Sentiment Analysis Model and Usefulness Score Model for Netflix Reviews Based on Machine Learning and Deep Learning Techniques

**Author**: Tianfeng GE

**Supervisor:** Luis Espinosa-Anke

Date:2024/08/26

Content

[Abstract 4](#_Toc176209446)

[1. Introductory 5](#_Toc176209447)

[1.1 Background and motivation 5](#_Toc176209448)

[1.2 Research Target 5](#_Toc176209449)

[1.3 Research Contribution 6](#_Toc176209450)

[2. Related Work 7](#_Toc176209451)

[2.1 Sentiment Analysis 7](#_Toc176209452)

[2.2 Usefulness Score 7](#_Toc176209453)

[3. Methodology 8](#_Toc176209454)

[3.1 Dataset 8](#_Toc176209455)

[3.2 Sentiment Analysis Model 9](#_Toc176209456)

[3.2.1 Analysis Data 9](#_Toc176209457)

[3.2.2 Data preprocessing 15](#_Toc176209458)

[3.2.3 Feature Extraction 19](#_Toc176209459)

[3.2.4 Model 23](#_Toc176209460)

[3.3 Usefulness Score Model 39](#_Toc176209461)

[3.3.1 Data preprocessing 39](#_Toc176209462)

[3.3.2 Feature Extraction 41](#_Toc176209463)

[3.3.3 Model 44](#_Toc176209464)

[4. Experiments and Results 61](#_Toc176209465)

[4.1 Evaluation Metrics 61](#_Toc176209466)

[4.2 Sentiment Analysis Model Result 66](#_Toc176209467)

[4.3 Usefulness Score Model Result 83](#_Toc176209468)

[5. Critical Discussion 101](#_Toc176209469)

[5.1 Comparison of Model Performance 101](#_Toc176209470)

[5.2 Comparison of Feature extraction methods 102](#_Toc176209471)

[5.3 Analysis of the effectiveness of LSTM architecture on the problem of training/validation loss fluctuation 103](#_Toc176209472)

[5.4 Advantages and Disadvantages of Ensemble Pretrained Embeddings in LSTM 105](#_Toc176209473)

[6. Conclusion 109](#_Toc176209474)

[Reference 111](#_Toc176209475)

[Acknowledgements 113](#_Toc176209476)

# Abstract

Based on the Netflix review dataset, this paper delves into the two key tasks of sentiment analysis and usefulness scoring. We adopt TF-IDF and BERT as feature extraction methods, construct logistic regression, random forest, XGBoost, and LSTM models, and tune the models by multiple methods to achieve sentiment analysis and usefulness score prediction of Netflix review data. For these two tasks, this paper provides a comprehensive evaluation and comparative analysis of the performance of the first three machine learning models through evaluation metrics such as accuracy, precision, recall, F1 score, and confusion matrix. For the LSTM deep learning model, this paper comprehensively evaluates the model performance and performance through classification reports, overall accuracy, and visualisation of loss versus accuracy curves during model training . The findings provide new insights into user feedback analysis and show the advantages and limitations of different models in practical applications.

# 1. Introductory

## 1.1 Background and motivation

“User-generated content (UGC) has evolved from a mere web phenomenon into a significant component of the digital world.” (Tomaiuolo, 2012, p. 3). With the popularity of social media and online platforms, UGC has taken an increasingly important place in the digital world. And Netflix, as the world's leading streaming platform, has a large number of user reviews, which not only influence other users' viewing decisions, but also provide valuable information for content creators and platform optimisation. Tomaiuolo (2012, p. 20) argues that “as UGC continues to grow in volume and complexity, data analytics has become an important tool for understanding user needs and behaviour.” In this context, sentiment analysis models and usefulness score models are important tools for meaningful interpretation of these comments.

## 1.2 Research Target

The main objectives of this study are:

1. **Sentiment analysis model:**

Split the dataset by date, train and test with temporally non-overlapping comments. Ultimately, score columns are predicted based on content columns to help platforms better understand users' sentiment tendencies.

1. Usefulness scoring model：

Giving usefulness ratings to reviews by training on the ‘thumbs up’ columns

## 1.3 Research Contribution

In this research, we combine two feature extraction methods, TF-IDF and BERT, and apply a variety of models such as logistic regression, random forest, XGBoost and LSTM. By systematically tuning these models, we delve into their performance in sentiment analysis and usefulness scoring tasks. Our study not only demonstrates the strengths and weaknesses of different models in processing Netflix user review data, but also provides a comprehensive analysis of the model results through precise evaluation metrics (e.g., accuracy, precision, recall, F1 score, etc.) and data visualisation. This process provides valuable insights for sentiment analysis and usefulness scoring and provides a strong reference for future research.

# 2. Related Work

## 2.1 Sentiment Analysis

Sentiment analysis is an important task in Natural Language Processing (NLP) aimed at extracting subjective information from text. *”Machine-learning-based techniques [*[*35*](https://www.mdpi.com/2079-9292/9/3/483#B35-electronics-09-00483)*] proposed for sentiment analysis problems can be divided into two groups: (1) traditional models and (2) deep learning models. ”* *(Dang, Moreno-García and De la Prieta, 2020).* Traditional methods such as dictionary-based methods and machine learning methods (e.g., support vector machines, plain Bayes) have been widely used. In recent years, deep learning techniques (e.g. LSTM and BERT) have made significant progress due to their powerful modelling of context.

## 2.2 Usefulness Score

Usefulness scores are commonly used to predict the value of user reviews. Traditional feature engineering methods and machine learning models (e.g., logistic regression, random forest) have performed well in this domain. In recent years, the introduction of deep learning models has further improved the prediction performance, especially in modelling complex textual features

# 3. Methodology

## 3.1 Dataset

We used the Netflix review dataset from Kaggle. This dataset records information about reviews submitted by Netflix users on the Google Play Store, including the following fields:

* *reviewid*: Unique comment identifier to uniquely identify each comment
* userName: Google Play Store username of the user who commented
* content: Reviews, the text of user-specific reviews for Netflix
* score: Ratings, User Ratings for Netflix
* thumbsUpCount: Likes, the number of ‘likes’ on the comment by other users
* at: Comment timestamp, date and time of comment submission
* reviewCreatedVersion: App version at the time of the review, the version number of the Netflix app at the time the user submitted the review
* appVersion: App Version, the current Netflix app version number that distinguishes between reviews and app versions.

应用程序

中度可信度描述已自动生成

P1. Examples of dataset

## 3.2 Sentiment Analysis Model

### 3.2.1 Analysis Data

”Data analysis is often the first step in any data science project. It involves exploring and understanding the data to uncover insights and patterns.” (Shan, Wang, Chen, & Song, 2016, p. xx). Data analysis is a necessary step in building an efficient sentiment analysis model. First, a basic check of the dataset, including looking at the first few rows of data to understand the structure of the data and checking for missing values, ensures the quality and completeness of the data, which lays a solid foundation for subsequent model training. Next, a sentiment column is created based on the score column, and the score values are converted to ‘Positive’, ‘Neutral’, or ‘Negative’ sentiment labels. The creation of sentiment labels transforms quantitative ratings into sentiment categories, enabling the model to perform effective classification tasks. The distribution of sentiment labels is counted and visualised to understand the number and distribution of comments across different sentiment categories. This can help identify imbalances in the data and guide data preprocessing and model optimisation strategies.

文本

描述已自动生成

P2. View Data in sentiment analysis model

图表, 条形图

描述已自动生成

P3. Sentiment Label Distribution Visualisation in sentiment analysis model

Subsequently, the distribution of the text length of each comment is analysed, including calculating the text length and outputting its descriptive statistics, as well as visualising it uses histograms and kernel density estimation plots. This reveals the features of the comments and provides a basis for the selection of model input features and text processing strategies. Finally, the distribution of text lengths under different sentiment labels is analysed, and the differences in text lengths across sentiment categories are compared through box-and-line plots . This can further help to understand the text characteristics of each sentiment category, to optimise the feature engineering of the model. Overall, in-depth data analysis not only helps to understand the data itself, but also provides important guidance and support for model construction, training and optimisation.

Below are the parsing and screenshots of output of the relevant charts:

* Histograms and KDE： The horizontal axis indicates text length, and the vertical axis indicates frequency. The graph contains histogram bars and a smooth kernel density curve. The histogram shows the frequency of different text length intervals, while the kernel density estimation graph provides a smooth curve of the text length distribution. This graph provides a clear view of the common intervals and trends in the distribution of text lengths.
* Boxplot：The horizontal axis represents the sentiment labels (e.g., Positive, Neutral, Negative) and the vertical axis represents the text length. Each box shows the statistical characteristics of text length under different sentiment labels (e.g., Median, Quartile, Minimum, Maximum, and Outliers). The box-and-line plots allow comparison of the distributional characteristics and variability of text length under different sentiment labels.

图表, 直方图

描述已自动生成

P4. Histograms and KDE of text length distribution in sentiment analysis model

图表, 箱线图

描述已自动生成

P5. Boxplot of text length distribution in sentiment analysis model

Below is the code for analyzing data for the sentiment analysis model: 文本

描述已自动生成

P6. The analysis data code for the sentiment analysis model

### 3.2.2 Data preprocessing

“It is essential because the quality of data preparation directly impacts the effectiveness and accuracy of the analytics.” (Svolba, 2006). Data preprocessing is a key step in building a sentiment analysis model, and its purpose is to ensure the quality and consistency of the data. First, this paper deleted rows with missing values ​​in the `userName` and `content` columns and removed the `reviewCreatedVersion` and `appVersion` columns that are not relevant to the sentiment analysis task. Next, text cleaning was performed, including removing HTML tags and non-alphabetic characters, and converting the text to lowercase. To ensure the uniformity of the data language, this paper only retained reviews containing pure English characters and removed all blank text. Subsequently, word segmentation was performed, and common stop words were removed to reduce noise.

After cleaning and preprocessing the data, this paper checked the data and verified the completeness and accuracy of the data by outputting missing value statistics, data preview, and data structure information. These steps ensured the consistency of the data and laid a solid foundation for the subsequent training of the sentiment analysis model.

文本, 表格

描述已自动生成

P7. Check the cleaned data

The time column `at` was converted to date time format and sorted. Next, this paper analyzed the time range of the `at` column and calculated the monthly and yearly review volume statistics to understand the time distribution of the data. Based on these time distribution statistics, this paper selected `2024-01-01` as the split point of the dataset and divided the data into training and test sets by date. The training set contains data before this date, while the test set contains data after this date. Finally, this paper extracted text and sentiment labels as training data and test data respectively and checked the size of the dataset and the amount of data before and after the split point. These steps ensured the comprehensiveness of the data and the reasonable distribution of time series, laying a solid foundation for model training and evaluation.

表格

描述已自动生成

P8. Time distribution statistics

Below is the data preprocessing code for the sentiment analysis model:

文本

描述已自动生成

P9. Data preprocessing code for sentiment analysis model:

### 3.2.3 Feature Extraction

##### TF-IDF (Term Frequency-Inverse Document Frequency):

“In automatic text retrieval systems, the TF-IDF term weighting scheme has been found to significantly improve the retrieval performance by effectively distinguishing between relevant and non-relevant documents.”(Salton & Buckley, 1988). TF-IDF is a statistical method commonly used in text analysis and information retrieval to evaluate the importance of words in a document collection. In sentiment analysis, TF-IDF is widely used to extract text features as feature vectors to help models better understand and classify sentiment in text. By filtering common meaningless words (such as "the", "is", etc.), TF-IDF provides an effective way to identify and highlight important words that are more representative of the document content, laying a solid foundation for model training and classification. The formula is:

* TF (Term Frequency): TF measures the frequency of a word in a document. TF is the number of times the word appears in a document divided by the total number of words in the document. The formula is:

TF(t,d)

* IDF (Inverse Document Frequency): IDF measures the prevalence of a word in the entire document collection. The lower the IDF value, the more likely the word appears in most documents, and the lower the amount of information. The higher the IDF value, the more likely the word appears in fewer documents, and the higher the amount of information. The formula is:

A high TF means that the word occurs frequently in the current document. A high IDF means that the word is uncommon in the entire document collection (i.e., has high information value). A high TF-IDF score means that the word occurs frequently in the document and is relatively rare in other documents, so the word contributes more to the feature recognition of this document.

**文本

描述已自动生成**

**P10.** **TF-IDF code for sentiment analysis model**

##### BERT (Bidirectional Encoder Representation Model):

"BERT’s bidirectional approach allows it to deeply understand both the left and right context of a word, making it more effective than previous models that only processed text in one direction. This results in significant improvements across a wide range of NLP tasks, including sentiment analysis, question answering, and text classification" (Devlin et al., 2019). BERT is a deep learning model based on the Transformer architecture, designed for natural language processing tasks. Unlike traditional word vector models, BERT captures contextual information through bidirectional encoding, enabling it to understand the semantics of a sentence more accurately. When encoding a word, BERT can consider the context on both the left and right sides at the same time, which makes it particularly good at dealing with complex language phenomena. Through pre-training and fine-tuning strategies, BERT not only provides context-sensitive word embeddings, but also converts text into high-dimensional contextual embedding vectors to better represent the meaning of the text. These embedding vectors perform well in tasks such as sentiment analysis, enabling the model to fully understand and process the sentiment tendencies of the text, greatly improving the accuracy of the analysis and the generalization ability of the model.

文本

描述已自动生成

P11.Bert code for sentiment analysis model

### 3.2.4 Model

In this study, we selected four models, namely Logistic Regression, Random Forest, XGBoost and LSTM (Long Short-Term Memory Network), to handle sentiment analysis and usefulness score. These models represent linear models, ensemble learning models, gradient boosting models and deep learning models, respectively, and can perform effective classification tasks under different feature extraction methods (TF-IDF and BERT).

##### Logistic Regression：

"Logistic regression is a powerful tool for modeling binary outcomes, especially when the relationship between the independent variables and the dependent variable is not strictly linear" (Pampel, 2000). Logistic regression is a classic linear classification model that is widely used in binary classification tasks. The core idea is to fit the data by maximizing the log-likelihood function to predict the probability that a sample belongs to a certain category. Although the logistic regression model is relatively simple, it performs well when dealing with linearly separable data sets, especially when dealing with high-dimensional sparse data (such as TF-IDF feature vectors), it has stable performance. Its main advantages include easy implementation, easy interpretation, and efficient computing power. However, logistic regression can only capture linear relationships in the data and has difficulty dealing with complex nonlinear patterns.

###### Logistic Regression on TF-IDF feature vectors

In this research, I first used a logistic regression model on the TF-IDF feature vector and used a variety of methods to tune and optimize the model. First, I used a standard logistic regression model to perform a preliminary classification of the data. On this basis, I introduced category weight adjustment to deal with the problem of category imbalance, especially to improve the classification performance of the "Neutral" category. In addition, I further optimized the model's hyperparameters, such as the regularization parameter C and penalty, through grid search (GridSearchCV) to find the best combination of model parameters. In particular, I tried different feature extraction and weight adjustment methods to improve the model's performance on high-dimensional sparse data (TF-IDF feature vectors). This series of tuning steps ensures the stability and accuracy of the model when dealing with category imbalance and high-dimensional data, laying a solid foundation for subsequent sentiment analysis tasks.

文本

描述已自动生成

P12. Logistic regression code for sentiment analysis model based on TF-IDF feature vector

文本

描述已自动生成

P13. Logistic regression code for sentiment analysis model based on TF-IDF feature vector 2

###### Logistic Regression on BERT feature vectors

In this research, we used a logistic regression model based on BERT feature vectors for feature learning and prediction. First, we set the max\_iter parameter to 1000 to ensure that the model is fully trained, and class\_weight='balanced' to deal with the class imbalance problem. After training, we predicted the test data and calculated the overall accuracy of the model. In addition, we generated a classification report that shows the precision, recall, and F1 score for each emotion category in detail. These indicators provide detailed performance information of the model on different emotion labels, helping us evaluate its predictive ability on each category.

To more intuitively show the classification effect of the model, we plotted a confusion matrix heat map. The confusion matrix shows the comparison between the model's prediction results and the actual labels in the form of a heat map, clearly showing the predicted and actual distribution of each category. In the heat map, the dark area indicates that the model's predictions on these categories are more concentrated, while the light area reveals the possible confusion of the model.

图形用户界面, 文本, 电子邮件

描述已自动生成

P14. Logistic regression code for sentiment analysis model based on Bert feature vector

##### Random Forest：

“Random forests create a multitude of decision trees and use the majority vote to determine the final classification result, which significantly improves the model’s accuracy and robustness.”(Liaw, Wiener, 2002). Random forest is an ensemble learning method that performs classification or regression by building multiple decision trees and voting on their predictions. Its advantages are high accuracy and strong resistance to overfitting, which makes it perform well when dealing with high-dimensional data and complex features. In addition, random forest can effectively handle missing values ​​and unbalanced data. However, the disadvantages of random forest include high computational complexity, which may lead to slower training and prediction speed, especially when dealing with large-scale data sets. At the same time, because the model consists of many decision trees, its internal structure is poorly interpretable, and it is difficult to understand the specific decision-making process of the model.

###### Random Forest on TF-IDF feature vectors

In this research, to improve the performance of sentiment analysis models, an ensemble learning method based on TF-IDF feature vectors was used, combining Random Forest with other classifiers. First, I defined three basic classifiers: Logistic Regression, Random Forest, and Gradient Boosting. These models were integrated through soft voting, where the voting results were based on the predicted probability of each classifier. During the training process, we used TF-IDF feature vectors, evaluated the overall accuracy, precision, recall, and F1-score of the ensemble model, and visualized the classification results through the confusion matrix. In this way, the ensemble model can comprehensively utilize the advantages of each classifier, overcome the shortcomings of a single model, and improve the classification effect when processing complex and high-dimensional data.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P15. Includes Random Forest’s TF-IDF feature vector-based sentiment analysis model ensemble learning code

###### Random Forest on BERT feature vectors

In this research, we applied a random forest classifier based on BERT feature vectors for sentiment analysis. To build the model, we set n\_estimators=100 to ensure that the random forest model can fully learn the complex patterns in the data and used class\_weight='balanced' to deal with the class imbalance problem. After the model training was completed, we predicted the test data and calculated the overall accuracy of the model. In addition, we generated a classification report that shows the precision, recall, and F1 score for each sentiment category in detail. To more intuitively show the classification effect of the model, we plotted a confusion matrix heat map.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P16. Random Forest code for sentiment analysis model based on Bert feature vector

##### XGBoost：

XGBoost (Extreme Gradient Boosting) is an efficient boosting algorithm that is widely used in classification and regression tasks. It improves the prediction accuracy of the model by gradually optimizing a series of decision trees, and introduces a regularization mechanism to prevent overfitting, thereby improving the generalization ability of the model. Chen and Guestrin (2016) pointed out that "XGBoost is designed to handle the bias-variance tradeoff effectively by incorporating regularization, which enhances model generalization and performance". This regularization and optimization mechanism makes XGBoost perform particularly well when processing complex data sets. However, the training process of XGBoost may be complicated and may be slow when resources are limited, but its advantages in accuracy and flexibility make it the preferred model in practical applications.

###### XGBoost on TF-IDF feature vectors

In the process of further optimizing the model using TF-IDF feature vectors, I expanded the ensemble learning method and added the XGBoost classifier (XGBClassifier), forming a four-model ensemble system including logistic regression, random forest, gradient boosting and XGBoost. XGBoost is a powerful boosting method that can further improve classification performance through reinforcement learning. This extended model is also integrated through soft voting, combining the predictions of different models.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P17. Includes XGBoost's TF-IDF feature vector-based sentiment analysis model ensemble learning code

###### XGBoost on BERT feature vectors

In this research, we applied the XGBoost model based on BERT feature vectors for sentiment analysis. First, I configured the parameters of the XGBoost classifier, where objective='multi:softmax' specifies a multi-class classification task, num\_class=3 indicates that there are three sentiment categories, eval\_metric='mlogloss' is used to evaluate the multi-class logarithmic loss of the model, and use\_label\_encoder=False avoids the warning of the label encoder. After the model was trained, we made predictions on the test set and calculated the overall accuracy of the model.

Next, we generated a classification report for the XGBoost model and plotted a heat map of the confusion matrix.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P18. XGBoost code for sentiment analysis model based on Bert feature vector

##### LSTM：

LSTM (Long Short-Term Memory) is a special type of recurrent neural network (RNN) designed specifically for processing and predicting long-term dependencies in time series data. LSTM can effectively capture and retain long-term contextual information by introducing a gating mechanism, which makes it perform well in processing sequence data. Compared with traditional RNN, LSTM solves the problems of gradient vanishing and gradient explosion through the combination of forget gate, input gate and output gate, greatly improving the training stability and performance of the model. As stated by Hochreiter and Schmidhuber (1997), "LSTM networks are able to learn long-term dependencies in sequences due to their ability to maintain a cell state over time, which mitigates the vanishing gradient problem" (Hochreiter & Schmidhuber, 1997). Although LSTM is very powerful in capturing complex patterns in time series data, its training process may take a long time, and the model complexity is also high. Despite this, LSTM still performs well in tasks such as natural language processing and time series prediction and is an effective tool for processing complex sequence data.

###### LSTM on TF-IDF feature vectors

In this research, we used a variety of LSTM models based on the TF-IDF variable to explore its performance in sentiment analysis and made optimizations and improvements. First, we built a basic LSTM model that used an embedding layer to map vocabulary to dense vectors, then used an LSTM layer for feature extraction, and finally used a fully connected layer to output the classification results. The model showed good basic performance after training. Then, we enhanced the model by adding a higher dropout rate to reduce overfitting and further improving the model's expressiveness through a second LSTM layer. This improvement improved the performance during training and validation. Subsequently, we introduced batch normalization to speed up training and stabilize the learning process of the network. This model showed higher accuracy and more stable training process when processing complex data. Finally, to deal with the problem of class imbalance, we used a bidirectional LSTM model and introduced class weights and learning rate scheduling strategies to better handle bias in the data. Each model recorded the changes in loss and accuracy during training and conducted a detailed performance evaluation. Through these steps, we have fully exploited the potential of the LSTM model for sentiment analysis and optimized its performance under various conditions.

文本

描述已自动生成

P19. LSTM code for sentiment analysis model based on TF-IDF feature vector

文本

描述已自动生成

P20. LSTM code for sentiment analysis model based on TF-IDF feature vector 2

###### LSTM on BERT feature vectors

In this research, we used LSTM networks to model sentiment data based on BERT feature vectors. First, we preprocessed the data by adding the time step dimension and reshaping the training and test data into a format suitable for LSTM input (i.e. (samples, timesteps=1, features)). When building the model, we designed a network with two LSTM layers. The first LSTM layer has 128 units and uses a dropout rate of 0.3 and a recurrent dropout rate of 0.3 to reduce the risk of overfitting and is normalized by a batch normalization layer. The second LSTM layer has 64 units, and the dropout rate and recurrent dropout rate are also applied, and another batch normalization layer is added immediately. Subsequently, we used 32 ReLU activation function units in the fully connected layer (Dense layer) and applied a dropout rate of 0.5. Finally, we handle the multi-classification problem through an output layer with a softmax activation function of 3 units.

The model uses the Adam optimizer, the learning rate is set to 0.001, and the loss function is sparse\_categorical\_crossentropy. We set 5 training cycles (epochs) during the training process, and used a batch size of 32 for training, while retaining 10% of the data for validation. After the training, we predicted the test data, calculated the overall accuracy of the model, and generated a classification report. To evaluate the performance of the model during the training process, we plotted the curves of training loss and validation loss, and training accuracy and validation accuracy. These charts show the performance trend of the model in each training cycle, helping us to intuitively understand the convergence and generalization ability of the model.

图片包含 文本

描述已自动生成

P21 LSTM code for sentiment analysis model based on Bert feature vector

## 3.3 Usefulness Score Model

### 3.3.1 Data preprocessing

In the process of building the usefulness score model, data preprocessing is crucial to ensure the accuracy and effectiveness of the model. First, we deleted records with missing values ​​in the userName and content columns, and removed the reviewCreatedVersion and appVersion columns that are not relevant to the analysis. Next, we performed text cleaning, removed HTML tags and non-alphabetic characters, and converted the text to lowercase to unify the data format.

We further filtered out reviews that only contain English characters and removed blank text. For the text content, we performed word segmentation and stop word removal operations: split the text into words, removed common stop words, and retained words that are useful for analysis. These processed texts are stored in the cleaned content column.

To simplify data processing, we retained the cleaned content and thumbsUpCount columns and removed missing values ​​in the thumbsUpCount column. When converting the numerical data of thumbsUpCount into a classification problem, we set a threshold: records with thumbsUpCount greater than 0 are marked as 1 (indicating usefulness), otherwise they are marked as 0 (indicating uselessness). Finally, we save the processed data as a CSV file for subsequent model training and analysis. These data preprocessing steps ensure the integrity and consistency of the data, laying a solid foundation for building the usefulness score model.

文本

描述已自动生成

P22. Data preprocessing code for Usefulness Score model

### 3.3.2 Feature Extraction

The feature extraction method is the same as sentiment analysis. Here we will not repeat the feature extraction method, but only describe how to use it:

##### TF-IDF:

First, we use TfidfVectorizer to convert the cleaned text data cleaned\_content into TF-IDF feature vectors. To limit the dimension of the features, we set max\_features=5000, that is, only the top 5000 most important feature words are retained. This step converts the text data into a numerical format suitable for machine learning models. Then, we separate the converted TF-IDF feature matrix X from the target variable thumbsUpCount (representing the usefulness score). Next, we divide the dataset into training and test sets, where 20% of the data is used for testing and 80% of the data is used for training the model. This process is completed by the train\_test\_split function, and the random seed random\_state=42 is set to ensure the reproducibility of the results. These feature extraction steps ensure the numerical processing of the data, providing a reliable foundation for subsequent model training and evaluation.

文本

描述已自动生成P23. TF-IDF code for the Usefulness Score model

##### BERT:

First, we converted the target variable thumbsUpCount into a classification problem, where the tokens with thumbsUpCount greater than 0 are 1 (useful) and the rest are 0 (useless). Next, we loaded the pre-trained BERT model and its tokenizers (BertTokenizer and BertModel). The tokenizer is used to convert text data into an input format acceptable to the BERT model, while the model is used to generate an embedded representation of the text. During the feature extraction process, we defined an embed\_text function, which uses the tokenizer to convert the text into BERT input and obtains the embedded representation of each text through the model calculation. Specifically, we extracted the hidden state of the [CLS] tag output by the BERT model as the representation vector of each text. To process large-scale data, we implemented a batch\_embed\_texts function to batch the text data to improve computational efficiency.

In actual operation, we batch the text data into BERT embedding vectors. To speed up the processing process, we used multi-threaded parallel processing technology to batch input the text data into the BERT model and collect the generated embedding vectors.

Finally, we separate the generated BERT feature vector bert\_embeddings from the target variable thumbsUpCount and use train\_test\_split to divide the dataset into training and test sets, with 20% of the data used for testing and 80% for training. These steps ensure the effective representation of text data in a high-dimensional semantic space, laying a solid foundation for subsequent model training and evaluation.

文本, 应用程序, 电子邮件

描述已自动生成

P24. BERT code for the Usefulness Score model

### 3.3.3 Model

The method of model selection is similar to sentiment analysis. I will not explain each model here, but only describe how to use it:

##### Logistic Regression:

###### Logistic Regression on TF-IDF feature vectors

In this research, we applied the Logistic regression based on TF-IDF feature vector for feature learning and prediction. First, we built a basic logistic regression model and trained the model using TF-IDF feature vectors. After the model training was completed, we made predictions on the test set and calculated the overall accuracy of the model. We also generated a classification report and confusion matrix to evaluate the performance of the model. The confusion matrix shows the prediction results of the model on each category, which helps us further understand the accuracy and misclassification of the model. In addition, we used the trained model to calculate the usefulness score for each text in the dataset, that is, the probability that the model predicts it to be useful.

To improve the performance of the model, we tried several improvement strategies. First, we used the class\_weight='balanced' parameter to deal with the class imbalance problem, thereby improving the model's recognition ability for minority classes. Next, to solve the model convergence problem, we increased the number of iterations of the logistic regression model to 200 and standardized the feature data to improve the stability and efficiency of the training process. The standardization operation helps to accelerate the convergence of the model and improve performance.

In general, these optimization steps significantly improved the performance of the logistic regression model in the usefulness score prediction task and provided a more reliable analysis tool for our text classification task.

文本

描述已自动生成

P25. Logistic regression code based on TF--IDF feature vector in Usefulness Score model

###### Logistic Regression on BERT feature vectors

In this research, we also used a logistic regression model to classify text based on BERT feature vectors. First, BERT was used to generate text feature vectors, and these features were used to train a logistic regression model. The model set max\_iter=200 to ensure sufficient training and used class\_weight='balanced' to deal with the problem of class imbalance. After training, the model made predictions on the test set, we calculated the overall accuracy, and generated a classification report to evaluate the model performance. The heat map of the confusion matrix further shows the details of the prediction results, helping us analyze the performance of the model in practical applications.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P26. Logistic regression code based on BERT feature vector in Usefulness Score model

##### Random Forest:

###### Random Forest on TF-IDF feature vectors

In this research, we applied a random forest classifier based on TF-IDF feature vectors for feature learning and prediction. First, we used the TF-IDF feature vectors as input data to train the random forest model. The model was fitted on the training set, then predicted on the test set, and the overall accuracy of the model was calculated. The performance of the model was evaluated through classification reports and confusion matrices, which showed the prediction precision and recall of different categories.

To further optimize the model, we adjusted the class weights of the random forest in the second stage to deal with the class imbalance problem. We calculated the weights of each category and applied them to the model training, which improved the performance of the model when dealing with imbalanced data. After training and evaluation, we also generated classification reports and confusion matrices to evaluate the performance of the adjusted model.

Finally, to improve the generalization ability of the model, we used GridSearchCV for hyperparameter tuning. First, the main hyperparameters of the random forest (such as n\_estimators and max\_depth) were determined through preliminary grid search. Then, the hyperparameter range was further refined based on the preliminary results, and the min\_samples\_split parameter was optimized. Through two rounds of grid search, we found the best hyperparameter combination and used it to train the final model. The performance of the final model on the test set was verified through classification reports and confusion matrices, and usefulness score prediction results were generated. These steps ensured that the model had good predictive performance and accuracy when processing real data.

文本

描述已自动生成

P27. Random Forest Code Based on TF-IDF Feature Vector in Usefulness Score Model

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P28. Random Forest Code Based on TF-IDF Feature Vector in Usefulness Score Model 2

###### Random Forest on BERT feature vectors

In this research, we also applied a random forest classifier based on BERT feature vectors for feature learning and prediction. First, we built a random forest classification model by using the feature vectors generated by BERT as input data. To deal with the class imbalance problem, we calculated the weights for each class and applied these weights in model training. These weights helped the model improve performance when dealing with data with uneven class distribution.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P29. Random forest code based on BERT feature vector in usefulness score model

##### XGBoost:

###### XGBOOST on TF-IDF feature vectors

In this research, we applied the XGBoost classifier based on TF-IDF feature vector for feature learning and prediction. First, we built an XGBoost classifier and set the scale\_pos\_weight parameter to deal with the class imbalance problem. This parameter balances the sensitivity of the classifier to different categories by adjusting the weights of positive and negative class samples. After the model was trained, we used the test set for prediction and calculated the overall accuracy and classification report to show the performance of the model on different categories.

Next, we performed hyperparameter optimization to further improve the model performance. We first calculated the class weights and defined a parameter grid, which included parameters such as max\_depth, learning\_rate, n\_estimators, and scale\_pos\_weight. Through GridSearchCV, we evaluated the effects of different parameter combinations in the cross-validation process and selected the best parameters. Finally, we used the optimized model to predict the original data, calculated the prediction probability, and added the usefulness score to the data frame. This process ensured that the XGBoost model could make full use of the TF-IDF features, provide accurate text classification results, and optimize the model's parameter settings for the best performance.

文本

描述已自动生成

P30. XGBoost code based on TF-IDF feature vector in usefulness score model

###### XGBoost on BERT feature vectors

In this reseach, we also applied the XGBoost classifier based on BERT feature vectors for feature learning and prediction. First, we calculated the class weights to deal with the problem of class imbalance in the dataset. We used the scale\_pos\_weight parameter to balance the weights of positive and negative class samples, which helped improve the performance of the model on the minority class. Next, we built an XGBoost classifier and applied the calculated class weights to the model. By training the model and predicting on the test set, we obtained the overall accuracy and classification report. These results show the classification effect of the XGBoost model when processing BERT feature vectors. The visualization of the confusion matrix further helps analyze the performance of the model on different categories. This approach combines the powerful text representation ability of BERT and the efficient classification performance of XGBoost, thereby improving the overall prediction ability of the model.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P31. XGBoost code based on BERT feature vector in usefulness score model

##### LSTM

###### LSTM on TF-IDF feature vectors

In this study, we applied LSTM based on TF-IDF feature vectors for feature learning and prediction. First, we built an LSTM model to process TF-IDF feature vectors. In the model architecture, we used the Embedding layer to map the sparse TF-IDF features to the dense embedding space. Next, we added two bidirectional LSTM layers, each containing 64 units, to capture the contextual information of the text data. To prevent overfitting, we applied 50% Dropout between the LSTM layers. Finally, the model was classified through a fully connected Dense layer, outputting the probability distribution of the category, using the softmax activation function. In the model compilation stage, we selected the adam optimizer and the categorical\_crossentropy loss function, which are suitable for multi-class classification problems. During the model training process, 5 epochs and a batch size of 64 were used, and class weights were applied to deal with the class imbalance problem. After training, we used the test data for prediction and evaluated the model performance through classification\_report and confusion matrix. During the training process, we also plotted the curves of training and validation loss and accuracy to observe the convergence of the model.

In further optimization, we increased the number of LSTM units and introduced L2 regularization to improve the generalization ability of the model. In addition, we set the Early Stopping callback to prevent overfitting and restore the optimal weights. By increasing the number of training cycles to 10 and applying the same category weights and batch size, we optimized the model performance and generated prediction results.

This series of steps ensures that the LSTM model can effectively learn from TF-IDF features and perform accurate usefulness classification.

文本

描述已自动生成

P32. LSTM code based on TF-IDF feature vector in usefulness score model

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P33. LSTM code based on TF-IDF feature vector in usefulness score model 2

###### LSTM on BERT feature vectors

In this study, we also applied LSTM based on BERT feature vectors for feature learning and prediction. First, we design a sequence model containing a bidirectional LSTM layer and a Dropout layer to process text features extracted from BERT. The model was compiled using the Adam optimizer and binary\_crossentropy loss function, and the training and validation losses and accuracy were recorded through the training process. To deal with the data imbalance problem, we applied SMOTE technology to oversample the training data and calculated category weights to balance the training process. Through grid search, we optimized the number of LSTM units and dropout rate and selected the hyperparameter configuration that performed best on the validation set. Finally, the evaluation on the test set showed that our LSTM model performed well in the classification task, with significantly improved accuracy. The confusion matrix and classification report further verified the effectiveness of the model.

文本

描述已自动生成

P34. LSTM code based on BERT feature vector in the usefulness score model

文本

中度可信度描述已自动生成

P35.LSTM code based on BERT feature vector in the usefulness score model 2

文本

低可信度描述已自动生成

P36.LSTM code based on BERT feature vector in the usefulness score model 3

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

P37.LSTM code based on BERT feature vector in the usefulness score model 4

# 4. Experiments and Results

## 4.1 Evaluation Metrics

We used a variety of evaluation metrics to comprehensively evaluate the performance of sentiment analysis models. These metrics include (TP is the number of correct predictions for the positive class, TN is the number of correct predictions for the negative class, FP is the number of wrong predictions for the negative class, and FN is the number of wrong predictions for the positive class):

##### Accuracy

Accuracy measures the proportion of all samples that are correctly classified by the model. The calculation formula is:

##### Precision

Precision measures the accuracy of the model's predictions for the positive class. The higher the precision, the more accurate the model is in predicting the positive class. Its calculation formula is:

##### Recall

Recall measures the model's ability to identify all positive classes. The higher the recall, the higher the model's coverage in identifying positive classes. Its calculation formula is:

##### F1 Score

The F1 score is the harmonic mean of precision and recall, which is used to comprehensively consider the accuracy and comprehensiveness of the model. Its calculation formula is:

##### Macro F1 Score：

**Macro F1 score calculates the F1 score of each category and takes the average. This indicator focuses on the performance of all categories and is suitable for situations where the categories are unevenly distributed.**

##### Weighted F1 Score：

Weighted F1 score calculates the F1 score for each category and takes a weighted average based on the number of samples in each category. This indicator better reflects the actual classification performance and is suitable for situations where the number of category samples varies greatly.

手机屏幕截图

描述已自动生成

P38. Evaluation Metrics

##### Confusion Matrix

The confusion matrix is ​​a table used to evaluate the performance of a classification model. It shows how each predicted class relates to the actual class, including:

* **TP(True Positive):**The number of correctly predicted positive classes
* **FP(False Positive):**The number of incorrectly predicted positive classes.
* **TN(True Negative):** The number of correctly predicted negative classes.
* **FN(False Negative):** The number of incorrect predictions for the negative class.

Through the confusion matrix, we can intuitively understand which categories the model performs well on and which categories it is lacking. For example, for logistic regression and random forest models, we can visualize the classification effect of the model through the code that generates the confusion matrix.

手机屏幕截图

中度可信度描述已自动生成

P39. Confusion matrix code

图表, 树状图

描述已自动生成

P40. Confusion Matrix Visualization

##### Loss Curve

The loss curve shows the change in the loss value of the model during the training and validation process in each epoch. The training loss reflects the degree of fit of the model to the training data, while the validation loss evaluates the performance of the model on unseen data. Through this curve, we can intuitively observe the convergence of the model. If the training loss continues to decrease, while the validation loss increases, it may indicate that the model has an overfitting problem.

图表, 折线图

描述已自动生成

P41. Loss Curve

##### Accuracy Curve

The accuracy curve shows how the accuracy of the model on the training and validation data changes with each epoch. The training accuracy reflects the model's ability to predict the training data, while the validation accuracy is used to evaluate the model's performance on unseen data. By plotting the training and validation accuracy curves, the learning progress of the model can be observed. A steadily rising accuracy usually indicates that the model performance is constantly improving, while fluctuations in the accuracy may suggest that the model parameters or training strategy need to be adjusted.

图表, 折线图

描述已自动生成

P42. Accuracy Curve

## 4.2 Sentiment Analysis Model Result

We show the performance of different models in sentiment analysis tasks. The following are the specific results of the model under TF-IDF and BERT feature extraction methods:

##### Performance of each model based on TF-IDF

|  |  |
| --- | --- |
| Selected Model | Accuracy |
| Logistic Regression Model | 83.12% |
| Logistic Regression Model +Class weight(neutral:3) | 79.50% |
| Logistic Regression Model +Class weight(neutral:2) | 81.73% |
| Logistic Regression Model + Hyperparameter Tuning (Grid Search) | 81.57% |
| Logistic Regression Model + Feature extraction optimization (ngram\_range=(1,2)) | 77.20% |
| Logistic Regression Model + Feature extraction optimization (ngram\_range=(1,3)) | 76.92% |
| Ensemble Learning (Logistic Regression,Random Forest,Gradient Boosting) | 81.30% |
| Ensemble Learning (Random Forest,Gradient Boosting,XGBoost) | 80.96% |
| LSTM Model | 79.83% |
| LSTM Model + Dropout | 82.52% |
| LSTM Model + Dropout and BatchNormalization | 81.85% |
| Bidirectional LSTM Model + Class weight | 74.15% |

图表, 条形图

描述已自动生成

P43. Logistic Regression Model

图表, 树状图

描述已自动生成

P44. Logistic Regression Model + Class Weight

图表, 树状图

描述已自动生成

P45. Logistic Regression Model + Class Weight 2

图表, 树状图

描述已自动生成

P46. Logistic Regression Model + GridSearchCV

图表, 树状图

描述已自动生成

P47. Logistic regression model + Feature extraction optimization

图表, 树状图

描述已自动生成

P48. Logistic regression model + Feature extraction optimization2

图表, 条形图

描述已自动生成

P49. Ensemble Learning (Logistic regression, Random forests and Gradient boosting)

图表

描述已自动生成

P50. Ensemble Learning (Random forests, Gradient boosting and XGBoost)

图表, 折线图

描述已自动生成

P51. LSTM Model

图表, 折线图

描述已自动生成

P52. LSTM Model + Dropout

图表, 折线图

描述已自动生成

P53. LSTM Model + Dropout and BatchNormalization

图表, 折线图

描述已自动生成

P54. Bidirectional LSTM model + Class weights

##### Performance of each model based on Bert

|  |  |
| --- | --- |
| Selected Model | Accuracy |
| Logistic Regression Model | 75.09% |
| Random Forest Model | 75.09% |
| XGBoost Model | 80.87% |
| LSTM Model | 82.63% |

图表, 条形图

描述已自动生成

P55. Logistic Regression Model

图表, 树状图

描述已自动生成

P56. Random Forest Model

图表

描述已自动生成

P57. XGBoost Model

手机屏幕截图

描述已自动生成

P58. LSTM Model

图表, 折线图

描述已自动生成

P59. LSTM Model Loss and Accuracy Curve

##### Result

Among all the models, the LSTM model with BERT feature vectors performed best with an overall accuracy of 82.63%. For negative sentiment classification, the LSTM model combined with Dropout and BatchNormalization achieved the highest F1 score of 0.88. The LSTM model with BERT feature vectors was on par with the LSTM model + Dropout in positive sentiment classification, both achieving an F1 score of 0.80. However, for neutral sentiment classification, no model performed well, and the LSTM model with BERT feature vectors and the XGBoost model performed the worst in this category. Considering the overall accuracy and the performance of each category, the LSTM model with BERT feature vectors performed well in both overall performance and negative sentiment classification. Although the model's performance in the Neutral class was not satisfactory, considering that the Neutral class performed poorly among all models, further optimization work may include more in-depth analysis or data augmentation of Neutral class samples, or trying more complex model structures to improve its performance.

## 4.3 Usefulness Score Model Result

We show the performance of different models in the usefulness score task. Here are the specific results of the model under TF-IDF and BERT feature extraction methods:

##### Performance of each model based on TF-IDF

|  |  |
| --- | --- |
| Selected Model | Accuracy |
| Logistic Regression Model | 76.48% |
| Logistic Regression Model +class\_weight=balanced | 72.47% |
| Logistic Regression Model + Increasing Iterations | 68.64% |
| Random Forest Model | 75.21% |
| Random Forest Model + Class Weight | 75.39% |
| Random Forest Model + GridSearchCV | 73.69% |
| XGBoost Model | 75.39% |
| XGBoost Model + GridSearchCV | 73.69% |
| LSTM Model | 64.15% |
| LSTM Model + Increase units and L2 regularization | 66.02% |

图表, 树状图

描述已自动生成

P43. Logistic Regression Model

图表, 树状图

描述已自动生成

P44. Logistic Regression Model + Class Weight

图表

中度可信度描述已自动生成

P45. Logistic Regression Model + Increase Iterations

图表, 树状图

描述已自动生成

P46. Random Forest Model

图表, 树状图

描述已自动生成

P47. Random Forest Model + Class Weight

图表, 树状图

描述已自动生成

P48. Random Forest Model + Hyperparameter Tuning with GridSearchCV

图表, 树状图

描述已自动生成

P49. XGBoost Model

图表, 树状图

描述已自动生成

P50. XGBoost Model + Tuning with GridSearchCV

图表, 折线图

描述已自动生成

P51. LSTM model

图表, 折线图

描述已自动生成

P52. LSTM Model, Increased units and L2 regularization

##### Performance of each model based on Bert

|  |  |
| --- | --- |
| Selected | Accuracy |
| Logistic Regression Model | 71.79% |
| Random Forest Model | 73.99% |
| XGBoost Model | 71.05% |
| LSTM Model | 68.64% |
| LSTM Model + Smote, Class weights | 74.70% |
| LSTM Model + Smote, Class weights2 | 75.25% |

图表, 树状图

描述已自动生成

P. Logistic Regression Model

图表, 树状图

描述已自动生成

P. Random Forest Model

图表, 树状图

描述已自动生成

P. XGBoost Model

图表

描述已自动生成

P. LSTM Model

图表, 折线图

描述已自动生成

P. LSTM Model 2

图表, 树状图

描述已自动生成

P. LSTM+Smote, Class weights Confusion Matrix

图表, 折线图

描述已自动生成

P. LSTM+Smote, Class weights Loss and Accuracy Curve

图表, 折线图

描述已自动生成

P. LSTM+Smote, GridSearch Loss and Accuracy Curve

图表, 树状图

描述已自动生成

P. Best LSTM+Smote, GridSearch Confusion Matrix

##### Result

In general, the models based on BERT feature representation performed better, especially the combination of BERT and LSTM model (accuracy 74.70%) and BERT and random forest model (accuracy 73.99%). These models have a more balanced recognition of the two types of labels in the classification task, especially in the performance of Class 1. In contrast, the performance of the model based on TF-IDF is slightly inferior, especially in the classification task of Class 1. Although the performance of random forest and XGBoost is acceptable, the overall effect is not at the best level.

# 5. Critical Discussion

## 5.1 Comparison of Model Performance

In the sentiment analysis and usefulness scoring tasks, although the BERT+LSTM model performs well in processing long texts and complex semantics and has higher accuracy and F1 scores than the traditional TF-IDF method, its superiority is also accompanied by some limitations. The BERT+LSTM model can effectively capture the deep information and long-term dependencies in the text with the pre-trained language representation ability of BERT and the sequence modeling ability of LSTM, which greatly improves the model's understanding of sentiment and usefulness. However, the complexity of this model also brings about an increase in computational overhead, and the training and inference time are significantly longer than the TF-IDF-based method. In addition, although the BERT+LSTM model performs best in overall accuracy and negative sentiment classification (Negative), its performance in neutral sentiment classification is still unsatisfactory, and it has not been significantly improved compared with the XGBoost model and other models. This shows that when dealing with certain categories, the BERT+LSTM model may need further tuning or combined with other techniques to improve the classification effect. Therefore, although the BERT+LSTM model has strong text processing capabilities, in practical applications it is still necessary to weigh its complexity and performance and optimize the model based on the requirements of specific tasks.

## 5.2 Comparison of Feature extraction methods

The TF-IDF method performs well in feature extraction of text data, mainly because it can effectively extract word frequency information and the relative importance of words, but its limitation is that it ignores contextual relationships and semantic information. In contrast, the BERT method captures richer contextual information and semantic relationships through a pre-trained deep language model, which gives it a clear advantage in feature representation. Although the BERT model has a large computational overhead in the training and inference stages and requires more time and resources, the refined feature representation it provides significantly improves the accuracy and reliability of sentiment analysis and usefulness scoring tasks. However, the high computational complexity and resource requirements of BERT may bring challenges in practical applications, and it is necessary to weigh its efficiency and application feasibility. In terms of the choice of feature extraction methods, although BERT has obvious advantages in accuracy, TF-IDF is still an effective alternative for scenarios with limited resources. Therefore, in different application scenarios, it is necessary to comprehensively consider the performance of the model, computing resources, and actual needs to select the most appropriate feature extraction method.

## 5.3 Analysis of the effectiveness of LSTM architecture on the problem of training/validation loss fluctuation

In this article, I found that the neural network training/validation loss had a peak problem during the construction of the sentiment analysis and usefulness scoring models. This indicates that the model overfits or underfits the data at a certain stage.

LSTM can effectively alleviate this problem through its unique mechanism. First, the gating mechanism of LSTM can selectively remember or forget information, thereby reducing the problems of gradient vanishing and gradient explosion, making the training process smoother and avoiding sudden peaks in the loss function. Secondly, LSTM is good at processing dependencies in long sequence data, especially in tasks such as sentiment analysis, where the emotional meaning of a word often depends on the previous and next contexts. LSTM can effectively capture this contextual dependency, thereby reducing the occurrence of loss peaks. In addition, the memory unit of LSTM can retain important information over multiple time steps, which is particularly important in sentiment analysis and usefulness scoring tasks that process long texts or complex semantics, allowing the model to better remember long-range dependencies related to sentiment or usefulness, thereby enhancing the stability and performance of the model.

The data often contains noise or class imbalance (such as fewer neutral class samples in sentiment analysis), which can also cause peaks in training/validation loss. I combined a series of techniques to optimize the process. The application of Dropout and Batch Normalization can effectively reduce overfitting, making the model training process for different batches of data more stable and reducing the fluctuation of validation loss. At the same time, in order to deal with the problem of data imbalance, by assigning class weights and using SMOTE, the LSTM model can learn the minority class more effectively, further reducing the instability of validation loss. The design of multi-layer LSTM architecture enhances the performance of the model. It extracts features layer by layer through multi-level information abstraction, better fits complex input data, especially the use of bidirectional LSTM, which helps to capture the forward and backward information of the text at the same time, thereby improving the understanding of semantics. The application of learning rate scheduler (ReduceLROnPlateau) and early stopping mechanism (EarlyStopping) helps prevent the loss function from oscillating and overfitting due to long training time, further smoothing the training process and reducing the peak of validation loss. The combination of these techniques makes LSTM more robust when dealing with complex data and imbalance problems, thereby enhancing the stability of the model.

## 5.4 Advantages and Disadvantages of Ensemble Pretrained Embeddings in LSTM

In building the sentiment analysis model and the usefulness score model, we used the pre-trained embedding generated by BERT as the input layer of the LSTM model and explored the advantages and disadvantages of integrating pre-trained embedding in the LSTM model.

Compared to the randomly initialized embedding layer, which only provides fixed and static word vectors and cannot dynamically adjust or capture contextual information, the BERT pre-trained embedding shows excellent contextual understanding ability. Because BERT uses a bidirectional Transformer architecture, it can consider both the left and right sides of the context when generating word embeddings. This context-aware ability is particularly important for the sentiment analysis task of LSTM, because sentiment often depends on the subtle relationship between words. For example, BERT can distinguish the different meanings of "very" in "very good" and "very bad", thereby capturing sentiment more accurately.

Randomly initialized embedding layers usually require a lot of training data to optimize their semantic expression ability. TF-IDF embedding is mainly based on word frequency and document frequency, lacking a deep understanding of word meaning and context. In contrast, integrating BERT pre-trained embeddings can capture richer semantic information. BERT is pre-trained on a large-scale corpus, so it can provide deeper and more accurate semantic representations, allowing LSTM to more effectively use these embeddings for sentiment classification, thereby improving model performance.

In terms of training efficiency, randomly initialized embedding layers require more training time and data to optimize, while TF-IDF embeddings have less computational overhead, but usually need to be used in combination with other features or models to achieve better results. Relatively speaking, BERT has been pre-trained on a large amount of data, so directly using its embeddings can significantly reduce training time and data requirements. The LSTM model only needs to fine-tune the pre-trained BERT embeddings instead of training the embedding layer from scratch.

Although integrating BERT pre-trained embeddings has a wealth of advantages, it also has some disadvantages that cannot be ignored. First, BERT has a large computational and memory overhead, especially when processing long texts or large-scale data, which usually requires higher computing resources and storage space, which may limit its application in resource-constrained environments. In contrast, randomly initialized embedding layers and TF-IDF embeddings have lower computational costs and are more suitable for scenarios with limited computing resources, but they are deficient in feature richness and contextual understanding. In addition, the corpus pre-trained by BERT may differ from the specific task domain, so in some cases additional domain fine-tuning is required to better suit the needs of specific tasks. Randomly initialized embedding layers can be trained from scratch for specific tasks, although this requires a large amount of domain-specific data to achieve optimal results.

In summary, integrating BERT pre-trained embeddings in LSTM models can significantly improve the model's contextual understanding ability and semantic expression richness, and reduce training time and data requirements. However, this approach also faces high computational and memory overhead, as well as potential domain adaptability issues. In contrast, although randomly initialized embedding layers and TF-IDF embeddings have lower computational costs, their capabilities in semantic expression and contextual understanding are relatively limited. Therefore, it is crucial to choose the right embedding method based on the specific application scenario and resource constraints, balancing performance improvements while considering the feasibility of computing resources.

# 6. Conclusion

This paper conducts a detailed evaluation of the performance of various models in sentiment analysis and usefulness scoring tasks. The results show that the combination of BERT and LSTM models shows the best results in these two tasks. LSTM's ability to process time series data and BERT's advantages in context understanding complement each other, making this deep learning model show significant superiority when processing complex text features.

In the sentiment analysis task, the BERT+LSTM model outperforms the traditional TF-IDF method in both accuracy and F1 score, proving the accuracy of deep learning methods in capturing the emotional information of comments. LSTM's sequence modeling ability effectively improves the understanding and prediction ability of long texts.

In the usefulness scoring task, the BERT+LSTM model also shows excellent performance, with significant improvements in accuracy and F1 score compared to TF-IDF+logistic regression and random forest models. This shows that deep learning models can not only understand the content of comments more accurately, but also more effectively predict the actual usefulness of comments.

Although the BERT+LSTM model has obvious advantages in overall performance, it has high computational overhead, and the training and inference time are significantly longer than the TF-IDF method. Future research can focus on exploring more advanced deep learning models, such as new models based on the Transformer architecture, which may provide more accurate feature representations. In addition, transfer learning techniques combined with pre-trained models are also worthy of in-depth study to further improve the performance and adaptability of the model. To enhance the generalization ability of the model, a variety of data augmentation techniques and optimization strategies should be considered, while finding a balance between model performance and computing resource requirements. By applying these new methods and techniques, the effectiveness and efficiency of the model can be further improved.

# Reference

***Tomaiuolo, N.G., 2012. UContent: The information professional's guide to user-generated content. Medford, New Jersey: Information Today, Inc.***

***Dang, N.C., Moreno-García, M.N. and De la Prieta, F., 2020. Sentiment Analysis Based on Deep Learning: A Comparative Study. Electronics, 9(3), p.483.***

***Shan, C., Wang, H., Chen, W., & Song, M. (2015). The Data Science Handbook. Data Science Bookshelf, New York.***

***Svolba, G., 2006.*** *Data Preparation for Analytics Using SAS. Cary, NC: SAS Institute.*

*Salton, G. and Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), pp.513-523.*

*Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), pp.4171-4186.*

*Pampel, F.C., 2000. Logistic Regression: A Primer. Sage Publications.*

*Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. R news, 2(3), 18-22.*

*Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 785-794.*

# Acknowledgements

I am deeply grateful to my advisor, Luis Espinosa-Anke, for his help in this research. He not only provided valuable ideas for the two models on the dataset, but also helped me complete the initial schedule, which enabled me to advance the research in an orderly manner according to the plan. In addition, he also suggested that I build things like GitHub repositories, which further improved the organization and reproducibility of the research. In particular, he provided me with important guidance and support during the model building and report writing process. I would like to express my sincere gratitude for his help, professional advice and patient guidance.

In addition, I would like to thank Kaggle for providing data support, which enabled me to obtain and use high-quality datasets for research. At the same time, I am also grateful to the library search of Cardiff University for providing paper reference support, which laid a solid foundation for my literature research and theoretical foundation.

Finally, I would like to thank everyone who helped and supported me during this process.