Sentiment Analysis on Movie Reviews

Kangbo Chen(Kanch152)

August 29, 2022

Abstract

The movie industry has been developing for decades and the success of a movie generally depends on people's attitudes. The review data where expressed audience opinion about a movie has become an increasingly important basis in evaluation. Sentiment analysis on review data can help extract and determine whether the audience holds a positive attitude about a movie or not, which is beneficial for people to make decisions in selection. Four kinds of models are trained in this project to do sentiment analysis tasks with Naive Bayes as the baseline model. Naive Bayes achieves an accuracy of 87%, which is even better than LSTM and TextCNN with random word embedding, while BERT and TextCNN with GloVe outperform the baseline slightly. Three possible reasons are discussed at the end including limited training data, mixed attitude, and pre-processing method.

Keywords: Movie review; Naive Bayes; LSTM; TextCNN; BERT

Contents

List of Figures

List of Tables

Introduction

As the Internet develops nowadays, social networking has become a common place for people to share lives in the form of videos, photos, short text, etc. The text is the most important part as it can not only appear alone, such as in blogging, Twitter, and movie reviews but is able to be contained in videos and photos as supplement material. Short text analysis has certain challenges like identification of sentiment, sarcasm, and use of slang words, thus, understanding the short text and gaining insight behind has attracted many researchers to work. Sentiment analysis, which is a natural language processing task to determine whether the short text is positive or negative, is one of those popular topics researchers are interested in.

Reviews expressing an opinion about a movie plays a vital role in the success of the movie because people generally choose a movie to watch based on the score and reviews from other audiences like on IMDB. Sentiment analysis of review text makes the task of Opinion Summarization easier by extracting the sentiment expressed by the reviewer. As the aim of this task is to examine and determine the sentiment polarity in this paper, emotions can be expressed as a binary value (1,0) meaning positive or negative and the words in the reviews are considered as features, therefore, the task is generalized as a classification problem where machine learning method can offer great help. Although these reviews are pairs of sentence containing many information and traditional machine learning methods may still have decent performance based on bag-of-words model regardless words order, other advanced methods might outperform as they can capture more contextual information.

Since the raw text cannot be feed into model directly, a series of procedures have been done to preprocess the documents including removing punctuation, stop words and non-alphabetic words, word embedding ,etc. A naive Bayesian model using Counter Vectorizer is implemented in this paper as baseline model. Convolutional Neural Networks(CNN) with both pre-trained GloVe word embedding and random tunable word embedding are constructed as well as the same for LSTM model. A base version of pre-trained Bert model has also been constructed and fine-tuned to compare with other models as advanced one because of its high performance in many researches. The best parameters for each model is acquired through gradient descent based on build-in loss function. The performance of models are evaluated using precision, f1 score and recall metrics.

The following content will first introduce the background knowledge of word embedding and model structure, data summerization, method used, experiment result, discussion and concludsion.

Data

The data used in this paper coming from Internet Movie Database (IMDb) with 50K movie reviews and binary sentiment labels published on Kaggle(Maas et al. (2011)). Part of the example data is shown in the following table2.1 where the left column is 'Reviews' content and the other column contains sentiment label for corresponding review.

Reviews	Sentiments
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with me.¡br /¿¡br /¿The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. .Bayespositive	
A wonderful little production. ¡br /¿¡br /¿The filming technique is very unassuming-very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece. ¡br /¿¡br /¿The actors are ...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet his parents are fighting all the time.¡br /¿¡br /¿This movie is slower ...	negative
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers us a vivid portrait about human relations. This is a movie that seems to be telling us what money, power and success do to people in the different situations we encounter. ...	positive
This show was an amazing, fresh and innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. By 1990, the show was not really funny anymore, and it's continued its decline further to the complete waste of time it is today.¡br /¿¡br /¿It's truly disgraceful how far this show has fallen. ...	negative

Table 2.1: Raw data

From the table above, it is clear that there are some non-alphabetic words, upper case word and punctuations in reviews, therefore, preprocessing is necessary to remove redundant words and change the letter into lower case. After pre-processing, examples of the raw data are indicated in the following table 2.2.

Reviews	Sentiments
reviewer mention watch episode ll hooked right exactly happen me thing strike brutality unflinching scene violence set right word trust faint...	positive
wonderful little production filming technique unassuming oldtimebbc fashion give comforting discomfort sense realism entire piece actor extremely ...	positive
basically s family little boy jake think s zombie closet parent fight time movie slow soap opera suddenly jake decide rambo kill zombiebr go film decide thriller drama ...	negative
petter mattei love time money visually stunning film watch mattei offer vivid portrait human relation movie tell money power success people different situation encounter variation ...	positive
amazing fresh innovative idea air year brilliant thing drop funny anyma ore continue decline complete waste time todaybr truly Bayesaceful far fall writing painfully bad performance bad mildly entertaining respite guesthost ...	negative

Table 2.2: Preprocessed data

The preprocessed data distribution is indicated in Figure 2.1 showing data is already balanced(25000 samples per class), therefore, both classes are equally weighted in models.

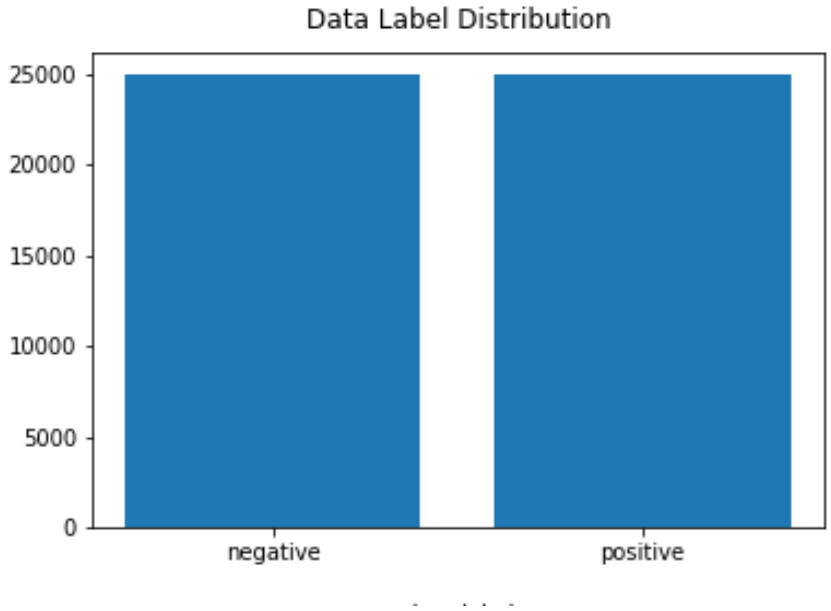

Figure 2.1: label distribution, 25000 for positive and 25000 for negative class

After counting the number of words in each review, a frequency distribution graph of text length is indicated in 2.2, where most texts are shorter than 500 words and the longest length of text contains 1277 words.

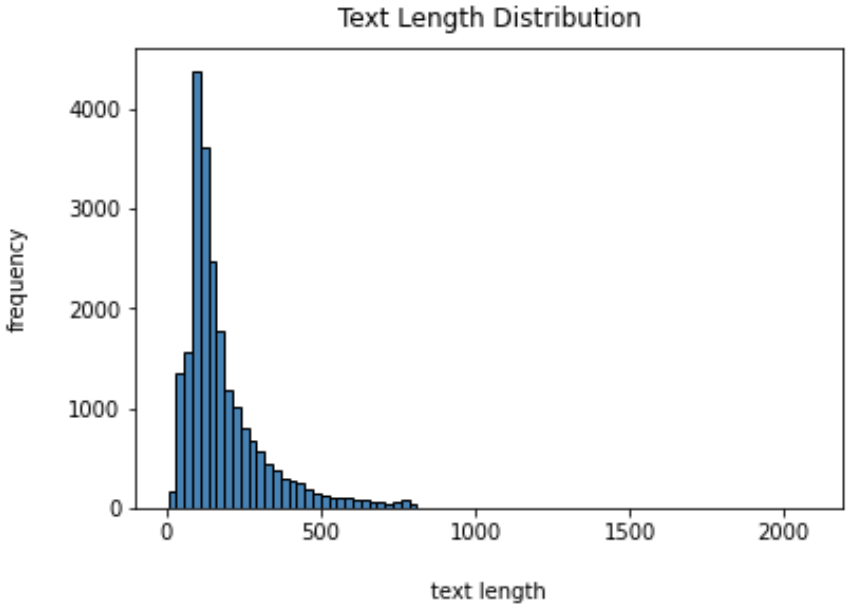

Figure 2.2: Text Length Distribution

Theory

3.1 Countvectorizer

Count Vectorizer is a method to convert text into numerical representation based on word frequency since the computer cannot understand and process text information. It uses the number of occurrences of single words in each document to represent words numerically. After vectorization, the text can be fed into the NLP models to train.

3.2 TF-idf Vectorizer

TF-idf stands for term frequency-inverse document frequency, which is another statistics method to vectorize text information. The vector of words results from the product of term frequency and inverse of document frequency, which is the frequency of a word in a document and the frequency of a word appearing in all documents respectively.

3.3 Naive Bayes

The Naive Bayes classifier is a simple and commonly used model to estimate the probability of data belonging to the different classes by posterior distribution. The Naive Bayes model assumes that the word features are independent of each other, which means the appearance of two words is not related. The following equation indicates the Bayes Theorem.

$$P(Class|X) = \frac{P(X|Class)P(Class)}{P(X)} \tag{3.1}$$

where $P(X|Class) = \prod_{j=1}^{m} P(x_j|Class)$, $P(Class)$ is the probility of one class, $P(X)$ refers to the probability of one set of word features.

There are many versions of the Naive Bayes classifier, the one that is applied in this project is Multinomial Naive Bayes and the objective is shown in the following equation.

$$Class_{predict} = \arg\max_{k} P(Class_k) \prod_{j=1}^{n} P(x_j|Class_k) \tag{3.2}$$

where the $Class_{predict}$ is the target class and k refers to the k-th class.

3.4 GloVe

Word embedding is used to represent text words in numerical vector space, which encode the meaning behind the word such that the words with similar meaning have a closer distance in vector space as well. Compared with high dimension in vectorizer, word embedding is denser with lower dimensional vector space, which saves storage space and reduces computation. The word embedding usually contains a large number of parameters to train and the training takes a relatively long time, therefore, a

pre-trained word embedding is generally introduced in the model instead. One of the most popular pre-trained word embeddings used in this project is GloVe standing for Global Vectors for Word Representation, which is developed by Pennington et al. (2014) at Standford. It is a distributed word representation produced by an unsupervised learning model. The corpus on which the representations are trained includes Wikipedia, Common Crawl, and Twitter with various vocabularies. There are also different dimensions of word vectors ranging from 25 to 300.

3.5 TextCNN

Convolutional Neural Networks, known as CNN, have achieventering markable results in the field of computer vision, such as image classification, object detection, etc, due to their strong ability to extract features. On the other hand, an increasing number of work making use of CNN for NLP, which is known as TextCNN, has been involved and shown to be effective after the work done by Yoon Kim in 2014. Kim (2014)

The model structure has been shown in Figure 3.1. It consists of several types of layers including an input layer, convolutional layer, max-pooling layer, and fully connected layer. The input layer requires the text data representation to be a fixed size matrix whose row dimension refers to the length of text and column is k-dimensional word vector for ith word in the text. The convolutional operation here is similar to that of image processing, but a one-dimensional filter with h certain window size is applied to generate new features.

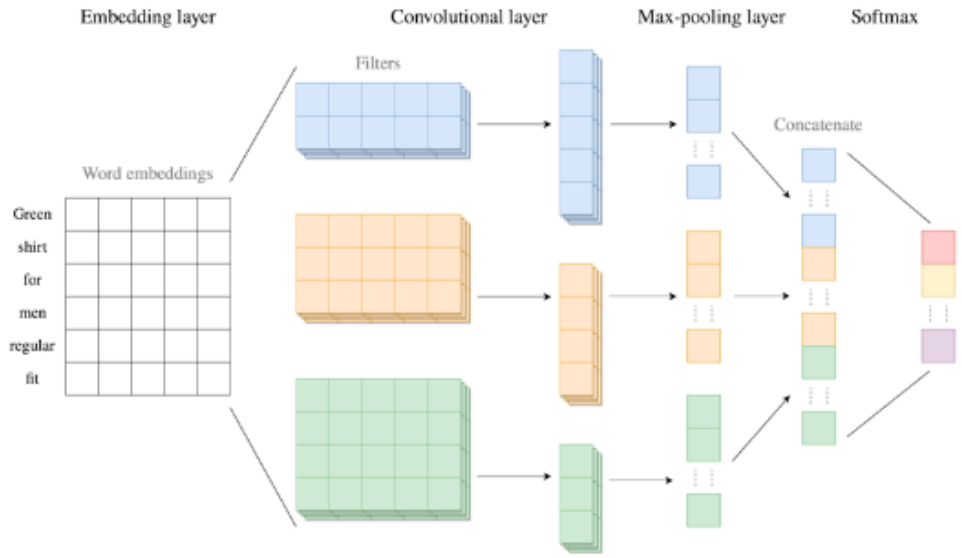

Figure 3.1: TextCNN architecture for text classification

TextCNN can also apply for pre-trained word vectors by introducing word embedding obtained from the unsupervised neural language model to improve the performance. The pre-trained word embedding in this paper comes from GloVe, a popular pre-trained word embedding. The hyperparameters in TextCNN are window size that controls the vision scope of filters and the number of channels for the corresponding window.

3.6 LSTM

Text words are treated as a sequence in the RNN-based model with aim of capturing dependencies between words, however, the vanilla RNN is only capable to carry short-term information due to gradient vanishing or exploding problems. This problem can be overcome by Long Short-Term Memory(LSTM) whose structure is specially designed to store long-term memory by introducing a memory cell that can store long-term information. The networks consist of a series unit in which there are three gates

including input gates, output gates, and forget gates to control the information flow into and out of each cell, as is shown in Figure 3.2. If the input gates output a value close to zero, the value from the net input will be blocked from entering the next layer. Forget gates will control to forget whatever value was remembered and the output gates control whether the cell should output the value in its memory. (Soutner and Müller (2013))

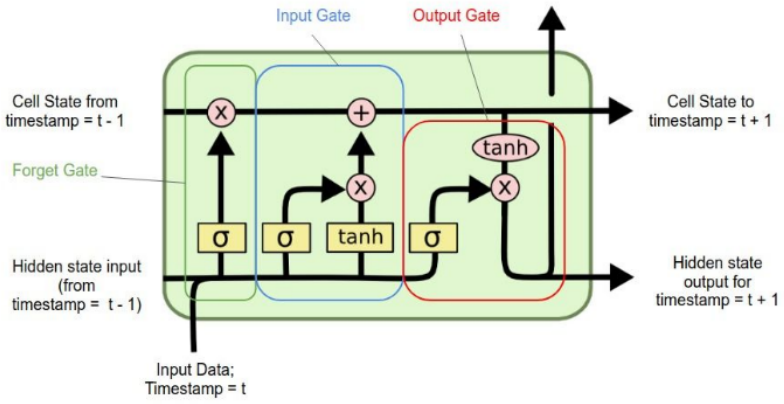

Figure 3.2: LSTM architecture

3.7 BERT

Bidirectional Encoder Representations from Transformers known as BERT is designed to pre-train deep bidirectional representations from text considering both left and right contextual environments in all layers and can be fine-tuned on various downstream natural language processing tasks. It is introduced by Jacod Devlin and his colleague from Google in 2018 and showed groundbreaking performance on 11 natural language processing tasks.Devlin et al. (2018) The overview architecture of the BERT model in this paper is indicated in Figure 3.3.

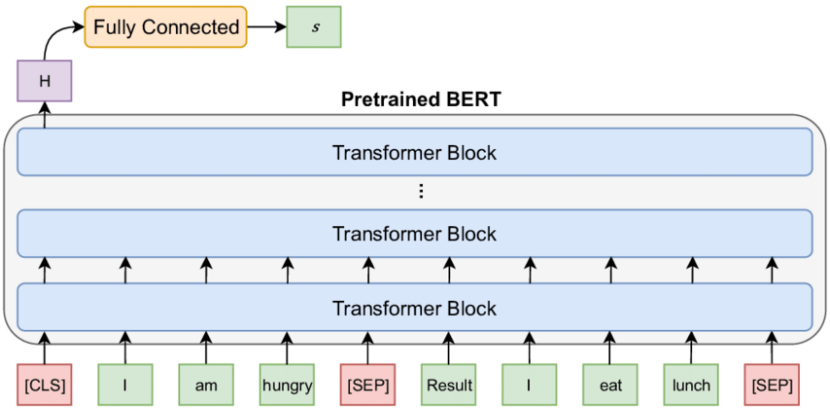

Figure 3.3: BERT architecture

BERT is a model based on Transformer which is developed by Vaswani and his colleague at Google Brain in 2017. Compared with sequential processing in RNNs, Transformer introduces attention mechanism to describe dependencies between input and output sequence, which allows more parallelization and higher accuracy in translation quality. Vaswani et al. (2017). Figure 3.4 shows the structure of the Transformer in which the left part mainly refers to the encoder and the right part refers to the decoder. The encoder part in Transformer is the building block of BERT where 12 encoders are used in the BERT-BASE model. Taking a further look at the encoder structure, it is composed of two sub-layers, a multi-head self-attention mechanism, and a fully connected feed-forward network, including residual

connection after the normalization layer. The self-attention structure is shown in Figure 3.5, where the left is Scaled Dot-Product Attention and the right Multi-Head Attention consists of several attention layers in parallel. Vaswani et al. (2017)

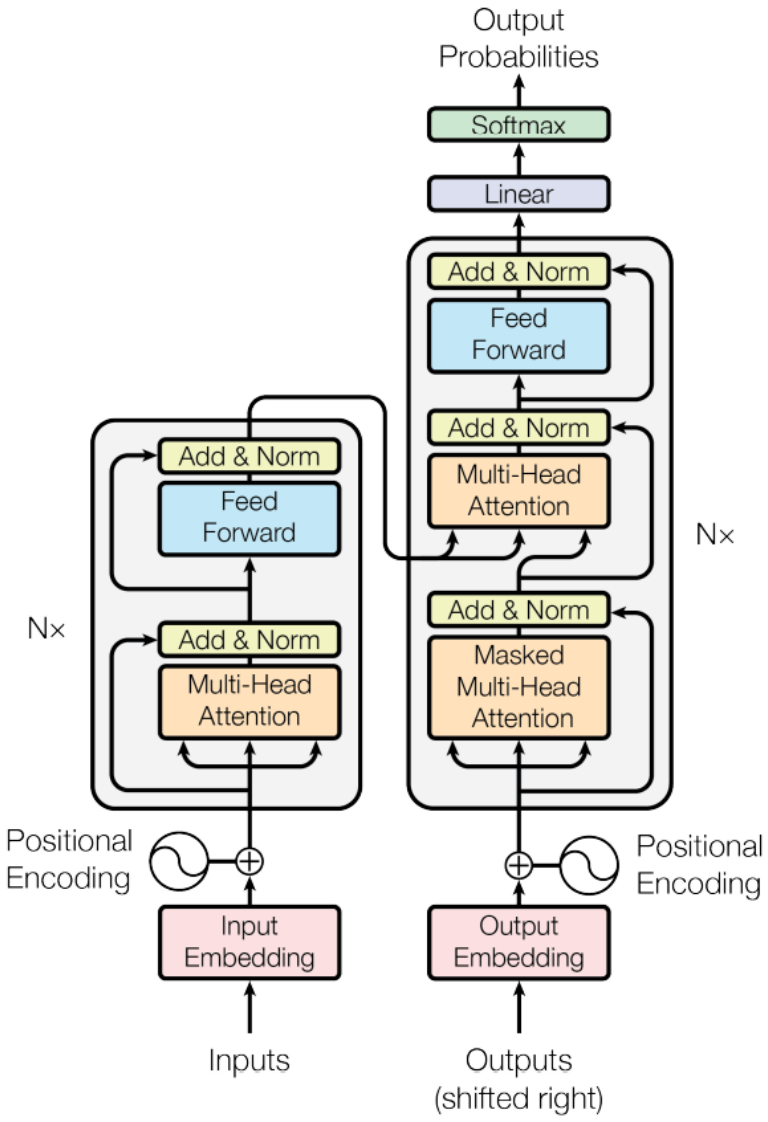

Figure 3.4: The Transformer architecture

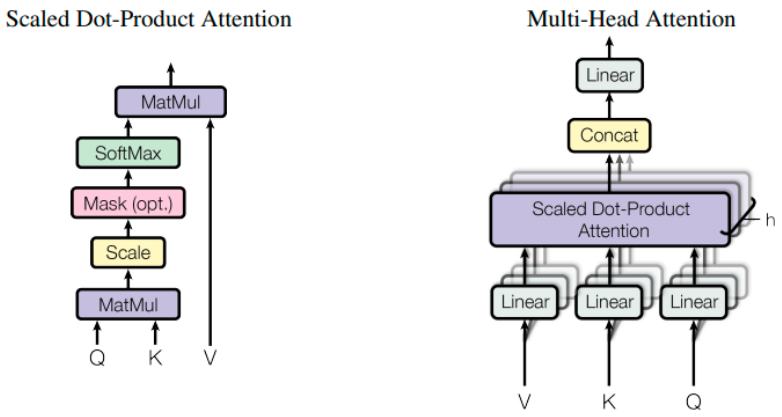

Figure 3.5: (left) Scaled Dot-Product Attention. (right) The multi-head self-attention architecture

There are two steps to training the BERT model, pre-training and fine-tuning as is shown in Figure 3.6. Pre-training phase consists of two unsupervised learning tasks that are Masked Language Model(MLM) and Next Sentence Prediction(NSP). In the MLM task, 15% of all word tokens are masked in each sequence at random before being fed into the model to predict which word they are in the vocabulary. This task trains the transformer parameters and embedding layer to recognize the masked word through the contextual environment. NSP task trains the model to identify sentence order relationships, for example, whether sentence B is the next sentence of A. 50 % of selected sentence B is the actual sentence of A, but the other half would be random sentences in the corpus (Vaswani et al. (2017)). The pre-training corpus used is the BooksCorpus (800M words) (Zhu et al. (2015)) and English Wikipedia (2,500M words).

During fine-tuning section, feed the BERT model with inputs and outputs data and fine-tune all the pre-trained parameters end-to-end in the Transformer. This returns us a final BERT model for the specific downstream task. Although the BERT model looks heavy with 110 million parameters to train, the self-attention mechanism within the Transformer allows fine-tuning faster than in the pre-training phase.

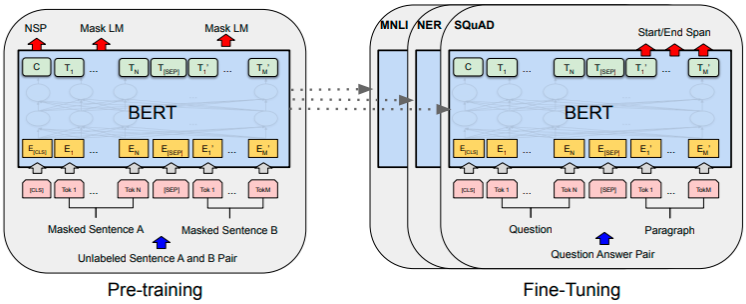

Figure 3.6: (left) Pre-training. (right) Fine-Tuning

Method

4.1 Data Separation

There are 50K reviews in the raw data set. 17500 reviews are divided as training data, 4500 reviews are used for validation, and 5000 as the test set.

4.2 Naive Bayes

The Naive Bayes model is constructed and fitted with training data as a baseline model to compare with other advanced models. The training text is firstly preprocessed, vectorized with a TF-IDF vectorizer, and then fitted into the Naive Bayes model. A 5-fold cross-validation phase is applied for the selection of hyperparameters using the grid-search function. Three parameters are selected as tuned hyperparameters, if *ngram-range* should be set to *(1,1)* or *(1,2)*, if *binary* should be set to *0* or *1* in TF-idf vectorizer, and if the smooth parameter α in Naive Bayes model should have set to a value between *0.1* and *1*.

4.3 LSTM

Before training of LSTM model, the sequence of text reviews is either padded or truncated to the length of 256 based on the specific number of words in each sequence. The number of pad length is decided by the previous statistical analysis of sequence length where the length of most documents is less than 256, which allow the representation to store most information and consider computational efficiency simultaneously. Apart from random trainable word embeddings, a pre-trained embedding using GloVe also applies in this project. GloVe-twitter-100d word embedding involves 27 billion tokens, 1.2 million vocabularies, and 100 dimensions in word representation. Since the review content is closer to twitter text in which people express their opinions and emotions than facts in Wikipedia, and has less computational burden than Common Crawl with 300 dimensions, GloVe is more suitable than the other two datasets in this project.

With the fixed length of sequences and generated work embedding, the whole network structure is ready to build. The final structure is decided as follows after many times of experiments, the first is the word embedding layer with the embedding weight matrix as mentioned above and the parameter 'trainable' is set to True, which means the embedding layer can be trained according to the training data. This is because the GloVe collection may not fit any situation, a tunable embedding allows a fine-tuned embedding based on a specific dataset. Then one bidirectional LSTM layer is added using 64 units followed by two dense layers with 256 hidden nodes and binary output. The drop-out rate after LSTM and hidden dense layer are set to 0.5. ReLU and Sigmoid functions are applied in the hidden layer and output layer respectively to acquire the final class label. The final structure is shown in the following Table 4.1.

Table 4.1: LSTM Network Structure

Layer (type)	Output Shape	Parameters
Input	(None, 256)	0
Embedding	(None, 256, 100)	15720000
Bidirectional	(None, 128)	84480
Dropout	(None, 128)	0
Dense	(None, 256)	33024
Dropout	(None, 256)	0
Dense	(None, 2)	514

From Table 4.2, it is clear that all 15,838,018 parameters in the model are trainable parameters, with 15,720,000 of these are embedding parameters. Those embedding parameters are also trainable to achieve optimal fine-tuned networks.

Table 4.2: LSTM Parameter Distribution

Name	Number of Parameters
Total Parameters	15,838,018
Trainable Parameters	15,838,018
Non-trainable Parameters	0

After initialization network structure, the model is compiled with the cross-entropy function, and the optimizer is set using Adam with learning rate equaling to 0.0005. (Plus, the output of the dense layer is expressed using one-hot label, categorical cross-entropy function is applied. The same setting applies to the following models as well). The models are trained for 10 epochs at most and mini-batches size is set to 400.

4.4 TextCNN

Similar to LSTM, initializing the word embedding layer is also the first step in TextCNN followed by 4 parallel convolutional layers with different window sizes. With filters in different dimensions, neural networks can extract phrase features with different lengths. In the final version of TextCNN, windows with sizes in 3,4,5,6 are applied and their filter channels are equal to 650, 650, 650, and 650 respectively. Since the information from convolutional filters is still large to process, Maxpooling layers are set after filters to compress information. Drop-out layers with parameters equaling 0.5 are put behind Maxpooling layers to improve the generalization ability of models. A concatenated layer is set after convolutional layers to combine the features from all window sizes with a single dense layer being put after it to output final class labels. 4.3

Table 4.3: TextCNN Network Structure

Layer (type)	Output Shape	Parameters
Input	(None, 256)	0
Embedding	(None, 256, 100)	9242300
Convolution (1D)	(None, 256, 650)	195650
Convolution (1D)	(None, 256, 650)	260650
Convolution (1D)	(None, 256, 650)	325650
Convolution (1D)	(None, 256, 650)	390650
Max pooling (1D)	(None, 1, 650)	0
Max pooling (1D)	(None, 1, 650)	0
Max pooling (1D)	(None, 1, 650)	0
Max pooling (1D)	(None, 1, 650)	0
Dropout	(None, 1, 650)	0
Dropout	(None, 1, 650)	0
Dropout	(None, 1, 650)	0
Dropout	(None, 1, 650)	0
Concatenate	(None, 1, 2600)	0
Flatten	(None, 2600)	0
Dropout	(None, 2600)	0
Dense	(None, 2)	5202

From Table 4.4, it is clear that all 10,420,102 parameters in the model are trainable parameters, with 9,242,300 of these are embedding parameters. Those embedding parameters are also trainable to achieve optimal fine-tuned networks.

Table 4.4: TextCNN Parameter Distribution

Name	Number of Parameters
Total Parameters	10,420,102
Trainable Parameters	10,420,102
Non-trainable Parameters	0

After initialization network structure, the model is compiled with the cross-entropy function, and the optimizer is set using Adam with learning rate equaling to 0.0005. The models are trained for 10 epochs at most and mini-batches size is set to 400.

4.5 BERT

As is mentioned in the Theory chapter, there are over 100 million parameters to train in the BERT BASE model and heavier networks in BERT LARGE. Thus, the BERT BASE model is more suitable in this situation due to the limited computational resources. Hugging Face provides a package for implemented BERT model making the construction of the model rather straightforward. The whole network starts by embedding layers different from TextCNN and LSTM as BERT hold different tokenization rules. Moreover, there are three different embedding components within the embedding layer considering the contextual environments of each word. It includes segment embedding, position embedding, and text embedding. Apart from word embedding, the encoder of BERT also generates an attention mask to represent which token is padded.

After word embedding layer, the transformer block is added t which token is analyzed the text reviews followed by a dense layer for the final classification. The final model parameters are shown in the Table 4.5.

Table 4.5: BERT Parameters

Layer (type)	Output Shape	Parameters
BERT(TFBertMainLayer)	multiple	109482240
Dropout	multiple	0
Classifier	multiple	1538

Table 4.6: BERT Parameter Distribution

Name	Number of Parameters
Total Parameters	109,483,778
Trainable Parameters	109,483,778
Non-trainable Parameters	0

Result

5.1 Error Metrics

The models are evaluated by error metrics including F1 score, Recall, Precision rate and confusion matrix.

5.1.1 Precision

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

where TP refers to True Positive, FP refers to False Positive.

5.1.2 Recall

$$Precision = \frac{TP}{TP + FN} \tag{5.2}$$

where TP refers to True Positive, FN refers to False Negative.

5.1.3 F1 Score

$$F1 = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \tag{5.3}$$

5.2 Naive Bayes

As a baseline model, a Naive Bayes classifier using Counter Vectorizer is fitted on the training set with default settings, and the Classification Report on the test set is indicated in Table 5.1 and the confusion matrix in Figure 5.1. From the classification report, the accuracy of the test set is 84% with 86% in the precision rate of the positive reviews and 83% for the negative reviews. The F1 scores are close for both classes, equaling 85% and 84% respectively.

Table 5.1: Classification Report for Naive Bayes Classifier with default setting(Count Vectorizer)

	Precision	Recall	F1 score	Support
negative	0.83	0.87	0.85	2534
positive	0.86	0.82	0.84	2466
accuracy			0.84	5000
macro avg	0.84	0.84	0.84	5000
weighted avg	0.84	0.84	0.84	5000

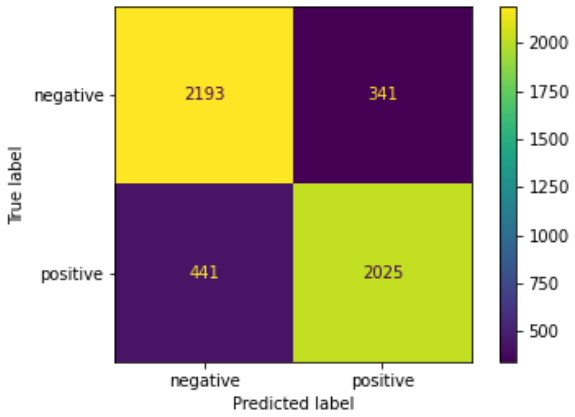

Figure 5.1: Confusion Matrix for Naive Bayes with Count Vectorizer

The classification report and the confusion matrix for the Naive Bayes model using TF-idf Vectorizer are illustrated in Table 5.2 and Figure 5.2. Compared with Count Vectorizer, the model with TF-idf Vectorizer shows a closer precision rate between the two classes with 84% for the negative label and 86% for the positive label, and a slightly higher accuracy rate reaching 85%.

Table 5.2: Classification Report for Naive Bayes Classifier with default setting(TF-idf Vectorizer)

	Precision	Recall	F1 score	Support
negative	0.84	0.86	0.85	2534
positive	0.86	0.84	0.85	2466
accuracy			0.85	5000
macro avg	0.85	0.85	0.85	5000
weighted avg	0.85	0.85	0.85	5000

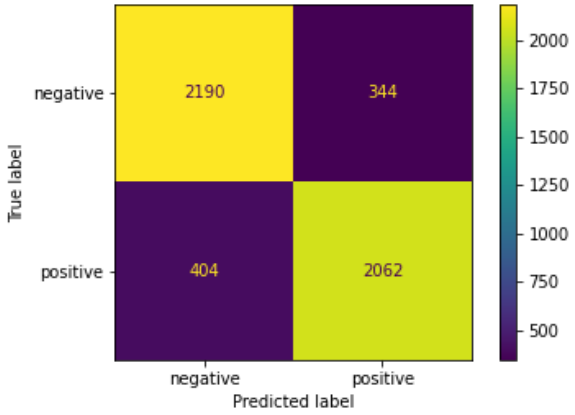

Figure 5.2: Confusion Matrix for Naive Bayes with TF-idf Vectorizer

Through the comparison between Count Vectorizer and TF-idf Vectorizer, it is clear that the latter produces better model performance, therefore, TF-idf Vectorizer is selected to proceed in the cross-validation step using grid search to find the optimal hyperparameters. Finally, the best hyperparameters selected include *binary = 1* and *ngram-range = (1,1)* in TF-idf Vectorizer, which means

the TF term in the vectorizer is binary and only unigrams are used. The classifier ends up with the smoothing parameter $\alpha = 1$. Plugging in optimal parameters into the model returns the final Naive Bayes model and its performance is indicated in Table5.3 and Figure5.3.

Table 5.3: Classification Report for Naive Bayes Classifier with Optimal Setting(TF-idf Vectorizer)

	Precision	Recall	F1 score	Support
negative	0.87	0.89	0.88	2534
positive	0.88	0.86	0.87	2466
accuracy			0.87	5000
macro avg	0.87	0.87	0.87	5000
weighted avg	0.87	0.87	0.87	5000

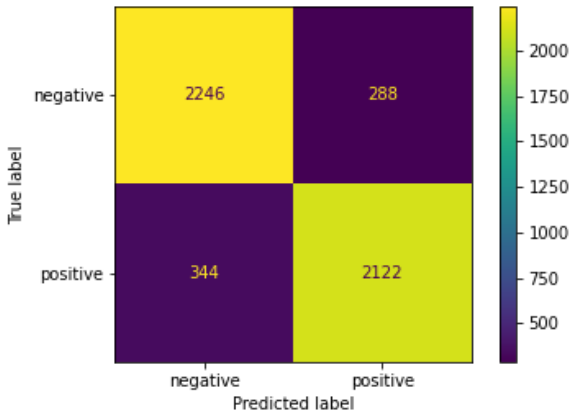

Figure 5.3: Confusion Matrix for Optimal Naive Bayes with TF-idf Vectorizer

From the classification report and confusion matrix of the optimal model, the precision rate for negative reviews improves significantly from 82% to 87% and positive labels also benefit from grid-search resulting in the accuracy rate increase by 2%.

5.3 LSTM

An LSTM model is constructed as introduced in chapter theory. The training data is fitted and the training process is indicated in Figure 5.4.

It is clear that the validation accuracy increases quickly in the first 3 epochs and is then stable before decreasing slightly during the last 3 epochs. The loss for validation follows an opposite trend with the accuracy value, but the model reaches optimal at the fifth epoch from both figures.

Table 5.4: Classification Report for LSTM using GloVe

	Precision	Recall	F1 score	Support
negative	0.87	0.86	0.87	2534
positive	0.86	0.87	0.87	2466
accuracy			0.87	5000
macro avg	0.87	0.87	0.87	5000
weighted avg	0.87	0.87	0.87	5000

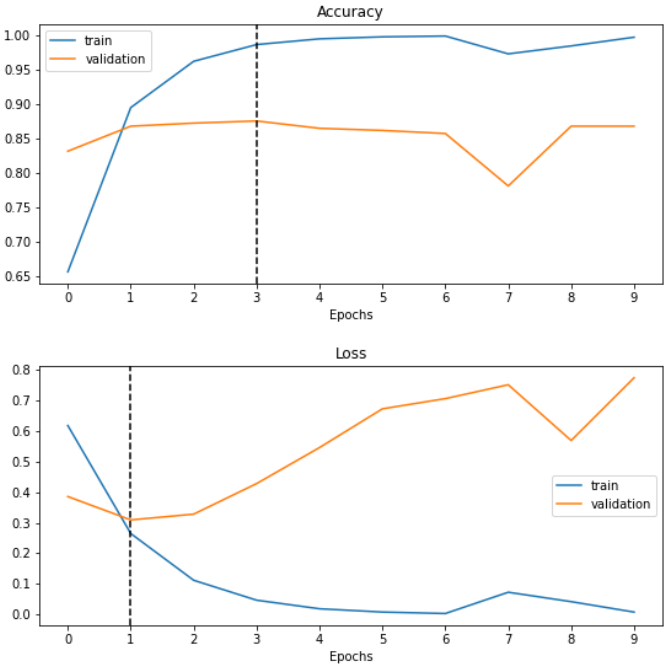

Figure 5.4: Training process for LSTM

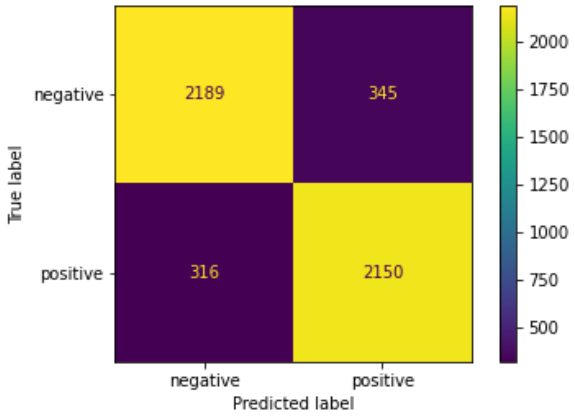

Figure 5.5: Confusion Matrix for LSTM

Table 5.4 and Figure5.5 illustrate the classification report and the confusion matrix of LSTM, from which we can see that 87% of negative reviews (2189 in confusion matrix) are classified correctly and the figure for positive reviews is 84% (2150 in confusion matrix). The F1 score for both labels is equal to 86%.

An LSTM model with random word embedding is constructed as introduced in chapter theory. The training data is fitted and the training process is indicated in Figure 5.6.

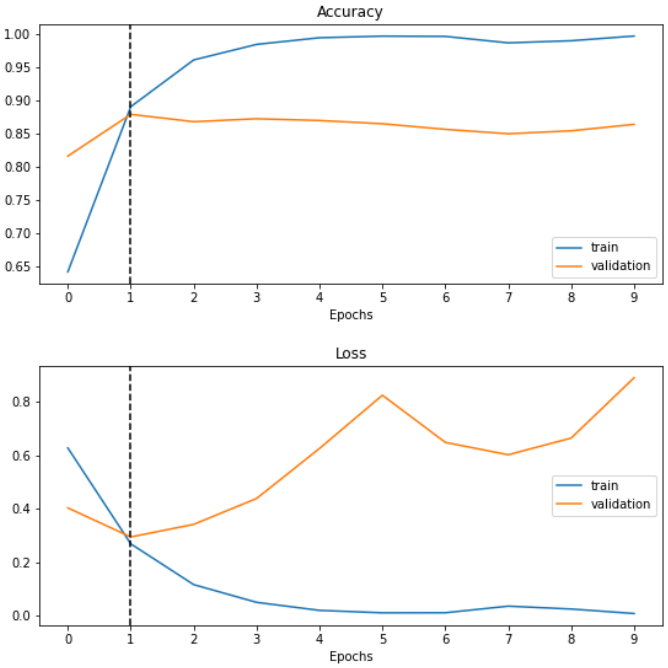

Figure 5.6: Training Process for LSTM with Random Word Embedding

It is clear that the validation accuracy increases quickly in the first 2 epochs and is then stable in the following epochs. The loss for validation follows an opposite trend with the accuracy value, but the model reaches optimal at the fifth epoch from both figures.

Table 5.5: Classification Report for LSTM using random word embedding

	Precision	Recall	F1 score	Support
negative	0.88	0.87	0.88	2534
positive	0.87	0.88	0.87	2466
accuracy			0.87	5000
macro avg	0.87	0.87	0.87	5000
weighted avg	0.87	0.87	0.87	5000

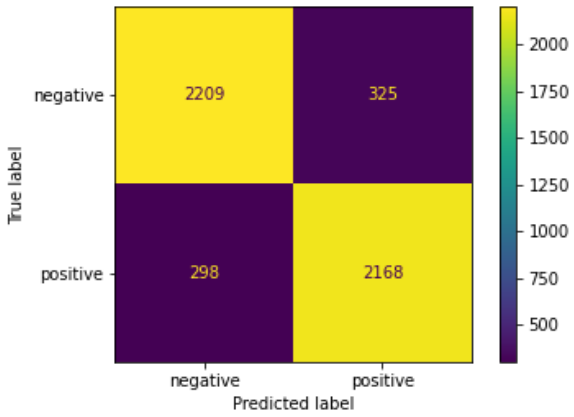

Figure 5.7: Confusion Matrix for LSTM with Random Word Embedding

Table 5.7 and Figure5.6 illustrate the classification report and the confusion matrix of LSTM, from which we can see that 88% of negative reviews (2200 in confusion matrix) are classified correctly and the figure for positive reviews is 87% (2185 in confusion matrix). The F1 score for both labels is equal to 87%.

5.4 TextCNN

A TextCNN model using GloVe word embedding is constructed as introduced in chapter theory. The training data is fitted and the training process is indicated in Figure 5.8.

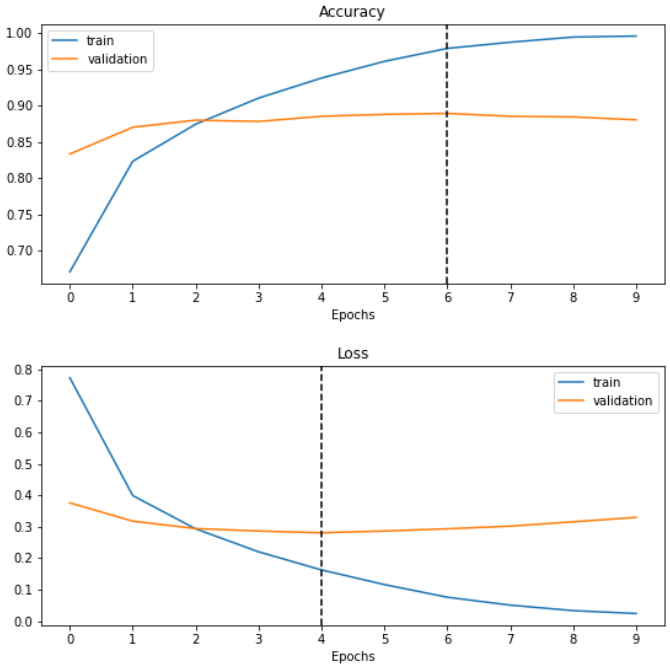

Figure 5.8: Confusion Matrix for TextCNN using GloVe

As can be observed, the validation accuracy increases slightly in the first 3 epochs and then remain stable in the following epochs, and the highest accuracy is 88.93% at the seventh epoch. The loss for validation follows an opposite trend with the accuracy value, and the minimum loss is achieved at

the fifth epoch. The optimal epoch during which the model achieves the highest accuracy rate on the validation set is selected and plugged in the training of the final model being used to classify the test set. The results are shown by Table 5.6 and Figure 5.9

Table 5.6: Classification Report for TextCNN using GloVe

	Precision	Recall	F1 score	Support
negative	0.89	0.87	0.88	2534
positive	0.87	0.89	0.88	2466
accuracy			0.88	5000
macro avg	0.88	0.88	0.88	5000
weighted avg	0.88	0.88	0.88	5000

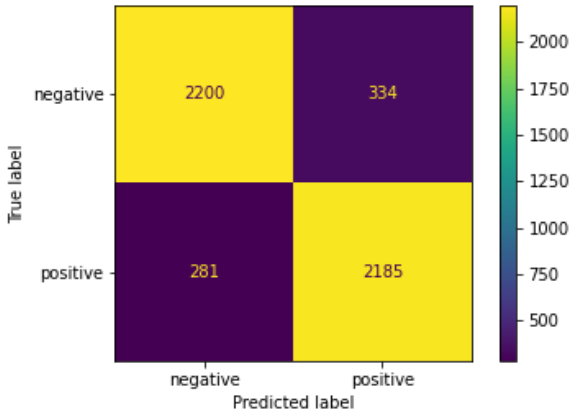

Figure 5.9: Confusion Matrix for TextCNN using GloVe

From the classification report, it is clear to see that the precision of negative reviews reaches 89% and the figure for positive labels is 87%, while the F1 scores for both labels are 88% equally. As for the confusion matrix, 334 positive reviews and 281 negative reviews are wrongly classified.

A TextCNN model using random word embedding is constructed as introduced in chapter theory. The training data is fitted and the training process is indicated in Figure 5.10.

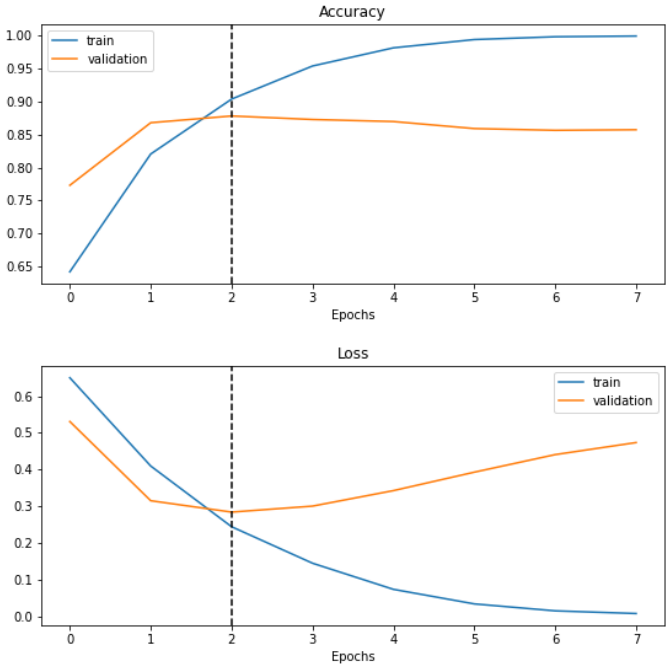

Figure 5.10: Confusion Matrix for TextCNN using Random Word Embedding

As can be observed, the validation accuracy increases slightly in the first 3 epochs, before slightly decreasing in the following epochs, and the highest accuracy is 87.82% at the third epoch. The loss for validation follows an opposite trend with the accuracy value, and the minimum loss is achieved at the third epoch as well. The optimal epoch during which the model achieves the highest accuracy rate on the validation set is selected and plugged in the training of the final model being used to classify the test set. The results are shown by Table 5.7 and Figure 5.11

Table 5.7: Classification Report for TextCNN using Random Word Embedding

	Precision	Recall	F1 score	Support
negative	0.88	0.85	0.86	2534
positive	0.85	0.88	0.86	2466
accuracy			0.86	5000
macro avg	0.86	0.86	0.86	5000
weighted avg	0.86	0.86	0.86	5000

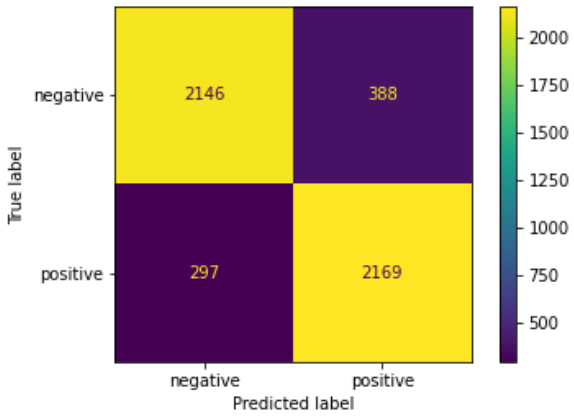

Figure 5.11: Confusion Matrix for TextCNN using Random Word Embedding

From the classification report, it is clear to see that the precision of negative reviews reaches 88% and the figure for positive labels is 85%, while the F1 scores for both labels are 86% equally. As for the confusion matrix, 411 positive reviews and 357 negative reviews are wrongly classified. Compared with GloVe word embedding, random word embedding achieves a lower accuracy rate no matter from the aspect of precison, recall, and F1 score.

5.5 BERT

A BERT model is constructed as introduced in chapter theory. The training data is fitted and the training process is indicated in Figure 5.13.

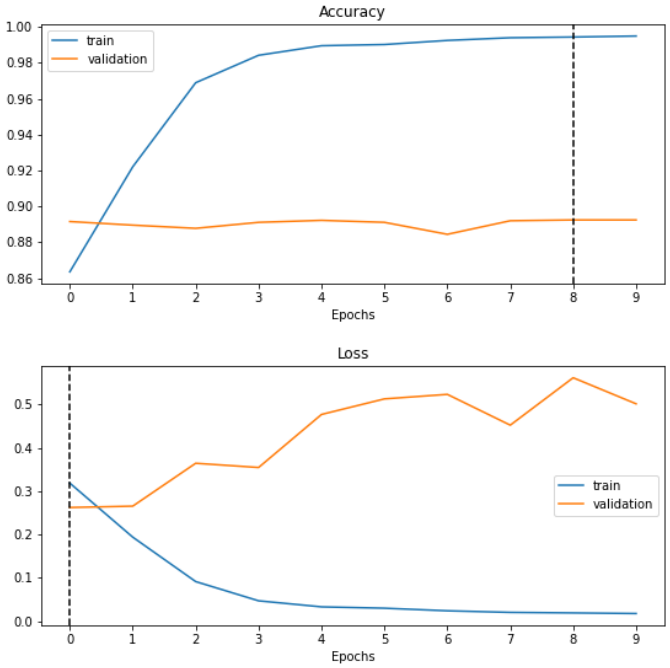

Figure 5.12: Training process for BERT

As can be observed, the validation accuracy remains stable over all epochs, and the highest accuracy is 87.82% at the ninth epoch. The loss for validation follows an opposite trend with the accuracy

value, and the minimum loss is achieved at the first epoch. The optimal epoch during which the model achieves the highest accuracy rate on the validation set is selected and plugged in the training of the final model being used to classify the test set. The results are shown by Table 5.8 and Figure 5.13

Table 5.8: Classification Report for BERT

	Precision	Recall	F1 score	Support
negative	0.90	0.89	0.89	2534
positive	0.88	0.89	0.89	2466
accuracy			0.89	5000
macro avg	0.89	0.89	0.89	5000
weighted avg	0.89	0.89	0.89	5000

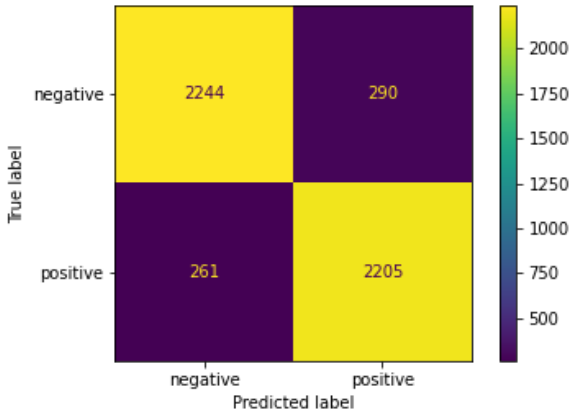

Figure 5.13: Confusion Matrix for BERT

From the classification report, it is clear to see that the precision of negative reviews reaches 90% and the figure for positive labels is 88%, while the F1 scores for both labels are 89% equally. As for the confusion matrix, 290 positive reviews and 261 negative reviews are wrongly classified.

Discussion

From the performance of the models evaluated above, the fine-tuned Naive Bayes model achieves surprisingly good classification results reaching 87.36% in accuracy, which is even higher than LSTM neural networks (87.08% and 87.02% respectively), and TextCNN with random word embedding. The BERT BASE model performs best compared with other models but does not outperform the baseline pretty much, only 1.63% higher than Naive Bayes and the computational cost including computational resources and time is far more than the baseline. The models' performance summary is indicated in Table 6.1. (Plus. the accuracy here is the percentage of correctly classified samples on overall samples)

Table 6.1: Model Performance on Test Set

Model Name	Representation	Accuracy(%)	F1 score(%)
Naive Bayes	TF-idf	87.36	87
LSTM	GloVe	87.08	87
LSTM	Random Embedding	87.02	87
TextCNN	GloVe	87.70	88
TextCNN	Random Embedding	86.30	86
BERT	Contextual Embedding	88.98	89

From the summary, it is clear that only TextCNN with GloVe embedding and the BERT model outperform the baseline. There could be reasons related to models, preprocessing, and data. First of all, the advanced models are complicated with much more parameters than the baseline, plus the limited number of training data, which could lead to neural networks over-fitting easily. Specifically, LSTM and TextCNN start converging after the first three epochs during training as the gaps between training error and validation errors expand afterward, while the validation loss for BERT even starts increasing after the first epoch.

The second possible reason behind this could be that some reviews include a positive or negative overview of the movie but hold the opposite views on one or two aspects in the following as well, which can be confusing even to readers. One negative review with positive prediction and corresponding processed sample from TextCNN and BERT results is indicated below.

Negative review with positive prediction: *Cheaply pieced together of recycled film footage, music and ideas, this film cannot really be called "well". But for me, when I watched it as a teenager, it was quite amusing. (I didn't know BATTLE BEYOND THE STARS before.) In retrospect, it has got something nostalgic, regarding the SF wave of the early eighties and the special effects of this time. Its trashy old-fashioned look and its naivety provide a certain attraction. To enjoy this movie I recommend to concentrate on the paternal relationship between the characters of Vince Edwards and David Mendenhall. In addition, I liked the idea that a bunch of scoundrels discovers its heroic qualities after been unwillingly confronted with the challenge to take care of a child.*

.

Processed: *cheaply pieced recycle film footage music idea film call watch teenager amusing not know battle star retrospect get nostalgic wave early eighty special effect time trashy oldfashione look naivety*

provide certain attraction enjoy movie recommend concentrate paternal relationship character vince edwards david mendenhall addition like idea bunch scoundrel discover heroic quality unwillingly confront challenge care child

From the text content, it is clear that a negative attitude is given at the beginning, however, the writer mentioned that he or she as a teenager felt 'quite amusing', and also admired 'heroic qualities' of characters at the end. Apart from the opposite attitude from writers, another reason behind could be the opposite emotional words of movie content review. An example of this is shown below.

Positive review with negative prediction: *There is only one film I can think of that might be as good or better than this one when it comes to Bugs Bunny and Daffy Duck–ALI BABA BUNNY. However, determining which is THE best is irrelevant–just watch them both and enjoy. I compared this to ALI BABA BUNNY because both feature Daffy at his absolute worst–greedy, nasty and very funny in the process. However, I think I prefer RABBIT SEASONING simply because Bugs is also pretty awful in this one–doing horrible things right back to Daffy every time Daffy tries a dirty trick. The film begins with Daffy leaving rabbit tracks right up to Bugs' hole in the hope that a hunter (naturally, it's Elmer) will blast the rabbit and leave Daffy alone! Not to be outdone, Bugs time and again takes all of Daffy's tricks and turns them around–and in most cases it involves Daffy getting shot in the face! It's all very, very clever and funny and I don't care how old you are, this cartoon will make you laugh unless you are a grouch. I especially love the great and unexpected ending, but I won't say more, as I don't want to spoil the surprise.*

.

Processed: *film think good well come bug bunny daffy duck ali baba bunny determine good irrelevantjust watch enjoy compare ali baba bunny feature daffy absolute worst greedy nasty funny process think prefer rabbit season simply bug pretty awful onedoe horrible thing right daffy time daffy try dirty trick film begin daffy leave rabbit track right bug hole hope hunter naturally elmer blast rabbit leave daffy outdone bug time take daffys trick turn aroundand case involve daffy getting shoot face clever funny not care old cartoon laugh grouch especially love great unexpected ending will not not want spoil surprise*

As can be observed, the writer holds a positive point of view toward the movie, but many negative words are included as introducing the movie characters, such as 'worst–greedy', 'pretty awful', 'dirty trick' and etc. These words could lead to a negative impact on model prediction.

The third reason could be produced from preprocessing phase. Some negative words like 'not' and 'cannot', could be removed during preprocessing, which might be able to produce the opposite meaning and disturb the training process. The short wrongly classified example below shows the impact of this.

Raw text with negative label: *The movie is not as funny as the director's preceding (and only other) movie, Shanghai Noon. Showtime did have its moments, but it did not satisfy me. Why it needed to be so foulmouthed, I don't know, but I give Showtime **/****

.

Processed text: *movie funny director precede movie shanghai noon showtime moment satisfy need foulmouthed not know showtime*

As can be seen in the gray box above, the raw text mentions that 'The movie is not as funny as...', and 'not satisfy me', but the negative words 'not' are removed from the raw text. From the processed data, the left words 'movie funny' and 'moment satisfy' seem to imply that the review is positive instead. Another example of this is the negative review with positive prediction mentioned above, where the 'cannot' in the first sentence which is also the only sentence indicating a negative attitude is removed. This phenomenon produces a great impact on those short reviews or the ones with many negative words.

Comparing the work with other research, it is mainly the preprocessing phase that different from the work in this project. Take the Naive Bayes model as an example, many researchers use the different corpus to tokenize rather than spacy in this project, for example, Yasen and Tedmori (2019)

applied NLTK during tokenization achieving 81.83% in accuracy for the Naive Bayes model. Apart from tokenization, the number of features selected also affects the model performance by about 2% in research from Mahyarani et al. (2021).

Conclusion

A set of IMDb data is used in training models in this project to analyze whether peoples hold a positive or negative point of view on movies. Four categories of models have been constructed and compared in this project. Naive Bayes classifier as the baseline model performs surprisingly well on the test set reaching 87.36% in terms of accuracy, while LSTM and TextCNN model with random word embedding achieve lower accuracy than the baseline. The Best model among these is the BERT BASE model with 88.98% of samples correctly classified followed by TextCNN with GloVe word embedding.

The surprising thing is that advanced models do not outperform the baseline much, even though there are two models that underperform the Naive Bayes. From the analysis of the wrongly classified samples, there could be three potential reasons behind it. The first is the limited number of samples, leading to complex model over-fitting quickly at the first several epochs. The second is the mixed attitude of writers causing the mixed positive and negative words fed into models. Finally, the removed negative words during preprocessing that changes the meaning oppositely could also be one potential reason.

Future Work

For future work, a larger data set and high-performance GPU can be applied to further verify whether advanced models have already reached their potential. Apart from the models evaluated in this project, more hyper-model models such as recurrent convolutional neural networks(RCNNs). The BERT LARGE model that requires more time and resources is also a potentially better model to try.

Bibliography

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

Yoon Kim. Convolutional neural networks for sentence classification in: Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp), 1746–1751.. acl. *Association for Computational Linguistics, Doha, Qatar*, 2014.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Meta Mahyarani, Adiwijaya Adiwijaya, Said Al Faraby, and Mahendra Dwifebri. Implementation of sentiment analysis movie review based on imdb with naive bayes using information gain on feature selection. In *2021 3rd International Conference on Electronics Representation and Algorithm (ICERA)*, pages 99–103. IEEE, 2021.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Daniel Soutner and Luděk Müller. Application of lstm neural networks in language modelling. In *International Conference on Text, Speech and Dialogue*, pages 105–112. Springer, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Mais Yasen and Sara Tedmori. Movies reviews sentiment analysis and classification. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 860–865. IEEE, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.