Zhejiang Gongshang University

SUSSEX
ARTIFICIAL INTELLIGENCE
INSTITUTE

Cover Sheet for MSc Intelligence and Adaptive Systems Project
Final Report

CANDIDATE NUMBER: **238394**	**DEPARTMENT:** **Sussex AI Institute ZJSU**
TITLE OF MODULE: **MSc Intelligent and Adaptive Systems Project**	
TITLE OF ASSIGNMENT: **Dissertation**	
Date due: **5 January 2022**	**Date submitted:** **5 January 2022**

DECLARATION:

In making this submission I declare that my work contains no examples of misconduct, such as plagiarism, collusion or fabrication of results.

I confirm that I discussed with my supervisor the ethical implications of my project and if an ethical review was required, it was submitted and approved before the commencement of data collection/experiments.

SIGNATURE: Shukai Che

- Work submitted late will be subject to the penalties set out at
- http://www.sussex.ac.uk/academicoffice/1-3-2.html
- Marks are provisional and subject to ratification by the relevant Examination Board
- Appeals can only be lodged against alleged procedural improprieties AFTER the ratification of marks
- Comments made by the tutor are intended to advise the student in developing their knowledge and abilities within the subject

NB. Tutors are normally required to return marks and feedback to students within 3 weeks of the date set for submission.

NB. Some assignments will be kept for ratification by the Exam Board. Students are encouraged to take a copy of their work before submission as it may not be returned.

Zhejiang Gongshang University

SUSSEX ARTIFICIAL INTELLIGENCE INSTITUTE

Coursework declaration – to be included with coursework when not personally submitting

In making this submission I declare that my work contains no examples of misconduct, such as plagiarism, collusion or fabrication of results.

Title of Module: .....MSc...Project.....

Title of Assignment: .....Dissertation.....

Date: ..... 5 January 2022.....

Zhejiang Gongshang University

SUSSEX ARTIFICIAL INTELLIGENCE INSTITUTE

PROJECT TITLE:

A Comparison of Naive Bayes, LSTM and Transformer for Sentiment Analysis using Nature Language Processing

Word Count：

10007

SUPERVISOR NAME:

David Weir

CANDIDATE NUMBER:

238394

DEGREE COURSE:

MSc Intelligence and Adaptive Systems Project

DATE:

5 January 2022

A Comparison of Naive Bayes, LSTM and Transformer for Sentiment Analysis using Nature Language Processing

Abstract

With the development of social networks and websites, there are more and more comments containing users' emotions, and the text information is more and more diverse and contains great value. Therefore, **sentiment analysis** has gradually become a research hotspot. The decision-maker gets the text of user comments and opinions from the website and further improves the decision based on the text analysis results. But user reviews are largely unstructured, so they need to be processed to extract valuable information. To extract information from the input data set, these comments and blogs can be grouped into groups of different emotional levels.

I choose three methods, **Naive Bayes (NB), Long Short-term Memory (LSTM)** and **Transformer**. Three different analysis methods are considered for sentiment classification. I will compare the performance of Naive Bayes, LSTM and Transformer on two different data sets, and calculate **Macro F1-score, Micro F1-score, accuracy, recall** and **precious** as **evaluation indicators**. The model with a large evaluation index has a strong emotion prediction ability. The first is a **hotel review dataset** with 20491 reviews and five sentiment categories, one for each review. The second is the **financial review dataset**, with 4837 comments and three sentiment categories, negative, positive and neutral for each review. The two data sets are not the same in number and type, so they are both imbalanced datasets.

Firstly, I'll examine how **each model** behaves on two different datasets. Then the performance of the **3 models** is compared on the same data set, which model performs best, and which model performs worst. The **research content** is shown below:

Naive Bayes

1) On which dataset does NB have a stronger predictive power, the financial review dataset or the hotel review dataset?

2) Whether the prediction power of the NB method increases when the sentiment category of the hotel review dataset changes from 5 categories to 3 and 2 categories?

3) When the financial review dataset is shuffled or not shuffled, the prediction ability of the NB method is better in which kind of dataset?

LSTM

1) Which dataset does Embedding+LSTM predict more accurately, financial review dataset or hotel review dataset?

2) Word2Vec+LSTM predicts more accurately on the financial review dataset than the hotel review

dataset?

3) In the financial review dataset, which model has better prediction ability, Embedding+LSTM or Word2Vec+LSTM?

4) In the hotel review data set, which model has better prediction ability, Embedding+LSTM or Word2Vec+LSTM?

Transformer

1) Which is a better predictor of the financial review dataset, BERT or RoBERTa?

2) Which is a better predictor of hotel review dataset, BERT or RoBERTa?

3) On the financial review data set and the hotel review data set, BERT predicted more accurately on which data set?

4) RoBERTa's predictions are more accurate on the financial review dataset or the hotel review dataset?

Comparison of three models

1) Which model, Naive Bayes, LSTM or Transformer, has the best predictive power on the financial review dataset?

2) Which model, Naive Bayes, LSTM or Transformer, has the best predictive power in the hotel review dataset?

3) On which of the two datasets do the 3 models perform better overall, considering the baseline and assessment indicators?

After doing the experiments, I concluded these research questions. There are fewer categories in the hotel review dataset, the data set is more balance, and the classification capabilities of Naive Bayes are improved. In category 2, the classification ability of Naive Bayes is best. Naive Bayes reviews predict results more accurately in the shuffled financial dataset. Naive Bayes categorizes more accurately on financial review datasets and hotel review datasets. In hotel data sets, Embedding+LSTM performs better than Word2Vec+LSTM. On both datasets, Word2Vec+LSTM performs better in finance and Embedding+LSTM also performs better in finance. On the same hotel review dataset, BERT did better. On the same financial review data set, RoBERTa did better. On both datasets, BERT and RoBERTa performed better and predicted more accurately on the Financial Review dataset. In the same dataset, Transformer has the best classification ability, LSTM is second and NB is the worst.

Content

1 Introduction

With the development of computer technology and the proliferation of new social networks and websites, the number of comments from users has increased dramatically. Every day, countless comments are posted on the Internet that covers a wide range of users' emotions. Sentiment analysis, also known as opinion mining, is text data based on user comments. It is a process of analyzing, processing, classifying and reasoning texts with subjective feelings and opinions. People can directly view the results of sentiment analysis to understand the attitude and emotional tendency of online users towards a certain organization, people and things. By using sentiment analysis, users' emotional tendencies can be quantified, their subjective emotions can be turned into objective data, and meaningful information can be obtained from objective data, which will provide great value for any product, event or person. The research on emotion analysis still has great value and significance in today's society **[1]**.

For text sentiment analysis, the aim is to transform unstructured target sentiment text into structured text that can be easily recognized and processed by the computer. Therefore, it is necessary to identify and judge the meaningful information units, to obtain the evaluation subject and evaluation view information. These comments and ideas are vectorized from textual data into a digital matrix. These matrices are then fed into different models, the reviews are classified, and the classification results are evaluated using different parameters, based on which the strengths and weaknesses of different analysis methods are evaluated **[2]**.

At present, the commonly used methods to obtain the information of evaluation point of view are mainly divided into the method based on emotion dictionary and rules, the method based on traditional machine learning, the method based on deep learning and the method of multi-strategy mixture. The foundation of the former two is the construction of the emotion dictionary, the quality of the emotion dictionary directly determines the subsequent emotion judgment. Machine learning-based methods train an emotion classifier with emotion labelled data and then predict the emotional tendency of new sentences in the test set **[3]**. The commonly used shallow machine learning classification algorithms include maximum entropy, naive Bayes and support vector machine. Deep learning is a subset of machine learning, and most sentiment analysis is based on CNN, RNN, LSTM and so on.

2 Research Question

In this study, I will examine how each model behaves on two different datasets firstly. Then the data results of the 3 models are studied.

Naïve Bayes

1) Change the number of categories of the hotel reviews dataset to make the dataset tend to be balanced, compare the performance of the model and analyze the reasons. (5 classes,3 classes,2

classes). Whether the prediction power of the NB method increases when the sentiment category of the hotel review data set changes from 5 categories to 3 and 2 categories?

2) Determine whether to shuffle the data set, observe the changes in accuracy, and analyze the causes. When the financial review dataset is shuffled or not shuffled, the prediction ability of the NB method is better in which kind of dataset?

3) Compare the performance of the two datasets on the NB model. On which dataset does NB have a stronger predictive power, the financial review dataset or the hotel review dataset?

LSTM

4) Studies the performance of Embedding+LSTM and Word2Vec+LSTM from Google on the same data set. In the financial review dataset, which model has better prediction ability, Embedding+LSTM or Word2Vec+LSTM? In the hotel review dataset, which prediction power is better, Embedding+LSTM or Word2Vec+LSTM?

5) The performance of Embedding+LSTM on two data sets is studied. The performance of Word2Vec+LSTM from Google on two datasets is studied. Word2Vec+LSTM predicted more accurately on the financial review datasets than the hotel review datasets? Which dataset does Embedding+LSTM predict more accurately in the financial review and hotel review datasets?

Transformer

6) The study compares the performance of Bert and Robert on the same data set. Which is a better predictor of the financial review dataset, BERT or RoBERTa? Which is a better predictor of hotel review dataset, BERT or RoBERTa?

7) Study BERT's performance on two data sets. The performance of RoBERTa on two data sets was studied. On the financial review dataset and the hotel review dataset, BERT predicts more accurately on which dataset? RoBERTa's predictions are more accurate on the financial review dataset or the hotel review dataset?

Compare Naïve bayes, LSTM and Transformer

8) Study the performance effects of naive Bayes, LSTM and Transformer on the same data set. Select the model that performs best. Which model, Naive Bayes, LSTM or Transformer, has the best predictive power on the financial review dataset? Which model, Naive Bayes, LSTM or Transformer, has the best predictive power in the hotel review dataset?

9) Change the types of data sets and study the prediction accuracy of a model for different types of data sets, naive Bayes, LSTM and Transformer. On which of the two datasets do the 3 models perform better overall, considering the baseline and assessment indicators?

3 Background and Related Work

3.1 Background

Sentiment analysis is a specific task in text classification, which is to analyze subjective texts with emotional colours (positive and negative meanings/positive and negative meanings) to determine the viewpoint, preference and emotional tendency of the text. For example, the text "this is a book to read" is positive, and "this book is ugly" is negative. The subjective texts studied include customers' comments on a product and the public's views on a hot news event. Through these texts, businesses can provide consumers with decision-making references, and relevant organizations can understand public opinion, but manual analysis costs a lot of money. The goal of text sentiment analysis is usually to use computer technology to mine the viewpoints and emotions expressed in the text, which can be divided into subject-related sentiment analysis and subject-independent sentiment analysis **[4]**. Topic correlation refers to that in addition to obtaining the emotional polarity of the text, relevant topics in the text should be extracted to focus on what kind of views and comments on the attributes of a certain event and item. Subject-independent sentiment analysis simply judges the emotional polarity of a document or sentence, regardless of the subject or attribute of the emotion.

Sentiment analysis is generally used to extract and analyze emotions contained in comments on news, blogs, post bars, forums, micro-blogs, and commodities (various products and services) on e-commerce websites, providing basic data for public opinion supervision, current affairs hot spot tracking, news recommendation, commodity evaluation and other applications **[5]**. In conclusion, sentiment analysis has a large number of application scenarios and provides a useful tool for market research. For example, the following application scenarios:

1) Social media and product reviews. Compared with other text data, social networks and product reviews tend to have a large number of texts, high real-time performance, rich information and greater potential value. For example, from product reviews, users' satisfaction with products can be understood and good marketing strategies can be formulated. Government departments can understand the emotional tendency of citizens on hot events and grasp the direction of public opinion, monitor public opinion in a more timely and effective manner and provide support for the formulation of relevant policies. However, social network and commodity comment texts are characterized by short text length, non-standard wording and many emerging popular words, so emotional analysis of these texts is more academic and technical challenges.

2) Quantify investments. Use AI technology to make investments, like buying and selling stocks. In quantitative investing, models are used to predict the future stock market and then make buying and selling decisions **[6]**. The common methods are using multi-factor model, time series prediction model and so on. In addition, many companies use sentiment analysis techniques to monitor in real-time whether a lot of negative news is being generated. Once discovered, it needs to be dealt with by the PR department. For example, a financial company will grab the text information of the public

account and forum related to the company in real-time and automatically judge whether there is a lot of negative emotion in it.

3) Competitive product analysis. Competitive product analysis is also a very important part. Through the emotion analysis technology to judge the competitors launched what products, public opinion of the product is what. Through this information, we can understand our competitors and make timely adjustments to our products.

3.2 Related Work

3.2.1 Lexicon-based Approach

The method based on the emotion dictionary mainly relies on the construction of the emotion dictionary, which means to obtain the emotion value of the emotion words in the document by using the emotion dictionary and then determine the overall emotion tendency of the document by weighted calculation. When using this method, the relationship between words is not considered, and the emotional value of words will not change with the change of application field and context **[7]**. Therefore, it is necessary to build a relevant emotion dictionary for specific fields to improve the accuracy of classification. In the absence of a large number of training data sets, the method based on dictionaries and rules can achieve relatively good classification results and is easy to understand. However, as network terms continue to emerge, emotion dictionaries need to be constantly updated and expanded to improve the accuracy of classification **[8]**. Emotion dictionary also has some limitations. Firstly, the discrimination and selection of emotional words depend on prior knowledge and experimental design. Secondly, it is necessary to construct corresponding domain emotion dictionaries for different domains, and the classification effect of cross-domain emotion analysis is not good.

3.2.2 Machine Learning Approach

The method based on machine learning refers to the selection of emotional words as the feature words, the text matrix, Logistic Regression, Naive Bayes, support vector machine (SVM) and another classification meth. The final classification effect depends on the choice of training text and correct emotion labelling. From the perspective of machine learning, text sentiment analysis is regarded as a supervised or semi-supervised classification problem. SVM, NB and ME are generally used as classifiers. The processing process includes text representation, feature extraction and classifier model establishment, and the main work is how to construct and learn features with the more representational ability **[9]**. The initial machine-learning algorithm uses an emotion dictionary to construct feature representation of text and then uses Naive Bayes (NB), Support Vector Machine (SVM) and Maximum Entropy (ME) model to classify positive and negative emotions **[10]**. Then, many people began to try to use machine learning methods for text sentiment analysis, and many methods were proposed.

For machine learning methods, a great difficulty is the acquisition of training data. Training samples can be obtained through manual annotation, but this method is very labour-intensive and cannot obtain a large amount of annotation data **[11]**. For microblogs, comments and other texts, emoticons in the text can be used to annotate the text. This annotation method will introduce some noise, but it can easily obtain a large amount of training data and still achieve good results.

3.2.3 Deep Learning Approach

Deep learning is a subset of machine learning, is a multi-layer neural network application in the study, covering multiple areas, involving more knowledge, to solve the previous machine learning is difficult to solve the problem of the image and voice processing and emotional analysis fields have achieved good effect, so being singled out as a discipline **[12]**. Emotion analysis methods based on deep learning mainly refer to text modelling, feature extraction and emotion classification based on the constructed deep network model. Deep neural network can have a strong ability of data feature representation. Because of the existence of multiple nonlinear hidden layers, a multi-level neural network can learn almost any distributed data feature. The biggest advantage of a deep network is that it can automatically learn multi-level feature representation, which is learned layer by layer and closer to semantics at a higher level. Automatic feature learning can save a lot of labour-intensive feature extraction and obtain a wide range of feature representations. At present, deep learning models include CNN, RNN, LSTM, Transformer, BiLSTM, Gate Recurrent Unit (GRU) and attention mechanism, etc. Text emotion analysis has its difficulties and challenges, such as language ambiguity, polysemy, emotion inversion, etc., these language characteristics may be difficult to be captured only according to the processing method of general text classification, resulting in a bias in emotion judgment **[13].** How to design a reasonable network structure and objective function for the specific difficulties of text sentiment analysis is the key to further improving the effect of deep learning on text sentiment analysis **[14]**.

4 Algorithm

4.1 Naïve Bayes

1) Joint probability and conditional probability

Joint probability: the probability of events occurring at the same time P (A and B). The probability of event A and event B occurring together. It is the probability of intersection microblogs of two or more events denoted P (A∩B).

Conditional probability P (B | A): B event A conditional probability is in the case of A known event has occurred to the probability that the event will occur.

2) Bayesian classifier and Gaussian Bayesian classifier

The Bayesian classifier is a machine learning classification model based on the naive Bayes formula, which is as follows. In a naive Bayes classifier for dichotomous sentiment analysis with words as the basic unit, the prior probability, likelihood probability and posterior probability may be in the following form:

Prior probability: according to the statistical results, judge the possibility P(A) of text classified as positive emotion appearing in all texts, where positive emotion is A result

Likelihood probability: when the text classification is positive emotions, the possibility of words in the text B, P (B | A). The occurrence of the word B is a cause, that is, there has been the result of "positive emotion", and based on the occurrence of this result, there will be the probability of the cause of the word B.

Posterior probability: The probability that the text is a positive emotion based on the occurrence of the word B (reason) in the text.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

A traditional naive Bayes sentiment analysis classifier is the process of using these prior probabilities and likelihood probabilities to solve the posterior probabilities. Among them, the solution of prior probability is relatively simple, and the number of all forward texts can be directly counted, which can be obtained by dividing the total number of texts. The likelihood probability can be solved in various ways: the relatively simple way is based on word frequency, counting the number of words appearing in all texts classified as a positive emotion, and counting the percentage of word B appearing in them. However, the disadvantage of this method is that the words' I 'and' you 'are not meaningful in the task of judging the emotion of the text, but their word frequency is very high, which will interfere with the judgment of the emotion analysis. To remedy such shortcomings, TF-IDF can be used as a likelihood probability on this basis. TF-IDF is a method used to judge the importance of words in documents, and its value range is [0,1]. The main idea of TF-IDF is that if a certain word or phrase appears frequently in one article and rarely in other articles, it is considered that the word or phrase has good classification ability and is suitable for

classification. TF represents the frequency of entries in document D **[15]**. The main idea of IDF is that if the number of documents containing the term t is smaller, that is, the smaller n is, the larger IDF is, it indicates that the term T has a good ability to distinguish categories. When TF-IDF is used as a likelihood probability, the probability of words contributing to classification is very high. Words such as' love 'and' amazing 'obviously contain positive emotion, and their likelihood probability in positive emotion texts will be higher **[16]**.

Gaussian Bayesian classifier

The emphasis of the Bayesian solution lies in the solution of likelihood probability, and there are many ways to calculate likelihood probability. In this paper, a new Gaussian Bayesian classifier based on WORD2VEC is proposed. In the Gaussian classifier for text classification, word frequency is no longer used to calculate the likelihood probability, but documents are converted into vectors, and the vector distribution of all documents under a category is assumed to satisfy the multidimensional Gaussian distribution. The probability density function of multi-dimensional Gaussian distribution is used to calculate likelihood probability.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

In this paper, the average word2veC of all words in the pre-processed document is taken as the vector of the document. This is because word2vec is closer between positive words such as good ', 'fancy', while these words are far from the negative vector of 'unhappy', 'dejected'. When there are more positive words than negative words in a sentence, that is, its semantics are closer to positive words, and the average vector of the whole sentence will be closer to positive words. So, you can use the average vector of all the words to represent the semantics of the entire sentence.

In our training data, all sentences were classified into different categories due to their semantic differences. That is, all sentences in the same category have similar semantics. When we represent the semantics of a sentence with vectors, the vector distances between vectors in the same category are shorter. Just as in clustering in **Figure 1**, vectors of sentences in the same category are clustered into clusters in higher dimensional space, as shown in the figure below. The distribution of all sentence vectors in the same category satisfies the multi-dimensional Gaussian distribution, that is, most vectors are concentrated in the center of the cluster, approximately close to the edge, and the number of vectors is less **[17]**.

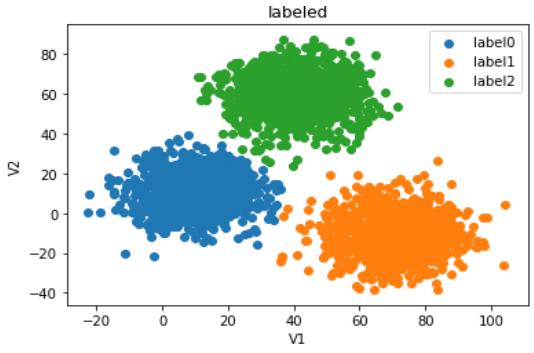


Figure 1 The document vector conforms to a Gaussian distribution

4.2 LSTM

A long short-term memory network (LSTM) is an improved model of RNN, which can learn long-term dependencies. Different from RNN, each LSTM unit consists of four basic structures, namely memory unit, input gate, output gate and forgetting gate. During a given time interval, memory cells record information, and the gate controls the flow of information between memory cells. There are two stages in this process: the hidden state and the cellular state **[18]**. The cell state simulates the memory of the LSTM unit, and the hidden state records the results of the unit. Thus, the hidden state and cell input determine the information stored in the memory cell, which is used to discard or store new information **[19]**. The specific steps are as follows:

Step1: Forgetting gate in **Figure 2** uses the sigmoid function to judge the information recorded in cell state in time step T. It receives the current input XT and the previously hidden state HT-1.

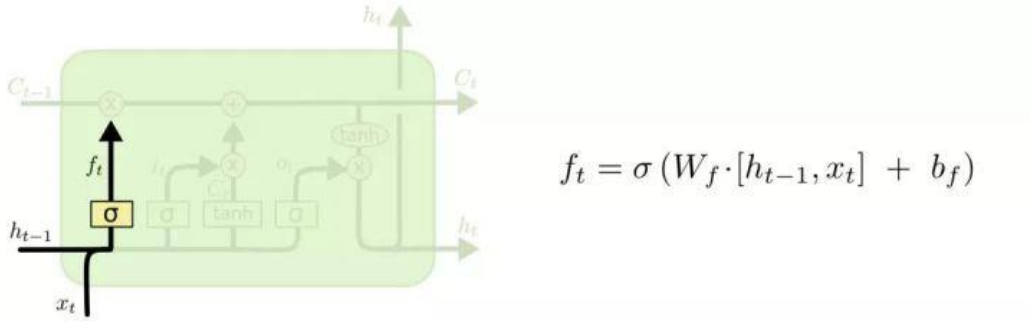


Figure 2 Forget gate

Source: https://blog.csdn.net/weixin_44901453/article/details/105383056

Step2: The input gate in **Figure 3** controls whether new information is to be stored in the cell state. It uses the sigmoid function to receive the current input XT and the previously hidden state HT-1. The Tanh function will generate candidate vectors, which will be further combined with the cell state to update the current cell state.

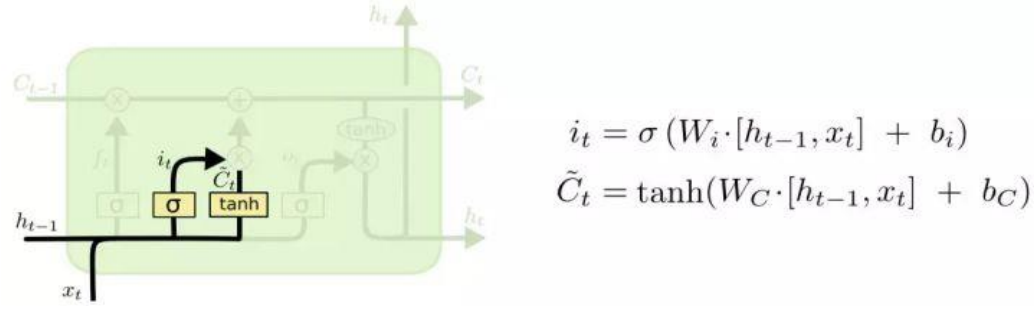
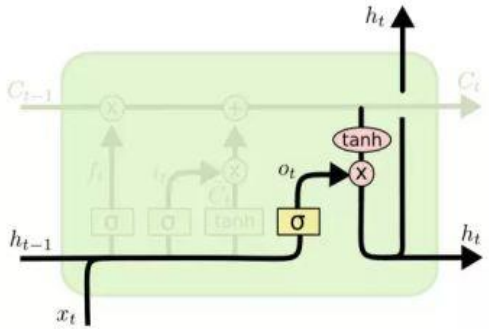


Figure 3 Input gate

Source: https://blog.csdn.net/weixin_44901453/article/details/105383056

Step3: The model determines the output based on the cell state in **Figure 4**. We first use a sigmoid function to determine which part of the cell state needs to be outputted, and then process the cell

state through the TANH layer, multiplying the two together to get the information we want to output.



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

Figure 4 Output gate
Source: https://blog.csdn.net/weixin_44901453/article/details/105383056

The forgetting gate controls gradients through it, helping to alleviate the problem of gradient disappearance or explosion in RNN.

4.3 Transformer

The transformer has two parts, encoder and decoder. BERT and Roberta's structure only takes the encoder part in Transformer. Bert and Roberta are a semantic extraction model. Each token entered into the model passes through many layers. The function of Layer is to more accurately represent the semantics of its input with vectors, and its input and output are vectors. In each layer, the multi-head attention mechanism is used to extract the semantics of the text. The attention mechanic is at the heart of Bert and Roberta **[20]**.

Each attention header contains three intermediate weight matrices Wq, Wk, and Wv. Each attention layer in **Figure 5** processes the input vector to generate a new vector as follows:
Step 1: Input X to multiply Wq, Wk and Wv matrices respectively to obtain Q, K and V.
Step 2: Q, multiply the K_T matrix to obtain the correlation between the words in X and scale (to prevent the result from being too large, divide by the root mean square of their dimensions).
Step 3: The degree of relevance in the second step is normalized by the softmax function to obtain the degree of relevance between each word and other words after normalization.
Step 4: Multiply the relevancy matrix of the third step by V, i.e., the weighted sum, to get the new vector coding of each word.

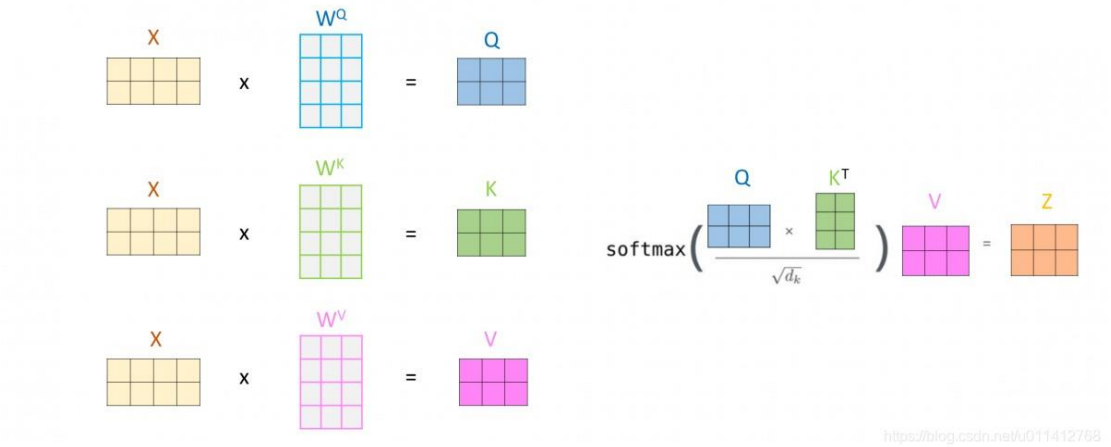


Figure 5 Attention mechanism

Bert and Roberta used multi-head attention. There are multiple sets of Wq, Wk, and Wv processing the input X simultaneously. Finally, all the attention output is grouped together and dimensionally reduced. Every attention can judge the semantic information contained in input X from a certain Angle. If the amount of attention is increased, the semantic information contained in input X can be comprehensively evaluated from multiple angles to enhance the semantic understanding of the model.

5 Methodology

5.1 Dataset

5.1.1 Introduction to Dataset

My dataset is from the Website Kaggle, and the dataset is publicly available. Data sets play an important role in subsequent model training. High-quality data sets can make the model better learn the features of the text, to improve the classification performance. I choose the hotel review data set and the financial review data set as the data set of naive Bayes, LSTM and Transformer models. In experiments, data sets need to be collected, preprocessed and analyzed.

1) Hotel review dataset

This dataset contains hotel reviews. Hotels play a crucial role in travelling and with the increased access to information new pathways of selecting the best ones emerged. With this dataset, consisting of 20k reviews crawled from TripAdvisor, this dataset can be used to predict the rating of hotel reviews. This dataset is imbalanced. Most comments are in the 150-300 range. The text length is long. The travel review data set contains 20,491 reviews. All comments were divided into five emotional intensities. On a scale of 1 to 5, reviews get more and more satisfying. Tag 1 represents the lowest customer satisfaction and tag 5 represents the highest customer satisfaction. The data set mainly contains satisfaction ratings and reviews in **Figure 6**. An example dataset is shown below:

label	Rating	Review
5	5	I love this hotel! excellent stay, delightful surprise stay Monaco.....
1	1	bad choice! The sound insulation of this hotel is very poor!.....
5	5	Nice choice in Iceland. This is the first time I came here, the waiters were very warm and there was a unique aroma on the quilt……

Figure 6 Examples of hotel review dataset

2) Financial review dataset

This dataset contains the sentiments for financial news headlines from the perspective of a retail investor. The dataset contains two columns, “Sentiment” and “News Headline”. The sentiment can be negative, neutral, or positive. It is also an imbalanced dataset. Most comments range from 10 to 100 words. The text length is short. The Financial Review dataset contains 4,837 reviews, a small number. All comments fall into three categories. The data set consists mainly of emotions, comments, and tags. Labels 1, 2 and 3 respectively represent Negative, Positive and neutral emotional intensity in **Figure 7**. An example data set is shown below:

label	Sentiment	Review
1	Negative	However, the growth margin slowed down due to the financial crisis.
3	Positive	Operating profit totalled EUR 21.1 mn, up from EUR 18.6 mn in 2007, representing 9.7 % of net sales.
2	Neutral	According to Gran, the company has no plans to move all production to Russia, although that is where the company is growing.

Figure 7 Examples of sentiments for financial news headlines

5.1.2 Dataset Analysis

1) Hotel review dataset

Out of 20,491 comments, the most satisfying tag 5, had 9,054 comments, or 44 per cent of all comments. Tag 4 had 6,039 comments or 29% of all comments. Hashtag 3 had 2,184 comments or 11% of all comments. Hashtag 2 had 1,793 comments or 9% of all comments. The least satisfied hashtag, 1, had 1,421 comments, or 7% of all reviews in **Figure 8** and **Figure 9**. The distribution of each category of the data set is uneven. The number of tags 5 is the largest, while the number of tags 3, 2 and 1 is almost the same, and they are all very small. This indicates that users tend to make positive comments, in reality, resulting in the uneven distribution of data sets. The distribution diagram of the dataset is as follows:

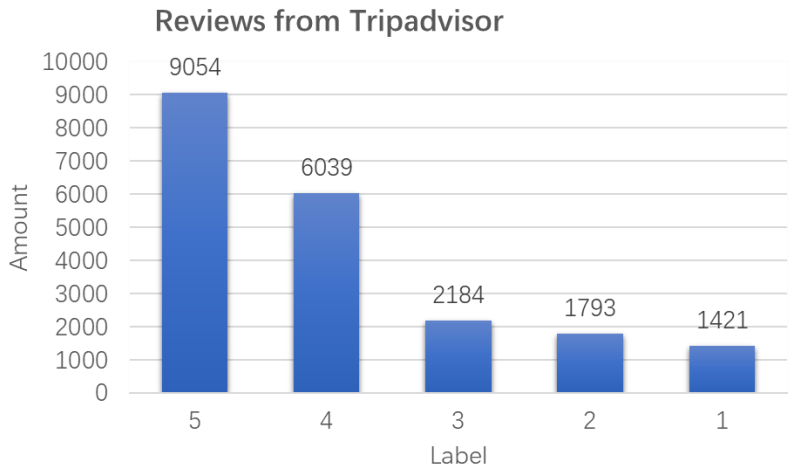


Figure 8 The amount of 5 categories

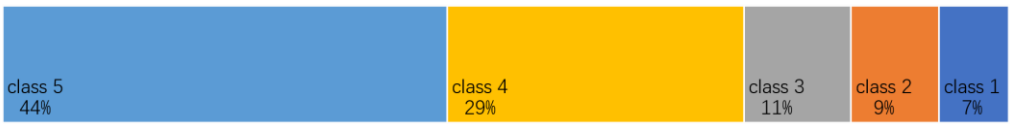


Figure 9 Ratio of 5 categories

Because the data set is imbalanced. I divided the data set into 5 classes, 3 classes, and 2 classes to make the data set relatively balanced, to study the influence of the number of classes on the accuracy of the model in **Figure 10**. Since the degree of satisfaction increases with rank, categories 1, 2, 3, and 4 are combined in turn. In the three cases, the data was most unbalanced when there were five categories. The data is most balanced when there are two categories.

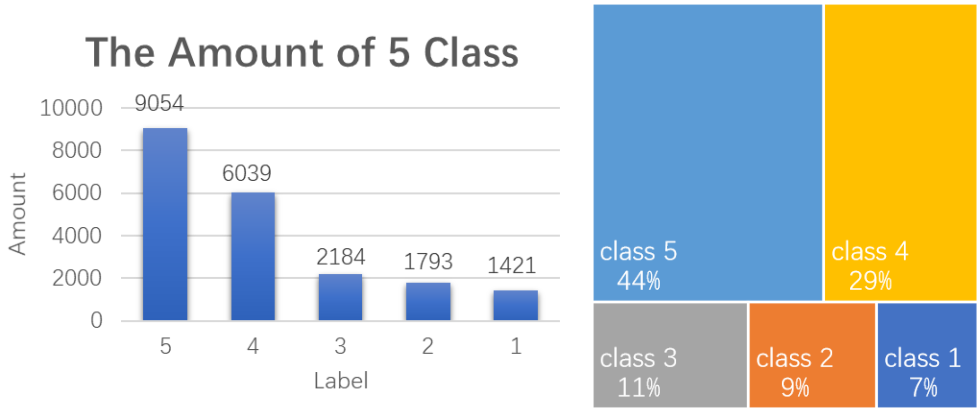


Figure 10 The quantity distribution of 5 categories

When reviews were divided into three categories, the proportion of the bulk data set tended to be more balanced than when reviews were divided into five categories in **Figure 11**. Categories 1, 2 and 3 are among the lowest levels of satisfaction. The number of comments on tag 5 is still much higher than the other two categories. But tag 4 and tag 1+2+3 have a similar ratio.

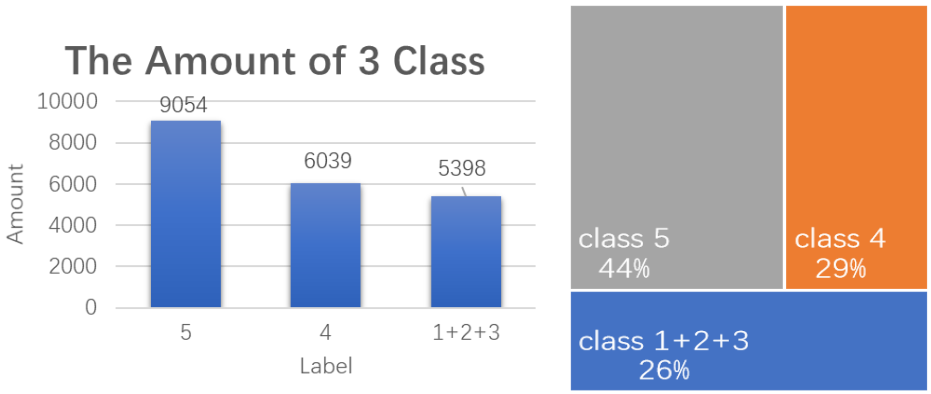


Figure 11 The quantity distribution of 3 categories

When the comments were divided into two categories, the proportion of the overall dataset was almost balanced in **Figure 12**. Comments in categories 1, 2, 3 and 4 are unsatisfied, while those in category 5 are satisfied. Tag 5 has roughly the same number of comments as tag 1+2+3+4.

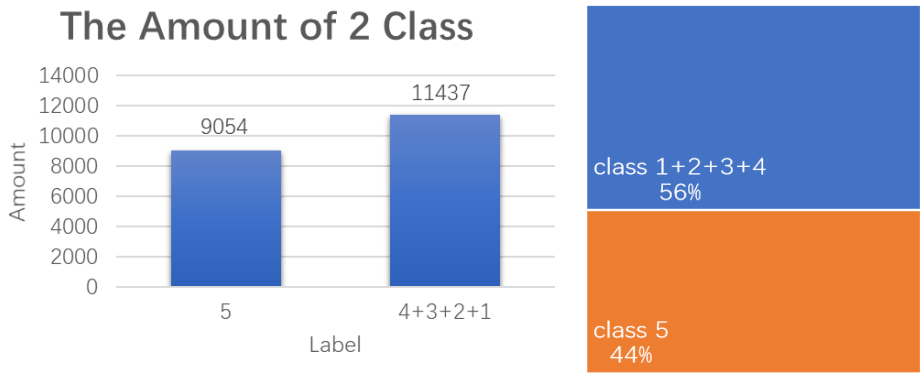


Figure 12 The quantity distribution of 2 categories

2) Financial review dataset

Of the 4,837 comments, neutrals were the most common, with 2,879, or 59 per cent of all comments. Negative comments were the fewest, with 604, or 12% of all comments. There were 1,363 positive comments, or 28 per cent in **Figure 13** and **Figure 14**. The quantitative distribution of data sets is as follows:

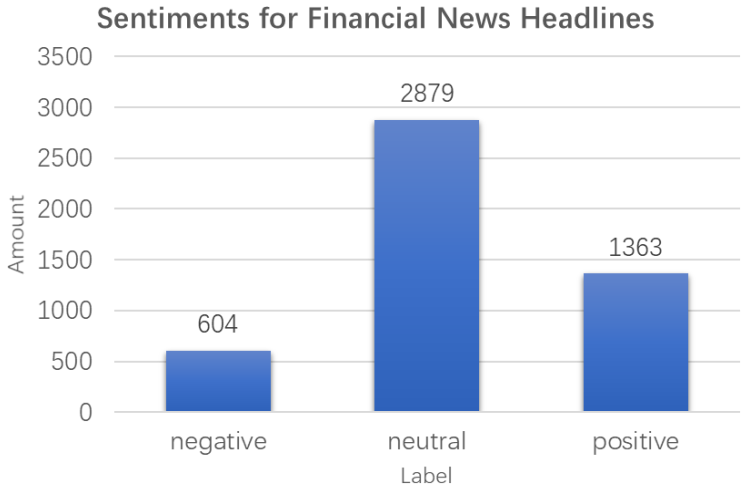


Figure 13 The quantity distribution of 3 categories

Figure 14 The quantity distribution of 3 categories

5.2 10-fold Cross-validation

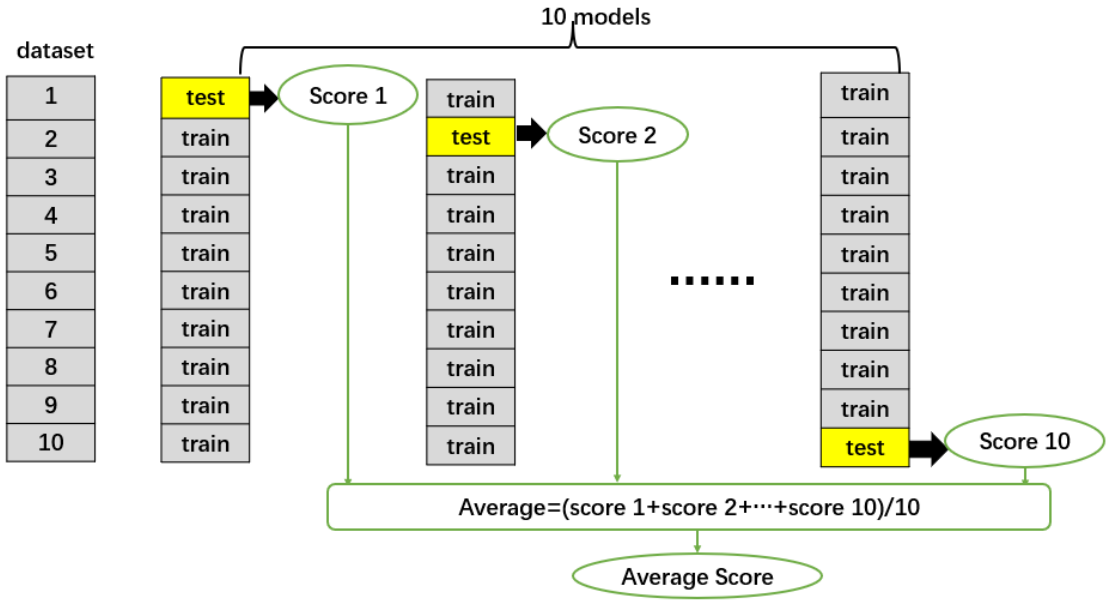


Figure 15 10-fold cross-validation

10-fold cross-validation in **Figure 15** is a way to test the accuracy of an algorithm. When the data set is small, it can make full use of limited data to find appropriate model parameters to prevent over-fitting and find accuracy. To reduce the contingency caused by the single partition of the training set and validation set, the existing data sets are fully used for multiple partitions to avoid the selection of accidental hyperparameters and models with no generalization ability due to special partition. Therefore, cross-validation can reduce contingency and improve generalization ability.

Steps:

1) Divide the data set into ten pieces. Each was randomly assigned different categories of data.

2) During the experiment, a fold was selected from the 10 folds as a temporary test set, and the folds were trained on the remaining 9 folds to calculate the performance on the temporary test folds. Repeat this with 9 folds as training data and 1 fold as test data. The corresponding evaluation index values were obtained for each experiment, including F1-Score, accuracy, recall and precious.

3) The average values of F1-score, accuracy, recall and precious of 10 times are used as the estimation of algorithm accuracy.

5.3 Naïve Bayes

The **code** of this method in **Figure 16** of using Gaussian Bayesian is built by myself.

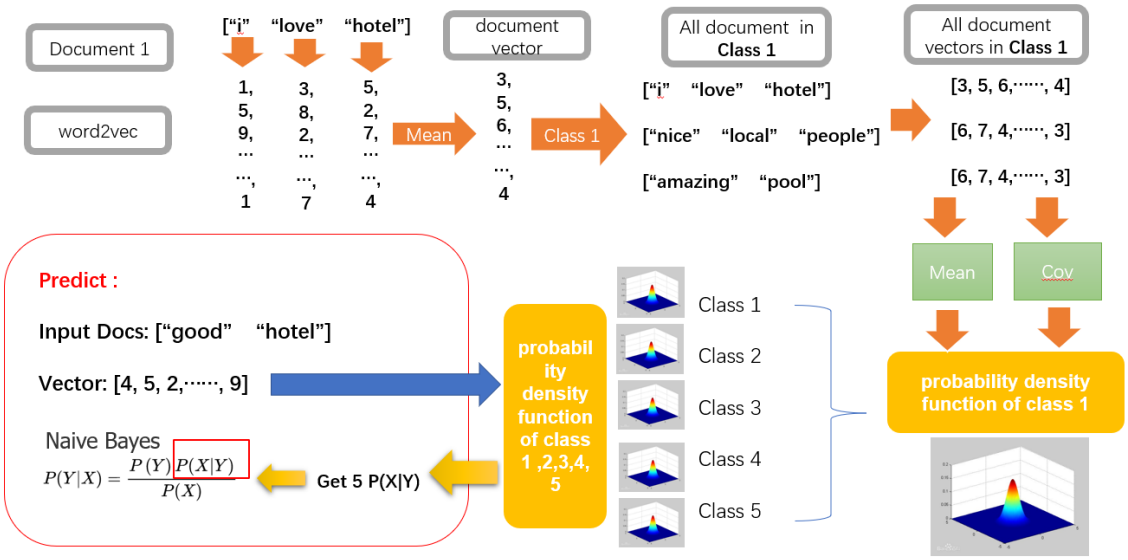


Figure 16 Gaussian Bayesian classifier processing

5.3.1 Data Preprocessing

All English text data that cannot be used directly need to be pre-processed before it can be input into the model for training or prediction. The main reason is that there will be a lot of noise in the original English text, which will increase the training amount of the model or make the prediction result inaccurate. The pre-processing involved in this paper includes the following steps in sequence:

'Best Hotel I Lived in Past 10 Years

1) Tokenization

word segmentation of all documents, that is, taking out all words in all documents in order. A document is made up of all the words in it, that is, the semantics of a document is made up of the semantics of all the words in it. Because a document is much longer than a word, it is much more difficult to directly extract the semantics of a document than to determine the semantics of a word. So, the first step is to slice the document in terms of words. The next step is to process each word, and finally, combine the semantics of all the words in a document to get the semantics of the document.

[' Best ', 'hotel', 'I', 'lived' and 'in', 'past', '10', 'years']

2) Num normalization

Replaces all numbers in the text with the identifier 'Num'. Sometimes there are a lot of numbers in the text. In practice, however, the specific value contained in the number often does not affect the classifier. Or the specific value of the number in the text is difficult to accurately extract semantics. Take, for example, the hotel review dataset used in this article, which reads' There are only two towels in the room, not enough 'and another reads' There are two televisions in the room, great'. The 'two' in the first number refers to negative emotion, and the 'two' in the second number refers to positive emotion. The exact value of the number cannot be used for classification, so replace it all

with 'NUM'.

[' Best ', 'hotel', 'I', 'lived' and 'in', 'past', 'NUM', 'years']

3) Wash out nonsensical punctuation marks, null values and stop words

In emotional analysis, some punctuation marks, such as exclamation marks, can serve to emphasize emotions. Some symbols, such as parentheses and other special symbols, have no meaning for the classification of the final emotion. Omission or mis collection sometimes occurs in data collection, and empty values or Spaces may appear, which are meaningless to the final emotion analysis, so it is necessary to clean and remove the empty values in the data set. Stop words refer to words that have no meaning, such as basic mood words, particles, etc. First, we need to import or download the stop words list, then filter the words sorted in the second step for stop words, and finally preserve the data that does not contain the stop words.

[' Best ', 'hotel', 'lived', 'past', 'NUM', 'years']

4) Lemmatize

A word in The English language has different tenses. In the emotion analysis task, we do not pay attention to word tenses. Taking the hotel review data set used in this paper as an example, the tenses of 'I have been to this hotel' and 'I am in this hotel' are inconsistent, but the difference in tenses does not affect the emotional semantics. During word restore, we used the POS_tag tool in the NLTK package to tag all the words in each document. After the part of speech is determined, the WordNet Lemmatize tool is used to restore the word to the prototype of the corresponding part of the speech

[' Best ', 'hotel', 'live', 'past', 'NUM', 'year']

5) Case Normalization

Converts all words to lowercase. The case of words has nothing to do with their semantics. However, in the subsequent processing, two words with the different cases but the same letters will be regarded as two different words by the model, thus affecting the model's judgment, so they are changed to lowercase.

[' best ', 'hotel', 'live', 'past', 'NUM', 'year']

5.3.2 Feature Extraction

For the preprocessed text data, Google word2vec is used to convert every word in each document into a 300-dimensional vector. This vector represents the semantics contained in the word. The closer two vectors are to each other, the more similar their semantics become. Average the vectors of all the words in a document, and you get a 300-dimensional vector. This vector can represent the combined semantics of all the words in a document, used to represent the semantics of the document. The semantic contained in a document is the feature of the text. Vectorization of the semantic of the text completes the feature extraction of the document.

5.3.3 Mean and Covariance Matrix Calculation

As mentioned above, all document vectors in a category satisfy the multidimensional Gaussian distribution. To calculate the probability density function of this Gaussian distribution, we need to know the distribution of descriptive data. To do this, we need to calculate the mean and covariance of each class. Taking the financial data set as an example, the data set contains three categories, and the text of each category is transformed into a 300-dimensional vector in the feature extraction stage. The calculated mean is also a 300-dimensional vector, and the covariance matrix is a 300-by-300 matrix. In this article, the GaussianNB tool in the Sklearn package is used to perform the computation. The tool automatically calculates the mean and covariance proofs and calculates the probability density function. Take the financial data set, which contains three categories. Each category will have its probability density function.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

5.3.4 Naive Bayes Computation

After the probability density function is calculated above, the training of the Bayesian model is completed. When we make a prediction, we calculate the probability that a given document vector will fall into each category. Because the denominator is P (B), the denominator can be omitted in the comparison, just the numerator. The numerator consists of likelihood probability and prior probability. The likelihood probability can be obtained by entering the vector into the probability density function of the corresponding class. The prior probability is the percentage of the data of the current category in the total data. How likely it is to get the current category by multiplying the two. Take the financial data set used in this article, which has three categories. When a new piece of data is entered for prediction, there are three possibilities, the most likely of which is the category the data belongs to.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

5.4 LSTM

I built the code for the LSTM method in **Figure 17**.

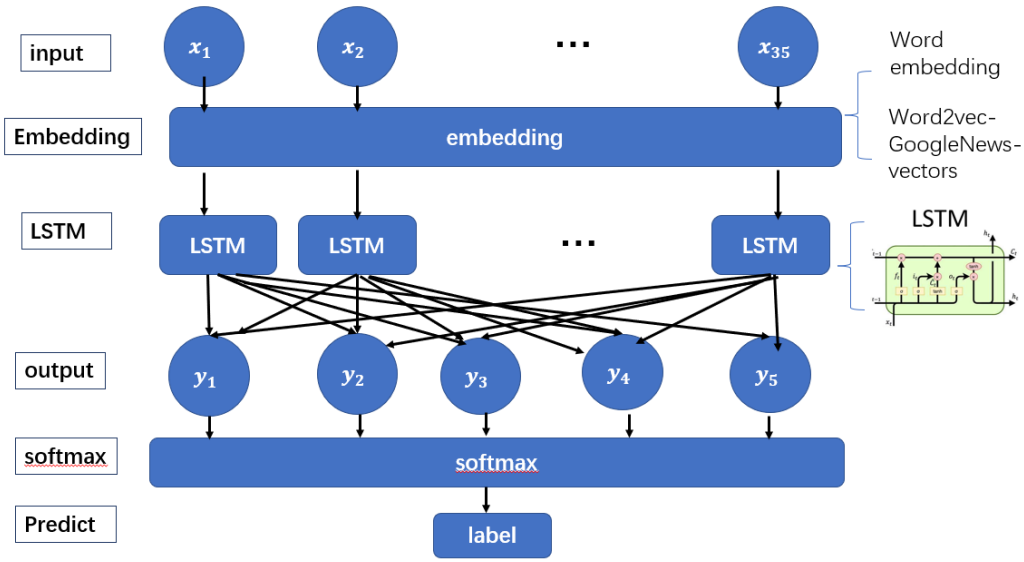


Figure 17 LSTM classifier processing

5.4.1 Data Preprocessing

It's the same process as Naive Bayes.

5.4.2 Embedding Layer

The text vectorization is carried out at the word embedding layer. Text vectorization is mainly to multiply the text data processing by the sequence vector form that can be directly received and processed by the LSTM layer. In this model, two methods: embedding and word2vec are used to vectorize text and generate word vectors. The results of the two methods are compared

The Word2Vec model used in this article is the Google News Word2vec model. This model uses the CBOW method for self-supervised learning. The specific method is to take one word from a text as the target word and the rest words as context words and train the model by using context words to predict the target word. The Word2vec model is pre-trained to use the model in a way similar to looking up a dictionary to translate words into corresponding vectors

This article also uses the embedding layer in the Keras package to vectorize words. The weights in the Embedding layer are random at first, and the word vector generated by such random weights can also be understood as random. Such vector is input into the model, and then the weight of the embedding layer is adjusted according to the loss function for backpropagation so that the

embedding layer can output a more accurate vector in the next training.

5.4.3 LSTM Layer

In the LSTM layer, LSTM neurons are mainly used to extract the corresponding semantic features from the embedding vector of each word. The output vector of the word embedding layer is the input vector of the LSTM layer. LSTM consists of a series of processing units, including input gate, output gate, forgetting gate and memory unit. These processing units can be thought of as neurons in a neural network. In the learning process, valuable information can be left according to the rules, and the information that does not conform to the rules can be discarded by the forgetting gate. These processors can effectively control the long-term and short-term memory of the input sequence. At any moment, each unit will receive the input from the previously hidden layer and the current input unit, which improves the model's ability to deal with the remote dependency problem.

5.4.4 Prediction Layer

I set up 100 LSTM neurons in the LSTM layer, and there will be 100 outputs in that layer. On this basis, a full connection layer is added. On the financial data set, because there are three categories, the number of neurons at the full connection layer is 3. On the hotel review data set, there are five categories and the number of neurons at the full connection layer is 5. All neurons in the full connection layer synthesize each output in the LSTM layer and output a score. This score represents the likelihood that entered reviews fall into this category.

The full connection layer is followed by a softmax layer, which uses softmax activation function mapping to output the classification results of emotion prediction data. The model uses cross-entropy as the loss function, and the optimization goal is to minimize the cross-entropy between the predicted output value of training samples and the actual sample value.

$$P(y|x) = \frac{e^{h(x,y_i)}}{\sum_{j=1}^{n} e^{h(x,y_i)}}$$

5.5 Transformer

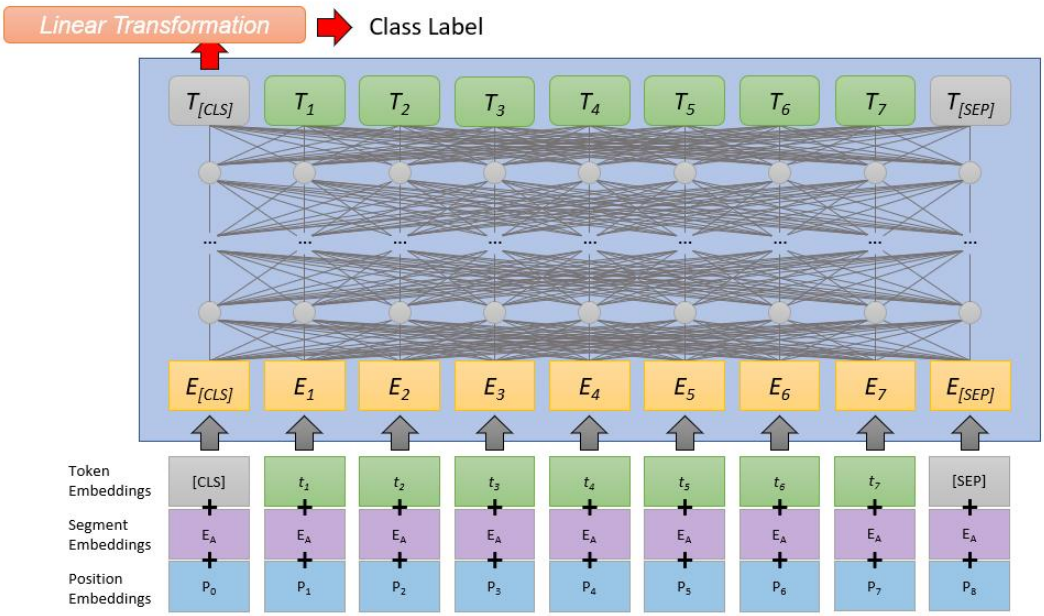


Figure 18 Transformer classifier processing

Source: Jimmy Lin (jimmylin@uwaterloo.ca), released under Creative Commons Attribution 4.0 International (CC BY 4.0): https://creativecommons.org/licenses/by/4.0/

The code for this model calls the existing Simple Transformer model. In this paper, Bert and Roberta are used for emotion analysis. Bert and Roberta are both Seq2Seq models based on Transformer. Roberta is an improved version of Bert developed by Facebook. The differences between them mainly lie in different pre-training tasks, training data and batch size used in training. Its model structure and internal algorithm are consistent **[21].**

Although BERT is based on Transformer, it only uses the Encoder part of the Transformer. Its input is a string of tokens, and its output is the encoding of each token. Its overall framework is made up of multiple encoder stacks of Transformer. The encoder of each layer is composed of a layer of Muti-head-attention and a layer of feed-forward. The Bert-base model used in this paper has 12 layers, and each layer has 12 attentions. The function of Attention is to encode the current word semantically by counting the correlation between the current word and all words.

5.5.1 Embedding Layer

Each entry into BERT's document goes through three embedding mechanisms. The Token embedding tokenizes the document. Insert the [CLS] identifier at the beginning and the [SEP] identifier at the end to mark the beginning and end of the input document. When the Segment embedding is input as two documents, 0 and 1 are used to distinguish the two sentences. However, in this research, only one document is input at a time, so all segment embedding is 0. Position embedding is used to mark the order of words in the input document. Because Bert used the attention

mechanism, in the attention view, all words in the document are input at the same time, without any order. Position embedding is required to add position information to the vector. Finally, vectors generated by the three embedding layers are added to get a vector input to BERT.

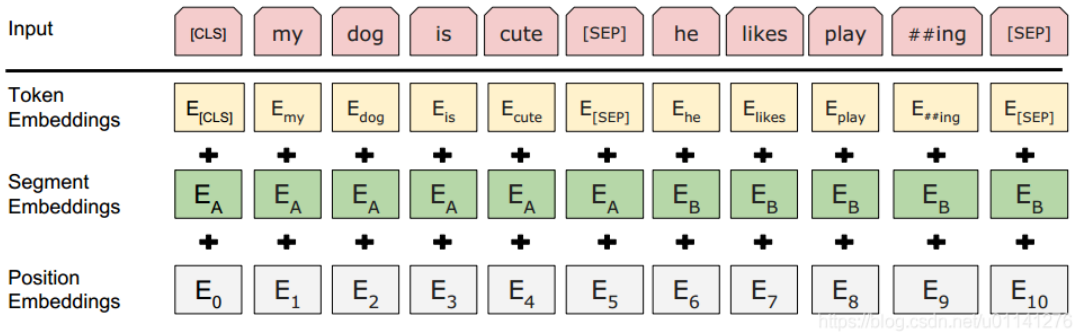


Figure 19 Embedding of Transformer

Source: Jimmy Lin (jimmylin@uwaterloo.ca), released under Creative Commons Attribution 4.0 International (CC BY 4.0): https://creativecommons.org/licenses/by/4.0/

5.5.2 Attention Layer

The attention layer in the model is responsible for the semantic extraction of all tokens in sentences. In the same layer, each attention header will analyze the current token from one aspect of semantics and calculate the correlation between the current token and all tokens. If there are 12 attention in one layer of our model, we can analyze the semantics of tokens from 12 aspects. If we input vectors with dimensions n*m, 12 attention headers will output 12 n*m vectors. A large vector of N *(12*m) is synthesized through data splicing. Finally, the vector dimension is reduced to N *m by data dimension reduction and output.

5.5.3 Output Layer

In this paper, BERT-Base and RoBERTa-Base models are used to perform emotion analysis tasks. Fine-tune was performed on the output layer of the original model to fit the existing task. Bert and Roberta are both seQ2SEQ models, whose output and input are of equal length, and each token input will have a corresponding output. However, we do not need to use all of the output for sorting tasks. The attention mechanism takes into account the correlation between the current input and all inputs, meaning that each output takes into account information from all token inputs. In this article, only the first output of the model, at the [CLS] identifier seat in the input, is used. Since the [CLS] identifier only represents the beginning of the input without any real semantics, after the attention mechanism, its output vector is a semantic vector that comprehensively considers all the tokens in the sentence. Linear Transform this vector to label **[22]**.

6 Experiment and Evaluation

6.1 Experiment

In the experiment, the corresponding research problems of NB, LSTM and Transformer are firstly studied. Each model has its questions to study. Finally, the performance of the three models is compared on two datasets.

	Financial reviews	Hotel reviews	Financial reviews, Hotel reviews
NB	Shuffle, not shuffle	5 class, 3 class, 2 class	3 class
LSTM	Embedding, W2V	Embedding, W2V	Embedding, W2V
Transformer	BERT, RoBERTa	BERT, RoBERTa	BERT, RoBERTa

Figure 20 The corresponding experimental diagram of Gaussian Naïve Bayes, LSTM and Transformer

Firstly, each model has its questions to study.

Gaussian Naïve Bayes

1) Change the number of categories of the dataset to make the dataset tend to be balanced, compare the performance of the model and analyze the reasons. (5 classes, 3 classes, 2 classes)
2) Determine whether to Shuffle the dataset, observe the changes in accuracy, and analyze the causes.
3) Compare the performance of the two datasets on the Gaussian Naïve Bayes model. (In the case that both datasets have 3 categories).

LSTM

4) Studies the performance of Embedding+LSTM and Word2Vec+LSTM from Google on the same data set.
5) The performance of Embedding+LSTM on two data sets is studied. The performance of Word2Vec+LSTM from Google on two datasets is studied.

Transformer

6) The study compares the performance of Bert and Robert on the same data set.
7) Study BERT's performance on two data sets. The performance of RoBERTa on two data sets was studied.

	Naïve Bayes	LSTM	Transformer
Financial reviews	Performance	Performance	Performance
Hotel reviews	Performance	Performance	Performance

Figure 21 Compare NB, LSTM and Transformer on Financial review dataset and hotel review dataset

Compare the performance of Naïve Bayes, LSTM and Transformer.

8) Study the performance effects of Naive Bayes, LSTM and Transformer on the same data set. Select the model that performs best. For the hotel review data set, the three models are tested in turn and the experimental results are obtained. For the financial review data set, the three models are tested successively, and the experimental results are obtained.

9) To change the dataset type and study the prediction accuracy of a model for different types of data set, naive Bayes, LSTM, Transformer. To the hotel review data set and the financial review data set, the three models are tested successively, and the experimental results are obtained.

6.2 Evaluation Index

I adopted accuracy, precision, recall and F1 as the evaluation indexes of the experiment. Set TP=true positive, TN=true negative, FP=false positive, FN=false negative.

1) Confusion matrix

A confusion matrix is the basis of many evaluation indicators.

True	Predict	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Figure 22 Confusion Matrix

True Positive (TP): the Positive sample is successfully predicted to be Positive.

True Negative (TN): The Negative sample is successfully predicted as Negative.

False Positive (FP): Incorrectly predict a Positive from a negative sample.

False Negative (FN): Positive samples are incorrectly predicted as Negative.

2) Accuracy

The accuracy index is an index used to evaluate the classification model. Accuracy is the percentage of the model that predicts the correct outcome. TP+TN is the number of correctly predicted samples, TP+TN+FP+FN is the total number of samples. Accuracy calculation formula:

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

3) Precision

Precision represents the proportion of samples where the prediction is positive, and the truth value is positive. When the cost of False Positive (FP) is high (the consequences are severe), that is, when FP is expected to be avoided as much as possible, the focus should be on improving precision. Calculation formula of accuracy:

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

4) Recall

The recall rate is the percentage of samples with positive truth values that are correctly predicted. When the cost of False Negative (FN) is very high (the consequences are very serious) and FN is expected to be avoided as far as possible, emphasis should be given to improving the recall index. Calculation formula of recall rate:

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

5) F1-score

F1-score is the harmonic mean of precision and recall. F1-score takes into account both the accuracy rate and recall rate of a classification model. It has a maximum value of 1 and a minimum value of 0. 1 represents the best output of the model, while 0 represents the worst output of the model. The formula of F1-score:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

6) Multi-classification evaluation indexes

In multi-classification problems, Micro-Average and Macro-Average are commonly used to evaluate the performance of models.

Macro - average method

Macro Average will calculate evaluation indexes such as precision, recall and F1-score for each class, and then Average them to get Macro Precision, Macro Recall and Macro F1. This method treats each category equally, but its value is affected by the rarity category. n indicates the total number of categories. I indicate the categories. The calculation method is as follows:

$$Macro\ Precision = \frac{1}{n}\sum_{i=1}^{n} P_i$$

$$Macro\ Recall = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$Macro\ F1 = \frac{1}{n}\sum_{i=1}^{n} F1_i$$

Micro - average method

According to the predicted value and true value, calculate (TP), false positive example (FP), false negative example (FN) for each class, i represents the class. Micro Recall = Micro Precision = Micro F1. The calculation method is as follows:

$$F1_{micro} = \frac{\sum TP_i}{\sum (TP_i + FP_i)}$$

When F1 is used, F1 gives the same weight to precision and Recall. But in practical application, we need to start from a given problem and consider which is more important, precision or Recall. For example, the consequences of misdiagnosing a sick person as healthy are far more serious than the consequences of misdiagnosing a healthy person as sick. In the case of multiple classifications, different prediction errors may have different meanings. For example, predicting X as Y will cost more than predicting W as R. Standard F1 does not take these things into account, so specific problems require the selection of appropriate model performance metrics.

6.3 Experimental Results

6.3.1 Results of Naïve Bayes

1) Because the two data sets are unbalanced, the value of F1-MICRO can better reflect the accuracy of the model in **Figure 23**. Based on naive Bayes' formula, if the number of class1 is large, the value of P (Y) is large and can have a big effect on the result. When the value of P (X | Y) the results become very small, the influence of model prediction ability will reduce.

2) On financial data sets, models with disrupted data sets perform better. Because there is continuity in the mood of financial commentary. For example, today most people are optimistic, more people will be infected with optimism and make positive comments. With the same label continuously, the training set and the prediction set obtained in the cross-validation method are very uneven, so the model with the scrambled data set performs better in **Figure 24**.

3) NB classifies financial data more accurately when the two data sets are in category 3. Because the financial data set documents are shorter, the information is more concentrated and NB classification is more accurate in **Figure 25**.

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
not shuffle	0.580539	0.33519	0.580539	0.404102	0.580539	0.390283	0.580539
shuffle	0.600695	0.445459	0.600695	0.497921	0.600695	0.490144	0.600695

Figure 23 The impact of shuffled data of hotel review dataset

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
5 class	0.464203	0.418341	0.464203	0.417288	0.464203	0.45174	0.464203
3 class	0.584401	0.585529	0.584401	0.589566	0.584401	0.595517	0.584401
2 class	0.711385	0.704895	0.711385	0.706266	0.711385	0.705316	0.711385

Figure 24 Influence of the number of classes of financial review dataset

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
Hotel reviews	0.584401	0.585529	0.584401	0.589566	0.584401	0.595517	0.584401
Financial reviews	0.600695	0.445459	0.600695	0.497921	0.600695	0.490144	0.600695

Figure 25 When there are 3 classed in 2 datasets, compare the index.

6.3.2 Results of LSTM

1) In the hotel data set, the Word2Vec+LSTM model from Google performs better. The Hotel data sets are all long texts with scattered information, which contains many contents. Meanwhile, Google's W2V is a general model, which is not accurate in judging the vocabulary in the professional field of finance, but accurate in judging the daily words in the hotel in **Figure 26**.

2) In financial data sets, the Embedding+LSTM model performs better. Because financial data sets are short in text and have concentrated lexical information, they are all information in the financial field in **Figure 27**.

3) In terms of evaluation indicators, the Embedding+LSTM model and Word2Vec+LSTM model from Google perform better on financial data sets. Because the financial data set has three categories, each with a base probability of 0.33, and the hotel data set has five categories, each with a base probability of 0.2. The base probability is high for fewer categories. Financial 0.73 is more than double 0.33. Hotel 0.61 is more than three times higher than 0.2. So the model is better at classifying hotel reviews than financial reviews.

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
LSTM+ embedding	0.564785	0.480268	0.564785	0.494592	0.564785	0.480925	0.564785
LSTM+ word2vec	0.611341	0.506238	0.611341	0.542801	0.611341	0.501956	0.611341

Figure 26 LSTM on hotel reviews

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
LSTM+ embedding	0.73503	0.681237	0.73503	0.700761	0.73503	0.672754	0.73503
LSTM+ word2vec	0.730304	0.624592	0.730304	0.688847	0.730304	0.600129	0.730304

Figure 27 LSTM on financial reviews

6.3.3 Results of Transformer

1) BERT did better on the hotel review dataset in **Figure 28**.

2) RoBERTa did better on the financial review dataset in **Figure 29**.

3) For BERT and RoBERTa, the evaluation indicators of both models are better on the financial review data set. The baseline of finance is 0.33, the baseline of the hotel is 0.2, finance 0.6 is three times of 0.3, hotel 0.8 is four times of 0.2. So, the two models do classify financial reviews more accurately.

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
BERT hotel reviews	0.684878	0.620094	0.684878	0.621418	0.684878	0.623426	0.684878
BERT Financial reviews	0.859794	0.840455	0.859794	0.819748	0.859794	0.867773	0.859794

Figure 28 BERT on both datasets

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
RoBERTa hotel reviews	0.666097	0.604024	0.666097	0.618001	0.666097	0.595678	0.666097
RoBERTa Financial reviews	0.866701	0.854733	0.866701	0.849662	0.866701	0.861387	0.866701

Figure 29 Roberta on both datasets

6.3.4 Compare the Performance of Naïve Bayes, LSTM and Transformer

I selected the model with the best performance in NB, LSTM and Transformer respectively for comparison in **Figure 30** and **Figure 31**. On the same data set: Transformer performance best, LSTM is second and NB is the worst. In two different data sets: just from the evaluation indicators F1-score and accuracy, all three models performed better in financial reviews. But there are 3 categories of finance, and the baseline is 0.33, and 5 categories of hotels, and the baseline is 0.5. Considering the baseline and evaluation index values, NB, LSTM and Transformer have better classification ability on the financial review datasets.

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
bayes+ word2vec	0.464	0.418	0.464	0.417	0.464	0.452	0.464
LSTM+ word2vec	0.611	0.506	0.611	0.543	0.611	0.502	0.611
BERT	0.685	0.620	0.685	0.621	0.685	0.623	0.685

Figure 30 Compare models on hotel review dataset

	accuracy	f1-macro	f1-micro	precision-macro	precision-micro	recall-macro	recall-micro
bayes+ word2vec	0.601	0.445	0.601	0.498	0.601	0.49	0.601
LSTM+ embedding	0.735	0.681	0.735	0.701	0.735	0.673	0.735
RoBERTa	0.867	0.855	0.867	0.850	0.867	0.861	0.867

Figure 31 Compare models on financial review dataset

7 Conclusion

Financial review data sets of over 4000 are divided into 3 categories. Hotel review data set of over 20,000, divided into five categories. Both data sets are imbalanced. Therefore, I pay more attention to micro F1-score and accuracy. I first studied the corresponding problems of NB, LSTM and Transformer3 models, then compared the performance of NB, LSTM and Transformer3 models, and designed the experimental process diagram.

Naive Baye

1) By changing the number of categories of hotel reviews (5 classes, 3 classes, 2 classes), the data set tends to be balanced. The NB classification ability is improved.

2) The number of financial data sets is small, and the text is short. After the financial data set was scrambled, the model classification was more accurate. Because reviews are copycat, positive reviews lead to more positive ones. Continuous identical labels are very uneven in the training and prediction sets taken in the cross-validation method, so the model with scrambled data sets performs better.

3) NB classifies financial data more accurately when the two data sets are in category 3. Because the financial data set documents are shorter, the information is more concentrated and NB classification is more accurate.

LSTM

4) The Word2Vec+LSTM model from Google performs better on the same financial review dataset. Embedding+LSTM performs better on the same hotel reviews dataset.

5) Word2Vec+LSTM performs better in the finance reviews dataset and Embedding+LSTM also performs better in the finance reviews dataset on two different data sets.

Transformer

6) On the same hotel data set, BERT did better. On the same financial data set, RoBERTa did better.

7) For BERT and RoBERTa, the evaluation indicators of both models are better on the financial review data set.

Compare the performance of NB, LSTM and Transformer on 2 datasets

8) In the same data set, Transformer has the best performance, LSTM the second, and NB the worst.

9) Considering the baseline and evaluation index values, NB, LSTM and Transformer have better classification ability in financial data sets.

To sum up, although Transformer has the best performance at present, the classification accuracy of naive Bayes and LSTM is not very low under the circumstance of limited special text and computer resources, and they can also be used as tools for sentiment classification. Naive Bayes classification takes the least time. Embedding+LSTM is faster than Word2vec+LSTM, however, Embedding +LSTM can get a good classification accuracy. When we need fewer resources to do the sentiment classification task, we can choose Embedding+LSTM. **Therefore, it is important to analyze the dataset and the specific needs first before making a decision on which models to use.**

8 Further Work

In the selection and processing of datasets, although I strive to balance the datasets and alleviate the imbalance classification of original data to some extent, due to personal time and technical limitations, the quality and type of datasets still have room for improvement. In the future, I will find more different types of datasets and data sets with more categories to compare the classification accuracy of different methods and recommend the appropriate methods to classify sentiment according to the dataset to be used in a particular scene.

In this study, I directly used the existing Bert model. In the future, I hope I can fine-tune Bert's parameters myself. According to different classification requirements, find the appropriate parameters.

References

[1] J.F. Sánchez-Rada, C.A. Iglesias, Social context in sentiment analysis: Formal definition, an overview of current trends and framework for comparison, Inf. Fusion. 52 (2019) 344–356.

[2] F. Hemmatian, M.K. Sohrabi, A survey on classification techniques for opinion mining and sentiment analysis, Artif. Intell. Rev. 52 (2019)1495–1545.

[3] W. Medhat, A. Hassan, H. Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Eng. J. 5 (2014) 1093–1113.

[4] O. Alqaryouti, N. Siyam, A.A. Monem, K. Shaalan, Aspect-based sentiment analysis using smart government review data, Appl. Comput. Informatics. (2019) 1–20.

[5] S.M. Jiménez-Zafra, M.T. Martín-Valdivia, M.D. Molina-González, L.A. Ureña López, How do we talk about doctors and drugs? Sentiment analysis in forums expressing opinions for medical domain, Artif. Intell. Med. 93 (2019) 50–57.

[6] L. Rognone, S. Hyde, S.S. Zhang, News sentiment in the cryptocurrency market: An empirical comparison with forex, Int. Rev. Financ. Anal. 69 (2020) 1–17.

[7] E. Georgiadou, S. Angelopoulos, H. Drake, Big data analytics and international negotiations: Sentiment analysis of brexit negotiating outcomes, Int. J. Inf. Manage. 51 (2020) 1–9.

[8] M. Erritali, A. Beni-Hssane, M. Birjali, Y. Madani, An approach of semantic similarity measure between documents based on big data, Int. J. Electr. Comput. Eng 6 (2016) 2454–2461.

[9] M. Kasri, M. Birjali, A. Beni-Hssane, Word2Sent: A new learning sentiment-embedding model with low dimension for sentence level sentiment classification, Concurr. Comput. Pract. Exp. (2020) 1–12.

[10] P.V. Ngoc, C.V.T. Ngoc, T.V.T. Ngoc, D.N. Duy, A C4.5 algorithm for english emotional classification, Evol. Syst. 10 (2019) 425–451.

[11] Y. Han, Y. Liu, Z. Jin, Sentiment analysis via semi-supervised learning: a model based on dynamic threshold and multi-classifiers, Neural Comput. Appl. 32 (2020) 5117–5129.

[12] Z.Y. Khan, Z. Niu, S. Sandiwarno, R. Prince, Deep learning techniques for rating prediction: a survey of the state-of-the-art, Artif. Intell. Rev. (2020) 1–41.

[13] H.H. Do, P. Prasad, A. Maag, A. Alsadoon, Deep learning for aspect-based sentiment analysis: A comparative review, Expert Syst. Appl. 118 (2019)272–299.

[14] R. Liu, Y. Shi, C. Ji, M. Jia, A survey of sentiment analysis based on transfer learning, IEEE Access. 7 (2019) 85401–85412.

[15] L. Mai, B. Le, Joint sentence and aspect-level sentiment analysis of product comments, Ann. Oper. Res. (2020)

[16] O. Araque, G. Zhu, C.A. Iglesias, A semantic similarity-based perspective of affect lexicons for sentiment analysis, Knowledge-Based Syst. 165 (2019) 346–359.

[17] L. Ren, B. Xu, H. Lin, X. Liu, L. Yang, Sarcasm detection with sentiment semantics enhanced multi-level memory network, Neurocomputing. (2020) 1–7.

[18] H. Liu, M. Zhou, Q. Liu, An embedded feature selection method for imbalanced data classification, IEEE/CAA J. Autom. Sin. 6 (2019) 703–715.

[19] W. Li, F. Qi, M. Tang, Z. Yu, Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification, Neurocomputing. 387 (2020) 63–77.

[20] L. Lazib, B. Qin, Y. Zhao, W. Zhang, T. Liu, A syntactic path-based hybrid neural network for

negation scope detection, Front. Comput. Sci. 14 (2020) 84–94.
[21] N.C. Dang, M.N. Moreno-García, F. De la Prieta, Sentiment analysis based on deep learning: A comparative study, Electronics. 9 (2020) 1–29.
[22] F. Hemmatian, M.K. Sohrabi, A survey on classification techniques for opinion mining and sentiment analysis, Artif. Intell. Rev. 52 (2019) 1495–1545.

Appendices(code)

Gaussian Naive Bayes

Write the code by myself

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from tqdm import tqdm
from nltk import pos_tag
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from genesis.models import KeyedVectors
from gensim. models.word2vec import Word2Vec
from sklearn.model_selection import KFold
from scipy.stats import multivariate_normal
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB

data = pd.read_csv('tripadvisor_hotel_reviews.csv')
# Get word parts
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

# Load model
wnl = WordNetLemmatizer()  # Lemmatize
stop = stopwords.words('english')
w2v_file_path = 'GoogleNews-vectors-negative300.bin'  # google w2v
w2v_model = KeyedVectors.load_word2vec_format(w2v_file_path, binary=True) # w2v model
# w2v_model = Word2Vec.load(w2v_file_path)

```

```
# Data preprocessing
text = [] # All vectorized comments
for i in tqdm(range(len(data))):  # The loop preprocesses each piece of data
    docs = data.iloc[i, 0] # i: number of lines, 0: class 1 text
    words = word_tokenize(docs) # tokenize
    tagged_sent = pos_tag(words) # Gets all parts of speech in a document
    vectorization_words = [] # Each word is vectorized into a 300-dimensional [0.1,0.2,0.3] vector, which is then stored as a doc list.

    for j in range(len(words)): # Every word in every document
        if words[j].isdigit(): # Check if it's a number
            w2v = w2v_model.wv['NUM']  # num normalization
            vectorization_words.append(w2v)
        # remove stopwords
        elif words[j] in stop or not words[j].isalpha():
            continue
        else:
            wordnet_pos = get_wordnet_pos(tagged_sent[j][1]) or wordnet.NOUN
            # lemmatization
            temp = wnl.lemmatize(tagged_sent[j][0], pos=wordnet_pos)
            # case normalization
            temp = temp.lower()
            # Each word is converted into a 300-dimensional vector
            if temp in w2v_model:
                w2v = w2v_model.wv[temp]
                vectorization_words.append(w2v)
    # Comment vectorization
    # Put each vectorized comment into text, which contains all comments.
    text.append(vectorization_words)

# Handling Labels
label = data['label']
# Change classes 1 and 2 to class 3,Class 1, 2, 3, 4, 5, become class 1+2+3,4,5
for i in range(len(label)):
    if label[i] == 1 or label[i] == 2:
        label[i] = 3

# A comment becomes a vector: take the average of all words (a 300-dimensional vector)
mean_docs = []
for t in text:
    t = np.mean(t,axis = 0)
    mean_docs.append(t)

# train model
```

```
results = []
# 10-fold cross-validation
for train_index, test_index in KFold(10).split(mean_docs, label):
    text_train = []
    label_train = []
    text_test = []
    label_test = []
    # training set
    for index in train_index:
        text_train.append(mean_docs[index])
        label_train.append(label[index])
    # testing set
    for index in test_index:
        text_test.append(mean_docs[index])
        label_test.append(label[index])

    # array, easy to do matrix operations
    text_train = np.array(text_train, dtype='float32')
    label_train = np.array(label_train, dtype='float32')
    text_test = np.array(text_test)
    label_test = np.array(label_test)

    print(text_train.shape)
    print(label_train.shape)

    # Gaussian Bayesian classifier
    classifier = GaussianNB()
    # fit=train
    classifier.fit(text_train, label_train)
    # predict label
    y_pred = classifier.predict(text_test)
    # performance
    result = {}
    result['accuracy'] = metrics.accuracy_score(label_test, y_pred)
    result['f1-macro'] = f1_score(label_test, y_pred, labels=list(set(label_test)), average='macro')
    result['f1-micro'] = f1_score(label_test, y_pred, labels=list(set(label_test)), average='micro')
    result['precision-macro'] = metrics.precision_score(label_test, y_pred, average='macro')
    result['precision-micro'] = metrics.precision_score(label_test, y_pred, average='micro')
    result['recall-macro'] = metrics.recall_score(label_test, y_pred, average='macro')
    result['recall-micro'] = metrics.recall_score(label_test, y_pred, average='micro')
    results.append(result)

print('\n\nAll results：')
df = pd.DataFrame(results)
```

```
print(df)
mean_df = df.mean()
mean_df = mean_df.to_dict()
print('Cross-validation results (average of all results)：')
print(mean_df)
results.append(mean_df)
df = pd.DataFrame(results)
print("The cross-validation result is placed in the last line of the result")
print(df)
df.to_csv('bayes_cross_validation_result_3_class.csv')
print("Results save in bayes_cross_validation_result_3_class.csv")
```

LSTM

Eembedding+LSTM

Write the code by myself

```
import pandas as pd
from sklearn.model_selection import KFold
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
from keras.models import load_model
import gensim
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from tqdm import tqdm
from nltk import pos_tag
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

data = pd.read_csv('fin_data_run.csv', encoding='unicode_escape')
data = data.sample(frac=1).reset_index(drop=True)


# The part of speech
def get_wordnet_pos(tag):
```

```
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

wnl = WordNetLemmatizer()
# nltk.download('stopwords')
stop = stopwords.words('english')
# nltk.download('punkt')
# nltk.download('averaged_perceptron_tagger')

# Data preprocessing
for i in tqdm(range(len(data))):
    doc = data.iloc[i, 0]
    words = word_tokenize(doc)
    tagged_doc = pos_tag(words)
    preprocessed_words = []
    for j in range(len(words)):
        if words[j].isdigit(): #
            preprocessed_words.append('NUM')  # num normalization（文本中所有数字用 NUM 代替）
        # remove stopwords and punctuation
        elif words[j] in stop or not words[j].isalpha():  # remove stopwords
            continue
        else:
            wordnet_pos = get_wordnet_pos(tagged_doc[j][1]) or wordnet.NOUN
            temp = wnl.lemmatize(tagged_doc[j][0], pos=wordnet_pos)  # lemmatization
            preprocessed_words.append(temp.lower())  # case normalization
    new_doc = ''
    for w in preprocessed_words:
        new_doc += w + ' '
    data.iloc[i, 0] = new_doc
text = data['text']
label = data['label']


# LSTM modelling
# Using tokenizer, each word in the text is represented by a number, which is the preprocessing of word vectorization
```

```
# texts_to_matrix: select MAX_NB_WORDS from MAX_NB_WORDS, select MAX_NB_WORDS from MAX_NB_WORDS
# We want to vectorize the cut_review data. We want to convert each cut_review to a vector of a sequence of integers
# Set the 50,000 most frequently used words
# set the maximum number of words per cut_review to 250 (more words will be cut, fewer words will be 0)
MAX_NB_WORDS = 50000
# The maximum length of new_doc for each preprocessed comment
MAX_SEQUENCE_LENGTH = 250
# The dimensions of each word
EMBEDDING_DIM = 100
# Make your  W2V dictionary
# Split words, convert them to numbers
tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(data['text'].values)
word_index = tokenizer.word_index
print(' %s different words.' % len(word_index))
# print(tokenizer.word_index)


X = tokenizer.texts_to_sequences(data['text'].values)
# Fill X so that the length of each column of X is the same, make it 250, fill it with 0
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
# Onehot expansion of multi-class tags
Y = pd.get_dummies(data['label']).values

print(X.shape) #  （20000，250）250 words
print(Y.shape) #  （20000，5）5 class


# 10-fold cross-validation
# The first of the models is the Embedding layer, which uses vectors of length 100 to represent each term
# The SpatialDropout1D layer has a random ratio of 0.2 input units for each update in training, which helps prevent overfitting
# The LSTM layer contains 100 memory cells
# The output layer is a fully connected layer with 10 categories
# The activation function is set to 'softmax' because it is multi-category
# As it is multiple categorization, the loss function is categorical crossentropy _crossentropy
results = []
k = 10  # k-fold cross-validation
for train_index, test_index in tqdm(KFold(k).split(X, Y)):  # x=docs, y=label
    # model
    model = Sequential()
```

```
    model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X.shape[1]))
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    print(model.summary())

    epochs = 5
    batch_size = 64
    history = model.fit(X[train_index], Y[train_index], epochs=epochs, batch_size=batch_size, validation_s
plit=0.1,
                        callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.0001)])

    # model = load_model('lstm_class.h5')
    result = {}
    y_pred = model.predict(X[test_index])
    y_pred = y_pred.argmax(axis=1)
    y_test = Y[test_index].argmax(axis=1)
    result['accuracy'] = metrics.accuracy_score(y_test, y_pred)
    result['f1-macro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='macro')
    result['f1-micro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='micro')
    result['f1-weighted'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='weighted')
    result['precision-macro'] = metrics.precision_score(y_test, y_pred, average='macro')
    result['precision-micro'] = metrics.precision_score(y_test, y_pred, average='micro')
    result['precision-weighted'] = metrics.precision_score(y_test, y_pred, average='weighted')
    result['recall-macro'] = metrics.recall_score(y_test, y_pred, average='macro')
    result['recall-micro'] = metrics.recall_score(y_test, y_pred, average='micro')
    result['recall-weighted'] = metrics.recall_score(y_test, y_pred, average='weighted')
    results.append(result)

print('\n\nAll results')
df = pd.DataFrame(results)
print(df)
mean_df = df.mean()
mean_df = mean_df.to_dict()
print('Cross-validation results (average of all results)：')
print(mean_df)
results.append(mean_df)
df = pd.DataFrame(results)
print("The cross-validation result is placed in the last line of the result")
print(df)
df.to_csv('lstm_cross_validation_result_fin.csv')
print("Results saveo in lstm_cross_validation_result_fin.csv")
```

Word2Vec+LSTM

Write the code by myself

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
from keras.models import load_model
from gensim.models import KeyedVectors
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from tqdm import tqdm
from nltk import pos_tag
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

data = pd.read_csv('fin_data_run.csv', encoding='unicode_escape')
data = data.sample(frac=1).reset_index(drop=True)

# The part of speech
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

wnl = WordNetLemmatizer()
# nltk.download('stopwords')
stop = stopwords.words('english')
# nltk.download('punkt')
```

```
# nltk.download('averaged_perceptron_tagger')

for i in tqdm(range(len(data))):
    sentence = data.iloc[i, 0]
    words = word_tokenize(sentence)
    tagged_sent = pos_tag(words)
    new_words = []
    for j in range(len(words)):
        if words[j].isdigit():
            new_words.append('NUM')  # num normalization
        elif words[j] in stop or not words[j].isalpha():  # remove stopwords
            continue
        else:
            wordnet_pos = get_wordnet_pos(tagged_sent[j][1]) or wordnet.NOUN
            temp = wnl.lemmatize(tagged_sent[j][0], pos=wordnet_pos)  # lemmatization
            new_words.append(temp.lower())  # case normalization
    new_sentence = ''
    for w in new_words:
        new_sentence += w + ' '
    data.iloc[i, 0] = new_sentence
text = data['text']
label = data['label']

# Using tokenizer, each word in the text is represented by a number, which is a preprocessing of word vect
orization
# Set the 50000 most frequently used words
# (texts_to_matrix will take the pre-max_nb_words column, will take the pre-max_nb_words column)
# the occurrence of too low-frequency words is not considered
MAX_NB_WORDS = 50000
MAX_SEQUENCE_LENGTH = 250
EMBEDDING_DIM = 300

tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower
=True)
tokenizer.fit_on_texts(data['text'].values)
word_index = tokenizer.word_index
print('%s different words.' % len(word_index))

w2v_file_path = 'GoogleNews-vectors-negative300.bin'  # w2v from google
w2v_model = KeyedVectors.load_word2vec_format(w2v_file_path, binary=True)
vocab_list = list(tokenizer.word_index.keys())
# vocab_list = [word for word, Vocab in w2v_model.key_to_index.items()]
word_index = tokenizer.word_index
# word_index = {"":0}
```

```
word_vector = {}
embeddings_matrix = np.zeros((len(vocab_list) + 1, w2v_model.vector_size))

for i in tqdm(range(len(vocab_list))):
    word = vocab_list[i]  # Each word
    # word_index[word] = i + 1 #
    if word in w2v_model:
        word_vector[word] = w2v_model[word]
        embeddings_matrix[i + 1] = word_vector[word]  # Word vector matrix
        continue
    word_vector[word] = word_vector[list(word_vector.keys())[-1]]  # Words: word vectors
    embeddings_matrix[i + 1] = word_vector[word]  # Word vector matrix

embedding_layer = Embedding(input_dim=len(embeddings_matrix),  # The length of the dictionary
                            output_dim=EMBEDDING_DIM,  # Word vector length (100)
                            weights=[embeddings_matrix],  # Key points: pre-trained word vector coefficients
                            input_length=MAX_SEQUENCE_LENGTH,  # Maximum length of each sentence
                            trainable=False  # Whether to update the word vector during training
                            )

X = tokenizer.texts_to_sequences(data['text'].values)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
Y = pd.get_dummies(data['label']).values

# cross-validation
results = []
k = 10  # 10-fold cross-validation
cont_times = 0
for train_index, test_index in tqdm(KFold(k).split(X, Y)):
    cont_times += 1
    # model
    model = Sequential()
    model.add(embedding_layer)
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    print(model.summary())

    epochs = 5
    batch_size = 64
    history = model.fit(X[train_index], Y[train_index], epochs=epochs, batch_size=batch_size, validation_s
plit=0.1,
                        callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.0001)])
```

```
126.
127.     # model = load_model('lstm_class.h5')
128.     result = {}
129.     y_pred = model.predict(X[test_index])
130.     y_pred = y_pred.argmax(axis=1)
131.     y_test = Y[test_index].argmax(axis=1)
132.     result['accuracy'] = metrics.accuracy_score(y_test, y_pred)
133.     result['f1-macro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='macro')
134.     result['f1-micro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='micro')
135.     result['f1-weighted'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='weighted')
136.     result['precision-macro'] = metrics.precision_score(y_test, y_pred, average='macro')
137.     result['precision-micro'] = metrics.precision_score(y_test, y_pred, average='micro')
138.     result['precision-weighted'] = metrics.precision_score(y_test, y_pred, average='weighted')
139.     result['recall-macro'] = metrics.recall_score(y_test, y_pred, average='macro')
140.     result['recall-micro'] = metrics.recall_score(y_test, y_pred, average='micro')
141.     result['recall-weighted'] = metrics.recall_score(y_test, y_pred, average='weighted')
142.     results.append(result)
143.
144. print('\n\nAll results')
145. df = pd.DataFrame(results)
146. print(df)
147. mean_df = df.mean()
148. mean_df = mean_df.to_dict()
149. print('Cross-validation results (average of all results)：')
150. print(mean_df)
151. results.append(mean_df)
152. df = pd.DataFrame(results)
153. print("The cross-validation result is placed in the last line of the result")
154. print(df)
155. df.to_csv('lstm_cross_validation_result_fin.csv')
156. print("Results saveo in lstm_cross_validation_result_fin.csv")
```

Transformer

BERT, RoBerta

Call an existing method

```
1.  import pandas as pd
2.  from simpletransformers.classification import ClassificationModel
3.  from sklearn.metrics import f1_score, accuracy_score
4.  from tqdm import tqdm
5.  from sklearn.model_selection import KFold
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics

data = pd.read_csv('tripadvisor_hotel_reviews.csv', encoding='unicode_escape')
data = data.sample(frac=1).reset_index(drop=True)
for i in range(len(data)):
    data.iloc[i, 1] = data.iloc[i, 1] - 1

k = 10  # 10-fold cross-validation
i = 0
results = []
for train_index, test_index in KFold(k).split(data):
    # Data
    train_df = data.iloc[train_index]
    eval_df = data.iloc[test_index]
    train_df.columns = ['text', 'labels']
    eval_df.columns = ['text', 'labels']
    # Creating a classification model(BERT,RoBERTa)
    # BERT
    model = ClassificationModel('bert', 'bert-base-uncased', num_labels=5, args={"output_dir": "./tmp/outputs_bert"+str(i)+"/",
                                                # "reprocess_input_data": True
                                                })
    # RoBERTa
    model = ClassificationModel('roberta', 'roberta-base', num_labels=5, args={"output_dir": "./tmp/outputs_roberta"+str(i)+"/",
                                                # "reprocess_input_data": True
                                                })

    # Train model
    model.train_model(train_df)

    # Evaluation model
    def f1_multiclass(labels, preds):
        return f1_score(labels, preds, average='micro')

    # result, model_outputs, wrong_predictions = model.eval_model(eval_df)
    result, model_outputs, wrong_predictions = model.eval_model(eval_df, f1=f1_multiclass, acc=accuracy_score)
    print('result', result)
    print('model_outputs', model_outputs)
    model_outputs = model_outputs.argmax(axis=1)
    print('model_outputs', model_outputs)
    print('wrong_predictions', wrong_predictions)
```

```
47.
48.     result = {}
49.     y_pred = model_outputs
50.     y_test = eval_df['labels'].values
51.     result['accuracy'] = metrics.accuracy_score(y_test, y_pred)
52.     result['f1-macro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='macro')
53.     result['f1-micro'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='micro')
54.     result['f1-weighted'] = f1_score(y_test, y_pred, labels=list(set(y_test)), average='weighted')
55.     result['precision-macro'] = metrics.precision_score(y_test, y_pred, average='macro')
56.     result['precision-micro'] = metrics.precision_score(y_test, y_pred, average='micro')
57.     result['precision-weighted'] = metrics.precision_score(y_test, y_pred, average='weighted')
58.     result['recall-macro'] = metrics.recall_score(y_test, y_pred, average='macro')
59.     result['recall-micro'] = metrics.recall_score(y_test, y_pred, average='micro')
60.     result['recall-weighted'] = metrics.recall_score(y_test, y_pred, average='weighted')
61.     results.append(result)
62.     i += 1
```