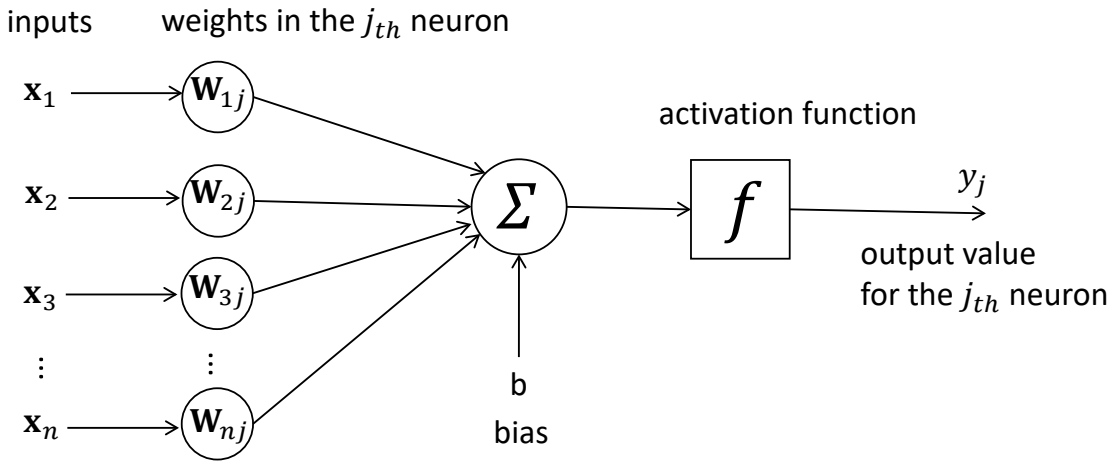
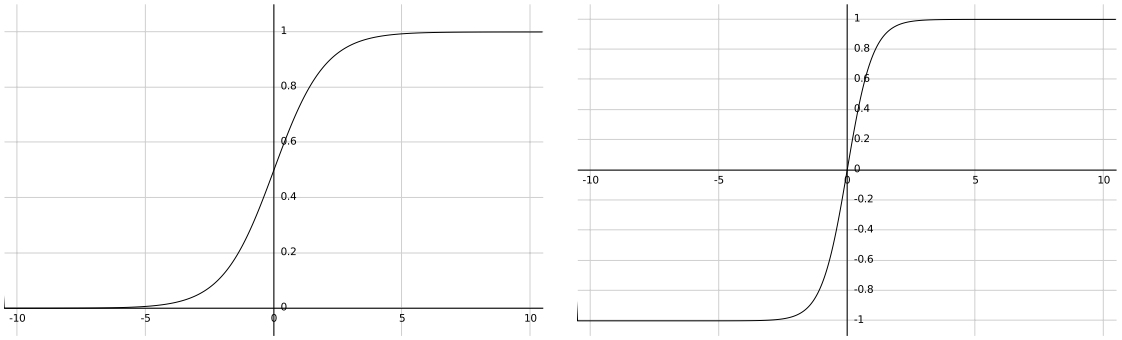


Figure 2.1. *The structure of a single neuron*



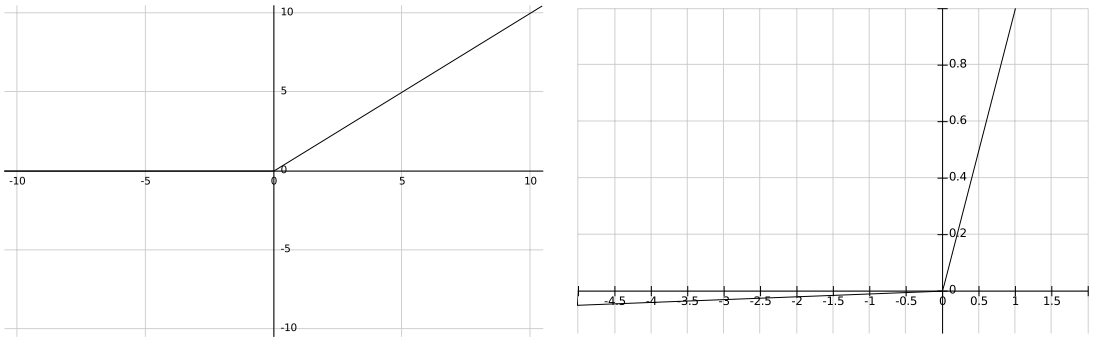
(a) *Sigmoid activation function* ***(b)*** *Tanh activation function*

Figure 2.2. *Sigmoid and Tanh activation function*

Activation functions are a basic component of an artificial neuron. A numerical value is accepted as the input in the activation function. The output is calculated after applying the mathematical calculation. There are a few common activation functions such as sigmoid, Rectified Linear Unit (ReLU)[55] etc.

The Sigmoid function is a common activation function for calculating probability. The mathematical formula of the sigmoid function is $Sigmoid(x) = 1/\left(1 + e^{-x}\right)$. The plotting of this activation function is shown in Figure 2.2a. It is seen that the sigmoid function takes a real-valued number and maps the input into a range between 0 and 1. In particular, when a large negative value is inputted into the sigmoid function, the output value is near 0. If a large positive value is inputted into the sigmoid function, one value near 1 is generated.

The Tanh function is another non-linearity activation function. The formula of this function is $(x) = 2/\left(1 + e^{-2x}\right) - 1$. The plotting of this activation function is shown in Figure 2.2b. The output range of the Tanh function is $(-1, 1)$. The Tanh activation function is similar with the sigmoid function, which it saturates. However, the output of the Tanh function is zero-centered.



(a) *ReLU activation function* ***(b)*** *Leaky ReLU activation function*

Figure 2.3. *ReLU and leaky ReLU activation function*

The ReLU [36] is a piece-wise linear activation function. The formula of the ReLU activation function is $ReLU(x) = \max(0, x)$ and it is plotted in Figure 2.3a. It shows that 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 compute, this feature delivers the benefit on computing speed. The second advantage is that ReLU does not saturate like the Sigmoid or Tanh activation function since the gradient of ReLU is constant. These two advantages lead to fast converging speed when a DNN model with ReLU activation function is trained. One variant of the ReLU function is the leaky ReLU activation function [33]. The mathematical formula of leaky ReLU is given in Equation 2.2. It is plotted in Figure 2.3b. The first improvement of leaky ReLU is that it does not have zero-slope parts. The second improvement is that leaky ReLU has faster training speed than ReLU. Thus, the leaky ReLU is considered as an alternative for the ReLU function.

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{2.2}$$

Neural network

The typical architecture of a neural network consists of input layer, hidden layer and output layer. A classic demonstration of a neural network is the multi-layer perceptron. The architecture of multi-layer perception is shown in Figure 2.4. It shows this multi-layer perception model consists of a input layer, a hidden layer and a output layer. Regarding the architecture of deep neural networks [28], it consists of more that one hidden layers.

2.1.2 Convolutional neural network

Convolutional Neural Network (CNN) is an important category of neural networks. CNN models usually consist of convolutional layers, hidden layers, pooling layers and fully connected layers.

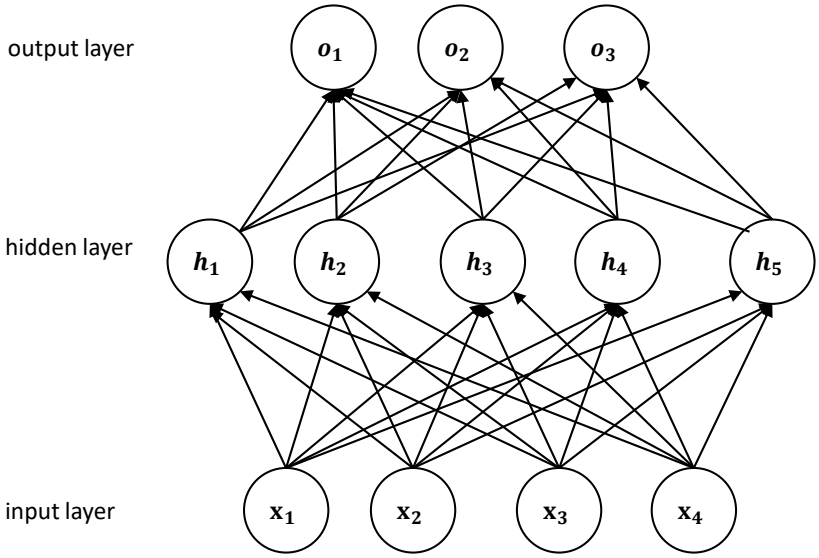


Figure 2.4. *Typical architecture of the multi-layer perceptron*

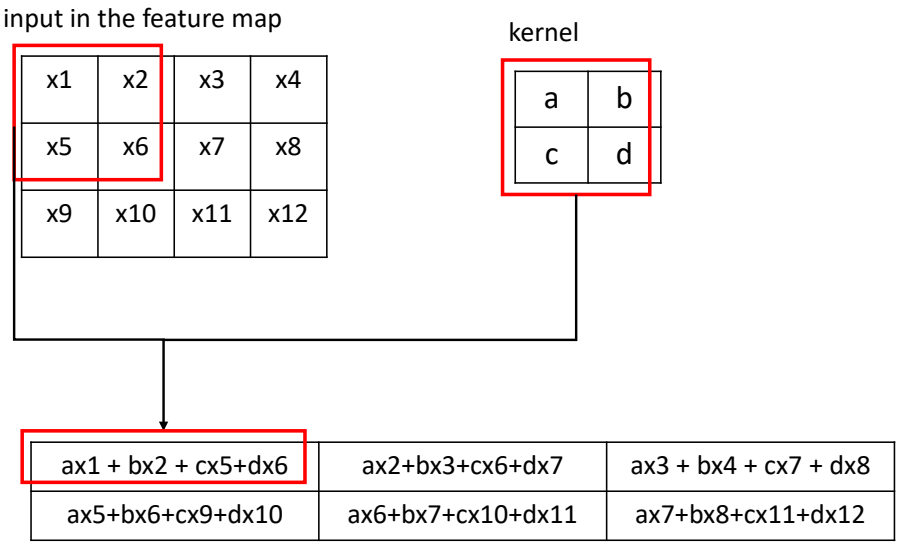


Figure 2.5. *Convolution operation illustration*

Convolutional layer

Convolutional layer is the key element for building a CNN model. Through the convolution operation, features in the convolutional window are learned. In addition, parameter sharing scheme is applied to reduce the number of parameters when operating the convolution. A demonstration of the convolution operation in CNN model is shown in Figure 2.5. It shows that the volume size of the feature map is 3×4, the convolution kernel size is 2×2 and the stride step is 1. 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.

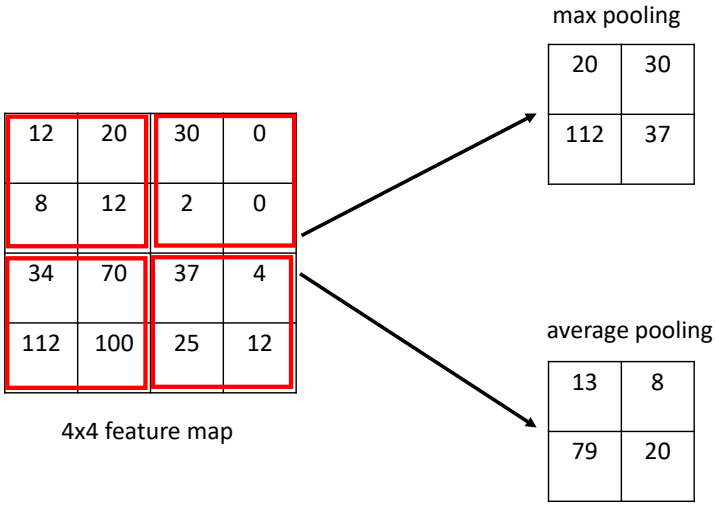


Figure 2.6. *Max pooling and average pooling operation comparison*

Pooling layer

Pooling layer is another important component in CNN models. The pooling operation in CNN is to aggregate the spatial feature. The max pooling and the average pooling operation are the two main pooling operations. The max pooling operation extracts the most obvious feature from the feature map. For example, the max pooling operation can detect edges for the image data. The average pooling operation extracts features in a smooth manner. It takes all values inside the convolutional window and computes the average value out of the window, which indicates that the average pooling operation takes into all values into account. The Figure 2.6 shows the difference between the average pooling and the max pooling operation. In the figure, the size of the feature map is 4 $\times$ 4. The convolutional window size is 2 $\times$ 2. During the max pooling operation, the largest value in each convolutional window is selected while the average value in each convolutional window is calculated in the average pooling operation.

Common CNN models

During the development of convolutional neural networks, many significant variant CNN models were proposed. AlexNet [26] won the champion in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [43] 2012 competition. The ImageNet dataset [7] contains 1000 classes images, including dogs, cats, flowers, and planes etc. The task for this competition is to classify images into correct classes. AlexNet model achieved 83.6% the highest accuracy. It decreased 9.4% error rate from the champion model in the previous year. Regarding the architecture of AlexNet, it is a typical modern CNN model, which consists of five convolutional layers, max-pooling layers, three fully connected layers. The VGG-16 model [6] won the ILSVRC champion in 2013. A 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. It was the deepest CNN model in 2013 and the VGG-16 model decreased the error rate to 7.3% in the ImageNet dataset and showed the pow-

erful learning ability of deep neural networks. Another contribution was the concept of repeating module to construct deep neural networks. This idea of using repeating block was inherited by many other models afterwards. Regarding the model architecture, the VGG-16 model comprised of 13 convolutional layers and 3 fully connected layers. And the size of reception filters for convolution operation was 3×3. Kaiming et al. proposed the ResNet model [12] in 2015. The ResNet model with 50 convolutional layers won the ILSVRC competition in the year. A innovation of ResNet model was that the shortcut operation was introduced 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 in DNN models can be passed to the next layers easier. The shortcut operation becomes the a useful technique for building CNN model later.

2.1.3 Recurrent neural network

Recurrent Neural Network (RNN) [34] is another important type of deep neural networks. RNN model is good at processing sequence data and text data is an important type 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 2.7. These four models are designed to deal with corresponding specific tasks. The tag prediction task [10] for one sentence in NLP area is the typical scenario for the many-to-many RNN model. Another typical application for the many-to-many RNN model is the machine translation task. The sentiment classification task is a representative task for the many-to-one RNN model. In the sentiment classification task, each word in the sentence is considered as one input, the predicted result of the sentiment class for the sentence is the only output.

When comparing to CNN models, a 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 shallow. The main reason for shallow layers in RNN is that RNN model is unfolding and calculated along with the time steps while CNN model calculate in the space instead.

RNN uses hidden states to store information which is generated in the previous time steps. The typical structure of RNN model unfolding computational graphs along with the time steps is shown in Figure 2.8. In this figure, the vector $\boldsymbol{x}$ is the input, the vector $\boldsymbol{o}$ is the predicted result, the loss function is $\boldsymbol{L}$, the true class is $\boldsymbol{y}$, the $\boldsymbol{h}$ denotes the hidden state, the weight matrix between the input and the hidden state is presented by $\boldsymbol{U}$. The weight matrix for the connection between the hidden state and the output is $\boldsymbol{V}$. $\boldsymbol{W}$ denotes the weight matrix between the hidden state in previous time step and the hidden state at current time step. The vectors $\boldsymbol{x}^{(t-1)}$, $\boldsymbol{x}^{(t)}$, $\boldsymbol{x}^{(t+1)}$ are three inputs at the $t-1$, t, $t+1$ three different time steps respectively. $\boldsymbol{x}^{(t-1)}$ connects the hidden state

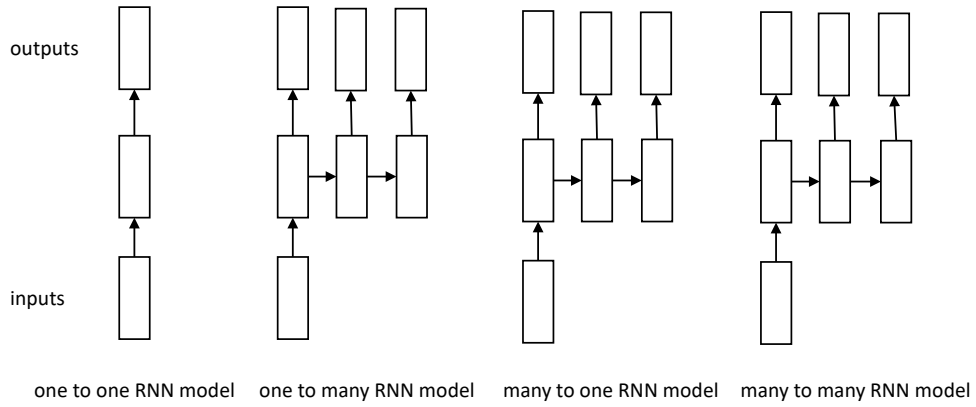


Figure 2.7. *Four types of RNN models*

$\boldsymbol{h}^{(t-1)}$ in the $t-1$ time step, $\boldsymbol{o}^{(t-1)}$ denotes the output in the $t-1$ time step. The vector $\boldsymbol{x}^{(t)}$ is the input in the t time step, $\boldsymbol{b}$ and $\boldsymbol{c}$ are bias vectors, ϕ is the activation function. The mathematical form of calculating the predicted value $\boldsymbol{o}$ is given in Equation 2.3. The total loss for the given sequence $\boldsymbol{x}$ with $\boldsymbol{y}$ as the targeted values is the sum of the losses over all the time steps.

$$\boldsymbol{a}^{(t)} = \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b} \tag{2.3}$$

$$\boldsymbol{h}^{(t)} = \phi\left(\boldsymbol{a}^{(t)}\right)$$

$$o^{(t)} = c + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

Long short term memory neural network

LSTM [13] is a well-known type of recurrent neural network and was proposed in 1997. One drawback of traditional RNN is that it is hard to capture the semantic and syntactic relation between two words with long distance in the text sequence data . This problem is known as the short memory issue. Long Short Term Memory neural network (LSTM) can mitigate this problem by using gate control mechanism to learn long term dependency in sentences. Besides, LSTM model can mitigate the gradient exploding problem. During the training phase of RNN model, gradients are usually exploded when the propagation of the gradients feeds forward. The design of the LSTM model overcomes the technical challenges of RNN to deliver on the promise of sequence prediction with neural networks.

Regarding the structure of LSTM cell, it is shown in Figure 2.9. Three gates are designed to control the information flow in neural networks, they are the input gate, the output gate and the forget gate respectively. The matrix $\boldsymbol{X}_t \in \mathbb{R}^{n \times d}$ is the input of the LSTM cell in the t time step, n is the number of samples, d is the number of dimensions. $\boldsymbol{I}_t$ is the value

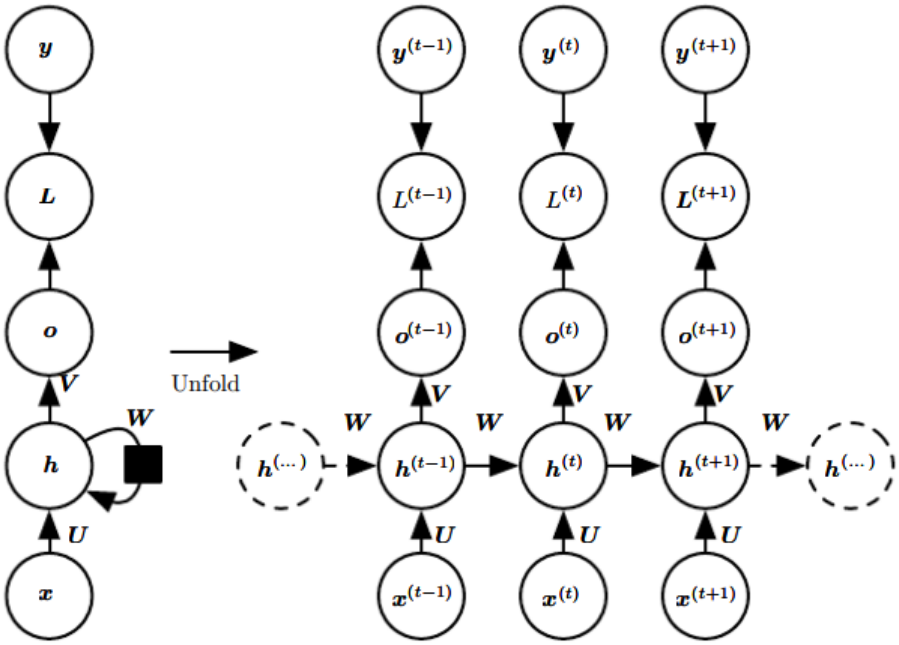


Figure 2.8. *The unfolding recurrent neural network along with time steps [9]*

matrix of the input gate in the t time step, $\boldsymbol{H}_{t-1}$ is the parameter matrix of the hidden states in the $t-1$ time step, the matrix $\boldsymbol{W}_{hi}$ represents the weight matrix between the hidden state and the input gate, $\boldsymbol{b}_i$ is the vector of bias parameter in the input gate. $\boldsymbol{F}_t$ is the value matrix of the forget gate in the t time step, $\boldsymbol{W}_{xf}$ is the weight matrix between the input gate and the forget gate, $\boldsymbol{W}_{hf}$ is the weight matrix between the hidden state and the forget gate, $\boldsymbol{b}_f$ is the vector of bias parameter in the forget gate. $\boldsymbol{O}_t$ is the value matrix of output gate in the t time step, $\boldsymbol{W}_{xo}$ is the weight matrix between the input gate and the output gate, $\boldsymbol{W}_{ho}$ represents the weight matrix between the hidden state and the output gate, $\boldsymbol{b}_o$ is the vector of bias parameter in the forget gate, ϕ is the activation function. The three gate and the hidden states variables use Equation 2.4 to update their values.

$$\boldsymbol{I}_t = \phi(\boldsymbol{X}_t\boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i) \tag{2.4}$$

$$\boldsymbol{F}_t = \phi(\boldsymbol{X}_t\boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$

$$\boldsymbol{O}_t = \phi(\boldsymbol{X}_t\boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o)$$

Bidirectional recurrent neural network

In feed-forward recurrent neural network, it is assumed that the syntactic and semantic information of the current word is determined by the previous appearing words. This indicates that the value of variables in the current time step is determined by the variables in the earlier time steps when the computational graph of recurrent neural networks is

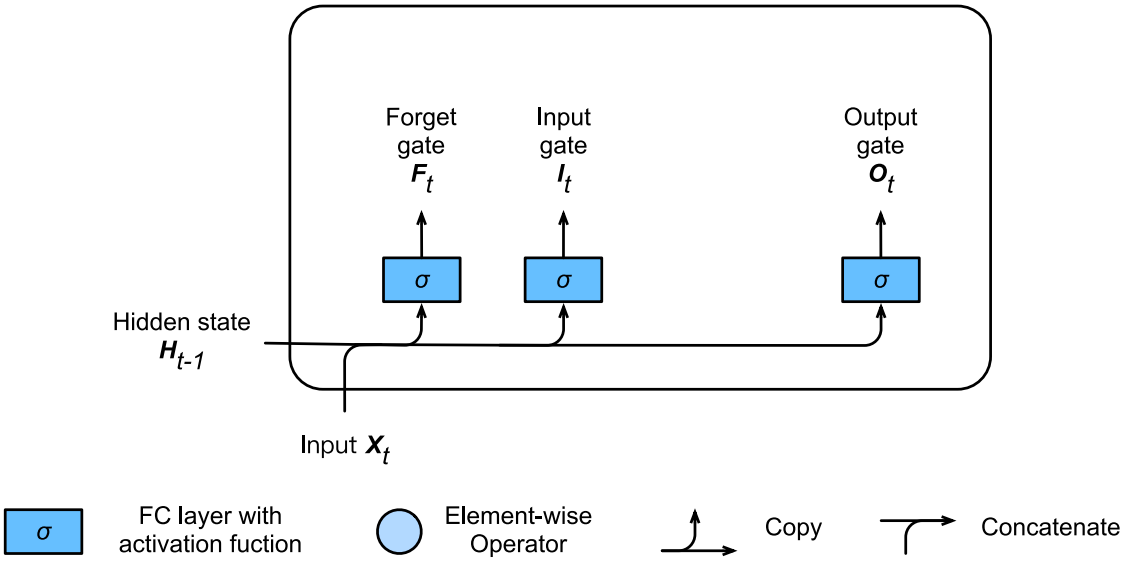


Figure 2.9. *LSTM cell structure [56]*

unfolding along with the time steps. However, the syntactic and semantic information of a word is determined by both the earlier appearing words and the later appearing words in many situations. Context information is an important factor for deep neural networks in 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 [44] was proposed to tackle this type of problem. Figure 2.10 illustrates the structure of a bidirectional recurrent neural network. In this figure, the hidden state for the forward direction in the t time step is $\overrightarrow{\boldsymbol{H}}_t \in \mathbb{R}^{n \times h}$, the hidden state for the backward direction in the t time step is $\overleftarrow{\boldsymbol{H}}_t \in \mathbb{R}^{n \times h}$, n is the number of samples which are inputted to the model, h is the number of hidden neurons, $\boldsymbol{X}_t$ is the matrix of the input value in the t time step, (f) indicates the forward direction, (b) indicates the backward direction in the bidirectional recurrent neural network, ϕ is the activation function, $\boldsymbol{W}_{hh}^{(f)}$ is the weight matrix between the hidden neurons in the forward direction, $\boldsymbol{W}_{xh}^{(f)}$ is the weight matrix between the input and the hidden neurons in the forward direction, $\overrightarrow{\boldsymbol{H}}_{t-1}$ is the parameter matrix of the hidden neurons in the $t-1$ time step in the forward direction, $\boldsymbol{b}_h^{(f)}$ is bias parameters in the hidden states in the forward direction. Regarding the backward direction, $\boldsymbol{W}_{xh}^{(b)}$ is the weight matrix between the input and the hidden neurons in the backward direction, $\overleftarrow{\boldsymbol{H}}_{t+1}$ is the parameter matrix of the hidden neurons in the $t+1$ time step in the backward direction, $\boldsymbol{W}_{hh}^{(b)}$ is the weight matrix between the hidden neurons in the backward direction, $\boldsymbol{b}_h^{(b)}$ is the bias parameters in the hidden neurons in the backward direction. The forward and backward hidden states are computed using Equation 2.5.

$$\overrightarrow{\boldsymbol{H}}_t = \phi(\boldsymbol{X}_t \boldsymbol{W}_{xh}^{(f)} + \overrightarrow{\boldsymbol{H}}_{t-1} \boldsymbol{W}_{hh}^{(f)} + \boldsymbol{b}_h^{(f)}) \tag{2.5}$$

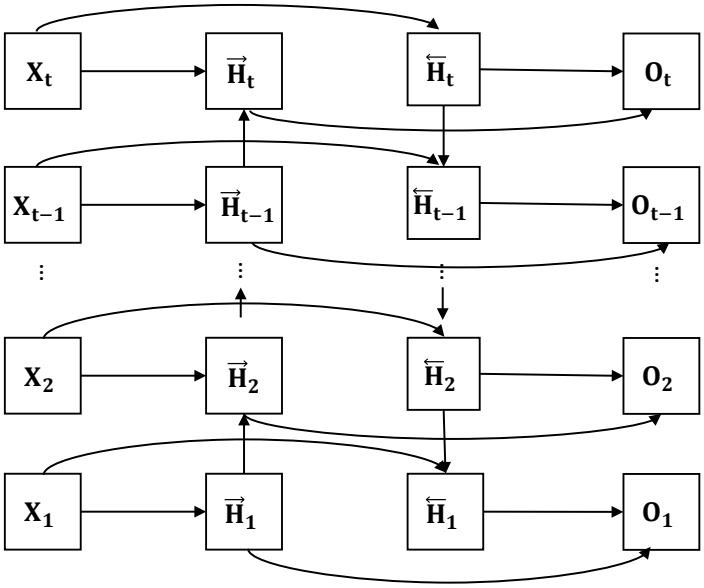


Figure 2.10. *The structure of the bidirectional recurrent neural network*

$$\overleftarrow{\boldsymbol{H}}_t = \phi(\boldsymbol{X}_t \boldsymbol{W}_{xh}^{(b)} + \overleftarrow{\boldsymbol{H}}_{t+1} \boldsymbol{W}_{hh}^{(b)} + \boldsymbol{b}_h^{(b)})$$

The forward and backward hidden states, $\overrightarrow{\boldsymbol{H}}_t$, $\overleftarrow{\boldsymbol{H}}_t$, are concatenated to obtain the hidden state $\boldsymbol{H}_t \in \mathbb{R}^{n \times 2h}$ in the t time step. The hidden state $\boldsymbol{H}_t \in \mathbb{R}^{n \times 2h}$ is inputted to the output layer, $\boldsymbol{W}_{hq}$ is the parameter matrix in the output layer, $\boldsymbol{b}_q$ is the vector of bias parameter in the output layer. The output layer computes the output $\boldsymbol{O}_t \in \mathbb{R}^{n \times q}$ with using Equation 2.6, q is the number of outputs.

$$\boldsymbol{O}_t = \boldsymbol{H}_t \boldsymbol{W}_{hq} + \boldsymbol{b}_q \tag{2.6}$$

2.1.4 Optimization algorithms

Objective function

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 decrease 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 is also named loss function in deep neural networks. The value of the objective function is the average value of the calculated losses when the data from the training data set is loaded to train a DNN model. The common form of objective function of a neural network is defined in Equation 2.7, $f(\boldsymbol{x}_i)$ is the loss function for the $\boldsymbol{x}_i$ sample in the training data set, n is the number of training samples, $f(\boldsymbol{x})$ is the objective function

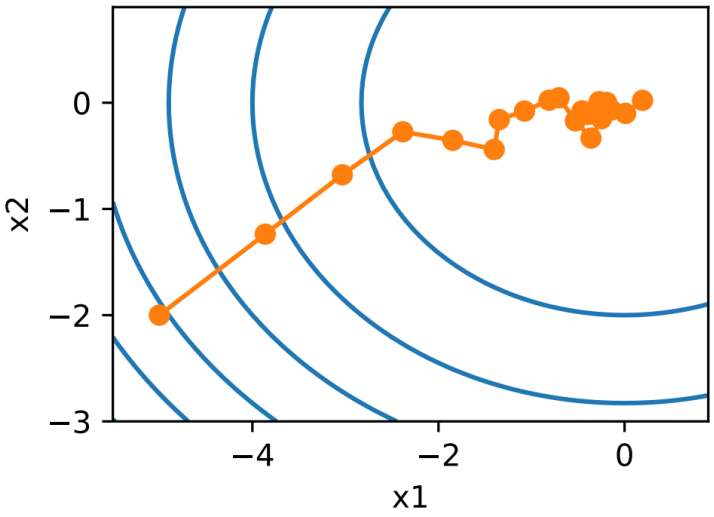


Figure 2.11. *Stochastic gradient descent optimization algorithm for the function* $f(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 2\boldsymbol{x}_2^2$ *[56]*

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 $\boldsymbol{x}$ in layers of DNN models with using Equation 2.8, where $\nabla f(\boldsymbol{x})$ is the gradient of x with function f.

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) \tag{2.7}$$

$$\nabla f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) \tag{2.8}$$

Stochastic gradient descent

During training processing, the optimization algorithms are critical for training deep neural networks since the training phase usually takes a few days or longer time to complete. Therefore, the optimization algorithm has an obvious effect on the cost time for the training. There are many optimization algorithms such as Stochastic Gradient Descent (SGD) [4], Adam [22]. In this thesis, SGD is used as the optimization algorithm.

A demonstration of applying SGD is shown in Figure 2.11. The input is a two-dimensional vector, $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$, the objective function is $f(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 2\boldsymbol{x}_2^2$. SGD optimization algorithm is used to find the minimal value for this function. The initial position for $\boldsymbol{x}$ is $(-5, -2)$, the iteration for updating $\boldsymbol{x}$ is 20 times. After the 20 times iteration, the function $f(\boldsymbol{x})$ finds the minimum point $(0, 0)$.

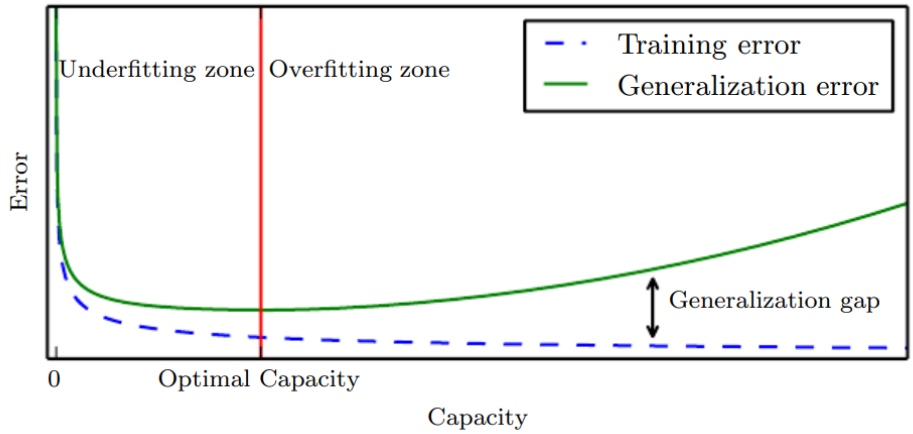


Figure 2.12. *Overfitting in deep neural networks [9]*

2.1.5 Back-propagation

Back-propagation [42] is used to compute the gradient in different layers in deep neural networks from the cost function. The information from the loss function is flowed backwards through back-propagation. Back-propagation exploits the derivative chain rule to compute derivatives. The data is inputted to feed forward propagation to update the value through different layers in DNN model and the value of the loss function is calculated later. When back-propagation is applied in recurrent neural network, the normal form of the back-propagation can not be applied directly since RNN is time sequence model. Therefore, back-propagation through time (BPTT) [54] was introduced for solving this problem. BPTT algorithm is the variant of back-propagation to compute the gradient. The recurrent neural network is required to expand along with the time steps in BPTT to obtain the dependencies in different time steps.

2.1.6 Regularization technique

Overfitting problem

Deep neural networks have strong representation learning ability. In the other hand, 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, which results in the bad prediction ability for test data even though the model achieves high performance in training data or validation data. Overfitting occurs when the gap between the training error and testing error is large. The illustration of the overfitting problem is shown in Figure 2.12. 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 data patterns.

There are many regularization techniques to improve the generalization ability for deep

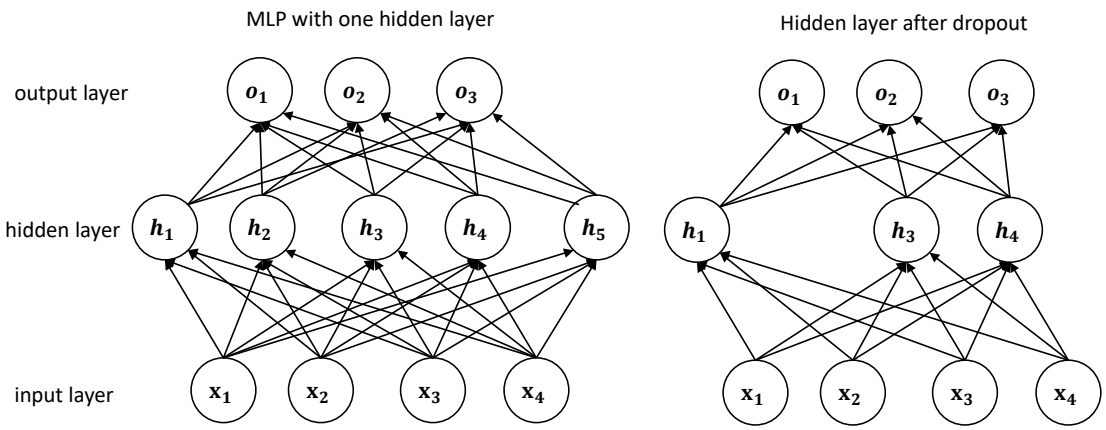


Figure 2.13. *The structure comparison of MLP with and without dropout technique*

neural networks. The common regularization methods include dropout, batch normalization, parameter norm penalties, early stopping mechanism, multitask learning technique etc.

Dropout

Dropout is a regularization technique to deal with the overfitting problem for DNN models. Dropout [46] technique is considered as a way of approximately combining many different neural networks together in an efficient way. Dropout is explained to drop out neurons in a hidden layer in a neural network. Dropping neural neurons out indicates temporarily removing the neurons and their corresponding incoming and outgoing connections in the neural network. The concept of dropout is illustrated in Figure 2.13. When the dropout technique is applied to the hidden layers in the multi-layer perceptron, there is a probability for neurons in the hidden layer to be skipped.

In [46], the authors experimented that convolutional neural network with dropout technique in fully connected layers had 14.32 % error rate in the CIFAR-10 dataset [25] while the original convolutional neural network without applying dropout contained 14.98 % error rate in the CIFAR-10 dataset. The experiments also showed that the model with applying dropout technique reduced the error rate from 43.48 % to 37.20 % in the CIFAR-100 dataset. The authors tested the dropout technique in the ImageNet dataset as well. The best method based on standard vision features achieved 26 % top-5 error rate in the ImageNet dataset at the time. However, the convolutional neural network with applying dropout achieved 16% top -5 test error. These experiments indicated that the dropout technique can improve the generalization ability for deep neural networks.

Batch normalization

Batch normalization [16] is another wide applied technique to improve the generalization ability of DNN. Batch normalization operation is usually inserted after fully connected layers or convolutional layers and before the activation functions. Through the batch normalization operation, the weight values of different layers in neural networks are normalized

with following 4 steps. The mini-batch inputs are $\mathcal{C} = [x_1, ...x_n]$, n is the number of inputs which are applied to activation functions. The first step of applying batch normalization is to calculate the mean value on the mini-batch inputs with using Equation 2.9, where $\mu_\mathcal{C}$ is the mean value on $\mathcal{C}$. Secondly, the variance of the mini-batch inputs is computed using Equation 2.10, where $\sigma^2_\mathcal{C}$ is the variance value on $\mathcal{C}$. The third step is to apply the normalization with using Equation 2.11, ϵ is a constant for numerical stability, the $\widehat{x}_i$ is the normalized value for the x_i. The fourth step is using the Equation 2.12 to scale and shift the variable $\widehat{x}_i$, where y_i is the final batch normalized value of x_i. γ and β are two learnable hyper-parameters.

$$\mu_\mathcal{C} \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.9}$$

$$\sigma^2_\mathcal{C} \leftarrow \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu_\mathcal{C})^2 \tag{2.10}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{C}}{\sqrt{\sigma^2_\mathcal{C} + \epsilon}} \tag{2.11}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BatchNormalization}_{\gamma,\beta}(x_i) \tag{2.12}$$

In [16], the authors experimented that the effect of batch normalization in the Inception deep neural network [48]. Their experiments showed that the Inception variant with batch normalization achieved 72.7% accuracy while the basic Inception model had 72.2% accuracy in the ImageNet dataset, which demonstrated that the performance of the Inception V3 model [49] was improved by 0.5% accuracy through the batch normalization technique in the ImageNet dataset. Furthermore, the training time for the Inception model was reduced when the batch normalization technique was applied according to their experiments result. A explanation for consuming less training time was that batch normalization can boost the speed of converging for training deep neural networks.

2.2 Natural language processing

2.2.1 Sentiment classification introduction

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

Table 2.1. *Examples of text review for sentiment classification*

examples	**sentiment**
“The strange thing is that it works”	positive
“Shattered image is not complex, it’s just boring and stupid”	negative
“Far from bewitching, the patience can be tested by the crucible”	neutral

neutral, positive or very positive. Table 2.1 shows examples of sentiment classification on reviews.

Use cases for sentiment classification

Sentiment classification has been widely applied in many areas with several applications, such as search engine, text information retrieval, machine reading and understanding etc. Three use cases for applying text classification methods are briefly described.

One example is the news category classification task, which text news related to different topics needs to be classified to their correct corresponding topics. There are many different topics including sports, finance, technology, health etc. Each topic is considered as one class. Sports news should be predicted 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 in 2017. The number of negative sentimental 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 [30] [29]. 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. In [32], authors introduced probability-based sentiment classification technique to classify reviews which were collected from mobile phone applications in Apple app store and Google play store. These reviews were classified to four different categories: feature requests, text rating, user experiences and bug reports. Sentiment classification can be applied in the politics area as well. When a new policy is proposed, the government can gather people’s opinion and discussion content from multiple sources to obtain feedback on the new policy. then the government can improve or adjust the policy in the next step.

2.2.2 Word embedding

Word embedding is a type of algorithms which map words or phrases from vocabulary to vectors in numeric values form. Each word is represented by a vector with using word

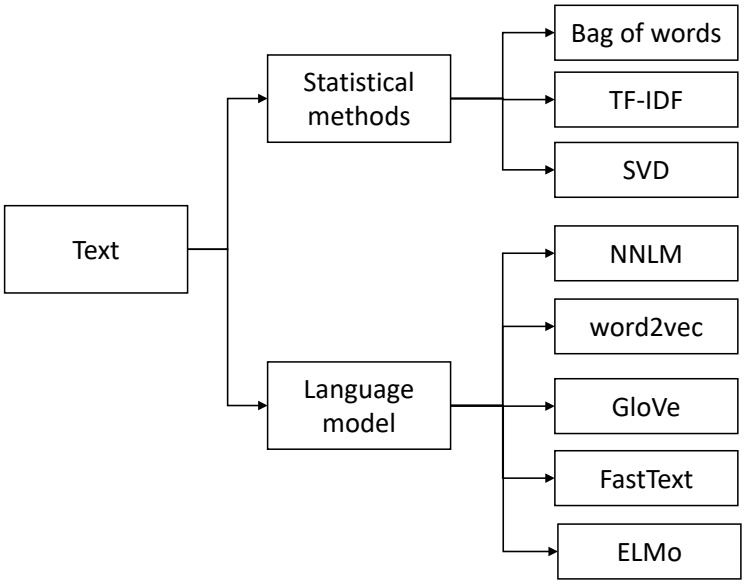


Figure 2.14. *Overview of word embedding technique*

embedding technique. The similarity of different words can be measured by the distance metrics such as calculating cosine similarity. The overview of word embedding technique is shown in the Figure 2.14, the bag of words method is a traditional word embedding technique. The Term Frequency-Inverse Document Frequency (TF-IDF) algorithm and Singular Value Decomposition (SVD) are another two typical statistical methods for generating word vectors. The language model based word embedding methods consider the semantic context information and have many advantages When compared to statistical word embedding technique. For example, the curse of high dimensionality and vector sparsity problem can be mitigated by utilizing the distributed word embedding technique. There are various algorithms to generate word embedding vectors for words in corpus. One knowing work is the Neural Network Language Model (NNLM) [2]. Matthew et al. [39] proposed the Embeddings from Language Models (ELMo) to represent word embedding in 2018. In this thesis, three different algorithms are elaborated. The first one is the Word2vec, the second one is the FastText and the third word algorithm is the GloVe.

Word2vec

The Word2vec [35] algorithm was first introduced in 2013. The Continuous Bag of Words (CBOW) and the Skipgram are the two variant models for computing word vectors. The Skipgram model predicts a target word with utilizing the nearby appearing words. In contrast, the CBOW model predicts the target word according to the surrounding context. The context is represented with using bag of words method which are contained in a specified 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 predicts the target word with using the information of all the surrounding words, including "selling", "these", "leather", and "jackets". The sum of these word vectors is applied to

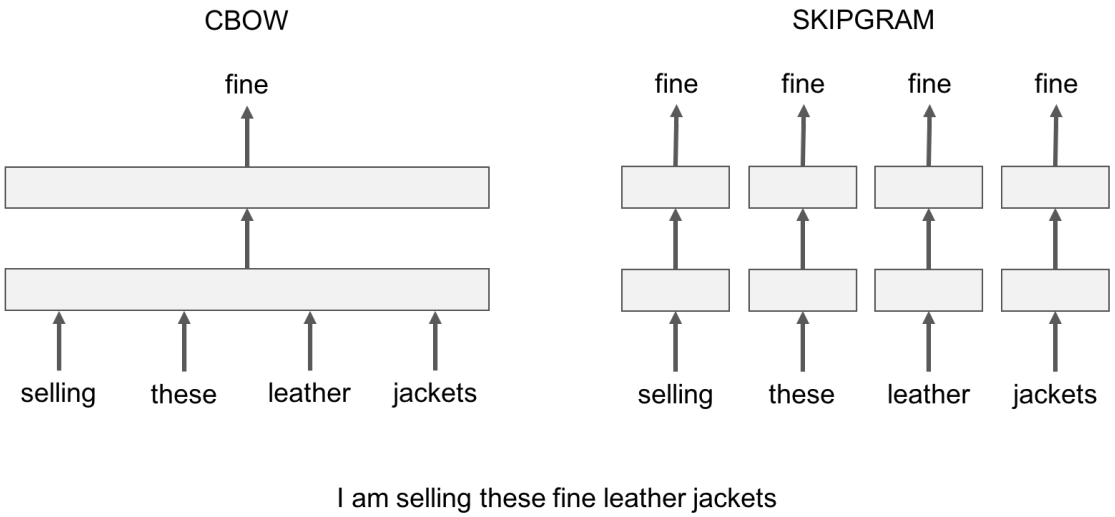


Figure 2.15. *The continuous-bag-of-words versus the skipgram method [14]*

predict the target word in the CBOW model. The Skipgram model takes a random near word instead of all words to predict the target word. In addition, the Skipgram model had better performance than the CBOW model in general according to the result of the experiments.

A demonstration of the effectiveness of wording embedding is illustrated in Figure 2.16. Two categories of English words are listed in this figure. The first words category is the countries, including "France", "Germany", "Spain" etc. The second category is the corresponding capital cities, including "Paris", "Berlin", "Madrid" etc. The word embedding for these English words was trained using the Skipgram model. The 300 dimensional word vectors for these English words were generated after finishing the training process. The Principal Component Analysis (PCA) [18] technique was applied to reduce the dimension size of word vectors from 300 to 2. These two values were treated as the coordinates for words to perform visualization in the figure. It is seen that the English words of these two concepts were clustered separately on each side. The relationship of matching words (countries and capital cities) between these two categories was learned, which indicates that the syntactic and semantic relationship of these words was kept in the word embedding form.

GloVe

GloVe [38] is a popular deep neural network based word embedding algorithm. It is a unsupervised learning algorithm for learning word vectors. GloVe was proposed after the Word2vec and had a few changes compared to the Skipgram model. The first change is that the GloVe algorithm adopted square loss instead of cross-entropy loss as the objective function to train model for generating the word embedding. Secondly, the GloVe algorithm considers the global statistics information of the English words based on the whole dataset while Word2vec algorithm only considers the information inside the fix size windows.

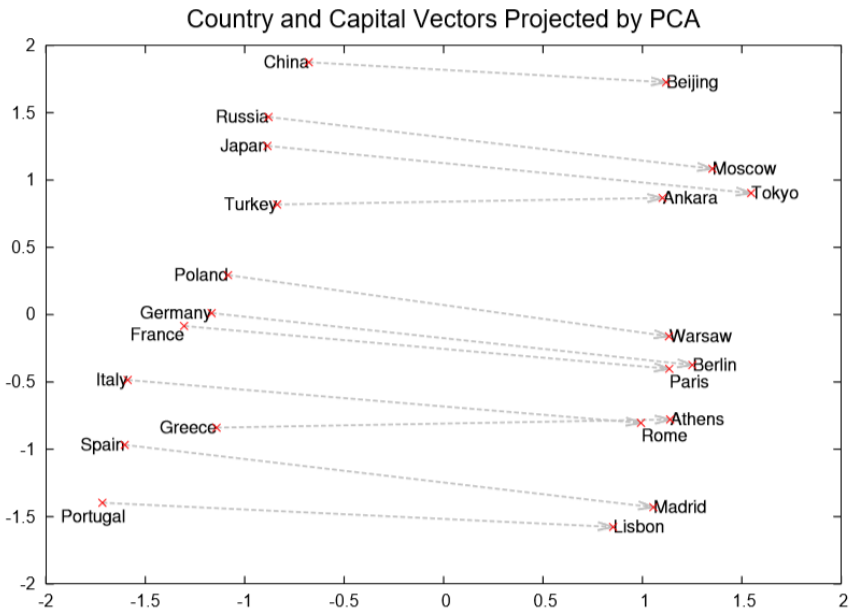


Figure 2.16. *Two-dimensional principal component analysis projection of the word vectors of countries and corresponding capital cities [35]*

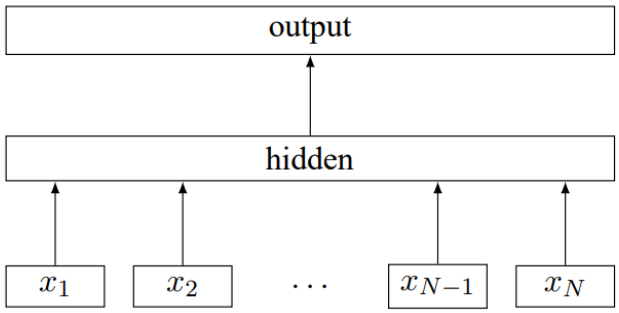


Figure 2.17. *FastText model architecture for generating the vector representation for a sentence with ngram features $x_1, \ldots, x_N$ [19]*

FastText

The FastText [19] is another common neural network model to generate accurate word vectors. A feature of the FastText model is that the training speed is fast. In the author's experiments, FastText model only needed 4 seconds to complete one epoch to perform sentiment classification on the Yelp Full review dataset. In contrast, 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 architecture of the FastText was simple, which contains one layer for the word embedding, one hidden layer and one layer for output.

GloVe and FastText are two standard word embedding technique. The comparison for these two algorithms is described in Table 2.2. The training corpus for GloVe are Wikipedia and Gigaword, 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

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

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

for word vectors is the dimensional size. A factor of the syntactic and semantic information of a word is the dimensional size. In [38], the experiment showed that the accuracy on the word analogy task was higher when the dimensional size of a word is larger. Normally, the 50-dimensional, 100-dimensional and 300-dimensional word embedding size are common. In this thesis, the 300 dimensional size for the word embedding was experimented for sentiment classification.

2.2.3 Overview of algorithms for sentiment classification

Zhang et al. [57] proposed a survey on the evolution of the 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 sentiment classification and analyzed why it is a difficult task. Many efforts have been devoted to exploring the sentiment classification from different perspectives. For example, the new text review dataset are introduced and the sentiment classification algorithms are proposed.

A perspective of categorizing sentiment classification algorithms is considering the length and form of reviews. Sentiment classification algorithms are divided into three small research direction from this perspective: the sentence-level sentiment classification algorithms, the paragraph-level sentiment classification algorithm, and the document-level sentiment classification algorithm. In the sentence-level sentiment classification, a whole sentence is treated as the input, the target is to predict the sentimental polarity of the sentence. In the paragraph-level sentiment classification, the vector representation of a paragraph is generated through integrating the sentence vectors, then the sentimental class for the paragraph is predicted. In 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. The paragraph vector is generated by integrating all paragraph vectors afterwards. Moreover, the evolution of text classification algorithms is commonly divided into three main approaches from the perspective of algorithms, linguistic rule-based approach, statistical machine learning approach, and deep neural network-based approach.

The linguistic rule-based algorithm is the earliest applied method to perform sentiment classification task. The linguistic rule-based method utilizes lexicon dictionary to classify the sentiment of text. In this approach, the sentimental words 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. A

representative algorithm of this method is the VADER [15].

The general process of applying statistical machine learning based algorithms for sentiment classification contains two steps. The first step is to extract handcrafted features from the text. The second step is to use statistical machine learning classifiers to predict the sentimental class for the specific text. There are many representative works. The common algorithms include Linear Regression (LR), support vector machine [37], Naive Bayes (NB) [41], hidden markov models [40], conditional random fields [47], latent dirichlet allocation [3] etc.

In [37], the authors applied SVM to classify the sentiment of text reviews instead of using topic based sentiment classification methods. The authors' experimented the unigrams and bigrams features, and the results showed that the SVM with unigrams algorithm achieved the state-of-art performance, 82.9% test accuracy on the internet movie database. In [8], the authors proposed a method which classified a term as positive or negative with utilizing the distribution of the frequency count and proportional presence count. Moreover, Konstantinas et al. [23] introduced a new method, which combined the prediction results of SVM and NB classifiers to improve classification performance for sentiment classification.

Deep neural network based algorithms have achieved the state-of-the-art performance in the sentiment classification 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 performing the sentiment classification task, which integrate the feature extraction and classification steps into a end-to-end processing.

There are many well-known deep neural networks for the text classification task. For example, Kim et al. [21] proposed the Text-CNN model, which applied convolutional neural network on the text. In 2013, Socher et al. [45] proposed a recursive deep neural model for text classification. The recursive deep neural network obtained the highest accuracy in the many text review datasets at the time and it achieved 80.7% accuracy on the fine-grained movie sentiment classification. In 2015, Zhang [59] proposed the character-level convolutional network to perform the text classification. The character-level convolutional network was the first model to use character instead of word to classify text. Previously, deep neural networks use word-level text input, the authors' experiment showed that the recurrent neural network had a strong learning representation ability for both characters and words.

2.3 Assessment criteria

There are many metrics to evaluate algorithms for different tasks in NLP area. For example, a common assessment criteria for the machine translation task is the perplexity criteria [1], which is the value of probability for generating one sentence. The aim of train-

ing models in the machine translation task is to reduce the value of perplexity. Another example is the BERTScore [58]. The BERTScore was proposed to compute cosine similarity score to match words between the candidate and references sentence, it can be used to evaluate algorithms in the image captioning and machine translation tasks.

To access the performance of algorithms for text classification, four critical metrics are calculated in this thesis, namely precision, recall, accuracy and F1-score. In addition, the confusion matrix is needed to obtain better understanding of the prediction result produced by three DNN models. The brief description of these metrics is described below.

2.3.1 Accuracy and F1-score

Accuracy

The mathematical function for the accuracy metric is given in Equation 2.13.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

- True Positive (TP), it indicates the classifier predicts 1 where the true class is 1.
- False Positive (FP), it indicates the classifier predicts 1 where the true class is 0.
- True Negative (TN), it indicates the classifier predicts 0 where the true class is 0.
- False Negative (FN), it indicates the classifier predicts 0 where the true class is 1.

F1-score

A common evaluation method to compare multi-class classification algorithms is to use precision and recall metrics. The mathematical form of precision is given in Equation 2.14. The definition of recall is given in Equation 2.15.

$$precision = \frac{TP}{TP + FP} \tag{2.14}$$

$$recall = \frac{TP}{TP + TN} \tag{2.15}$$

Precision is also named positive predictive value. Recall is also named sensitivity. 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.

The F1-score is the harmonic mean of precision and recall. The equation of the F1-score

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)

is formulated in Equation 2.16. The F1-score is described as a weighted average value of the precision and recall. The best value of the F1-score is 1 and the worst 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.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.16}$$

2.3.2 Confusion matrix

The confusion matrix is used as a method to visualize the classification result. For the binary classification case, true (positive) and false (negative) 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

The whole processes of the experiments were elaborated in this chapter. In the first section, the TripAdvisor dataset is first introduced, including the main process for making this dataset. The statistics overview of the TripAdvisor dataset and the Stanford Sentiment Treebank dataset (SST) is presented as well. In the second section, the architecture of the three deep neural networks is elaborated. The third section is describing the environment of the experiments.

3.1 Datasets

3.1.1 TripAdvisor dataset

Data collection

The TripAdvisor dataset [1], is made for this thesis. The first step for making the TripAdvisor dataset is collecting the hotel reviews data. A Python web crawling framework, Scrapy [24], was used to crawl hotel review data from the TripAdvisor website. There are two critical 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 reviews data is large 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 with not encountering obstacles. The second step is to clean the raw dataset including discarding incomplete hotel reviews. For example, some hotel reviews miss the date of writing the review in the raw data format. The third step for making the dataset 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 considered as the training set, 10% of the dataset was treated as the validation set, and the remaining 20% was the test set.

A original hotel review in the TripAdvisor website is shown in Figure 3.1. It demonstrates that this hotel review was given a 5 score by a user named "000Glenn" in November 2012. The review content of this data record has three separated paragraphs. The title of this review is "Outstanding". The hotel is "Sofitel Legend The Grand Amsterdam". The text

[1] The TripAdvisor dataset can be accessed through this link: `https://drive.google.com/open?id=19TdLOrqjAQ11lpYRx1r2GcYo79fZXloE`

Figure 3.1. *A original hotel review on the TripAdvisor website [50]*

format of this hotel review was saved into the JavaScript Object Notation (JSON) format. The stored JSON data is listed in Table 3.1. It shows that a JSON 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 hotel name, the score rating of the review, reviewing date, the Uniform Resource Locator (URL) of this review, the URL of the hotel, and the review content. To make the text classification dataset, only text reviews and their corresponding classes are needed to keep, other information is dropped. Thus, the "review" and the "score" columns were extracted as the input and class in the experiments.

Overview of the TripAdvisor dataset

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

The distribution of the number of reviews for each class in the TripAdvisor dataset is shown in Figure 3.2. The *very positive* class has the largest number of hotel reviews among all other four classes, which contains more than 40000 data samples. The *positive* class has over 30000 data samples and it is the second largest number of hotel reviews. However, the *very negative* class contains less than 10000 data items. 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 a hotel review for each class in the TripAdvisor dataset is shown in Figure 3.3. It shows that the a hotel review in the *very negative* class has the largest number of the sentences on average, which consists of 8.65 sentences for a review. In contrast, a review in the *very positive* class has the least number of the sentences, which has 6.63 sentences. The NLTK [31] library was used as the library for calculating the number of sentences from the whole text review. The average number of words for each sentimental

Table 3.1. *A sample of the raw hotel review data in the TripAdvisor dataset*

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. It's at the bottom of the Red Light district, but don't let that put you off - this is the heart of the old centre, and the hotel's locality south of the Damstraat bridge which traverses O.Voorburgwal is actually quite peaceful at night. Our room was large and well - appointed with a canal view. The bed, oh the bed. The most comfortable bed I've had the pleasure to sleep in. Hermes toiletries. Spotlessly clean. Service and food are outstanding and what you expect of a hotel of this calibre. Concierge was excellent. In summary, I cannot recommend this hotel highly enough. A hotel which richly deserves it's 5-star rating.

Table 3.2. *The matching relation between the sentimental score and class*

score	**sentiment**	**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 statistics of all sentimental classes in the TripAdvisor dataset*

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

class in the TripAdvisor dataset is shown in the Figure 3.4. It is seen that the average number of the words for a review in the *very negative* class is 176.85 while that in the *very positive* class is 114.02.

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

data	**value**
dataset	TripAdvisor
$\|C\|$	5
$\|V\|$	99686
average number of sentences in a review	7.08
average number of words in a review	128.46
training set	70000
validation set	10000
testing set	20000

3.1.2 Stanford sentiment treebank dataset

The SST dataset [45] is a common dataset for text classification. All reviews in the SST dataset are related to the movie content. There are two different classification tasks for the SST dataset. The first type is the five-way fine-grained classification and the second one is the binary classification task. For the first type, the five class labels are: *very positive*, *positive*, *neutral*, *negative*, and *very negative*. For the second type, it only contains *positive* and *negative* classes. In the experiments, the five-way fine-grained classification type was selected. The overall statistics of the SST dataset is listed in Table 3.6. It is seen that the training set has 8544 data samples, the validation set has 1101 data records and the testing set contains 2210 data records. Five review samples in the SST dataset are listed in Table 3.7. The first column is the movie review, the second column is the sentimental score and the third column is the corresponding class for the sentimental score.

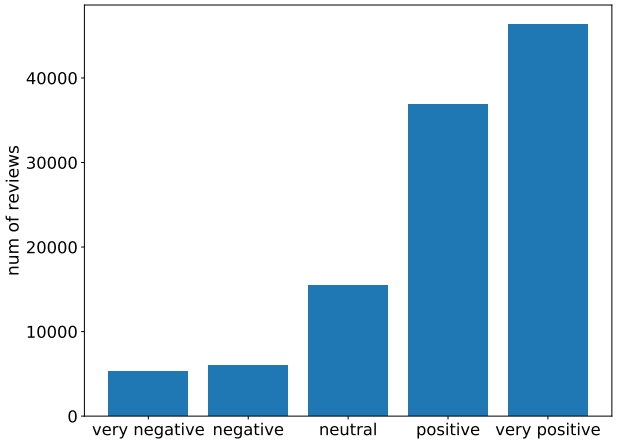


Figure 3.2. *The distribution of the number of reviews for each class in the TripAdvisor dataset*

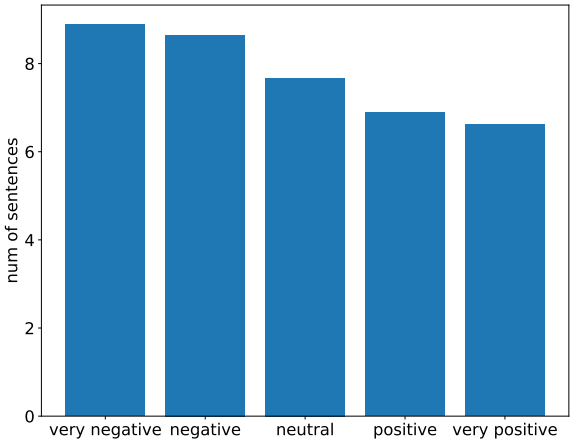


Figure 3.3. *Average number of sentences per class in the TripAdvisor dataset*

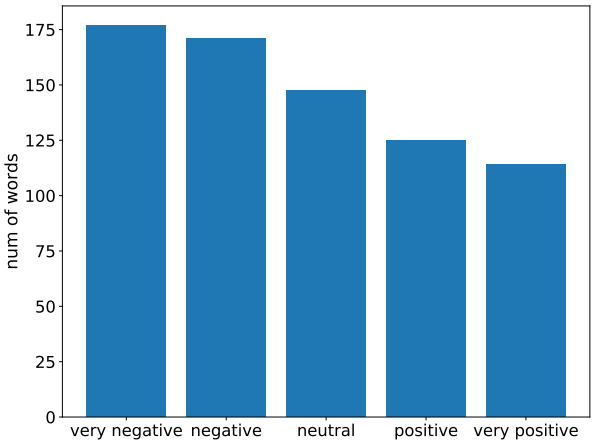


Figure 3.4. *Average number of words per class in the TripAdvisor dataset*

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

hotel reviews	**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

Table 3.6. *Overall statistics of the SST dataset*

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

Table 3.7. *Data examples from the SST dataset*

movie 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

3.2 Deep neural networks for sentiment classification

There are many well-designed deep neural networks [27] [11] [52] for text sentiment classification. Three representative deep neural networks were evaluated for sentiment classification 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 general steps for applying deep neural networks on sentiment classification. The first step is word representation, the second step is feature extraction and the third step is classification. In the first step, the pre-trained word embedding mechanism is applied. The FastText and GloVe pre-trained word embedding technique are tested in the experiments. In the second step, deep neural networks extract feature from the word embedding, the vector representation is generated for the text review. In the third step, fully connected layers are used to classify vectors to obtain the final prediction result. One example of applying deep neural networks for sentiment classification is illustrated in Figure 3.5. From the figure, the review, "I love my new iPhone", was passed through the three steps. The sentimental class expressed from the review is finally predicted.

3.2.1 Text convolutional neural network

The Text-CNN model was proposed in [21]. 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 for text classification. The architecture of the model is shown in Figure 3.6. It illustrates how the Text-CNN model predicts the sentiment of a text review into negative or positive sentimental class. First of all, the review, "I like this movie very much!", was mapped into the word vectors in

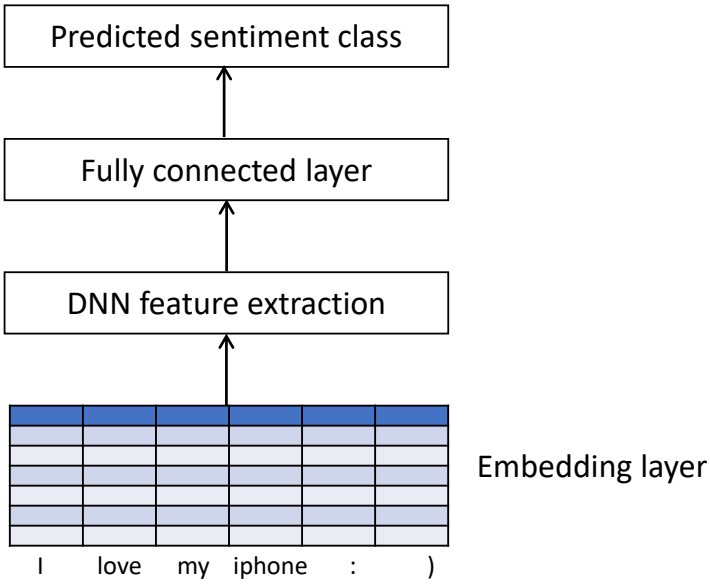


Figure 3.5. *The steps of applying DNN model on sentiment classification*

the embedding layer. The dimension size of the word vector was 5. The whole sentence was converted into a 7×5 matrix in the numeric form. Afterwards, the convolution operation was operated on the matrix, the three kernel sizes of the convolution were 2, 3 and 4. There were 2 filters in each convolutional layer. Each filter was treated as one channel. In the next step, 2 feature maps were generated from the previous two channels and the max-pooling layer was followed. The 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 fully connected layer. In the authors' experiments, the Text-CNN model achieved the highest accuracy with 88.1% in 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 VD-CNN model was proposed in [6]. Firstly, the VD-CNN model operated the input text at character-level instead of words. Secondly, small convolutions with different kernel sizes and pooling operations were utilized in the VD-CNN model for text classification. 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 different length span relations among words in one sentence. Max pooling operation extracted the obvious feature from the windows-size length text while average pooling operation contained the average information from the sentence. In the authors' experiment, the max pooling layer had better performance than the average pooling layer. Thirdly, neural networks were shallow in previous CNN models for text classification, which were up to 6 convolutional layers. In contrast, the

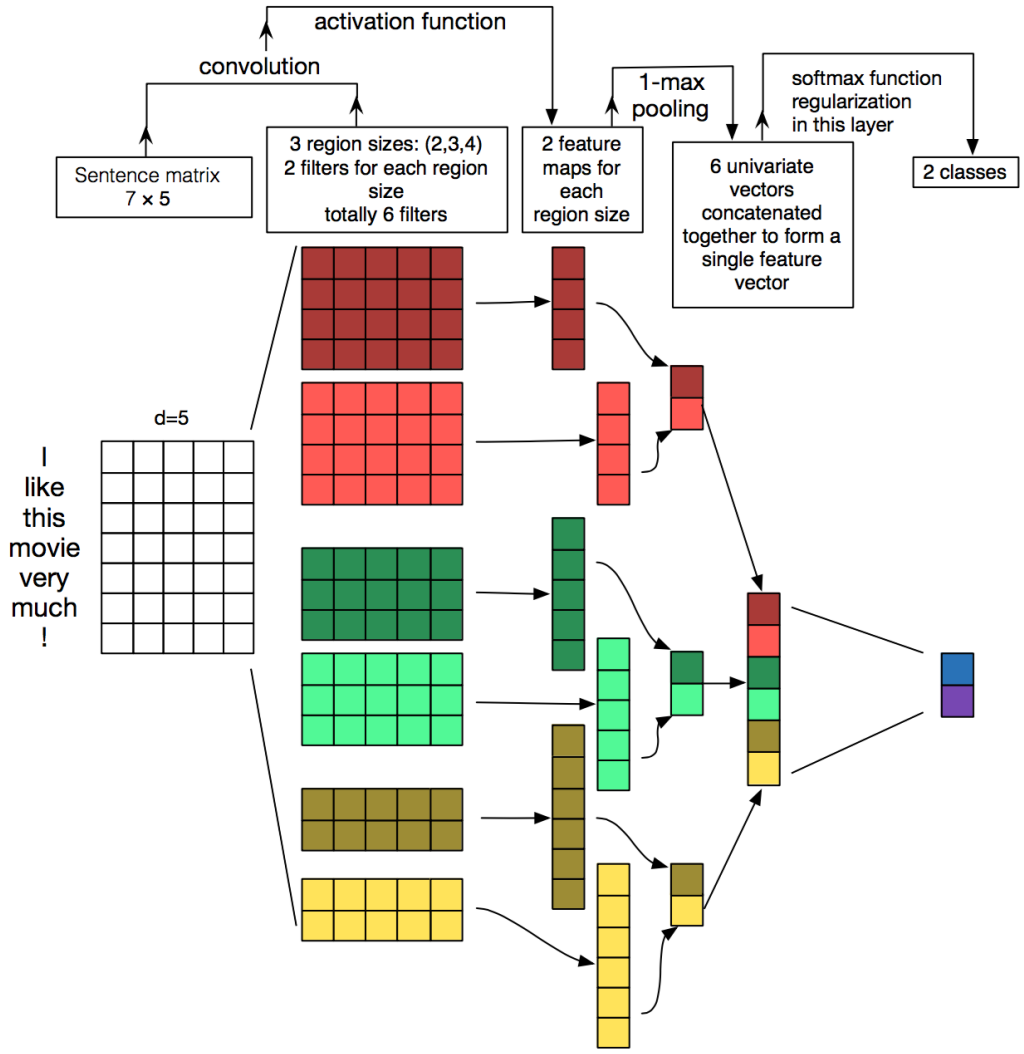


Figure 3.6. *The Text-CNN model architecture [21]*

VD-CNN model was deeper than previous deep neural networks in text classification area at the time. Another 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 model [12]. Shortcut operation was added to mitigate the gradient vanishing problem in the ResNet model. In the VD-CNN model, the shortcut had the same functionality to mitigate the gradient vanishing because of the depth of the model.

The architecture of the VD-CNN model with 29 convolutional layers is shown in Figure 3.7. It shows that the text review was first converted into the word vectors according to the lookup table. A 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. Then one max-pooling layer with the size as 8 was added. The function of the pooling layer was to change the dimension of

the output through the max pooling operation. Three fully connected layers with ReLU activation function were applied in the VD-CNN model. The number of neurons in the last fully connected layer was as the same as the number of outputs, which indicates that the number of neurons depended on the number of classes on different text classification tasks.

The VD-CNN model was tested on several classification tasks, including sentiment classification, topic classification, and news categorization. The authors' experiments showed that the VD-CNN model obtained the state of the art results when compared to other models in the situation of not applying data augmentation technique at the time. Specifically, the VD-CNN model with 29 convolutional layers achieved 35.28% error rate and the VD-CNN model with 9 convolutional layers had 37.63% test error in the Yelp full dataset [2]. In the experiments of 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.

3.2.3 Bidirectional long short term memory neural network

The BiLSTM model [53] combined bidirectional recurrent neural networks and convolutional layers into one model for text classification. There were some work [51] [17] which combined RNN and CNN for solving specific tasks in computer vision area in previous studies. The BiLSTM model 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. The sentence, "video games are more involving than this mess", was demonstrated. First of all, the English words were passed to the embedding layer and they were converted to the word vectors. Two paddings were added on each side to increase the length. Two convolution operations and one max-pooling operation were applied after the embedding layer respectively. The concatenation operation was followed afterwards to generate the sentence encoding. The three fully connected layers were added after the sentence encoding. Then the sentimental class of the sentence was calculated through the softmax activation function. There were a few different variants of the BiLSTM model. For example, the CNN-LSTM-GloVe is one variant which was a model with GloVe pre-trained word vectors, max pooling layer and LSTM cell. Another variant is the CNN-GRU-FastText model. This model used FastText pre-trained word vectors, max pooling layer, and Gate Recurrent Unit (GRU) [5] together. The authors tested the BiLSTM model in the SST1 dataset. The experiments showed that the CNN-LSTM-Word2vec model achieved the best performance with 51.50 % accuracy. The BiLSTM variant model contained 49.1% accuracy when it only utilized recurrent neural network without a combination of the convolutional layer.

[2]The Yelp dataset can be accessed through this link: `http://www.yelp.com/dataset_challenge`

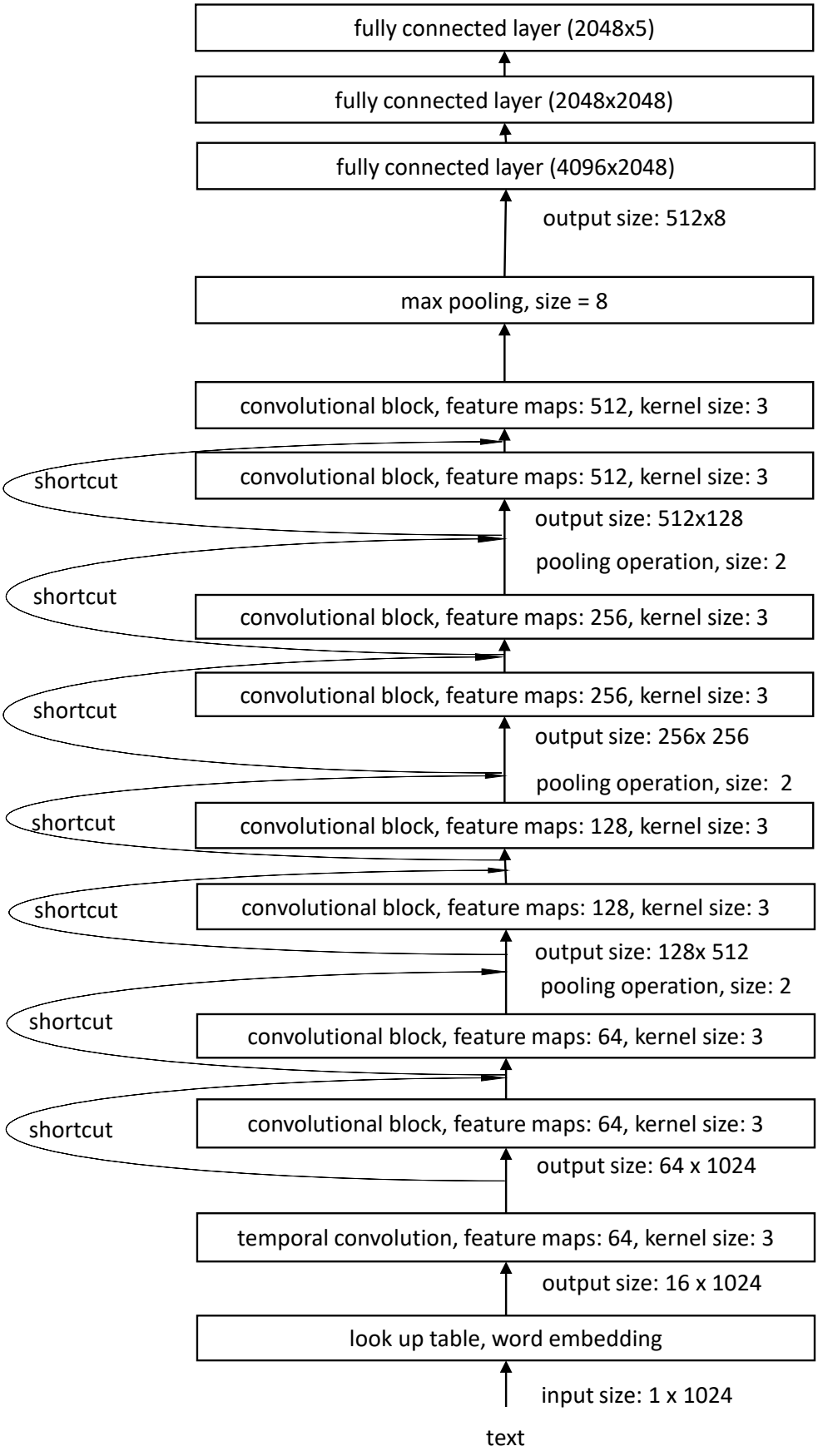


Figure 3.7. *The architecture of the VD-CNN model for text classification*

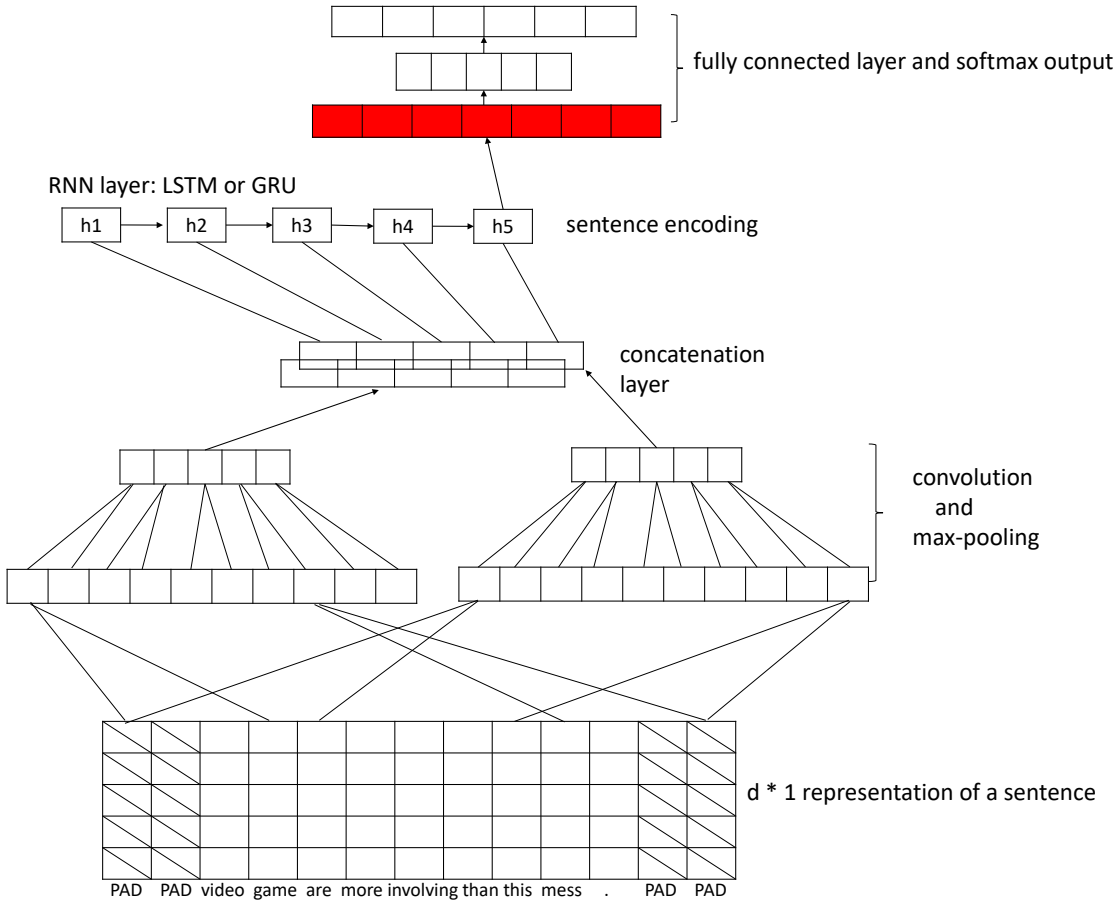


Figure 3.8. *BiLSTM model architecture with using an example sentence*

3.2.4 Loss function for sentiment classification

The loss function for sentiment classification adopts cross-entropy loss. The mathematical formula of the cross-entropy loss for sentiment classification is given in Equation 3.1. The predicted result is denoted by $\hat{y}_c$, where C is the number of classes. In the experiments, the C value is 5, namely *very positive*, *positive*, *neutral*, *negative*, *very negative*. y_c is the one hot vector of the true label. The predicted result is yielded after the softmax activation function. The index of the largest value is selected as the final predicted result. The mathematical form for calculating the probability is given in Equation 3.2, where $\boldsymbol{z}$ is the output vector.

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

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

Distance ranking metric

The formula 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]. $i \in [Text{-}CNN, BiLSTM, VD\text{-}CNN]$. $\hat{y}_i$ is the predicted class, y_i is the actual class. When a review has the high *distance* value, it indicates that the predicted sentiment result was misclassified by most deep neural networks. When the *distance* value is 12, it is the worst prediction result, which indicates that all three models predict the sentiment of a review into the opposite value. For example, the true sentiment class of a review is 5, the *very positive* sentiment. These three DNN models predicted the sentiment of the review as 1, the *very negative* sentiment. The *distance* value in this situation is 12. If the *distance* value is 0, the result indicates that the review was correctly classified by all three deep neural networks.

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

3.3 Experiments

3.3.1 Data preprocessing

For the data preprocessing, the word input length of a review is set as 1024. If the length is less than 1024, the padding is added to the review to increase its length. If the length is over 1024, the review is truncated into 1024 words. For the preprocessing of unknown characters, the English word or character is deleted if it is not included in the following dictionary. The dictionary is a total of 69 tokens and it is shown below. For example, if the word in the review contains some characters which are not English but other languages such as Russian or Finnish, the word is deleted.

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

3.3.2 Experimental environment

The experimental environment includes many factors such as the operating system, deep learning framework etc. The experiment results depend on the environment. For example, the training and testing time for deep neural networks can have different values on different computer machines. The detailed environment for the experiments is listed in Table 3.8. The operating system is the Ubuntu 18.04, Python was selected as the programming language. The code [3] implementation was completed on PyTorch [20] framework. All experiments were performed on a single Nvidia TITAN Xp GPU.

[3] The source code of this thesis can be accessed through this link. `https://github.com/yipersevere/Deep-neural-network-based-algorithms-Performance-Comparison-for-Sentiment-Classification`

PyTorch is a universal 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, researchers can easily debug the neural network source code to check the computational graph of a deep neural network. In the graph mode, deep neural networks are optimized for deployment 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 NLP application, there is the torchtext library. The torchtext library provides pre-trained GloVe and FastText word vectors.

Table 3.8. *Configuration of the experiments*

experimental environment	**tools**
programming language	Python 3.5
operating system	Ubuntu 18.04
PyTorch	1.0.0
torchtext	0.4.0
GPU	Nvidia TITAN xp
Compute Unified Device Architecture (CUDA) library	9.0
memory	32 GB

4 EVALUATIONS

4.1 Experimental results

Model training

The three deep neural networks were trained using SGD as the optimization algorithm. The setting of the model training hyper-parameters is listed in Table 4.1. Regarding the regularization technique applied in the three models, the dropout was adopted in the last fully connected layer and the dropout rate was set to 0.5 to prevent the overfitting problem when training models 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. The three models were trained to minimize the cross-entropy loss. The three models were trained for 60 epochs. The early stop mechanism was applied when training models. The training processing was terminated after the loss was not improved for over 10 epochs. 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. In the backward propagation, the loss was computed and the loss information was feed backward to the different layers in the model to update the value of parameters.

Table 4.1. *The setting of training hyper-parameters*

model hyper-parameters	**value**
batch size	32
word embedding dimension	300
dropout rate	0.5
initial learning rate	0.01
epochs	60

The number of parameters of the three models and the costing time for one epoch are listed in Table 4.2. All values were calculated under the same experimental environment. It shows that the VD-CNN model has 17 million parameters, which is the largest number of parameters. The BiLSTM model has the least number of the parameters, 3.5 million. The Text-CNN model has almost as the same amount of parameters as the BiLSTM. Regarding the time of one epoch for training and validation, the VD-CNN model needed

to take 619.5 seconds, the BiLSTM model cost 249.0 seconds the second longest time. The Text-CNN model consumed only 69.6 seconds, which is the least time among the three models. A reason for the longest costing time by the VD-CNN model is that the VD-CNN model is a large model and contains the largest number of parameters. Thus, the VD-CNN model cost a longer time than other models for training. In addition, the BiLSTM model cost around 4 times longer time than the Text-CNN model to complete one epoch. RNN can not utilize parallel computing because of the time dependency of text sequence data. This feature results in that RNN has low efficiency on the usage of GPU when compared to CNN.

Table 4.2. *The three models in training phase*

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

Experimental results

The experimental results for the three DNN models are described in Table 4.3. The performance of these three DNN models was compared in the TripAdvisor dataset and the SST dataset. The effect of word embedding technique (FastText and GloVe) for initializing word vectors was investigated among the three models in these two datasets as well.

The Table 4.3 shows that the BiLSTM model outperformed the other two methods in 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. In addition, the Text-CNN model and BiLSTM model had 0.68 F1-score in the TripAdvisor test dataset, the VD-CNN model achieved 0.67 F1-score in the TripAdvisor dataset, which was the lowest F1-score among all three models.

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

The three DNN models were tested in the 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 26.97% accuracy with using FastText word embedding and 25.58% accuracy with using GloVe word embedding. The BiLSTM model

contained 36.35% the highest accuracy in the SST dataset.

Table 4.3. *The performance of the three models evaluated in 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 investigated in the experiments as well. The confusion matrices of these three DNN models were calculated. These confusion matrices were calculated with using GloVe instead of FastText word embedding, and these confusion matrices were calculated in the TripAdvisor dataset instead of the SST dataset.

The visualization result of the confusion matrix for the Text-CNN model is shown in Figure 4.1. The confusion matrix of the VD-CNN model is shown in Figure 4.2. For the BiLSTM model, the confusion matrix of the experimental result is shown in Figure 4.3. It is seen that the *negative* class contained the largest number of misclassified hotel reviews in all these three models. Specifically, 69% of the *negative* class reviews were wrongly predicted to other classes for the Text-CNN model, 61% of the negative class reviews were misclassified for the VD-CNN mode and 59% of the *negative* class reviews were misclassified for the BiLSTM model. In addition, for the neutral sentimental class reviews in the TripAdvisor dataset, 35% of these reviews were wrongly classified as the positive class in the Text-CNN model. Both the VD-CNN model and the BiLSTM model mislcssified 30% out of the *neutral* sentimental reviews into the positive class in the TripAdvisor test set. This indicated hotel reviews are more prompted to misclassify the hotel review toward the positive polarity sentiment than the negative polarity sentiment. One possible explanation is the unbalance of the sentimental classes in the TripAdvisor training dataset. Furthermore, the *very positive* sentiment class achieved the highest classification accuracy when compared to the other four sentiment classes.

4.2 Discussion

The reasons for misclassifying hotel reviews in the TripAdvisor dataset by the three DNN models were analyzed. Table 4.4 presents the top 5 misclassified reviews in the TripAdvisor test dataset by the three models according to the descending order of the $distance$ metric. The analysis of the misclassified hotel reviews on Table 4.4 indicated that long

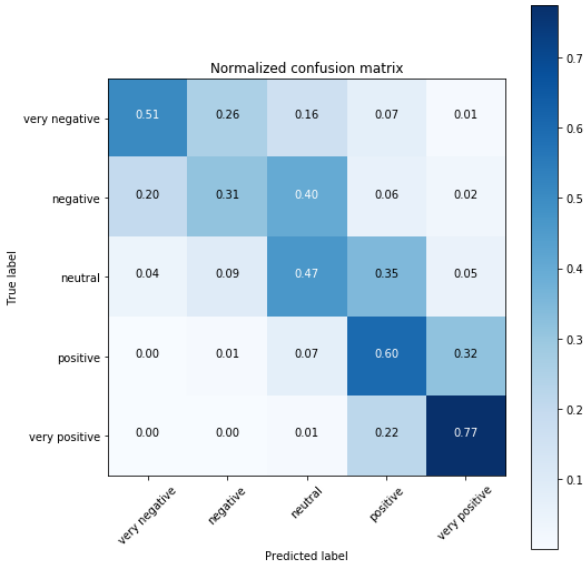


Figure 4.1. *Confusion matrix of the Text-CNN model in the TripAdvisor test dataset*

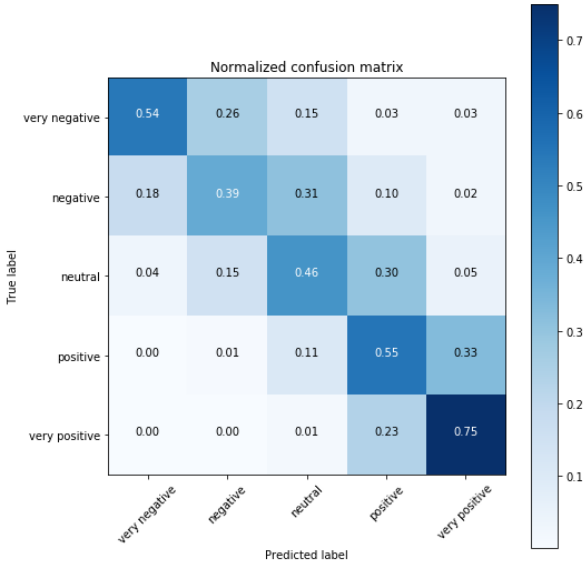


Figure 4.2. *Confusion matrix of the VD-CNN model in the TripAdvisor test dataset*

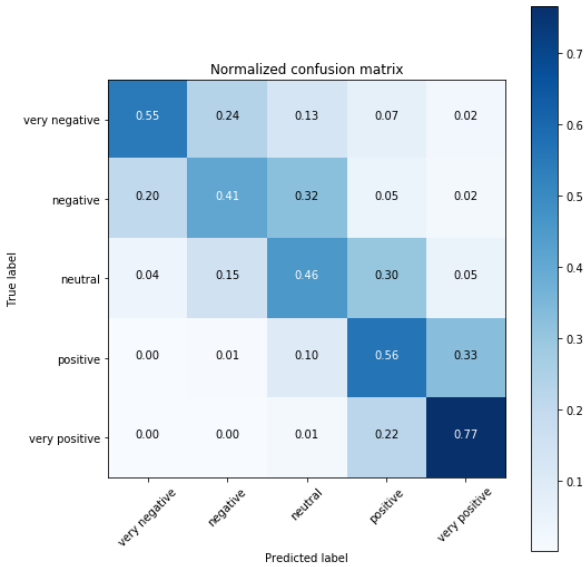


Figure 4.3. *Confusion matrix of the BiLSTM model in the TripAdvisor test dataset*

hotel reviews with more contradictory sentimental words were more prompted to be classified wrongly. For example, a hotel review from the Table 4.4 is listed 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 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 true sentiment of this review is the *very positive* class. The Text-CNN model and the VD-CNN model wrongly predicted this review as the *very negative* sentiment. The BiLSTM model misclassified this review as the *negative* sentiment. The value of the distance metric of this review is 11. This review contains both negative and positive words. The negative words include "noisey", "Not happy", and "appoligised". The positive words in the review contains "great", "enjoyed", and "good". These positive sentimental words expressed that the guest was satisfied with staying in the hotel while the negative words expressed the negative feeling at the same time. This sentiment contradiction increases the difficulty for correctly classifying the review for the three models. Even though the overall sentiment of this long hotel review is positive, the three DNN models failed to capture the semantic relationship of these words in this long text. As a result, these three models completely misclassified the hotel reviews as the negative class. For the other three hotel reviews listed in the table, they are long text review and contain contradictory words as well.

Another example of failing sentiment classification is the last hotel review shown in the Table 4.4. The true class for this review is the very negative class. However, this review does not contain any sentimental words. It is more prompted to narrative one story than expressing the reviewer's opinion. The three DNN models wrongly predicted it as the *very positive* class.

Table 4.4. *Top 5 misclassified hotel reviews in the TripAdvisor dataset*

hotel review	actual 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. We fell 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 [...]"	4	11
"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	11
"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 time I came back to my room in the middle of the day and the windows were wide open with no person in the room. I did call and I said that I never want this to happen again because birds, debris [...]"	4	9
"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	9
"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	6

5 CONCLUSION

In this thesis, three different deep neural networks (Text-CNN, VD-CNN, and BiLSTM) were evaluated on their performance for sentiment classification on TripAdvisor and SST datasets. Two word embedding initialization technique (FastText and GloVe) were compared in the experiments. Moreover, the misclassified hotel reviews in the TripAdvisor dataset were analyzed to understand the reason for misclassification by different networks.

Deep neural network models performance comparison

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 both TripAdvisor and SST dataset.

Word embedding initialization comparison

Through the experiments in this thesis, it becomes known that the word embedding initialization methods have an effect on the prediction result. In the Text-CNN and the BiLSTM model, the GloVe method outperformed the FastText method in both TripAdvisor and SST dataset. In the VD-CNN model, the GloVe method had slightly higher accuracy than the FastText in the TripAdvisor dataset but performed a little worse in the SST dataset.

Misclassified hotel reviews analysis

By analyzing the most misclassified hotel reviews from all three DNN-based methods in the TripAdvisor dataset, it can be seen that the long length hotel reviews with more contradictory sentimental words were more prompted to be misclassified. When words with strong contradiction (expressing positive or negative sentiments) exist in one review, it increases the complexity of the review and makes it more difficult for the DNN based algorithms covered in this thesis to correctly detect the sentiment of that review.

Future work

This thesis researched the sentiment classification with deep neural networks. There are a few limitations in this research work. The first limitation is the unbalance distribution of different sentimental reviews in the TripAdvisor dataset in the experiments. It is seen that the "very positive" and "positive" classes contain most reviews while the "negative" and "very negative" classes have small amount of reviews. One inspiration is to use sampling technique to make the sentimental classes balanced. The second limiting factor is the data size. The VD-CNN model achieved the worst performance in the experiments. A possible reason is that the TripAdvisor dataset is small for the VD-CNN model. We can test the VD-CNN model on a larger version of TripAdvisor dataset. Another possible work is to apply other word embedding techniques such as ElMo word representation to initialize word vectors. There are several other methods for generating vector representation for one sentence or paragraph that can be experimented on. Another possible future work is to use computational linguistics knowledge such as dependency parsing to analyze the intrinsic structure of the misclassified reviews and propose new features to improve the performance for the deep neural networks in the hotel review sentiment classification task.

REFERENCES

[1] D. Bahdanau, K. Cho and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2015).

[2] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research* 3.Feb (2003), 1137–1155.

[3] D. M. Blei, A. Y. Ng and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research* 3.Jan (2003), 993–1022.

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

[5] 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).

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

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, 248–255.

[8] K. V. Ghag and K. Shah. SentiTFIDF – Sentiment Classification using Relative Term Frequency Inverse Document Frequency. 2014.

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

[10] K. Hashimoto, C. Xiong, Y. Tsuruoka and R. Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587* (2016).

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

[12] 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.

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

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

[15] 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.

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

[17] J. Johnson, A. Karpathy and L. Fei-Fei. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 4565–4574.

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

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

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

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

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

[23] K. Korovkinas, P. Danenas and G. Garsva. SVM and Naıve Bayes Classification Ensemble Method for Sentiment Analysis. 2017.

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

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

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

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

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

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

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

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

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

[33] A. L. Maas, A. Y. Hannun and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. *Proc. icml*. Vol. 30. 1. 2013, 3.

[34] 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.

[35] 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.

[36] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, 807–814.

[37] 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.

[38] 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.

[39] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

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

[41] 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.

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

[43] 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.

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

[45] 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.

[46] 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.

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

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

[49] 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.

[50] 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.

[51] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang and W. Xu. CNN-RNN: A Unified Framework for Multi-label Image Classification. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 2285–2294.

[52] 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.

[53] 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.

[54] 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.

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

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

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

[58] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi. BERTScore: Evaluating Text Generation with BERT. *arXiv preprint arXiv:1904.09675* (2019).

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