The University of Manchester

COMP30040-Report

Classification of app reviews for requirements engineering using machine learning models.

Supervisor: Dr. Liping Zhao
Author: Tianpeng Huang
Student ID: 10653872

April 21, 2023

Contents

1 Introduction **2**
1.1 Context and Motivation . . . 2
1.2 Aims and Objectives . . . 3
1.3 Report Structure . . . 3

2 Background and Related Work **4**
2.1 Background to User Review Classification . . . 4
2.2 Related Approaches to User Review Classification . . . 4
2.2.1 Lexicon Based Method . . . 4
2.2.2 Machine Learning Method . . . 5
2.2.3 Advanced language model . . . 6

3 Research Methodology **7**
3.1 Data Collection . . . 7
3.1.1 Inter-Annotator Agreement . . . 7
3.2 Data Preprocessing . . . 8
3.2.1 Tokenization . . . 8
3.2.2 Case Folding . . . 8
3.2.3 Stop Words Removal . . . 8
3.2.4 Stemming and Lemmatization . . . 9
3.2.5 Data Augmentation . . . 9
3.3 Lexicon-Based Model . . . 10
3.3.1 Keywords Matching . . . 10
3.4 Machine Learning Model . . . 10
3.4.1 Feature Extraction . . . 10
3.4.2 K-Nearest Neighbors . . . 12
3.4.3 Naive Bayes . . . 13
3.4.4 Support Vector Machines . . . 14
3.5 Advanced Language Model . . . 16
3.5.1 BERT Model . . . 16
3.5.2 RoBERTa Model . . . 18
3.5.3 XLNet Model . . . 19
3.5.4 GPT2 . . . 21
3.6 Evaluation Metrics . . . 23
3.6.1 Confusion Matrix . . . 23
3.6.2 Precision . . . 24
3.6.3 Recall . . . 24
3.6.4 F1-Score . . . 24
3.6.5 Accuracy . . . 24
3.7 Programming Environment and Libraries . . . 24
3.7.1 Programming Environment . . . 24
3.7.2 Libraries . . . 24

4 Building Machine Learning Models **25**
4.1 Design . . . 25
4.2 Part1-Review Collection . . . 25
4.2.1 Cohen's Kappa coefficient . . . 25
4.2.2 Dataset Analysis . . . 26
4.3 Part2-Data processing . . . 27
4.3.1 Data Preprocessing . . . 27
4.3.2 Dataset Segmentation . . . 27
4.4 Part3-Model Implementation . . . 27
4.4.1 Rule-based Model Implementation . . . 28
4.4.2 Machine Learning Model Implementation . . . 28
4.4.3 Advanced Language Model Implementation . . . 28
4.5 Part4-Model Evaluation . . . 29
4.5.1 Overfitting and Underfitting . . . 29

4.5.2 Training loss and Validation loss . . . 29

5 Results Analysis **31**
5.1 Lexicon-Based Model Results . . . 31
5.2 Machine Learning Model Results . . . 32
5.3 Advanced Language Model Results . . . 34
5.4 Discuss the impact of factors on model performance . . . 36
5.4.1 Impact of preprocessing on performance . . . 36
5.4.2 Impact of dataset balance on performance . . . 37
5.4.3 Impact of dataset size on performance . . . 38

6 Research Evaluation **39**
6.1 Comparison of My Project with Related Work . . . 39
6.2 Reflection . . . 39
6.3 Limitations . . . 40

7 Conclusion **41**
7.1 Achievements . . . 41
7.2 Future Work . . . 41

List of Figures

1.1 Availability of apps in major app stores as of the third quarter of 2022 . . . 2

2.1 Lexicon Based Method . . . 5
2.2 Models for machine learning . . . 5

3.1 Example of Tokenization . . . 8
3.2 Wordcloud of stop words . . . 8
3.3 Luhn's Hypothesis . . . 9
3.4 Example of stemming and lemmatization . . . 9
3.5 Example of Data Augmentation . . . 10
3.6 Example of BOW . . . 11
3.7 Example of Term Frequency-Inverse Document Frequency . . . 11
3.8 KNN(K=3) . . . 12
3.9 Calculate Distance . . . 12
3.10 SVM . . . 14
3.11 Margin Maximisation . . . 15
3.12 Transformer Architecture . . . 16
3.13 BERT Model . . . 16
3.14 BERT Structure . . . 17
3.15 BERT embedding . . . 17
3.16 Next Sentence Prediction . . . 18
3.17 Improvements of RoBERT . . . 18
3.18 RoBERTa's batch size experiment . . . 19
3.19 RoBERTa increase data and training steps experimental renderings . . . 19
3.20 An example of the Transformer-XL model with four segments. . . . 20
3.21 Two Stream Self Attention . . . 20
3.22 Schematic diagram of calculating the output of the third word in different permutations . . . 21
3.23 GPT2 architecture . . . 22
3.24 Binary Confusion Matrix . . . 23
3.25 Multiclass Confusion Matrix . . . 23

4.1 Structure of the project implementation . . . 25
4.2 Dataset1 Analysis . . . 26
4.3 Dataset2 Analysis . . . 26
4.4 Dataset3 Analysis . . . 26
4.5 Dataset after preprocessing . . . 27
4.6 Model loss value . . . 29

5.1 Model Performance Results Line Chart in Keyword Matching . . . 31
5.2 Model Performance Results Line Chart, KNN (top left), Naive Bayes (top right), SVM (bottom) 33
5.3 Model Performance Results Line Chart, BERT (top left), RoBERTa (top right), XLNET (bottom left), GPT2 (bottom right) . . . 35
5.4 Model Performance Results Line Chart of Different Pre-processing Techniques . . . 36
5.5 Model Performance Results Line Chart of balanced and unbalanced datasets . . . 37
5.6 Model Performance Results Line Chart of Different dataset size . . . 38

List of Tables

1.1 Aims and Objectives 3

3.1 Example of Inter-Annotator Agreement 7
3.2 Keywords Dictionary 10

4.1 Dataset1 IAA 25
4.2 Dataset1 IAA 25
4.3 Number of labels in each dataset and text length 27

5.1 Keyword Matching Model Results 31
5.2 KNN Model Results 32
5.3 Naive Bayes Model Results 32
5.4 SVM Model Results 32
5.5 BERT Model Results 34
5.6 RoBERTa Model Results 34
5.7 XLNet Model Results 34
5.8 GPT2 Model Results 34
5.9 Model Accuracy with Different Pre-processing Techniques 36
5.10 Model Accuracy with balanced and unbalanced datasets 37
5.11 Model Accuracy with Different Dataset size 38

6.1 Model Performance Accuracy(Comparison of My Project with Related Work) 39

Abstract

Applications have become ubiquitous, with almost 3 million apps available on app stores such as Google Play catering to a wide range of needs and services. These apps are used and reviewed by billions of users, making user feedback an important source of information for developers. To help developers quickly identify and address issues and improve the overall user experience, this paper introduces eight models for classifying app reviews into four categories: bug reports, feature requests, user experiences, and ratings. These models belong to three categories, namely rule-based models (keyword matching), machine learning models (KNN, SVM, and NB), and advanced language models (BERT, RoBERTa, XLNet, and GPT2). We conduct a series of experiments to compare the performance of these models and investigate the impact of different preprocessing techniques, dataset balance and size on classification performance.

Our study compared various text classification models, including high-level language models, machine learning, and keyword matching. The high-level language model had the best performance, with an average F1 score of 0.85 across four models, while the machine learning models achieved an average F1 score of 0.736. In contrast, the keyword matching model had the worst performance, with an F1 score of only 0.31. Larger datasets improved performance, with average F1 scores of 0.74 and 0.62 for large and small datasets, respectively. Interestingly, the unbalanced dataset had better performance than the balanced dataset, with average F1 scores of 0.68 and 0.62, respectively. Removing some stop words performed slightly better than removing all stop words, with average F1 scores of 0.61 and 0.59, respectively. Our findings have important implications for improving text classification models in various applications.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Liping Zhao, for introducing me to the world of Machine Learning and Natural Language Processing, and for providing her invaluable support and guidance throughout my dissertation journey. Dr. Zhao's expertise in the field of NLP and her insightful feedback were instrumental in shaping the direction of my research. I am also grateful for her contribution to the project and her counsel throughout this year, as her input was always helpful and greatly appreciated. It was a pleasure working with Dr. Zhao, and I am fortunate to have had her as my supervisor.

In addition, I would like to extend my heartfelt thanks to my parents for their unwavering love, support, and encouragement throughout my academic journey. Their constant encouragement and belief in me have been a source of inspiration and motivation, and I am truly grateful for their presence in my life.

Finally, I am grateful to the University of Manchester for hosting four amazing years of my life. The University provided me with the necessary resources and facilities, and the academic environment was conducive to my growth and development as a student. I am proud to be an alumnus of this prestigious institution.

Introduction

1.1 Context and Motivation

Applications have become an integral component of our daily lives, and it is challenging to envision a world without them [1]. With over 3 million applications available on app stores such as Google Play and the Apple App Store (Figure 1.1), there is virtually an app for every conceivable purpose. In many software markets, technical issues are no longer the primary determinant of an app's success. Instead, the user experience has emerged as the key factor, with evaluations playing a pivotal role [2]. For prospective users who have not yet engaged with an app, its reviews can serve as an invaluable source of information. Both positive and negative feedback can significantly influence the app's trajectory and may ultimately determine its success or failure. In essence, user reviews has become a crucial determinant in the success or failure of an application.

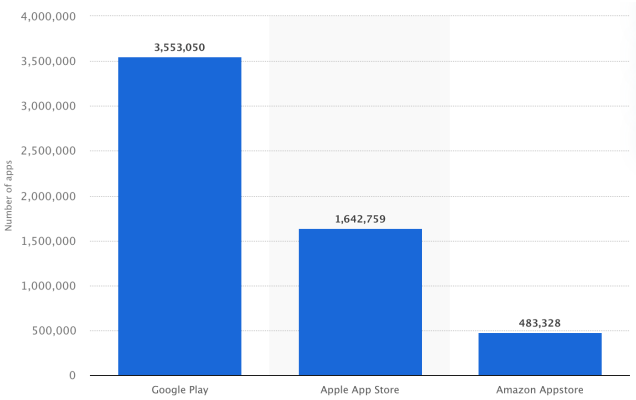

Figure 1.1: Availability of apps in major app stores as of the third quarter of 2022

In developing a tool to classify app reviews, the initial step involves automatically categorizing them based on the type of information they convey. Specifically, we assess various classifiers for organizing reviews into four categories: bug reports, feature requests, user experiences, and ratings. **Bug reports** pinpoint issues within the app necessitating resolution, such as crashes, incorrect behaviors, or performance problems. Conversely, **feature requests** embody users' aspirations for currently absent functionality or content, along with suggestions for enhancing the app's features in subsequent releases. **User experiences** encapsulate the overall interactions of users with the app and its features in particular contexts, thus supplying invaluable documentation pertaining to the app, its requirements, and its features. Lastly, **ratings** comprise simplistic textual representations of the numeric star rating, which offer limited insight as they merely convey praise, criticism, or discouragement [3].

The classification of reviews can significantly contribute to streamlining the distribution of various types to appropriate software project stakeholders [3]. For instance, bug reports can be directed to developers and testers for product maintenance, while feature requests can be allocated to requirements analysts. Meanwhile, user experience reports can be provided to business operators, enabling them to make well-informed decisions regarding future application updates and enhancements. Consequently, this targeted distribution of review types facilitates more efficient decision-making processes and supports the overall improvement of applications.

In order to investigate the performance of various factors on eight distinct classifier models (Keywords, KNN, SVM, NB, BERT, RoBERTa, XLNet, GPT2), this study aims to compare and analyze the impact of different preprocessing techniques, dataset balance, and size on classification performance. To evaluate the performance of the classification models, a multi-metric assessment strategy will be implemented. The metrics employed include accuracy, precision, recall, F1-score, and confusion matrices.

Accuracy quantifies the percentage of accurately classified instances, while precision calculates the ratio of true positives to all predicted positives. Recall determines the proportion of true positives among all actual positives, and the F1-score represents the harmonic mean of precision and recall. Confusion matrices offer an in-depth insight into the classifications made by the models, facilitating a comprehensive analysis of their performance. By employing these metrics, a thorough evaluation of the classification models will be achievable, ultimately leading to an enhanced understanding of their respective strengths and weaknesses.

1.2 Aims and Objectives

The subsequent table is adapted from the original project description. For each task, additional criteria have been incorporated, which will serve as the basis for determining the completion of each respective task:

Aims and objectives	Criteria for success
Collect real reviews from app stores (Apple AppStore or Google Play) and extract their metadata. Select a few representative reviews from these reviews, manually analyze their content, and flag them as bug reports, feature requests, user experience, or ratings. Organize them into datasets.	Metadata collection should be efficient and accurate in order to correctly collect relevant information such as English reviews, application dependencies, etc. Manual categorization should be performed by reviewers trained in the taxonomy standards so that comments are correctly and consistently categorized.
Construct 3 different-sized, different-balanced datasets for this project, and perform preprocessing and data analysis on the contents of the datasets.	The dataset is clean and the data format is consistent. There are also datasets that are relevant to my project and have sufficient data volume. In addition, the data preprocessing is accurate, and each preprocessing is kept in the same dataset file.
Explore 8 models (Keywords, KNN, SVM, NB, BERT, RoBERTa, XLNet, GPT2) to classify user reviews and compare the performance of these models.	Run the program correctly and efficiently, and get reasonable results. The results were assessed using a variety of assessment methods (accuracy, precision, recall, F1-score, and confusion matrices).
A series of analyzes are performed on the results to compare the performance of these models and to investigate the impact of different preprocessing techniques, dataset balance and size on classification performance.	The analysis should be thorough, with detailed insights into the performance of different models and the impact of various factors on classification performance. It should be supported by statistical measures and visualizations (tables, histograms) that make the findings clear and understandable.

Table 1.1: Aims and Objectives

The key objectives of this project are as follows:

- **Data:** Three different types of datasets come from user reviews on Google play.
- **Software:** Use 8 models to classify user reviews
- **Report:** This report describes the research methodology, analyzes the results and highlights key findings.

1.3 Report Structure

This report contains seven chapters:

- **Chapter 1** provides an overview of the project and introduces the main topics covered in the report.
- **Chapter 2** offers a comprehensive analysis of the project's background, along with an introduction to related work in the field.
- **Chapter 3** outlines the methodology used throughout the project, providing a detailed description of each step in the development process.
- **Chapter 4** introduces the structure of the project, focusing on the details of each stage of project development.
- **Chapter 5** give the results of the model's performance and discuss the reasons for it in detail
- **Chapter 6** comparing the implementation of this project and related work, as well as reflections and limitations on the project
- **Chapter 7** conclude of the report are presented, providing insights into the possible future direction of the project.

Background and Related Work

This chapter aims to provide an overview of the development and significance of user review classification technology, outlining its evolution in history. Next, a related work on user review classification is presented, which explores existing research on this topic.

2.1 Background to User Review Classification

The history of user review classification technology dates back to the early days of the internet, when the rising popularity of online communities and e-commerce platforms made user reviews an essential part of consumer decision-making [4]. These reviews offer insights into product quality, customer satisfaction, and overall experiences, which has led to the development of techniques and algorithms to better understand, classify, and analyze them. Here is a brief overview of the evolution of user review classification technology:

Initially, computer technology was not highly developed, and early approaches to user review classification involved manual annotation and analysis of user-generated reviews. This labor-intensive and time-consuming process proved infeasible for large-scale applications.

In the late 1990s and early 2000s, the emergence of online platforms such as Amazon and eBay led to a proliferation of user reviews. During this period, rudimentary rule-based and keyword-based techniques were employed for classifying and analyzing reviews [5]. These methods relied on manually crafted rules and keyword lists to discern sentiment or topics.

By the mid-2000s, as user reviews became more prevalent and data volume increased, machine learning techniques like Naive Bayes, decision trees, and support vector machines (SVM) were adopted for user review classification tasks [6]. These algorithms offered enhanced accuracy and scalability compared to rule-based approaches; however, they still depended on manual feature engineering and often faced issues with overfitting and limited adaptability.

From the late 2000s to the early 2010s, advancements in natural language processing (NLP) techniques prompted the rise of sentiment analysis as a popular application for user review classification [7]. Techniques such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and other unsupervised methods were employed for topic modeling and sentiment analysis. During this period, feature extraction methods like bag-of-words and TF-IDF gained widespread adoption.

In the mid-2010s, the advent of deep learning facilitated the increased use of neural networks [8], such as convolutional neural networks (CNN) and recurrent neural networks (RNN), including long short-term memory (LSTM) and gated recurrent units (GRU), in NLP tasks like user review classification. These models automatically learned features from text data, eliminating the need for manual feature engineering and improving overall performance.

From the late 2010s to the early 2020s, pre-trained language models like BERT, GPT, ELMo, and RoBERTa revolutionized NLP and user review classification [9]. These models, trained on vast amounts of text data, could be fine-tuned for specific tasks, yielding considerable improvements in classification accuracy, sentiment analysis, and topic modeling. Attention mechanisms and transformers emerged as popular architectures for user review classification tasks.

In conclusion, the evolution of user review classification technology has progressed from simple rule-based techniques to advanced deep learning models. The field has experienced a shift from manual feature engineering to automatic feature learning, enabling more precise and efficient analysis of user reviews. The development and proliferation of pre-trained language models have further augmented the capabilities of user review classification technology, paving the way for new research avenues and applications.

2.2 Related Approaches to User Review Classification

2.2.1 Lexicon Based Method

Lexicon-based classification is a method for categorizing user comments into predefined groups using a set of rules. Various rule-based algorithms have been developed for sentiment analysis of user reviews.

Pang, Lee, and Vaithyanathan proposed a multi-level rule-based system for sentiment analysis of movie reviews, which extracted features based on the presence of subjective adjectives and adverbs [10]. Their approach involved constructing a lexicon of subjective terms using existing resources like WordNet and applying hand-crafted rules to identify sentiment-bearing phrases. This study demonstrated that rule-based classification could yield reasonable performance in sentiment analysis tasks, laying a foundation for future research.

Hu and Liu proposed a two-step rule-based classification approach for mining product features and associated sentiment from user reviews [11]. This approach involved identifying product features through noun phrase extraction and frequency-based filtering, followed by determining sentiment polarity using a sentiment lexicon and syntactic patterns. Their method was effective in extracting both explicit and implicit product features and determining the sentiment expressed towards these features, demonstrating the efficacy of rule-based classification in mining fine-grained sentiment information from user reviews.

However, rule-based classification systems have limitations, such as reliance on predefined rules that may not be universally applicable and difficulty in capturing linguistic nuances like sarcasm or irony. To address these issues, researchers have explored combining rule-based classification with other technologies.

Thakor investigated the integration of ontologies and rule-based classification for sentiment analysis in user reviews [12]. Their approach combined domain-specific ontologies with hand-crafted syntactic and semantic rules to identify sentiment-bearing phrases and their polarity. By incorporating domain knowledge through ontologies, the method was able to resolve ambiguity and improve classification performance. This study highlighted the potential of combining rule-based classification with domain-specific knowledge representation to enhance sentiment analysis in user reviews.

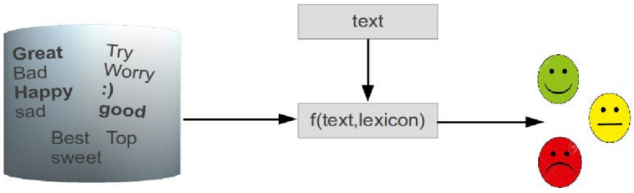

Figure 2.1: Lexicon Based Method

In conclusion, rule-based classification has demonstrated promise for user review classification, offering interpretability and the ability to incorporate domain knowledge. Although challenges such as handling ambiguity and scalability persist, ongoing research is addressing these issues and advancing the development of more sophisticated rule-based classification techniques.

2.2.2 Machine Learning Method

Machine learning-based classification has emerged as a prevalent approach for sentiment analysis of user reviews. Various machine learning algorithms, such as Naive Bayes, SVM, CNN, and RNN, have been developed for this purpose. Machine learning techniques applied to text classification can be broadly categorized into three types: supervised learning, unsupervised learning, and semi-supervised learning [13]. Supervised learning algorithms map input features (e.g., words or phrases in a document) to output labels (e.g., categories) using labeled training data, while unsupervised learning algorithms discover underlying structures or patterns within input data without relying on labeled data (e.g., clustering similar documents together). Semi-supervised learning algorithms combine elements of both supervised and unsupervised learning, utilizing a small amount of labeled data to guide the learning process and leveraging a larger amount of unlabeled data to enhance performance.

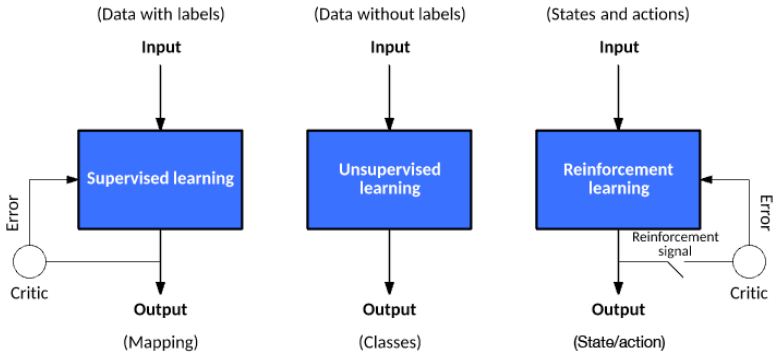

Figure 2.2: Models for machine learning

Naive Bayes, a probabilistic classifier based on Bayes' theorem with strong independence assumptions between features, has demonstrated surprisingly good performance in text classification tasks, including user review classification, despite its simplicity [14]. Its effectiveness in handling large feature spaces and computational efficiency make it suitable for large-scale applications.

SVM, a discriminative classifier, aims to find the optimal hyperplane maximizing the margin between different classes in the feature space. Widely used in text classification tasks, including user review classification, SVM consistently exhibits strong performance [14]. It effectively handles high-dimensional data and accommodates both linear and non-linear classification problems using various kernel functions.

Recently, neural networks, particularly deep learning models, have gained popularity due to their ability to learn complex patterns and representations from substantial amounts of data. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been applied to user review classification tasks with promising results[15]. Deep learning models can automatically learn features and representations from raw text data, eliminating the need for manual feature engineering. However, they typically require larger amounts of labeled data and computational resources compared to traditional machine learning algorithms.

Overall, the performance of machine learning algorithms in user review classification tasks depends on factors such as the size and quality of the training data, the complexity of the problem, and the choice of features and model parameters. While traditional machine learning algorithms like Naive Bayes and SVM have demonstrated strong performance in various applications, deep learning models show potential for further improving classification performance by automatically learning features and representations from raw text data.

2.2.3 Advanced language model

Pre-trained models are deep learning models that have been trained on extensive text corpora to learn general language representations. These models can be fine-tuned for specific tasks, such as user review classification, with relatively small amounts of labeled data.

Models like BERT (Bidirectional Encoder Representations from Transformers) are classified as Masked Language Models (MLMs) [16]. MLMs are trained to predict masked words within a context, enabling the model to learn bidirectional context representations. BERT's architecture comprises multiple layers of self-attention mechanisms, which facilitate learning complex relationships between words in a sentence.

Sun et al. proposed a fine-tuning approach for BERT in sentiment analysis tasks, including user review classification [17]. Their study demonstrated that BERT outperforms traditional machine learning methods and other deep learning models in classification accuracy. The authors also emphasized the significance of task-specific pre-processing and parameter tuning for optimal performance.

Models like GPT (Generative Pre-trained Transformer) fall under the autoregressive category [18]. Autoregressive models are trained to predict the next word in a sequence, given the context of previous words. GPT's architecture is based on the Transformer model, which employs self-attention mechanisms and parallelization for efficient training.

Xu et al. explored the application of GPT-2 for user review classification, specifically focusing on aspect-based sentiment analysis [19]. The authors fine-tuned GPT-2 on a dataset of user reviews, demonstrating its effectiveness in identifying aspects and sentiments. The study revealed that GPT-2 achieved superior performance compared to other deep learning models and traditional machine learning methods.

Models like RoBERTa (A Robustly Optimized BERT Pretraining Approach) combine elements of MLMs and autoregressive models [20]. RoBERTa is a BERT variant that improves upon its pre-training methodology by utilizing more extensive training data, longer training times, and dynamic masking. These modifications result in enhanced performance on various natural language processing tasks.

Zhang et al. examined the use of RoBERTa for user review classification in the e-commerce domain [20]. The authors employed RoBERTa in a multi-task learning setup, where the model was fine-tuned to classify reviews based on various attributes, such as sentiment, relevance, and helpfulness. The study demonstrated that RoBERTa outperformed other state-of-the-art models, showcasing its potential in user review classification tasks.

Pre-trained models like BERT, GPT-2, and RoBERTa have exhibited impressive performance in user review classification tasks, often surpassing traditional machine learning methods (e.g., SVM, Naive Bayes) and other deep learning models (e.g., CNN, RNN). These models effectively learn complex language representations and context-dependent features from large-scale text corpora [21]. Furthermore, they can efficiently transfer learning capabilities, requiring relatively small amounts of labeled data for fine-tuning specific tasks. This leads to improved performance in various natural language processing tasks, including sentiment analysis, aspect extraction, and multi-task classification. However, pre-trained models also possess limitations compared to traditional machine learning and rule-based approaches. Training and inference can be computationally resource-intensive, posing challenges for resource-constrained researchers and practitioners [22]. Moreover, adapting pre-trained models to specific domains or languages can present difficulties, particularly in terms of large-scale availability.

Research Methodology

My project has followed a systematical approach consisting of the following steps:

- **Step 1:** Data collection
- **Step 2:** Data Preprocessing
- **Step 3:** Lexicon-Based Model
- **Step 4:** Machine Learning Model
- **Step 5:** Advanced Language Model
- **Step 6:** Evaluation Metrics

These steps are detailed in the following sections.

3.1 Data Collection

The first step in starting a project requires a large-scale, high-quality dataset. Kaggle provided me with a raw dataset of the latest 200,000 user reviews of the top 20 apps in the Google Play Store, which are the most popular of the moment, such as Facebook, Twitter, WhatsApp, and Instagram, among others. The dataset is updated daily, ensuring that the user reviews provided are current and relevant for analysis purposes.

However, it should be noted that this dataset from Kaggle is not perfect, it contains a lot of non-English and non-representative comments. Filtering and cleaning of data are required for subsequent work. Furthermore, this dataset does not provide labels for classification tasks, which also requires manual annotation of labels.

3.1.1 Inter-Annotator Agreement

To create a good dataset, I selected 6000 representative reviews from 200000 reviews. Since the original dataset has no category labels, all reviews in the dataset were manually analyzed. Due to the size of the dataset, I sought additional help and hired an undergraduate computer science student to help me through the process. He was compensated for his time and energy. Each comment in the sample is assigned to two people. We read each review carefully, taking care to indicate its type. Specifically, we categorize each review as a bug report, feature request, user experience, or rating. The two of us discussed the definition and examples of each label before manual classification to ensure accurate classification. Through this rigorous process, we were able to create a solid data set that would serve as a solid foundation for the subsequent progress of the project.

Inter-Annotator Agreement (IAA) is a measure of the level of agreement or consensus between multiple annotators or coders who independently annotate or label the same set of data [23]. It is a way to assess the reliability and consistency of human annotation or labeling. The table below is a simple example of how the two of us do manual labeling:

Review	People1	People2	Agree?
I like this app, 5 stars.	Rating	Rating	Yes
Please add dark mode!	Feature request	Feature request	Yes
App not good, many bugs!	Rating	Bug report	No

Table 3.1: Example of Inter-Annotator Agreement

IAA is typically calculated using a statistical measure, such as Cohen's kappa coefficient, which quantifies the level of agreement between annotators beyond what would be expected by chance. A higher IAA score indicates a higher level of agreement among annotators, which indicates that the reliability of the data set is higher.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{3.1}$$

P(A) is the observed agreement, that is, the proportion of items in which the two raters agree. P(E) is the expected agreement by chance and can be calculated by summing the products of the two raters' proportions for

each category. The value of κ ranges from -1 to 1, where negative values indicate less agreement than expected by chance, values near zero indicate no more agreement than expected by chance, and values near 1 indicate almost perfect agreement [23]. The calculation of Cohen's Kappa coefficient involves a confusion matrix. I use the sklearn.metrics.cohen_kappa_score function in Python to calculate Cohen's kappa coefficient.

3.2 Data Preprocessing

In any NLP related project, the first thing that needs to be done after collecting a dataset is text preprocessing. This is a critical step in building great projects.

3.2.1 Tokenization

In text analytics and natural language processing, tokenization is a crucial pre-processing step [24]. It entails dividing a given text into tokens, which are smaller units. Depending on the level of granularity required for a particular activity, tokens are typically words, phrases, or sentences. Tokenization assists in transforming unstructured text data into a structured format, which facilitates processing and analysis.

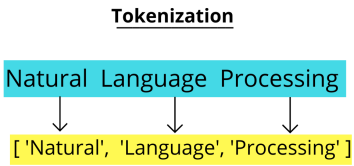

Figure 3.1: Example of Tokenization

3.2.2 Case Folding

Case folding is a text normalization technique used in natural language processing (NLP) and text analytics to convert all characters in a given text to a single case, usually lowercase [24]. However, case normalization can lead to a loss of valuable, important information. Take Turkey and turkey as an example. The first is the country. The second is a bird. Those are completely different meanings. This simple example shows us how much meaning or information is carried by capitalization alone. And how much caution we should apply when using normalization techniques.

3.2.3 Stop Words Removal

Stop words are words that occur frequently in a language and are considered to be of little value in NLP tasks such as text classification or sentiment analysis [25]. Examples of stop words include "the", "and", "or", "of", and "a". These words are often removed from text data during preprocessing to improve the accuracy and efficiency of NLP algorithms.

Figure 3.2: Wordcloud of stop words

These stop words are classified as words exceeding the upper limit in Luhn's hypothesis [26]. Stop words are considered to be very common and thus do not contribute significantly to the content of the article.

While stop word removal can help to improve the accuracy and efficiency of the model, it is important to be aware of cases where stop word removal may not be appropriate or may lead to loss of information. For example,

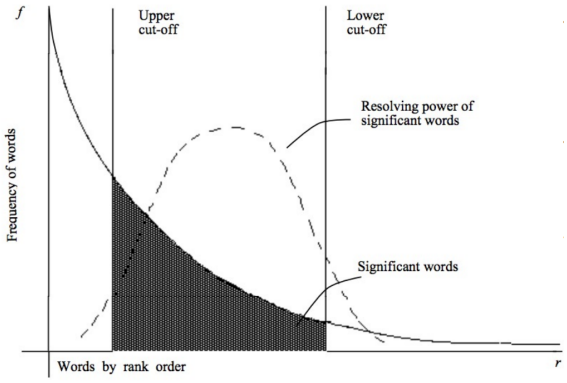

Figure 3.3: Luhn's Hypothesis

in some tasks, such as sentiment analysis or text classification, stop words may contain valuable information about the sentiment of the text, and their removal may lead to a loss of accuracy. (eg. must, can't, not, etc)

3.2.4 Stemming and Lemmatization

Natural Language Processing procedures like stemming and lemmatization are used to normalise text and get words and documents ready for machine learning processing. You may wish to admit, for instance, that the terms "like" and "liked" are the same word in different tenses when using NLP. The next step is to use either stemming or lemmatization to combine the two words into a single word root. Because of this, "like" and "liked" are regarded similarly, as opposed to how "like" and "car" are for the model.

Stemming is the process of reducing words to their root form, called the stem, by removing suffixes and prefixes [24]. This is done to reduce the number of unique words in a corpus and simplify the analysis of text data. For example, the stem of the word "running" is "run", and the stem of "happiness" is "happi". There are various algorithms for stemming, such as Porter stemming, Snowball stemming, and Lancaster stemming.

Lemmatization [24], on the other hand, is the process of converting words into their base or dictionary form, called the lemma. Unlike stemming, which simply removes affixes, lemmatization considers the context and morphological analysis of the word to return the root form. For example, the lemma of "running" is "run", and the lemma of "happiness" is "happy". Since lemmatization produces more accurate and meaningful results than stemming, it is often preferred in applications where precise meaning and context are important, such as sentiment analysis or machine translation.

Original	Stemming	Lemmatization
New	New	New
York	York	York
is	is	be
the	the	the
most	most	most
densely	dens	densely
populated	popul	populated
city	citi	city
in	in	in
the	the	the
United	Unite	United
States	State	States

Figure 3.4: Example of stemming and lemmatization

3.2.5 Data Augmentation

In this project, obtaining large labeled datasets presented significant challenges due to the time-consuming and expensive nature of the process. As a result, the dataset collected comprised only 4,000 user reviews.

Data augmentation proves particularly beneficial in text classification tasks, as it helps mitigate the limitations associated with small quantities of labeled data for training purposes.

Data augmentation [27] using synonyms entails substituting select words in a sentence with their synonyms while preserving the sentence's meaning. This technique is employed in natural language processing (NLP) to enhance the diversity of datasets and bolster the performance of machine learning models. Synonym replacement can be executed using a predefined thesaurus or through the application of word embeddings and machine learning algorithms.

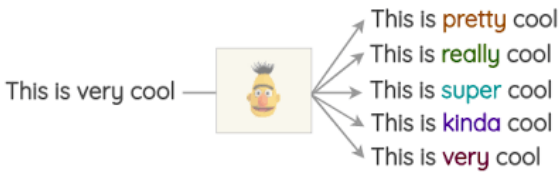

Figure 3.5: Example of Data Augmentation

A more diverse training dataset yields a more robust and accurate model. Data augmentation techniques assist in addressing the issue of overfitting, a phenomenon where a model demonstrates strong performance on training data but struggles with new, unseen data. By augmenting the training data with variations of the original data, the model encounters a broader range of examples and is less likely to overfit. Furthermore, data augmentation can be utilized to balance class distribution in imbalanced datasets and to minimize bias in the training data. Overall, data augmentation serves as a crucial tool for enhancing the performance and generalization capabilities of machine learning models.

3.3 Lexicon-Based Model

3.3.1 Keywords Matching

Automatic categorization of user reviews can be achieved through a basic technique, which involves searching for specific keywords within the text. This approach requires the manual creation and maintenance of a list of relevant keywords that are expected to be present in different types of user reviews, such as bug reports, feature requests, user experiences, or ratings. The text is then scanned to determine whether it contains any of these predefined keywords [7]. This technique is a straightforward means of automatically categorizing user reviews. A list of potential keywords for each sort of evaluation is provided in Table 3.2.

Label	**Keywords**
Bug reports	slow, defect, crash, solve, bug, fix, problem, issue
Feature requests	improve, miss, need, suggest, want, prefer, request, should, wish, add, please, could, hope,
User Experiences	help, support, assist, when, experience, situation
Ratings	cool, love, hate, great, good, nice, very, bad, worst

Table 3.2: Keywords Dictionary

Keyword matching is a straightforward and easy-to-implement method for text classification, but it is constrained by several limitations and drawbacks, ultimately limiting its applicability. For instance, keyword matching models are incapable of handling synonyms and near-synonyms effectively, which can lead to classification errors. Moreover, the performance of such models is heavily reliant on the quality of the keyword list, and incomplete or inaccurate lists can adversely affect the performance of the classifier. Additionally, keyword matching models do not account for contextual information, thereby overlooking the influence of different contexts on the meaning and semantics of the same keyword. As a result, to enhance the accuracy and generalization capability of text classifiers, more sophisticated algorithms and models.

3.4 Machine Learning Model

3.4.1 Feature Extraction

Feature extraction is a very important step in machine learning [28]. Its purpose is to extract the most useful features from the original data in order to train a model that can accurately predict the target variable. Let me introduce two feature extraction techniques of the machine learning model in this project: **Bag-of-words** and **Term Frequency-Inverse Document Frequency**.

Bag-of-words

This approach represents text as an unordered collection of words, disregarding grammar and word order, but maintaining the frequency of each word. Count the occurrence frequency of each word in all texts to obtain a word frequency table. According to the word frequency table, each text is represented as a vector, and the dimension of the vector is the total number of words. Each dimension corresponds to a word, and the value in the vector indicates how often that word occurs in the text.

	she	loves	pizza	is	delicious	a	good	person	people	are	the	best
She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0
She is a good person	1	0	0	1	0	1	1	1	0	0	0	0
good people are the best	0	0	0	0	0	0	1	0	1	1	1	1

Figure 3.6: Example of BOW

The BOW model has the advantage of being simple to use and suitable for most natural language processing tasks [28]. It can convert text into numerical features for easy training and prediction by machine learning models. However, the BOW model also has some disadvantages, such as its inability to take into account the semantic and contextual information of words, so it may not perform well in some tasks that require semantic and contextual information to be taken into account.

Term Frequency-Inverse Document Frequency

This is also a commonly used feature extraction method in the field of machine learning theory. It is used to measure the importance of words in a collection of documents. TF-IDF consists of two parts: Term Frequency (TF) and Inverse Document Frequency (IDF) [28].

Term frequency represents the number of times a word occurs in a document. It reflects the importance of words in the document. In general, the more times a word occurs, the more important it is in the document. The word frequency calculation formula is:

$$\mathrm{tf}(t,d) = \frac{\text{number of times term } t \text{ appears in document } d}{\text{total number of terms in document } d} \tag{3.2}$$

Inverse document frequency represents the prevalence of a word in a collection of documents. If a word appears in many documents, it is probably a relatively common word that is less useful for differentiating documents. The formula for calculating inverse document frequency is:

$$\mathrm{idf}(t,D) = \log \frac{\text{total number of documents in the corpus } N}{\text{number of documents containing term } t \text{ in the corpus } D} \tag{3.3}$$

A higher TF-IDF value indicates that a word is more important in a particular document and less important in other documents. But there are also the same shortcomings as BOW, such as the inability to consider the semantics and context information of words, and the inability to handle some complex natural language processing tasks.

Figure 3.7: Example of Term Frequency-Inverse Document Frequency

3.4.2 K-Nearest Neighbors

KNN (K-Nearest Neighbors) is a supervised machine learning algorithm that can be used for classification and regression tasks [29]. In classification problems, KNN predicts the class of a new sample by analyzing the class of its K nearest neighbors in the feature space.

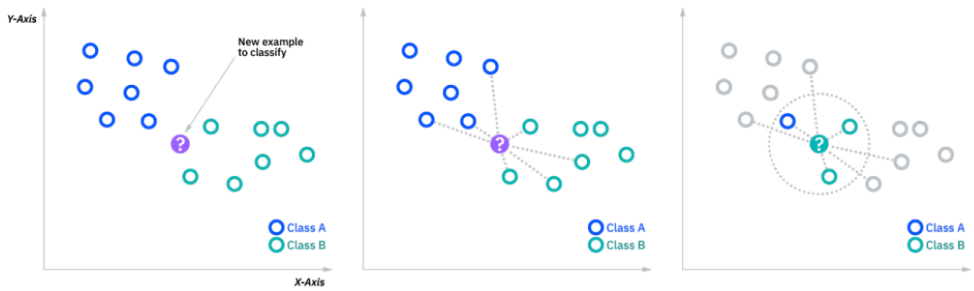

Figure 3.8: KNN(K=3)

Given a training data set and a new input instance, the K-Nearest Neighbour algorithm locates the K instances that are closest to the instance in the training data set. Since the majority of these K instances belong to a particular class, the input instance is classified into this class (similar to the idea that the minority obeys the majority in real life).

In the KNN Algorithm, given a training data set, will convert user reviews into numeric vectors. Commonly used methods are Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding (such as Word2Vec or GloVe).

There are a number of measurement techniques, including the conventional Manhattan distance, Euclidean distance, and others, to determine the distance between two locations in space.

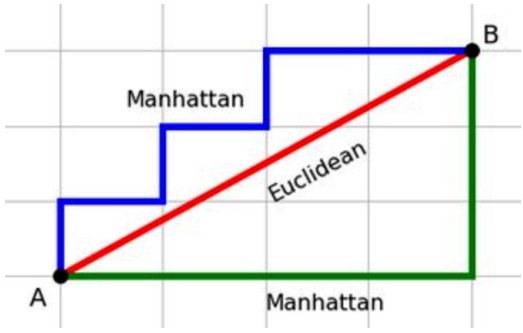

Figure 3.9: Calculate Distance

In n-dimensional multidimensional space, the Manhattan distance between two points can be calculated as:

$$d_1(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i| \tag{3.4}$$

In an n-dimensional multidimensional space, the Euclidean distance between two points can be calculated as:

$$d_2(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{3.5}$$

For each text to be classified, find out the K nearest neighbors in the training set, and then vote according to the category of these neighbors. Assign the category with the most votes to the text to be classified. In the KNN algorithm, weights can be assigned to each neighbor so that neighbors with closer distances have a greater influence on the classification results.

The KNN algorithm has good adaptability because it performs classification or regression according to the distribution of actual data points [29]. And it makes predictions directly based on the similarity of data points, so it adapts well to non-linear data. However, the disadvantage of the KNN algorithm cannot be ignored. Its computational complexity is high because the distance between the test data point and all the training data points needs to be calculated in the prediction stage. Also sensitive to noisy data and outliers; need to choose an appropriate K value and distance metric.

3.4.3 Naive Bayes

Naive Bayes is a simple probabilistic classifier based on Bayes' theorem [30], suitable for text classification tasks. Its basic principle is to calculate the probability of a given text appearing under each category, and then select the category with the highest probability as the classification of the text.

Like other machine learning models, the Naive Bayes model has to go through a series of pre-processing, feature extraction and other prerequisites.

To derive the Naive Bayes formula for text classification, we first start with Bayes' theorem, which provides a way to calculate conditional probabilities. Bayes' theorem is defined as:

$$P(C|D) = \frac{P(D|C) \times P(C)}{P(D)} \tag{3.6}$$

- C is the class/category
- D is the document/text
- $P(C|D)$ is the probability of class C given document D (posterior probability)
- $P(D|C)$ is the probability of document D given class C (likelihood)
- $P(C)$ is the probability of class C (prior probability)
- $P(D)$ is the probability of document D (evidence)

Assume that document D consists of words w1, w2, ..., wn. Based on the Naive Bayes assumption of conditional independence between features (words), we can rewrite the likelihood P(D—C) as the product of individual word probabilities given the class:

$$P(D|C) = P(w_1, w_2, \ldots, w_n|C) \approx P(w_1|C) \times P(w_2|C) \times \ldots \times P(w_n|C) \tag{3.7}$$

We can now substitute this likelihood back into the Bayes' theorem formula:

$$P(C|D) \approx \frac{P(w_1|C) \times P(w_2|C) \times \ldots \times P(w_n|C) \times P(C)}{P(D)} \tag{3.8}$$

Since the evidence P(D) remains the same for all classes, we can ignore it during the classification process. Therefore, we can rewrite the formula as:

$$P(C|D) \propto P(w_1|C) \times P(w_2|C) \times \ldots \times P(w_n|C) \times P(C) \tag{3.9}$$

To prevent numerical underflow, which can occur when multiplying many small probabilities, we typically use the log of the probabilities instead. Thus, the formula becomes:

$$\log(P(C|D)) \propto \log(P(w_1|C)) + \log(P(w_2|C)) + \ldots + \log(P(w_n|C)) + \log(P(C)) \tag{3.10}$$

Sometimes, a word that has not appeared in the training data set may appear in our test data set. At this time, the calculation result of Naive Bayes will be 0. If we multiply 0 by the formula, then we can't expect to get the normal result. Therefore, we also need to slightly improve the formula for calculating a certain word in a certain category. Using Laplace smoothing, you can avoid the case where the probability is 0. Don't be intimidated by the name, Laplace smoothing is a very simple smoothing method: add 1 to the numerator and add the entire value range to the denominator, so that you can add a very small offset to the entire fraction, Avoid zero.

$$P(w_i|C) = \frac{count(w_i, C) + 1}{\sum_{w \in V}(count(w, C) + 1)} \tag{3.11}$$

The Naive Bayes formulation for text classification is now ready to use. Given a new review, the derived formula is used to calculate the posterior probabilities of all classes, and the class with the highest posterior probability is assigned to the review.

Although the Naive Bayesian algorithm has the advantages of simplicity and easy implementation, fast training speed, good performance on many problems, and robustness to missing data. But it also has some disadvantages. First of all, Naive Bayes assumes that the features are independent of each other, but in practical problems, this independence assumption may not hold true, resulting in affected performance. Second, Naive Bayes has strict assumptions about the distribution of input data, which may lead to performance degradation when the actual data distribution does not match the assumed distribution. In addition, due to the reliance on estimates of prior probabilities and class probabilities, it may lead to inaccurate predictions when the amount of training data is small or the classes are imbalanced [30].

3.4.4 Support Vector Machines

Support Vector Machine [31] (SVM) is the third machine learning method of this project, which can also be applied to user review classification tasks. The SVM model also undergoes a series of prerequisites such as preprocessing and feature extraction. The following is an explanation of the basic principles and key concepts of SVM.

For a linearly separable problem, SVM seeks to find a maximum margin classifier, i.e., maximizing the minimum distance between data points of the two classes and the decision boundary. This distance is called the margin. The decision boundary can be represented as:

$$w \cdot x + b = 0 \tag{3.12}$$

where w is the normal vector (a vector perpendicular to the hyperplane), x is the data point, and b is the bias term. Based on the principle of margin maximization, we can formulate the optimization problem as follows:

$$\text{maximize}(M) \tag{3.13}$$
$$\text{subject to: } \|w\| = 1 \tag{3.14}$$
$$y_i(w \cdot x_i + b) \geq M, i = 1, \ldots, N \tag{3.15}$$

The above optimization problem requires maximizing the margin M while satisfying the classification constraints for all data points. To solve this problem, we can transform it into an equivalent dual problem, i.e., minimizing the Lagrangian function with respect to w and b:

$$\text{minimize } L(w, b) = \frac{1}{2}\|w\|^2 \tag{3.16}$$
$$\text{subject to: } y_i(w \cdot x_i + b) \geq 1, i = 1, \ldots, N \tag{3.17}$$

By solving the above quadratic programming problem, we can obtain the optimal w and b, and thus determine the decision boundary. In this process, the data points closest to the decision boundary (satisfying the equation $y_i(w \cdot x_i + b) = 1$) are called support vectors. Support vectors determine the position of the decision boundary, providing SVM with good generalization ability.

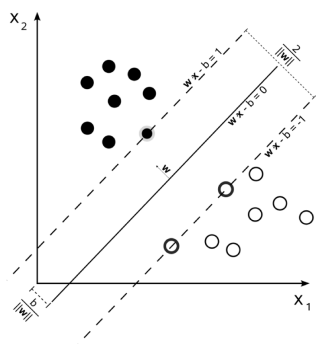

Figure 3.10: SVM

In Support Vector Machines (SVM), the maximum margin classifier refers to finding a decision boundary (hyperplane) in linearly separable problems that maximizes the minimum distance between the two classes of data points. The goal of this classifier is to find a decision boundary with the largest margin, thereby improving the generalization ability of the model [31].

However, in real-world problems, data is often not linearly separable, which requires the introduction of the concept of soft margin. A Soft margin allows some data points to be on the wrong side of the decision boundary, meaning that the classifier is allowed to make mistakes to some extent. To achieve this, we introduce a slack variable ξ and restate the optimization problem as:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{3.18}$$
$$\text{subject to: } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \ldots, N \tag{3.19}$$

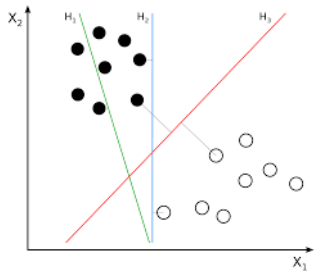

Figure 3.11: Margin Maximisation

Here, C is a regularization parameter used to balance the trade-off between maximizing the margin and classification errors. By adjusting the value of C, we can control the complexity of the model and the risk of overfitting.

For non-linearly separable problems, SVMs employ a kernel function to map the original data to a higher-dimensional space, rendering the data linearly separable in the new space. Common kernel functions include Linear Kernel, Polynomial Kernel, and Radial Basis Function Kernel (RBF). The choice of kernel functions and parameter tuning significantly influence the performance of SVM models.

Our project involves the multi-classification of user reviews. However, SVM is inherently a binary classifier. We can extend SVM to multi-classification problems through One-vs-One or One-vs-All strategies. The One-vs-One strategy entails training an SVM classifier for each pair of categories, ultimately determining the category of the data point through a voting mechanism. Conversely, the One-vs-All strategy involves training an SVM classifier for each category to distinguish it from all other categories, eventually selecting the classifier with the highest confidence.

The advantage of the SVM model lies in its ability to generalize well while finding the maximum margin [31]. Furthermore, SVMs can handle non-linear data through kernel tricks, rendering them suitable for various types of problems. Nonetheless, the disadvantages of SVMs include slower processing of large datasets and sensitivity to parameter tuning and feature scaling, which can result in fluctuations in model performance.

3.5 Advanced Language Model

3.5.1 BERT Model

Bidirectional Encoder Representation for Transformer (BERT) is an NLP model developed by Google Research in 2018 [17]. Figure 3.12 demonstrates that the Transformer architecture, also known as encoder-decoder architecture, comprises both an encoder stack and a decoder stack, whereas BERT is only an encoder stack of the Transformer architecture. The architecture complexity of the two variations, BERT-base and BERT-large, varies. The standard model's encoder contains 12 layers, while the Large encoder has 24 layers.

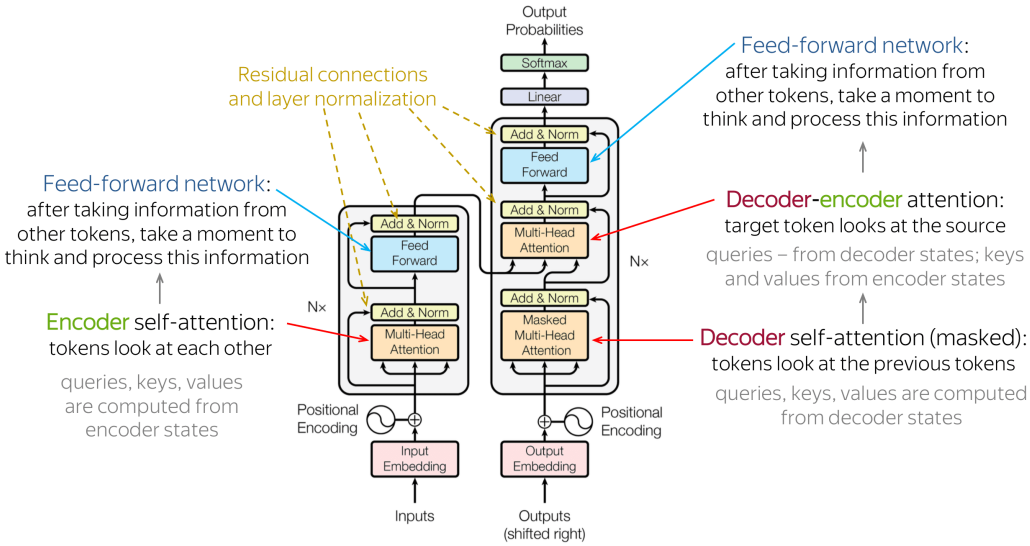

Figure 3.12: Transformer Architecture

BERT was developed using a huge text corpus for training, which allows the architecture or model to learn the heterogeneity in data patterns and perform effectively across a variety of NLP applications. Being bidirectional, BERT gathers knowledge from both the left and the right sides of the context of a token during training.

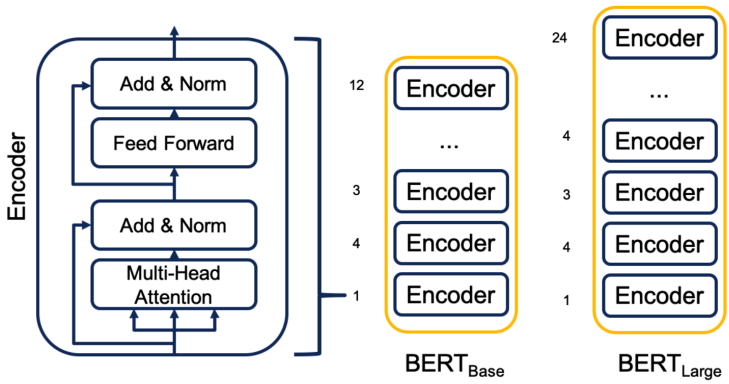

Figure 3.13: BERT Model

The overall BERT framework consists of two phases: pre-train and fine-tune. pre-train models are trained on unlabelled labeled data, fine-tune models are first initialised with pre-train model parameters, and then all parameters are trained with downstream labeled data.

The input to the encoder for BERT is a sequence of tokens, which are first converted into vectors and then processed in the neural network. For starters, every input embedding is a combination of 3 embeddings [17]:

- **Token Embeddings** are word vectors, the first word is the [CLS] flag and can be used for subsequent classification tasks.
- **Segment Embeddings** are utilised to differentiate between two sentences, pre-training is not just for LM but also for the classification task with two sentences as input

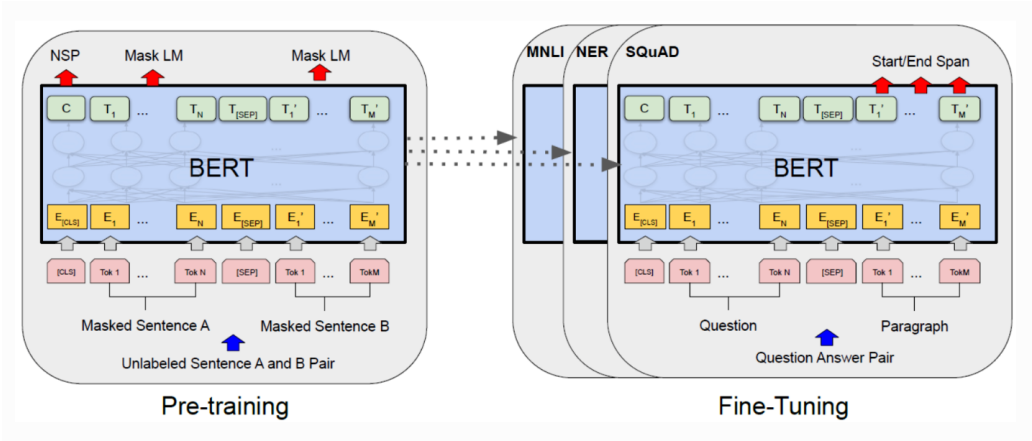

Figure 3.14: BERT Structure

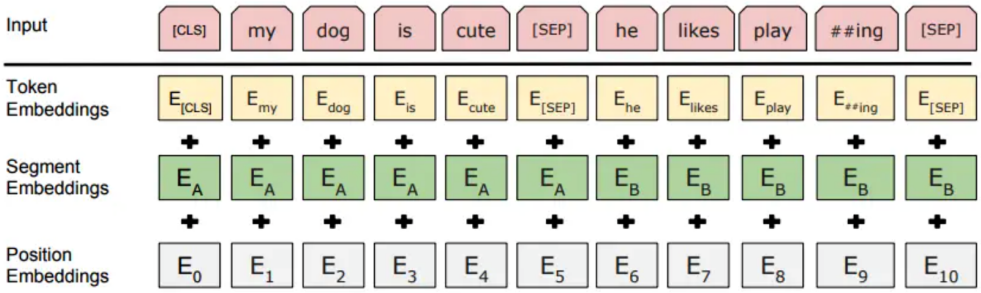

Figure 3.15: BERT embedding

- **Position Embeddings** differs from the Transformer from the previous article in that it is a learned function rather than a trigonometric one.

BERT is pre-trained on two NLP tasks

Masked Language Modelling

Masked Language Modeling (MLM) is a pre-training task in which a certain proportion of words in the input corpus are masked, and the model is trained to predict the masked words based on the surrounding context [17]. This concept is analogous to the word completion tasks commonly encountered in secondary education. The architecture of the Transformer makes it well-suited for MLM tasks, as it can effectively learn contextual relationships similar to traditional language modeling algorithms and Recurrent Neural Networks (RNNs). In BERT's experiments, 15% of the WordPiece tokens are randomly masked. During the model training process, a sentence is fed into the model multiple times for parameter learning. However, instead of consistently masking the same words, Google devised a method that, after identifying the words to be masked, applies different masking strategies to diversify the training input.

- 80% of the time, the identified word is directly replaced with the [MASK] token. For example, the sentence "my dog is cute" would be transformed into "my dog is [MASK]."
- 10% of the time, the identified word is replaced with a random word. In this case, the word "cute" could be replaced with a random word such as "apple," resulting in the sentence "my dog is apple."
- 10% of the time, the original token is retained, leaving the sentence unchanged. For instance, the sentence "my dog is cute" would remain as "my dog is cute."

Next Sentence Prediction

Next Sentence Prediction (NSP) is a fundamental task in natural language processing, aimed at training a model to predict whether a given sentence B is the actual next sentence following sentence A in a text corpus [17]. This task is accomplished through the pre-training of models such as BERT, which are fed pairs of sentences A and B, with the aim of capturing inter-sentence coherence and context. During pre-training, 50% of the time,

sentence B is the true next sentence, while the other 50% of the time, a randomly selected sentence from the corpus is used as sentence B.

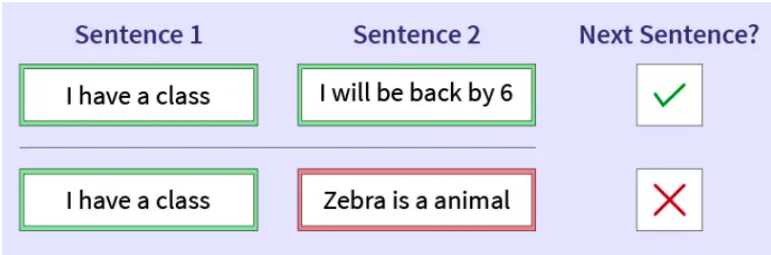

Figure 3.16: Next Sentence Prediction

The NSP task serves to enable the model to learn the relationship between sentences and improve its understanding of the context in which they are used [32]. By combining this objective with the masked language modeling (MLM) objective, BERT can learn bidirectional context representations, which have proven effective in enhancing its performance on various downstream tasks such as question answering and sentiment analysis. The integration of NSP into the pre-training process has been shown to be particularly effective in improving BERT's ability to comprehend and generate coherent sentences, thereby making it an essential component of state-of-the-art language models.

3.5.2 RoBERTa Model

In terms of model structure, the RoBERTa model has no innovation to a large extent compared with the BERT model, and it is more like a further exploration of BERT in pre-training. These small improvements allow RoBERTa to outperform BERT on certain natural language processing tasks. Next, I will explain the improvement of RoBERTa on the BERT tutorial [20].

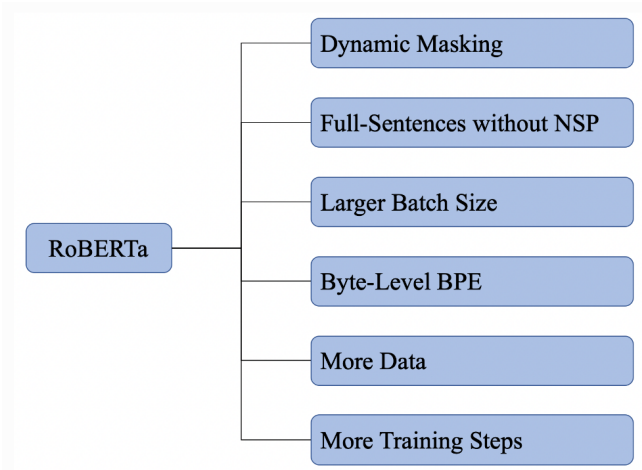

Figure 3.17: Improvements of RoBERT

Dynamic Masking

The masking process in BERT is static, meaning that the same tokens are masked throughout the entire pre-training process. In RoBERTa, dynamic masking is introduced, which changes the masking pattern for each training instance in every epoch [20]. In other words, the tokens to be masked are chosen randomly for each instance and each epoch, rather than being fixed at the beginning of the pre-training process. This dynamic masking approach encourages the model to learn more diverse and robust representations of the input data, as it exposes the model to a wider range of contexts and token combinations.

Full-Sentences without NSP

In the original BERT model, the Next Sentence Prediction (NSP) task was used as a pre-training objective to help the model learn the relationship between sentences. However, subsequent research found that the NSP task did not significantly contribute to the downstream performance as initially believed. RoBERTa, a variant

of BERT, omits the NSP task from its pre-training process, focusing solely on the Masked Language Modeling (MLM) objective [20]. Nonetheless, RoBERTa is still capable of learning inter-sentence relationships due to the way it processes input text during pre-training.

The "Full Sentences" approach in RoBERTa involves continuously extracting sentences from one or multiple articles to fill the model input sequence. In this method, an input sequence may span across multiple article boundaries. To be more specific, the model extracts sentences from an article until it reaches the end, and if the input sequence still has room for more text, it proceeds to extract sentences from the next article. The content from different articles is separated using the SEP delimiter.

The key to achieving this inter-sentence learning lies in RoBERTa's pre-training methodology. By using a continuous stream of text without considering sentence boundaries, RoBERTa is able to process multiple concatenated sentences together. This approach allows the model to understand relationships between sentences, making it more effective for various natural language processing tasks.

Larger Batch Size

In the context of deep learning, batch size refers to the number of training samples used in a single update to the model's weights. Larger batch sizes have several benefits, which contribute to the improved performance of RoBERTa compared to the original BERT model [20].

RoBERTa observes the performance of the model on pre-training tasks and down-stream tasks by increasing the Batch Size during training. It is found that increasing the Batch Size is beneficial to reduce the Perplexity of the retained training data and improve the down-stream index.

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Figure 3.18: RoBERTa's batch size experiment

Byte-Level BPE

Byte-Pair encoding (BPE) is a way of representing words and generating word lists. "word", especially for longer words. For example, the word wonderful may be split into two sub-words 'wonder' and 'ful'. Unlike BERT, RoBERTa uses a Byte-based BPE, with a total of around 50K words in the word list, this way there is no need to worry about unregistered words, as it will break the word down at the Byte level.

More Data and More Training Steps

RoBERTa uses more training data than BERT, and Figure 1 shows the experimental effect of RoBERTa as the training data increases and the number of training steps increases, and it is clear that the model's performance in the down-stream improves as both increase [20].

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Figure 3.19: RoBERTa increase data and training steps experimental renderings

3.5.3 XLNet Model

eXtreme Language understanding NETwork(XLNet) is also a pre-trained language model based on the Transformer architecture [33]. But it is an improved version of Transformer, called Transformer-XL. Its main

innovation lies in the use of permutation language modeling and two stream self attention technology, which can capture richer contextual information and achieve excellent performance in natural language processing tasks.

Transformer-XL

Transformer-XL is an advanced model based on the original Transformer architecture, designed to address the challenges associated with handling long sequences and long-distance dependencies [34]. While the Transformer-XL shares many structural similarities with the standard Transformer, it introduces innovations such as relative position encoding and a segmented recurrence mechanism, resulting in improved performance compared to its predecessor.

Relative position encoding allows the model to learn more adaptable position information, enabling it to capture long-distance dependencies more effectively. The segmented recurrent mechanism, on the other hand, facilitates the capture of extended contextual information by dividing long sequences into multiple subsequences. This mechanism preserves the hidden state from the previous subsequence while processing each subsequent subsequence, thereby enhancing the model's ability to understand the context.

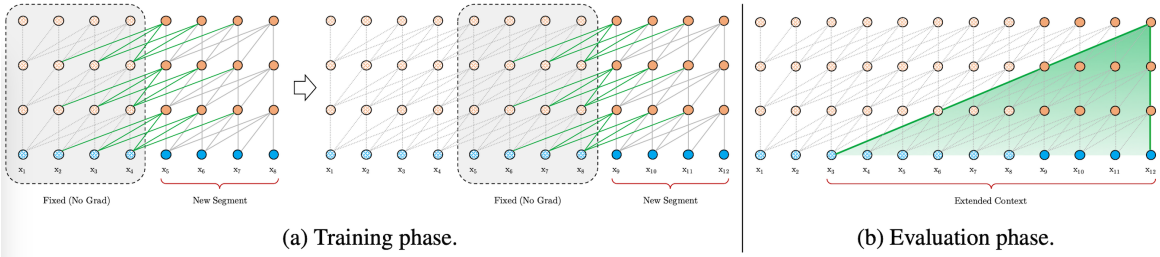

Figure 3.20: An example of the Transformer-XL model with four segments.

Two Stream Self Attention

Two-Stream Self-Attention is a self-attention mechanism especially suitable for the XLNet model [33]. It is used to support Permutation Language Modeling and to avoid information leakage when predicting the current word. Two-stream self-attention includes content vectors and query vectors, which are used to store the contextual information of words and calculate self-attention scores, respectively. By calculating the dot product of the query vector and the content vector, the attention score is obtained, and after being normalized by the Softmax activation function, it is multiplied by the corresponding content vector to obtain the weighted content vector. Finally, all weighted content vectors are summed to obtain the contextual representation of the current word. The two-stream self-attention mechanism allows the model to utilize the information of other words when predicting the current word while avoiding information leakage, thus better adapting to full-permutation language modeling.

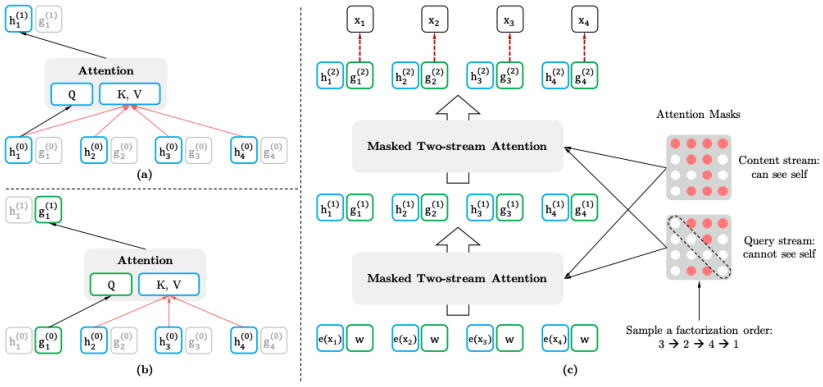

Figure 3.21: Two Stream Self Attention

Permutation Language Modeling

Unlike BERT's Masked Language Model, XLNet uses full permutation language modeling(PLM). In this approach, the model learns for each word of the input sequence a conditional probability distribution, the probability of that word given other words. By using all possible permutations to learn the input sequence,

XLNet is able to capture bidirectional contextual information while avoiding the problem of dynamic masks in BERT. The following are the main ideas of full permutation language modeling [33]:

1. Full Permutation Language Modeling (PLM) involves generating all possible permutations of a given input sequence. For instance, when provided with the input sequence "[A, B, C]", full permutation language modeling generates six permutations: "[A, B, C]", "[A, C, B]", "[B, A, C]", "[B, C, A]", "[C, A, B]", and "[C, B, A]".

2. Central to PLM is the process of learning the probability distribution of each word given the presence of other words. For example, in the permutation "[A, B, C]", the model needs to learn P(A—B, C), P(B—A, C), and P(C—A, B). By modeling all possible permutations, the model can effectively capture bidirectional contextual information.

3. PLM allows models to grasp bidirectional contextual information by learning the probability distribution of each word, considering other words in the sequence. This approach differs from the Masked Language Modeling (MLM) utilized in BERT, which captures contextual information by dynamically masking a portion of the words within the input sequence.

4. As full permutation language modeling does not rely on dynamic masking, it avoids the issues encountered in BERT. During BERT's pre-training phase, masking words can lead to discrepancies between input distributions in the pre-training and fine-tuning phases. PLM addresses this problem by eliminating the need for masked input sequences during the pre-training phase, ultimately resulting in a more consistent and accurate representation of contextual information.

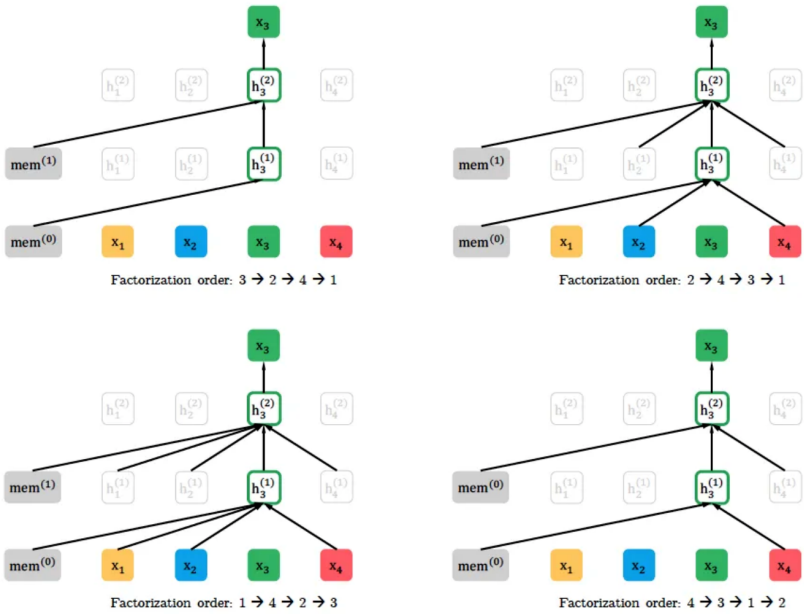

Figure 3.22: Schematic diagram of calculating the output of the third word in different permutations

3.5.4 GPT2

The Generative Pretrained Transformer 2 (GPT-2) model is also based on the Transformer architecture [35]. GPT-2 and GPT share a similar model structure; however, GPT-2 incorporates several enhancements. The primary distinction between the two is the increased model size of GPT-2, which features more layers, hidden units, and attention heads, enabling it to learn more intricate and diverse patterns. Furthermore, GPT-2 employs a larger training dataset that encompasses multiple languages and domains, in addition to incorporating technical improvements in both training and generation strategies. These advancements contribute to the superior performance of GPT-2 in comparison to its predecessor, GPT, particularly in handling complex natural language processing tasks.

GPT-2 also uses a two-stage training method. First, the model performs unsupervised pre-training on a large amount of text data and learns a wealth of language knowledge. Then, for specific tasks, the model is adapted to the requirements of specific tasks through a supervised fine-tuning process.

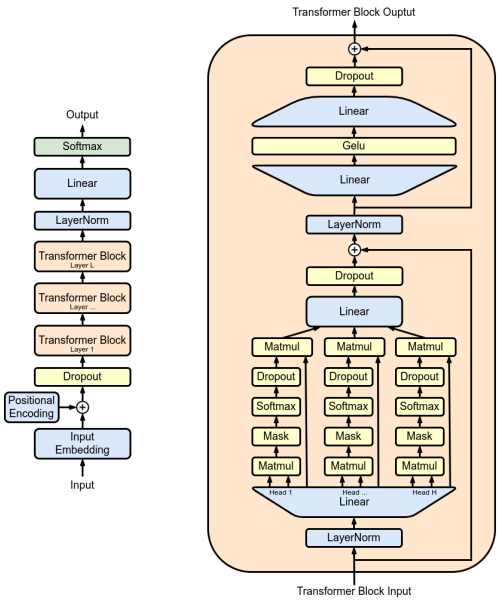

Figure 3.23: GPT2 architecture

Unsupervised pre-training

Unsupervised pretraining (Pretraining), specifically, given an unlabeled corpus $U = \{u_1, u_2, \ldots, u_n\}$, we train a language model and perform maximum (logarithmic) likelihood on the parameters estimate:

$$L_1(U) = \sum_i \log P\left(u_i \mid u_1, \ldots, u_{k-1}; \Theta\right) \tag{3.20}$$

where k is the size of the context window, B is the conditional probability,Θ is the parameter of the conditional probability, and the parameter is updated by stochastic gradient descent.

The training process is also very simple, which is to take the word embedding of n words W_e plus the positional embedding W_p which is then fed into the Transformer and the n outputs are each predicted to be the next word in that position It can be seen that the GPT is a one-way model, with the inputs to the GPT denoted by h_0 where 0 represents the input layer and h_0 is calculated as follows:

$$h_0 = UW_e + W_p \tag{3.21}$$

W_e is the Embedding matrix of the token, and W_p is the Embedding matrix of the position code. Use voc to represent the vocabulary size, pos to represent the longest sentence length, and dim to represent the Embedding dimension, then W_p is a matrix of pos$\times$dim, and W_e is a matrix of voc$\times$dim. In GPT, the author randomly initializes the position embedding matrix and lets the model learn by itself instead of using sine and cosine functions for calculation.

After getting the input h_0, you need to pass h_0 into the Transformer Decoder of the GPT2 in order to get h_n. n is the number of layers of the neural network. Finally, get h_n and then predict the probability of the next word.

$$h_l = transformer_{block}\left(h_{l-1}\right), \forall l \in [1, n] \tag{3.22}$$

Supervised Fine-Tuning

After pre-training, we also need to perform Fine-Tuning for specific tasks. Suppose the input C of the supervised data set X is a sequence $x^1, x^2, \ldots, x^m$, and the output is a label of a category y, such as a text classification task.

We input $x^1, \ldots, x^m$ into the Transformer model to get the output h_l^mat the last moment of the top layer, and classify it through our newly added Softmax layer (parameter is W_y), and finally use cross entropy to calculate the loss, so that according to the standard Data adjustment Transformer parameters and Softmax parameters W_y. This is equivalent to maximum likelihood estimation:

$$P\left(y \mid x^1, \ldots, x^m\right) = \text{softmax}\left(h_l^m W_y\right) \tag{3.23}$$

W_y represents the parameters when predicting the output, and the following functions need to be maximized during fine-tuning:

$$L_2(C) = \sum_{x,y} \log P\left(y \mid x^1, \ldots, x^m\right) \tag{3.24}$$

Normally, we should adjust the parameters so that L_2 is the largest, but in order to improve the training degree and the generalization ability of the model, we use Multi-Task Learning. GPT also considers the pre-trained loss function when fine-tuning, and at the same time makes it the maximum likelihood L_1 and L_2.

$$L_3(C) = L_2(C) + \lambda \times L_1(C) \tag{3.25}$$

The L_1 used here is still the loss (likelihood) from the previous language model, but instead of the previous unsupervised data U, the data used is the current task's data C, and using only X of it, without the labels y.

3.6 Evaluation Metrics

3.6.1 Confusion Matrix

The confusion matrix is a widely used tool for evaluating the performance of a classifier in a classification task [36]. It provides a table that displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for each class in the classification task. In the context of user review classification, a confusion matrix would be used to show how many reviews were correctly or incorrectly classified as positive or negative. Specifically, a confusion matrix for a binary user review classification task would typically include four cells: true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Figure 3.24: Binary Confusion Matrix

In addition to binary classification tasks, confusion matrices are also commonly used in multi-class classification tasks. In these cases, the confusion matrix would typically be a square matrix that displays the number of true positives, true negatives, false positives, and false negatives for each class in the classification task. As the current project involves multi-class classification of user reviews, the use of a confusion matrix is particularly relevant. By displaying the performance of the classification model for each class, the matrix provides a detailed overview of the strengths and weaknesses of the model.

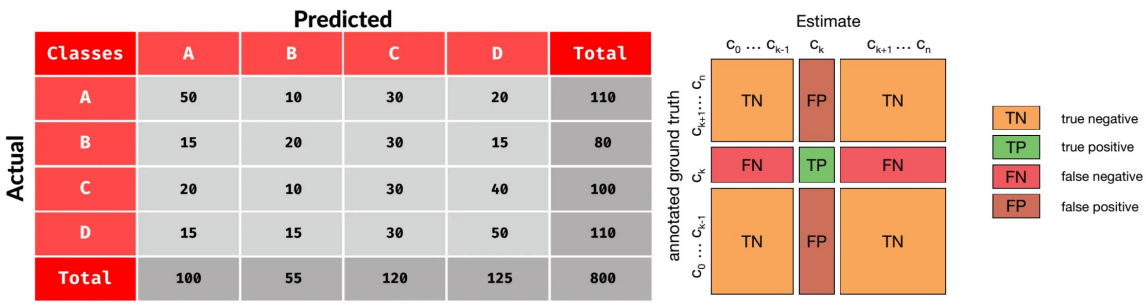

Figure 3.25: Multiclass Confusion Matrix

The confusion matrix is very useful in evaluating the performance of the classification model and diagnosing the problems of the classification model. Calculating the evaluation index of the classification model can help us understand the performance of the model, optimize the parameters of the model, and improve the performance of the model.

3.6.2 Precision

Precision measures the proportion of true positives to the total number of predicted positive reviews [37]. In the context of user review classification, precision is an important metric as it measures how accurately the model predicts positive reviews.

$$Precision = \frac{TP}{TP + FP} \tag{3.26}$$

3.6.3 Recall

Recall, on the other hand, measures the proportion of true positives to the total number of actual positive reviews [37]. In user review classification, recall is a critical metric as it measures how effectively the model is able to identify all positive reviews.

$$Recall = \frac{TP}{TP + FN} \tag{3.27}$$

3.6.4 F1-Score

F1-score is a widely used evaluation metric that takes both precisions and recalls into account, providing a comprehensive assessment of model performance [37]. By combining these two metrics, the F1-score offers a balanced evaluation of model effectiveness that is particularly useful in complex classification tasks.

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.28}$$

3.6.5 Accuracy

Accuracy is another commonly used evaluation metric that measures the proportion of correctly classified samples among all samples. In the context of user review classification, accuracy is a useful metric as it provides an overall assessment of the model's ability to correctly classify reviews [37].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.29}$$

3.7 Programming Environment and Libraries

3.7.1 Programming Environment

In this project, we are utilizing the Google **Colab** platform, which offers free access to GPUs and various data science libraries such as PyTorch and Transformer. **Python3** is our preferred programming language due to its extensive machine learning libraries, including Scikit-learn, TensorFlow, Keras, and PyTorch. Among these libraries, we are specifically utilizing **PyTorch**, a popular deep learning framework known for its dynamic calculation graph and automatic derivation function. Its ease of use and powerful pre-training models make it a valuable tool for our machine learning implementation.

3.7.2 Libraries

Various Python libraries have been utilized to process and analyze data and to train and evaluate machine learning models. **Pandas** has been used to handle various data types and to perform data processing and analysis efficiently. **Numpy**, as a library for processing large array data, has been used for numerical calculations and operations. **Matplotlib** has provided visualization tools and functions to generate different types of charts and graphics. **Scikit-learn** has been used for feature extraction, data preprocessing, model training, and evaluation. **Transformers** has been employed to utilize pre-trained NLP models like BERT, GPT, RoBERTa, etc. Lastly, **Hugging Face** has provided an open-source Python library called transformers, which contains many pre-trained language models, and also provides convenient APIs and tools to use and train these language models. These libraries have been instrumental in ensuring the efficiency and accuracy of data processing and analysis and have helped in achieving the project goals effectively.

Building Machine Learning Models

4.1 Design

The processes used to implement the current project are shown in Figure 4.1. It is crucial to remember that every step of the implementation process has its own special obstacles and hurdles. Consequently, it is vital to adjust and optimise the implementation process in order to guarantee the accomplishment of each assignment.

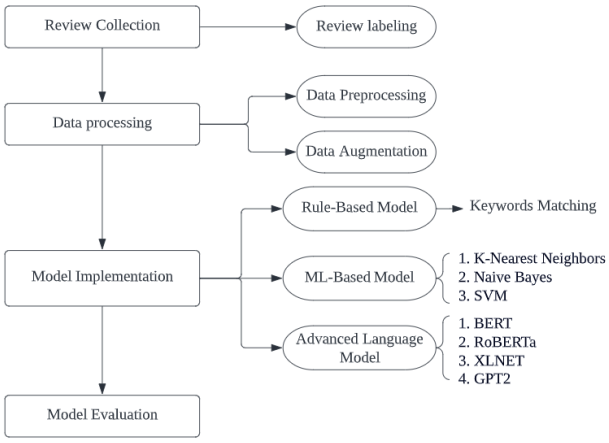

Figure 4.1: Structure of the project implementation

4.2 Part1-Review Collection

4.2.1 Cohen's Kappa coefficient

Tables 4.1 and 4.2 present the results of the labelling of the dataset by two annotators. In order to assess the consistency of the work of the two individuals, Cohen's Kappa coefficient was calculated. As Dataset 3 was augmented with data from Dataset 1, it was not necessary to calculate the coefficient for this dataset. Using the formulas presented in Section 3.3.1, the calculated values for kappa1 and kappa2 were 0.938 and 0.971, respectively. These values indicate a high level of agreement between the two annotators and suggest that the datasets collected from both Dataset 1 and Dataset 2 are reliable.

Dataset1		Annotator 1				
		Bug Report	Feature Request	Rating	User Experience	Total
Annotator 1	Bug Report	988	8	3	12	1011
	Feature Request	4	981	4	2	991
	Rating	2	8	977	18	1005
	User Experience	12	3	5	973	993
	Total	1006	1000	989	1005	4000

Table 4.1: Dataset1 IAA

Dataset2		Annotator 1				
		Bug Report	Feature Request	Rating	User Experience	Total
Annotator 1	Bug Report	1393	4	5	8	1410
	Feature Request	20	1237	7	2	1266
	Rating	3	8	715	10	736
	User Experience	2	6	7	573	588
	Total	1418	1255	734	593	4000

Table 4.2: Dataset1 IAA

4.2.2 Dataset Analysis

In Figures 4.2, 4.3, and 4.4, three combined pie charts and histograms show the data analysis for the three datasets. Each graph's right side displays the frequency distribution of user comment lengths, with the left side showing the percentage of each category in the dataset. Furthermore, Table 4.3 gives precise information on the quantity and length of user comments. For the creation and assessment of the user review categorization models, these visualisations and tables provide a thorough knowledge of the traits and distribution of the datasets.

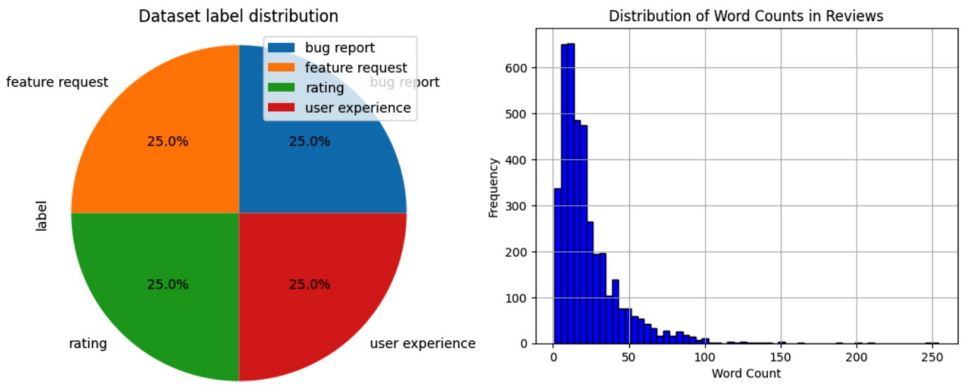

Figure 4.2: Dataset1 Analysis

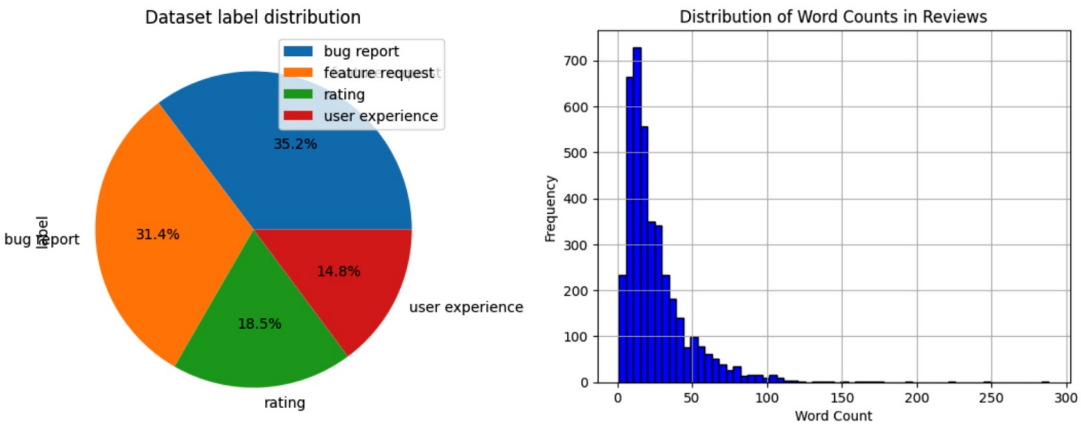

Figure 4.3: Dataset2 Analysis

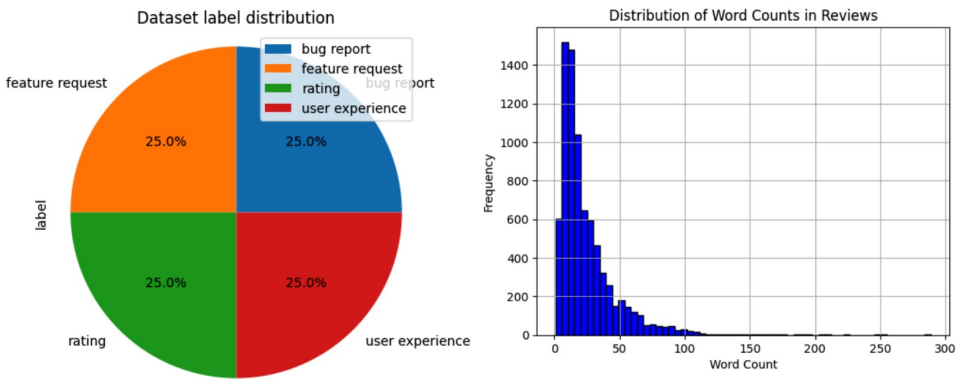

Figure 4.4: Dataset3 Analysis

Any machine learning project must include data analysis since it offers insights into the dataset's properties, makes it possible to choose the best features and model parameters, and identifies any potential problems or biases in the data. Researchers can better understand the distribution of data points, the frequency of various categories, the length and structure of user comments, and other pertinent characteristics by conducting data analysis on the dataset.

	Bug Report	Feature Request	Rating	User Experience	Average length	Median length	Mode length
Dataset1	1000	1000	1000	1000	121.94	87.00	57.00
Dateset2	1410	1257	740	593	129.67	96.00	54.00
Dataset3	2000	2000	2000	2000	125.58	90.00	53.00

Table 4.3: Number of labels in each dataset and text length

4.3 Part2-Data processing

4.3.1 Data Preprocessing

One of the goals of this project is to explore the impact of different preprocessing on the performance of the model, so I performed 5 different and separate preprocessing on the initial data set and saved it in the data set. Figure 4.5 shows the contents of the data set. We can see that after preprocessing, the new data set adds four parts.

review	label	new_label	remove_all_stopwords	remove_some_stopwords	stemming	lemmatization
Not able to add freinds.	bug report	0	able add freinds show son	not able add freinds show s	not abl to ac	not able to add freinds it show something went wrong recommend not
Calls keep dropping for	bug report	0	calls keep dropping reaso	calls keep dropping no reas	call keep dro	call keep dropping for no reason and is super annoying
Can't access to my acco	bug report	0	cant access account solve	cant access account solve i	cant access t	cant access to my account solve this issue immediately
I don't know what's wro	bug report	0	dont know whats wrong v	dont know whats wrong wh	i dont know	i dont know whats wrong with my own whatsapp it not working and if
Cannot record audio wh	bug report	0	cannot record audio takir	cannot record audio taking	cannot recor	cannot record audio while taking video
can't turn off notificatio	bug report	0	cant turn notifications ap	cant turn off notifications	cant turn off	cant turn off notification if i do so the app will crush and notification
My videos are turning ir	bug report	0	videos turning green shoo	videos turning green after s	my video are	my video are turning into green after i shoot them please help
My snaps aren't sending	bug report	0	snaps arent sending reaso	snaps arent sending reason	my snap are	my snap arent sending and for some reason the time on my snapchats
App is disappointing mo	bug report	0	app disappointing messag	app disappointing message	app is disapp	app is disappointing message are taking to long to send
When I try to make filte	bug report	0	try make filter video hang	try make filter video hangi	when i tri to	when i try to make filter video it hanging and video lagging cant make
I cant find my friend on	bug report	0	cant find friend snapchat	cant find friend snapchat	i cant find m	i cant find my friend on snapchat
hello Snapchat support	bug report	0	hello snapchat support te	hello snapchat support tea	hello snapch	hello snapchat support team i get due to repeated failed login attempt
I'll give one star to Snap	bug report	0	ill give one star snapchat	ill give one star snapchat b	ill give one s	ill give one star to snapchat because my snapchat is not working prope

Figure 4.5: Dataset after preprocessing

A series of experiments were conducted to compare the performance of these models and to investigate the impact of different pre-processing techniques, dataset balance and size on classification performance. In Chapter 5, we can see that each model needs to train the following 7 sets of data:

1. original review(balanced 4000 review dataset)
2. remove_all_stopwords (balanced 4000 review dataset)
3. remove_some_stopwords (balanced 4000 review dataset)
4. stemming (balanced 4000 review dataset)
5. lemmatization (balanced 4000 review dataset)
6. original review(unbalanced 4000 review dataset)
7. original review(balanced 8000 review dataset)

4.3.2 Dataset Segmentation

In order to train, tune and evaluate the performance of machine learning models on the datasets in this project, a random split is performed on each dataset to create training, validation and test sets. The split ratio is 80:10:10. This approach helps to ensure that the model has good generalization ability and avoids overfitting to the training data. Proper dataset segmentation is crucial for accurately assessing the performance of the model on unseen data and ensuring that it can be applied effectively in real-world scenarios.

4.4 Part3-Model Implementation

The core of this project is the implementation of various models to classify app user reviews. The models used can be classified into three categories, namely rule-based models (keyword matching), machine learning models (KNN, SVM, and NB), and advanced language models (BERT, RoBERTa, XLNet, and GPT2). The theoretical knowledge of these models has been presented in Chapter 3, so further elaboration is unnecessary. The subsequent section will focus on the specific details of some of these models.

4.4.1 Rule-based Model Implementation

In Chapter 3, the keyword matching model was introduced in detail. To implement this model, a vocabulary of keywords for each category needs to be built. Whenever a new review is entered, it is compared to all the keywords in the vocabulary, and the category with the most keyword matches is assigned to the review.

4.4.2 Machine Learning Model Implementation

Three machine learning models are implemented in this project, namely KNN, SVM, and Naive Bayes, the theories of which are also introduced in Chapter 3. When training and validating machine learning models, it is important to pay attention to tuning and evaluation. This includes selecting appropriate hyperparameters.

4.4.3 Advanced Language Model Implementation

BERT, RoBERTa, XLNet, and GPT2 are all advanced language models that are based on Transformer architecture. These models can be implemented through two stages of pre-training and fine-tuning. However, it is important to note that the optimization and loss functions used in neural networks also play a crucial role in the performance of these models. It is necessary to explain those two pieces of knowledge.

Loss Function

The loss function, sometimes referred to as the cost function or objective function in neural networks, measures the discrepancy between anticipated output and actual target values. The main objective of training a neural network is to reduce this loss function, which results in predictions that are more accurate. Cross-entropy loss [38] is commonly used for classification problems, particularly for multi-class classification. The formula for cross-entropy loss is:

$$L(y, \hat{y}) = -\sum y_i \log(\hat{y}_i) \tag{4.1}$$

where $L(y, \hat{y})$ represents the loss, y_i denotes the actual target value, $\hat{y}_i$ indicates the predicted output, and N is the number of samples.

The gradient problem in neural networks occurs when the gradient of the loss function becomes very small, leading to slow learning or getting stuck in local minima during training. To avoid this problem, various optimization algorithms have been developed, such as stochastic gradient descent (SGD) and its variants. In addition to the optimization algorithm, the choice of the loss function is also important. The cross-entropy loss function is commonly used in classification tasks, as it measures the difference between the predicted and actual class probabilities. The sigmoid function is often used as the activation function for binary classification tasks, but its derivative becomes very small when the input is either very large or very small, which can exacerbate the gradient problem. To avoid this issue, the cross-entropy loss function can be used without the sigmoid or softmax classifier within the update equation, as its derivative does not involve the sigmoid function.

Optimizer

Optimizers are a class of algorithms commonly used in neural networks to minimize loss functions by updating the parameters of neurons using the deep learning technique of gradient descent [39].

$$\theta = \theta - \eta \frac{\partial Loss(\theta)}{\partial \theta} \tag{4.2}$$

The learning rate (η) in gradient descent optimization algorithms is a critical hyperparameter that controls the magnitude of gradient updates. Selecting an appropriate learning rate is essential for achieving optimal optimization results. A learning rate that is too low might lead to slow convergence, vanishing gradients, and overfitting, whereas an excessively high learning rate can cause exploding gradients, substantial error value oscillations, and hinder model convergence.

Smith proposed the LR range test method to address this issue, which involves training the model across multiple epochs while increasing the learning rate from a minimal threshold. An acceptable learning rate should yield satisfactory performance in both training and validation accuracy. Additionally, decay strategies can be employed to adjust the learning rate during training to ensure successful model convergence.

Despite the widespread use of gradient descent algorithms for optimizing machine learning models, they have limitations. They cannot guarantee that the final result is a global minimum and may be trapped in a local minimum or a saddle point. Consequently, numerous advanced optimization algorithms based on gradient descent have been proposed. In this project, we employ the Adam optimizer, which incorporates momentum and adaptive learning rate techniques. Adam [40] is a popular optimization algorithm that dynamically adjusts the

learning rate as the gradient decreases, allowing the loss function to converge more rapidly and efficiently. The momentum technique aids the optimizer in maintaining a consistent direction during training, while the adaptive learning rate technique ensures the learning rate is adjusted appropriately for each parameter. Together, these techniques increase the likelihood of finding the global minimum of the loss function and enable the model to fit more quickly and effectively.

Adamw is an extension of the Adam optimizer that incorporates weight decay regularization to enhance the optimization performance of deep neural networks. Weight decay regularization introduces a penalty term to the loss function that encourages the model to utilize smaller weight values, which can prevent overfitting and improve generalization performance.

4.5 Part4-Model Evaluation

Performance indicators (accuracy, precision, recall, F1-score) are important elements for evaluating model performance. One thing to watch out for, though, is overfitting and underfitting, which can have an impact on the performance metrics derived from the confusion matrix, and thus the model's implementation.

4.5.1 Overfitting and Underfitting

Overfitting [41] arises when a model excessively learns the training data, capturing noise and random fluctuations instead of the underlying patterns. Consequently, the model becomes less generalizable and exhibits poor performance on unseen data due to its high variance. In the context of user review classification, overfitting can lead to inaccurate classifications, as the model might be overly sensitive to specific words or phrases in the training data that do not generalize well to new reviews.

Conversely, underfitting [41] occurs when a model fails to capture the underlying patterns in the training data, often attributable to its simplicity. In this case, the model demonstrates high bias and performs poorly on both training and testing data. In the realm of user review classification, underfitting can result in the model's inability to identify pertinent features and patterns necessary for accurate classification, ultimately leading to suboptimal performance.

4.5.2 Training loss and Validation loss

In the model, training loss and Validation loss are two commonly used performance indicators for evaluating the training situation and generalization ability of the model. Train loss refers to the loss of the model on the training set, and can also be understood as the error of the model on the training set. Val loss refers to the loss of the model on the validation set, and can also be understood as the error of the model on unseen data. In deep learning, due to a large number of model parameters, the model is prone to overfitting. By monitoring the changes in the values of the two losses, the overfitting of the model can be discovered in time, so that measures can be taken to adjust and optimize it.

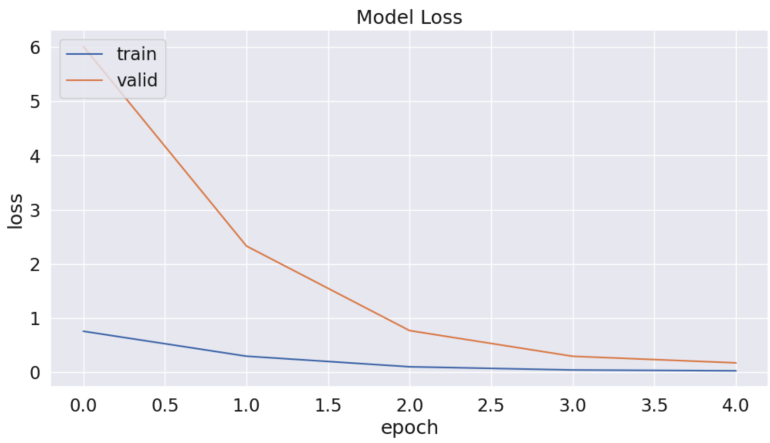

Figure 4.6: Model loss value

- Train loss down, Validation loss down: training is normal, the network is still learning, best case.
- Train loss decreased, Validation loss: increased or unchanged: a little overfitting overfitting, you can stop training, use overfitting methods such as data enhancement, regularization, dropout, max pooling, etc.

- The training loss is stable, and the Validation loss is declining: There is a problem with the data, check whether the data label is correct, whether the distribution is consistent, and whether it is shuffled.
- The training loss is stable, and the Validation loss is stable: when the learning process encounters a bottleneck, you can try to reduce the learning rate or the number of batches
- Increase in train loss, increase in Validation loss: Improper network structure design, unreasonable parameters, data sets need to be cleaned, etc., the worst case.

Results Analysis

5.1 Lexicon-Based Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.31	0.70	0.23	0.34	0.57	0.55	0.56	0.4	0.39	0.39	0.32	0.07	0.31
2	0.29	0.70	0.27	0.39	0.56	0.53	0.53	0.4	0.35	0.37	0.41	0.02	0.29
3	0.31	0.70	0.26	0.38	0.58	0.56	0.57	0.38	0.4	0.39	0.43	0.02	0.31
4	0.32	0.69	0.31	0.43	0.54	0.51	0.52	0.43	0.34	0.38	0.37	0.11	0.32
5	0.34	0.68	0.3	0.42	0.58	0.59	0.58	0.41	0.39	0.4	0.36	0.08	0.34
6	0.34	0.70	0.25	0.37	0.66	0.54	0.59	0.37	0.34	0.36	0.21	0.09	0.34
7	0.31	0.70	0.24	0.36	0.57	0.56	0.56	0.4	0.36	0.38	0.33	0.07	0.31

Table 5.1: Keyword Matching Model Results

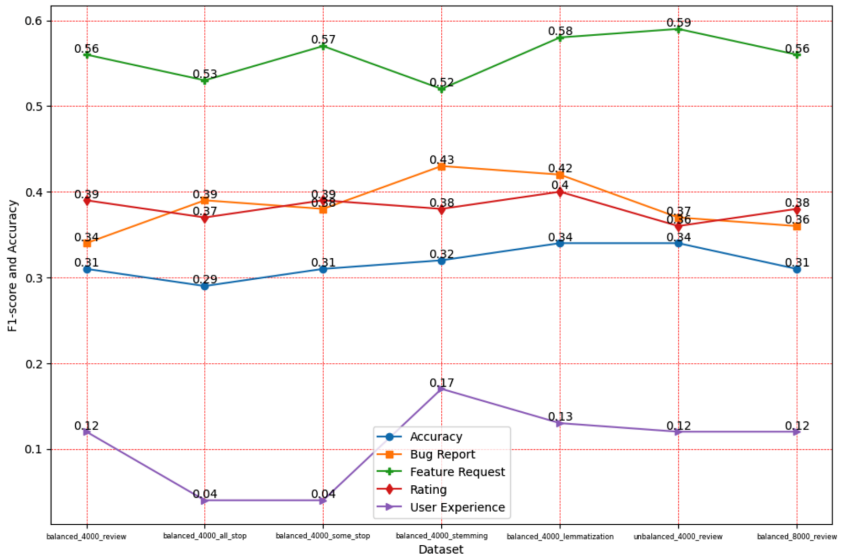

Figure 5.1: Model Performance Results Line Chart in Keyword Matching

According to Figure 5.1, the performance of the four categories, as measured by the average F1-score, is ranked in the following order: Feature Request performs the best, followed by Rating, Bug Report, and finally User Experience. This ranking is primarily dependent on the quality of the keyword dictionary utilized.

As observed in the figure, Accuracy does not exhibit significant variation across different datasets, remaining consistently between 0.31 and 0.34. This suggests that when employing keyword matching methods, the overall accuracy is relatively low and largely unaffected by the dataset. Specifically, the highest Accuracy score of 0.34 is achieved using the "stemming " dataset, while the lowest score of approximately 0.31 is observed with the "remove all stopwords" dataset.

The decrease in accuracy when removing stopwords may be attributed to the presence of stopwords in the dictionary. However, stemming and lemmatization can alter the form of words, leading to an increase in accuracy if irrelevant words are preprocessed into words in the dictionary, and a decrease otherwise.

The changes induced by different datasets do not exhibit large fluctuations as observed in other models. This is because the keyword matching method determines the text category solely based on a predefined keyword list, without considering the contextual and semantic information of the text for classification. In smaller datasets, keyword matching methods may be adversely affected by a limited number of data samples and unbalanced category distribution. Conversely, in larger datasets, the diversity and complexity of text data may also constrain the performance of keyword matching methods.

5.2 Machine Learning Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.42	0.51	0.37	0.43	0.55	0.32	0.4	0.35	0.8	0.49	0.42	0.17	0.24
2	0.41	0.55	0.34	0.42	0.74	0.32	0.44	0.32	0.84	0.46	0.43	0.13	0.2
3	0.42	0.62	0.39	0.48	0.75	0.27	0.39	0.32	0.88	0.47	0.47	0.12	0.19
4	0.45	0.55	0.41	0.47	0.56	0.33	0.42	0.36	0.82	0.5	0.52	0.2	0.29
5	0.44	0.57	0.38	0.46	0.56	0.33	0.42	0.35	0.83	0.5	0.51	0.2	0.29
6	0.48	0.61	0.39	0.48	0.68	0.51	0.58	0.3	0.88	0.45	0.56	0.09	0.15
7	0.6	0.68	0.59	0.63	0.82	0.55	0.66	0.47	0.81	0.58	0.61	0.45	0.52

Table 5.2: KNN Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.69	0.67	0.88	0.76	0.79	0.66	0.72	0.77	0.55	0.64	0.6	0.67	0.63
2	0.68	0.67	0.88	0.76	0.76	0.68	0.72	0.71	0.53	0.61	0.57	0.61	0.59
3	0.69	0.67	0.91	0.77	0.79	0.67	0.73	0.73	0.54	0.62	0.6	0.62	0.61
4	0.61	0.62	0.83	0.71	0.7	0.57	0.63	0.76	0.32	0.45	0.49	0.72	0.59
5	0.68	0.68	0.89	0.77	0.77	0.66	0.71	0.76	0.46	0.57	0.54	0.69	0.61
6	0.66	0.66	0.88	0.76	0.62	0.81	0.7	0.95	0.27	0.42	0.91	0.19	0.32
7	0.78	0.72	0.89	0.8	0.85	0.77	0.8	0.85	0.68	0.76	0.72	0.77	0.74

Table 5.3: Naive Bayes Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.68	0.71	0.8	0.75	0.71	0.66	0.68	0.68	0.65	0.66	0.59	0.59	0.59
2	0.66	0.71	0.76	0.73	0.72	0.68	0.7	0.64	0.62	0.63	0.56	0.57	0.56
3	0.66	0.71	0.79	0.75	0.72	0.68	0.7	0.65	0.63	0.64	0.55	0.54	0.54
4	0.68	0.71	0.8	0.75	0.68	0.66	0.67	0.69	0.63	0.66	0.6	0.6	0.6
5	0.69	0.73	0.82	0.77	0.74	0.65	0.7	0.69	0.67	0.68	0.61	0.62	0.62
6	0.77	0.78	0.85	0.81	0.8	0.86	0.83	0.74	0.61	0.67	0.72	0.53	0.61
7	0.83	0.83	0.89	0.86	0.86	0.84	0.85	0.84	0.8	0.82	0.79	0.79	0.79

Table 5.4: SVM Model Results

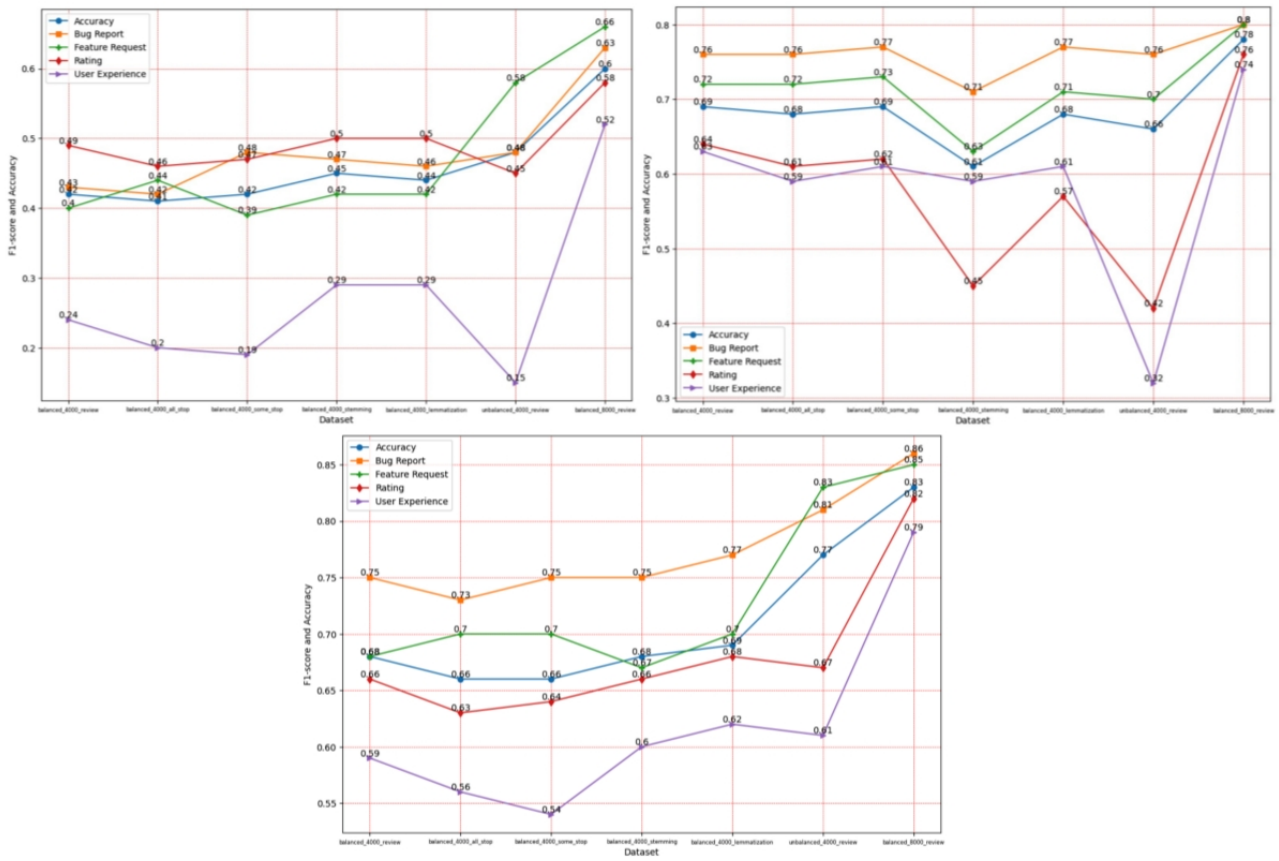

Figure 5.2: Model Performance Results Line Chart, KNN (top left), Naive Bayes (top right), SVM (bottom)

This section presents the performance results of three machine learning models, namely SVM, Naive Bayes, and KNN, in user review classification tasks. The results are displayed in tables and line graphs. It is evident that SVM achieves the best performance, followed by Naive Bayes, and finally KNN.

The superior performance of SVM in this project can be attributed to its ability to handle high-dimensional, sparse data more effectively. SVM finds the optimal hyperplane in the feature space, making it well-suited for high-dimensional data. In contrast, KNN suffers from the "curse of dimensionality," and Naive Bayes relies on the assumption of feature independence, which may not hold for text data [42]. Furthermore, SVM is efficient with sparse data as it only depends on a small subset of training data (support vectors) to define the decision boundary, while KNN's distance-based approach is less effective and computationally intensive for sparse data. SVM also maximizes the margin between classes, providing a robust decision boundary less sensitive to noise and outliers, unlike Naive Bayes and KNN. Lastly, SVM's kernel trick enables it to handle non-linear classification problems, which is not inherently possible with Naive Bayes and KNN [42].

For the four labels in user reviews, Naive Bayes and SVM models yield the highest F1 score for bug reports, followed by feature requests, ratings, and finally user experience. In the KNN model, the highest F1 score is observed for ratings, followed by bug reports, feature requests, and lastly user experience. These rankings are closely related to the content of user reviews.

The accuracy trends of the three models across seven different datasets are consistent. The accuracy decreases when "remove all stop words" is applied compared to the original dataset, but recovers when "stemming" is employed. The accuracy further decreases with "lemmatization" and increases on both imbalanced and large datasets. However, the accuracy of the Naive Bayes model in the imbalanced dataset is reduced. The reasons for these observations are discussed in sections 5.4.1 and 5.4.2.

5.3 Advanced Language Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.73	0.78	0.76	0.77	0.81	0.69	0.75	0.7	0.81	0.75	0.62	0.66	0.64
2	0.7	0.71	0.77	0.74	0.74	0.69	0.71	0.69	0.75	0.72	0.65	0.6	0.63
3	0.72	0.82	0.73	0.77	0.75	0.71	0.73	0.66	0.76	0.7	0.63	0.67	0.65
4	0.68	0.73	0.74	0.74	0.75	0.7	0.73	0.65	0.73	0.69	0.58	0.55	0.57
5	0.7	0.82	0.77	0.8	0.72	0.71	0.71	0.69	0.73	0.71	0.55	0.58	0.56
6	0.85	0.88	0.88	0.88	0.89	0.85	0.87	0.75	0.86	0.8	0.77	0.76	0.76
7	0.91	0.96	0.93	0.95	0.88	0.95	0.91	0.89	0.94	0.91	0.93	0.84	0.89

Table 5.5: BERT Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.8	0.89	0.81	0.85	0.83	0.82	0.82	0.74	0.81	0.78	0.73	0.74	0.73
2	0.73	0.85	0.69	0.76	0.72	0.75	0.73	0.7	0.8	0.75	0.63	0.66	0.64
3	0.76	0.89	0.77	0.82	0.77	0.79	0.78	0.71	0.81	0.76	0.65	0.65	0.65
4	0.72	0.84	0.74	0.79	0.72	0.71	0.72	0.69	0.75	0.72	0.62	0.65	0.64
5	0.77	0.9	0.75	0.82	0.79	0.77	0.78	0.71	0.82	0.77	0.65	0.71	0.68
6	0.85	0.94	0.84	0.89	0.84	0.89	0.86	0.72	0.87	0.79	0.82	0.77	0.79
7	0.93	0.96	0.96	0.96	0.93	0.93	0.93	0.92	0.94	0.93	0.92	0.9	0.91

Table 5.6: RoBERTa Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.73	0.88	0.77	0.82	0.74	0.78	0.76	0.62	0.76	0.68	0.69	0.63	0.66
2	0.69	0.79	0.73	0.76	0.71	0.72	0.72	0.6	0.75	0.66	0.66	0.58	0.61
3	0.71	0.85	0.74	0.79	0.72	0.76	0.74	0.61	0.71	0.66	0.65	0.62	0.63
4	0.69	0.84	0.71	0.77	0.71	0.75	0.73	0.52	0.75	0.62	0.66	0.57	0.61
5	0.73	0.85	0.77	0.81	0.75	0.76	0.76	0.64	0.79	0.71	0.68	0.61	0.65
6	0.82	0.93	0.81	0.86	0.84	0.87	0.85	0.63	0.85	0.72	0.76	0.74	0.75
7	0.87	0.9	0.92	0.91	0.89	0.89	0.89	0.84	0.89	0.86	0.87	0.8	0.83

Table 5.7: XLNet Model Results

		Bug Report			Feature Request			Rating			User Experience		
	Accuracy	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.6	0.67	0.69	0.68	0.71	0.68	0.7	0.58	0.48	0.53	0.47	0.55	0.51
2	0.6	0.65	0.58	0.61	0.82	0.52	0.64	0.53	0.67	0.59	0.52	0.67	0.59
3	0.6	0.59	0.7	0.64	0.69	0.59	0.64	0.56	0.71	0.63	0.57	0.44	0.49
4	0.56	0.64	0.5	0.56	0.7	0.51	0.59	0.46	0.73	0.57	0.5	0.49	0.49
5	0.61	0.69	0.6	0.64	0.64	0.73	0.68	0.52	0.65	0.58	0.62	0.45	0.52
6	0.67	0.73	0.75	0.74	0.61	0.77	0.68	0.66	0.53	0.59	0.67	0.42	0.52
7	0.73	0.74	0.78	0.76	0.85	0.69	0.76	0.66	0.75	0.7	0.69	0.71	0.7

Table 5.8: GPT2 Model Results

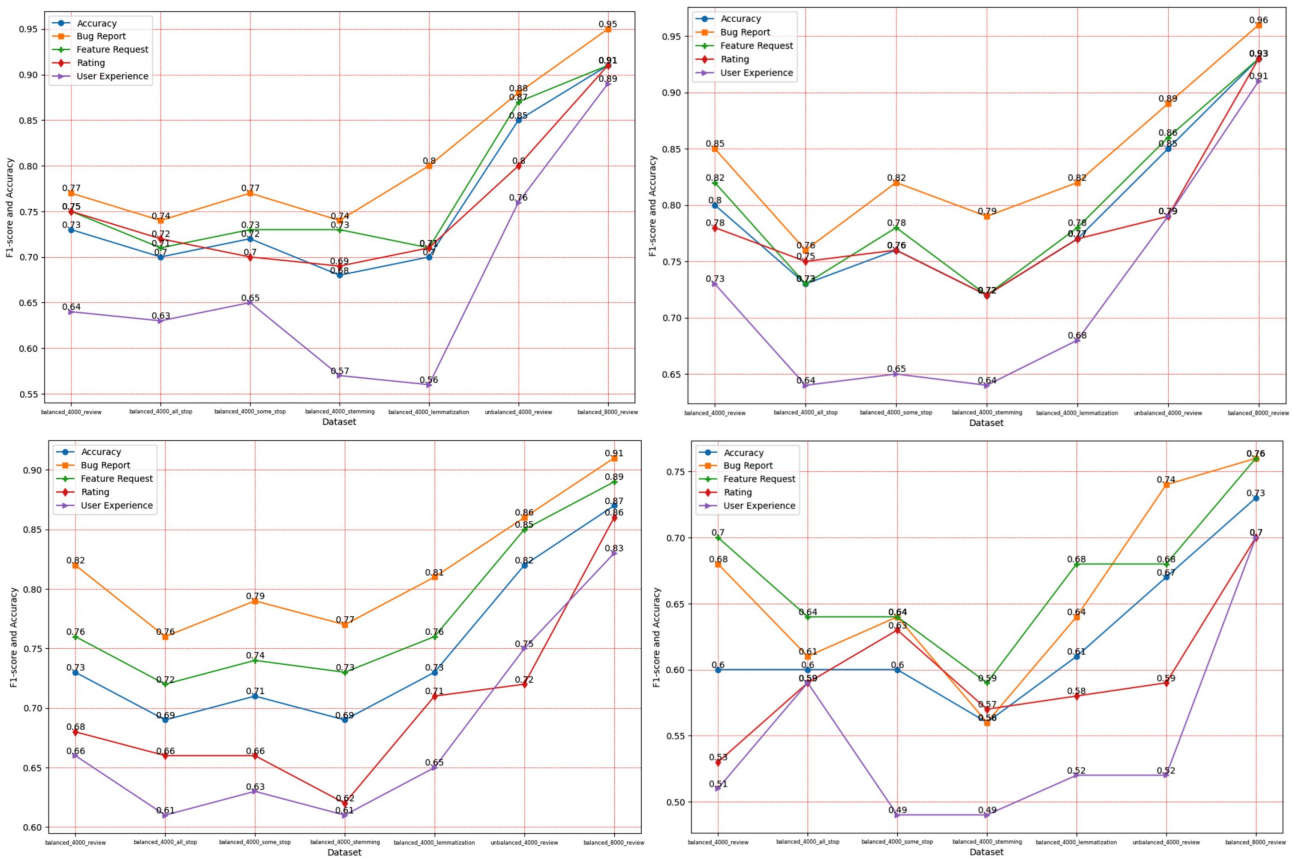

Figure 5.3: Model Performance Results Line Chart, BERT (top left), RoBERTa (top right), XLNET (bottom left), GPT2 (bottom right)

This section presents the performance results of four advanced language models in user review classification tasks, illustrated through tables and line graphs. The findings indicate that RoBERTa achieves the best performance, followed by BERT, XLNet, and lastly GPT-2.

RoBERTa is a robustly optimized variant of BERT that outperforms it in NLP tasks, including text classification, due to several key optimizations [43]. It is trained on a larger dataset and for a longer duration, enabling it to learn more complex patterns. RoBERTa also modifies BERT's pre-training process by removing the NSP task, using larger batch sizes, and employing dynamic masking, which enhances the model's contextual understanding. Moreover, RoBERTa adopts byte-level Byte Pair Encoding (BPE) tokenization, improving its handling of out-of-vocabulary words. Collectively, these factors contribute to RoBERTa's superior performance in text classification tasks compared to BERT.

BERT, GPT-2, and XLNet are popular Transformer-based models employed in NLP tasks, with BERT generally considered superior for text classification. BERT's advantages include bidirectional context processing, which captures context more effectively than GPT-2's unidirectional approach and XLNet's permutation-based method. BERT's pre-training objectives, such as MLM and NSP, contribute to better contextual representations. Additionally, BERT's architecture allows for easy fine-tuning by adding a classifier layer, making it a flexible choice for various applications [44]. Despite BERT's advantages, it is important to note that model performance can be highly dependent on the specific task and dataset, and in some cases, GPT-2 and XLNet might outperform BERT.

Across the four labels in user reviews, BERT, RoBERTa, and XLNet models exhibit the highest F1 score for bug reports, followed by feature requests, ratings, and finally user experience. In contrast, GPT-2 displays the highest F1 score for feature requests, followed by bug reports, ratings, and lastly user experience. These rankings are closely related to the content of user comments.

The accuracy trends of the four models are consistent across the seven different datasets. Accuracy decreases in the "remove all stopwords" dataset compared to the original dataset but recovers in the "stemming" dataset. However, accuracy decreases with "stemming" and increases with "lemmatization." Subsequently, the accuracy rate improves significantly in both imbalanced and large datasets. The reasons for these observations are discussed in Section 5.4.1.

5.4 Discuss the impact of factors on model performance

5.4.1 Impact of preprocessing on performance

Model	Keys Words	KNN	SVM	Naive Bayes	BERT	RoBERTa	XLNet	GPT2
Review	0.34	0.42	0.68	0.69	0.73	0.80	0.73	0.60
Remove All Stopwords	0.39	0.41	0.66	0.68	0.70	0.73	0.69	0.60
Remove Some Stopwords	0.38	0.42	0.66	0.69	0.72	0.76	0.71	0.60
Stemming	0.43	0.45	0.68	0.61	0.68	0.72	0.69	0.56
Lemmatization	0.42	0.44	0.69	0.68	0.70	0.77	0.73	0.61

Table 5.9: Model Accuracy with Different Pre-processing Techniques

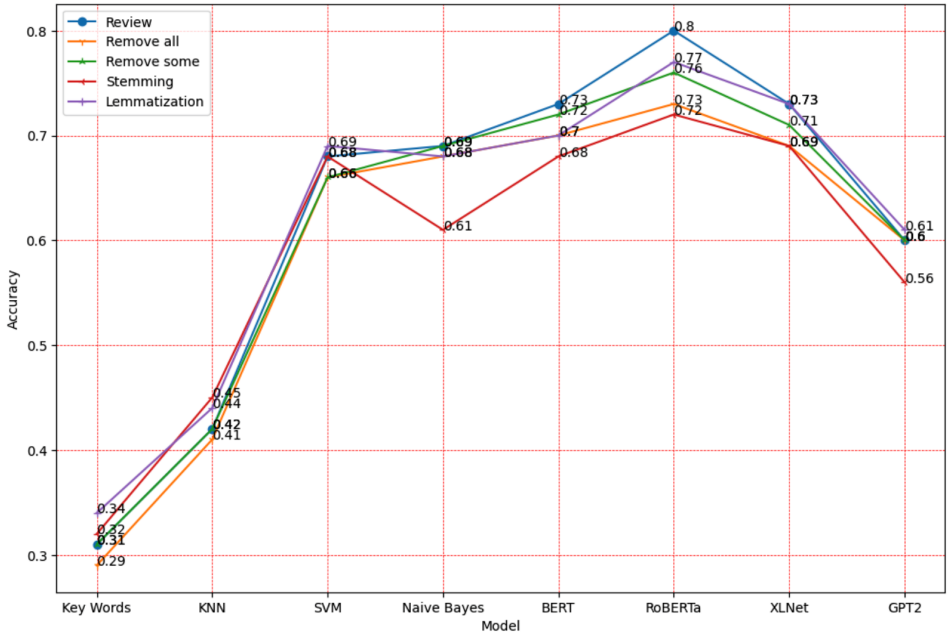

Figure 5.4: Model Performance Results Line Chart of Different Pre-processing Techniques

Figure 5.4 shows the accuracy performance of different text preprocessing methods (original review, remove all stop words, remove some stop words, stemming, and lemmatization) on different models.

For original reviews, models such as BERT and RoBERTa perform well, demonstrating their ability to handle raw text data effectively [17]. When all stopwords are removed, performance decreases slightly for most models, suggesting that stopwords contribute to capturing semantics and context. Selectively removing stopwords yields better results for models like BERT, RoBERTa, and XLNet, indicating that some stopwords play a more crucial role in understanding semantics and context [45].

Applying stemming leads to mixed results, with performance dropping for some models, such as SVM and Naive Bayes, possibly due to the loss of semantic information. In contrast, lemmatization generally improves the performance of models like BERT, RoBERTa, and XLNet compared to the original review, as it preserves more semantic information.

For Keyword, KNN, SVM, and Naive Bayesian models, retaining stop words in the results of the original review with slightly better accuracy. In BERT, RoBERTa, and XLNet models, partial stopword removal and lemmatization lead to improved performance, suggesting the importance of some stopwords in these models. However, stemming generally performs poorly across all models, likely due to the loss of semantic information.

Lastly, the GPT-2 model exhibits lower accuracy in these experiments, which could be attributed to its generative nature, rendering it less suitable for classification tasks compared to other pre-trained models.

In conclusion, the influence of the text preprocessing method chosen varies depending on the model. Deep learning models that have already been trained, like BERT and RoBERTa, can frequently handle raw text input well and may profit from more advanced preprocessing methods like lemmatization. The performance of some classical machine learning models, however, may be improved with less preprocessing or stemming. In actual use, it's critical to select the best preprocessing method based on the particular model and task.

5.4.2 Impact of dataset balance on performance

Model	Keys Words	KNN	SVM	Naive Bayes	BERT	RoBERTa	XLNet	GPT2
balanced	0.34	0.42	0.68	0.69	0.73	0.80	0.73	0.60
unbalanced	0.37	0.48	0.77	0.66	0.85	0.85	0.82	0.67

Table 5.10: Model Accuracy with balanced and unbalanced datasets

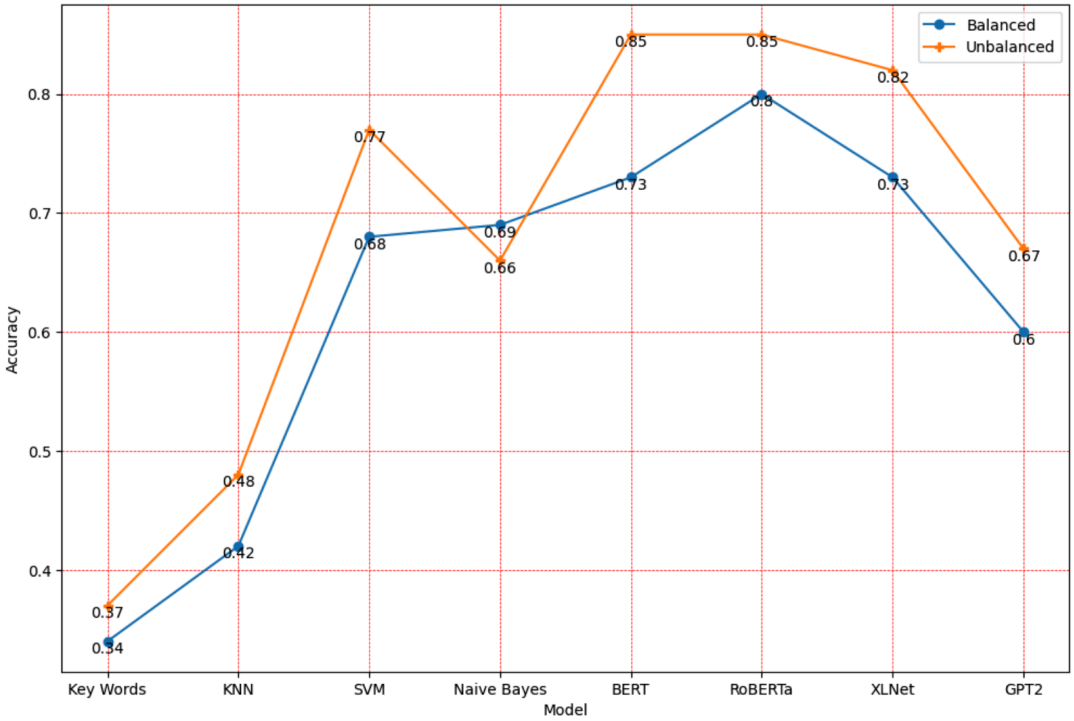

Figure 5.5: Model Performance Results Line Chart of balanced and unbalanced datasets

Figure 5.5 shows the performance of different models when trained on balanced and unbalanced datasets. We can know the following.

Keyword matching, KNN, and SVM models perform better on unbalanced datasets, reflecting their ability to benefit from the natural data distribution. Conversely, Naive Bayes demonstrates superior performance on balanced datasets, potentially due to its assumptions being better suited for such data. Pretrained models like BERT, RoBERTa, XLNet, and GPT-2 also exhibit improved performance on unbalanced datasets, suggesting their effectiveness in capturing and leveraging real-world data distribution.

The poor performance of the Naive Bayes model on imbalanced datasets can be attributed to its underlying assumptions and data distribution. The Naive Bayes model is premised on the assumption of conditional independence, implying that features are mutually independent. In a balanced dataset, each class has approximately the same number of samples, enabling Naive Bayes to learn the feature distribution for each class. Conversely, in an imbalanced dataset, the sample size of certain classes may greatly exceed that of others, leading to biased learning of class feature distributions by the Naive Bayes model [46]. This bias may result in suboptimal performance in classes with fewer samples, consequently reducing the model's overall performance.

In summary, most models perform better on the unbalanced dataset, likely because it resembles real-world data distribution. However, results may vary depending on the task and dataset, so it is essential to consider their specific characteristics when choosing a training strategy.

5.4.3 Impact of dataset size on performance

Model	Keys Words	KNN	SVM	Naive Bayes	BERT	RoBERTa	XLNet	GPT2
Small dataset	0.34	0.42	0.68	0.69	0.73	0.80	0.73	0.60
Big dataset	0.36	0.60	0.83	0.78	0.91	0.93	0.87	0.73

Table 5.11: Model Accuracy with Different Dataset size

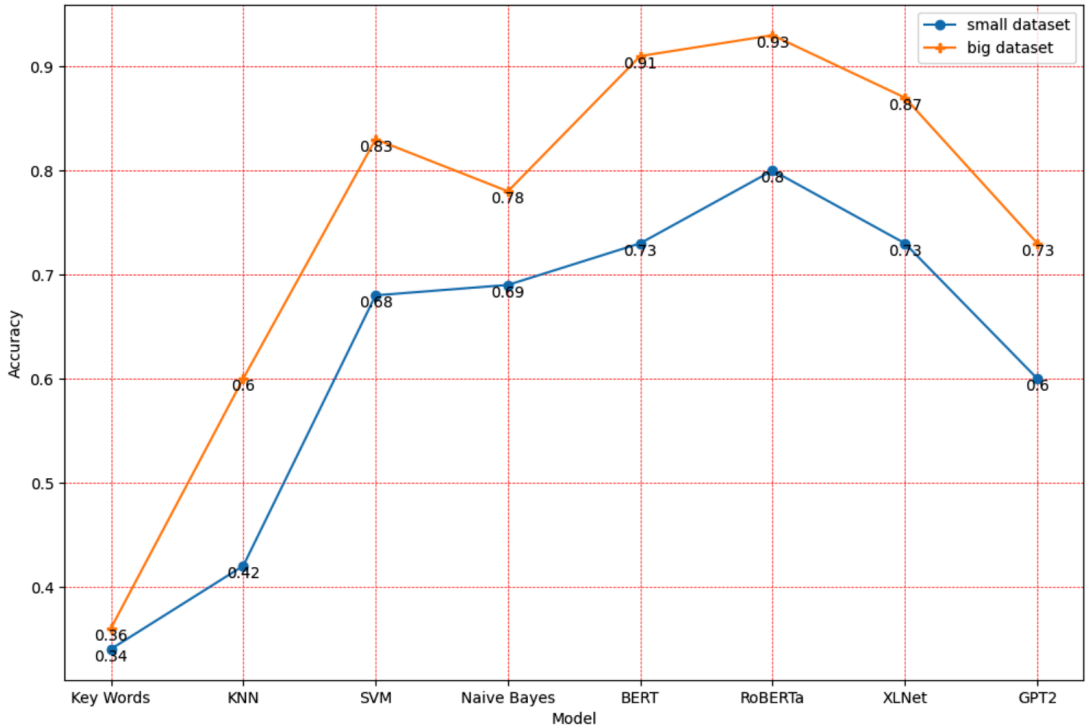

Figure 5.6: Model Performance Results Line Chart of Different dataset size

Figure 5.11 shows the accuracy performance of different models for the size of the dataset. We can observe that all models perform better on large datasets than on small ones.

Larger datasets have several advantages for training machine learning models. Firstly, they contain more samples and features, which helps capture richer and more diverse patterns, leading to improved generalization. Secondly, larger datasets reduce the risk of overfitting, where the model captures noise instead of true underlying patterns. Thirdly, larger datasets improve model stability, reducing the impact of random fluctuations and noise on model performance. Lastly, larger datasets have a more balanced class distribution, improving the model's ability to handle class imbalance issues and achieve better performance across classes [47].

In summary, all models perform better on the big dataset, emphasizing the importance of having sufficient data for training machine learning and deep learning models.

Research Evaluation

6.1 Comparison of My Project with Related Work

In this section, we compare the performance of our project with related work to demonstrate our understanding of the existing literature and identify research gaps addressed by our project. Comparing our project with related work helps evaluate different approaches, validate findings, and demonstrate our expertise in the field, thereby contributing to the advancement of the field and enhancing the credibility of our work. To better reflect, we select the results obtained on the data set with the best performance (original 8000 reviews) and compare the performance of related work models.

	Keywords Matching	KNN	Naive Bayes	SVM	BERT	RoBERTa	XLNET	GPT2
My Project	36	60	83	78	91	93	87	73
Relate Work	64	61.78	62	61.16	98.07	98.36	97.89	94.78

Table 6.1: Model Performance Accuracy(Comparison of My Project with Related Work)

Pang, Lee, and Vaithyanathan used a rule-based method based on keyword matching to achieve an accuracy rate of 64% in sentiment analysis of movie reviews, which is a binary classification task. In contrast, our dataset has 4 labels, making it a more complex multi-classification task. Therefore, it is understandable that our accuracy rate of 36% is lower than that of related work.

Sarno proposed using social media text to predict personality using three classification methods: Naive Bayes, KNN, and SVM. The dataset for this study is the MyPersonality dataset, which includes 10,000 status updates from 250 users. These statuses have been labeled into five large personality dimensions, making it a 5-category task, which is more complex than the four labels task of our project. The implementation of the KNN model is similar, but the distance calculation uses the Cosine similarity method, and the accuracy rate is higher than our project at 61.78%. Sarno also proposed a Naive Bayes variant called Multinomial Naive Bayes, which is better suited for discrete feature data, achieving an accuracy rate of 62%. The SVM model did not show significant improvement, with an accuracy rate of 61.16%.

Compared to related work, the four high-level language models used in our project were implemented according to the original paper and its appended codes. However, the final results of the related work models were better than ours, which can be attributed to factors such as their larger and better datasets, good hardware support, better parameter choices, and other factors.

6.2 Reflection

The completion of this project has been satisfactory, with all objectives successfully achieved. However, certain areas merit reflection in order to optimize future projects and learning processes.

Initially, time management was not executed in a well-structured manner. A significant amount of time was spent collecting and labeling new data in pursuit of high-quality datasets, rather than utilizing the provided datasets to train the classifier model. In hindsight, a more balanced approach could have been adopted by using the teacher-provided datasets to initiate the project and making modifications as needed.

Moreover, an excessive amount of time was dedicated to the literature review and acquiring knowledge about advanced language models. A more efficient approach would have entailed integrating literature reading, knowledge acquisition, and project development. This could involve setting initial project goals based on available literature and incrementally expanding those goals as more advanced literature is collected and understood.

At times, the pursuit of ambitious ideas led to the implementation of complex, albeit interesting, methods. Despite the enthusiasm for these methods, they were either not a good fit for the project or difficult to execute, ultimately slowing progress and increasing stress. It is crucial to carefully evaluate the practicality and relevance of techniques before investing substantial time and energy in them.

Nonetheless, the project's outcome was successful, as it required the implementation of at least three classifier models, but eight classifier models were completed. In-depth experiments were conducted to compare the performance of these models and investigate the impact of various preprocessing techniques, dataset balance, and size on classification performance. The extensive efforts invested in collecting a new dataset contributed to improved model performance, making the endeavor worthwhile.

This project has provided valuable experience in working with advanced language models and reinforced essential skills. Notably, literature review speed, programming ability, and time management skills have significantly improved. Reflecting on these experiences will contribute to a more efficient and effective approach to future projects and academic endeavors.

6.3 Limitations

Even as I perfected my project, I inevitably encountered some limitations, which required us to reflect and pave the way for future research exploration:

1. The performance of machine learning models, in particular, is heavily dependent on the quality and quantity of training data. My dataset only contains 4000 user reviews, even with data augmentation it's only 8000. The project faced limitations in available labeled data compared to large corporate databases, which may have affected the performance of some models.

2. The user reviews in the training dataset used in this project are the top 20 apps in Google Play. These apps don't cover all app categories, and may not perform well in user reviews for some uncommon categories of apps.

3. In real life, a single review might mention multiple aspects of a product. This project will only classify a user comment into only one label.

4. Advanced language models, while powerful, tend to be less interpretable than rule-based and traditional machine learning models. This project did not delve deeply into the interpretability aspect, which could be a consideration for future work, especially in applications where understanding the reasoning behind classifications is critical. While these models have demonstrated impressive capabilities in natural language understanding, they are also known to be black boxes, which makes it difficult to understand the reasoning behind their classifications and predictions.

5. Although we used standard evaluation metrics to compare the performance of the models, there may be other relevant measures that could provide additional insights. For instance, a more comprehensive analysis could include metrics such as precision-recall trade-offs, time is taken for training and inference, or the robustness of models to adversarial attacks.

By acknowledging and addressing these reflections and limitations, future work can build upon the insights gained from this project to further improve user review classification techniques and better understand the trade-offs involved in selecting the most suitable models for different applications.

Conclusion

7.1 Achievements

This project aimed to compare the performance of eight different user review classification models across three categories: rule-based models, traditional machine learning models, and advanced language models. Through extensive evaluation and analysis, we have observed the strengths and weaknesses of each model, providing valuable insights into the most suitable approaches for various scenarios.

A crucial aspect at the beginning of this project is the collection of a high-quality dataset. In this study, our facilitator and I manually annotated a raw dataset of user reviews. To ensure the annotation process was carefully designed and executed, annotators were provided with well-defined labeling guidelines and appropriate training. Maintaining the quality of annotations is critical for achieving accurate and effective classifier models.

Rule-based models, such as keyword matching, offer a straightforward and interpretable method for classifying user reviews. Although they can efficiently accomplish simple tasks and require minimal training data, they cannot capture complex linguistic patterns and semantic relationships. Consequently, these models may struggle with more intricate classification tasks, resulting in suboptimal performance (Accuracy: 30%).

Traditional machine learning models, including KNN, SVM, and Naive Bayes, provide a more robust solution for review classification than rule-based models (accuracy: 40% to 85%). Capable of handling noisy data and considering relationships between features, these models yield improved performance. However, they necessitate a considerable amount of labeled training data and may struggle to capture the nuances of natural language, particularly when addressing reviews containing sarcasm or implicit meaning.

Advanced language models, such as BERT, RoBERTa, and XLNet, have demonstrated superior performance in user review classification tasks (accuracy: 80% to 93%). Although the GPT2 model performed well (accuracy: 60% to 73%), it did not match the performance of the other three advanced language models and was even outperformed by some traditional machine learning models. The exceptional performance of other advanced language models can be attributed to their ability to leverage pre-training on large-scale corpora, enabling them to capture intricate linguistic patterns and contextual information. As a result, these models excel in handling diverse and complex classification tasks, even when faced with subtle language cues or minimal training data. However, they are computationally intensive and may require significant resources for fine-tuning and deployment.

In light of these findings, it is essential to carefully consider the specific requirements and constraints of a given classification task when selecting an appropriate model. For simpler tasks with clear-cut rules, rule-based models may suffice. Traditional machine learning models may be more suitable for tasks with abundant training data and less complex linguistic patterns. For the most challenging tasks involving intricate language nuances and limited training data, advanced language models are the most promising choice.

In conclusion, the choice of model will depend on factors such as computational resources, available training data, and the complexity of the classification task. By leveraging the insights gained through this project, practitioners can make informed decisions when selecting the best model for their user review classification needs, leading to more accurate and useful results in real-world applications.

7.2 Future Work

Building on the findings and learnings from this project, there are several avenues for future work to further enhance user review classification techniques and deepen our understanding of the strengths and weaknesses of various models. Some potential directions include:

1. Larger, more diverse, and more high-quality labeled datasets are collected, which can help improve model performance.

2. As advanced language models become more prevalent, their interpretability and explainability gain importance. Future research could focus on developing techniques to extract insights from these models, enabling users to understand the rationale behind classifications and fostering trust in the model's predictions.

3. Developing a multi-label classification task, a single review may mention multiple aspects of a product. In this case, assigning the comment to multiple labels will more accurately describe the content of the comment.

4. Investigating methods for reducing the computational and resource requirements of advanced language models, such as model distillation or pruning, can help make these powerful models more accessible and practical for deployment in a wider range of settings.

5. Combining the strengths of different model categories, such as advanced language models and traditional machine learning models, through ensemble techniques or model stacking could lead to improved classification performance. Future research could investigate optimal ways to combine models and exploit their complementary strengths.

6. Developing comprehensive evaluation frameworks and benchmarks for user review classification can help standardize comparisons between models and establish best practices for the field. This includes not only performance metrics but also considerations such as computational efficiency, robustness to adversarial attacks, and interpretability.

7. Ensuring that user review classification models are ethically designed and free from biases is crucial. Future work could focus on designing models that are fair, transparent, and accountable, as well as developing techniques for detecting and mitigating biases in the training data and model predictions.

By pursuing these avenues of future work, researchers and practitioners can continue to advance the field of user review classification, developing increasingly accurate, efficient, and interpretable models to better serve the needs of various applications and stakeholders.

Word Count: 14756

References

[1] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pages 281–290. IEEE, 2015.

[2] Harry Budi Santoso and Martin Schrepp. Importance of user experience aspects for different software product categories. In *User Science and Engineering: 5th International Conference, i-USEr 2018, Puchong, Malaysia, August 28–30, 2018, Proceedings 5*, pages 231–241. Springer, 2018.

[3] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[4] Junwen Lu, Xinrong Zhan, Xintao Zhan, and Lihui Shi. A book-influence-evaluation method based on user ratings of e-commerce platform. *Electronics*, 11(24):4198, 2022.

[5] Roshan Fernandes and GL Rio D'Souza. Semantic analysis of reviews provided by mobile web services using rule based and supervised machine learning techniques. *International Journal of Applied Engineering Research*, 12(22):12637–12644, 2017.

[6] Aagam Shah, Komal Kothari, Umang Thakkar, and Satvik Khara. User review classification and star rating prediction by sentimental analysis and machine learning classifiers. In *Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2018*, pages 279–288. Springer, 2020.

[7] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21:311–331, 2016.

[8] John Jianjun Zhu, Yung-Chun Chang, Chih-Hao Ku, Stella Yiyan Li, and Chi-Jen Chen. Online critical review classification in response strategy and service provider rating: Algorithms from heuristic processing, sentiment analysis to deep learning. *Journal of Business Research*, 129:860–877, 2021.

[9] Zhixue Zhao, Ziqi Zhang, and Frank Hopfgartner. A comparative study of using pre-trained language models for toxic comment classification. In *Companion Proceedings of the Web Conference 2021*, pages 500–507, 2021.

[10] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*, 2002.

[11] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[12] Pratik Thakor and Sreela Sasi. Ontology-based sentiment analysis process for social media content. *Procedia Computer Science*, 53:199–207, 2015.

[13] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. *Mining text data*, pages 163–222, 2012.

[14] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[15] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

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

[17] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[21] Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. *arXiv preprint arXiv:1903.09722*, 2019.

[22] Xinsong Zhang, Pengshuai Li, and Hang Li. Ambert: A pre-trained language model with multi-grained tokenization. *arXiv preprint arXiv:2008.11869*, 2020.

[23] Ron Artstein. Inter-annotator agreement. *Handbook of linguistic annotation*, pages 297–313, 2017.

[24] Wael Etaiwi and Ghazi Naymat. The impact of applying different preprocessing steps on review spam detection. *Procedia computer science*, 113:273–279, 2017.

[25] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666. IEEE, 2003.

[26] Sheetal A Takale and Sushma S Nandgaonkar. Measuring semantic similarity between words using web documents. *International Journal of Advanced Computer Science and Applications*, 1(4), 2010.

[27] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[28] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. *Feature extraction: foundations and applications*, pages 1–25, 2006.

[29] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pages 986–996. Springer, 2003.

[30] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

[31] SVM Vishwanathan and M Narasimha Murty. Ssvm: a simple svm algorithm. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2393–2398. IEEE, 2002.

[32] Yi Sun, Yu Zheng, Chao Hao, and Hangping Qiu. Nsp-bert: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction. *arXiv preprint arXiv:2109.03564*, 2021.

[33] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[34] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[35] Jia-Hong Huang, Luka Murn, Marta Mrak, and Marcel Worring. Gpt2mvs: Generative pre-trained transformer-2 for multi-modal video summarization. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, pages 580–589, 2021.

[36] Jingsai Liang. Confusion matrix: Machine learning. *POGIL Activity Clearinghouse*, 3(4), 2022.

[37] Guy S Handelman, Hong Kuan Kok, Ronil V Chandra, Amir H Razavi, Shiwei Huang, Mark Brooks, Michael J Lee, and Hamed Asadi. Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods. *American Journal of Roentgenology*, 212(1):38–43, 2019.

[38] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813, 2019.

[39] Kozo Takayama, Mikito Fujikawa, and Tsuneji Nagai. Artificial neural network as a novel method to optimize pharmaceutical formulations. *Pharmaceutical research*, 16:1–6, 1999.

[40] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[41] Will Koehrsen. Overfitting vs. underfitting: A complete example. *Towards Data Science*, pages 1–12, 2018.

[42] Bayu Yudha Pratama and Riyanarto Sarno. Personality classification based on twitter text using naive bayes, knn and svm. In *2015 international conference on data and software engineering (ICoDSE)*, pages 170–174. IEEE, 2015.

[43] Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121. IEEE, 2020.

[44] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[45] Akrivi Krouska, Christos Troussas, and Maria Virvou. The effect of preprocessing techniques on twitter sentiment analysis. In *2016 7th international conference on information, intelligence, systems & applications (IISA)*, pages 1–5. IEEE, 2016.

[46] Nicola Burns, Yaxin Bi, Hui Wang, and Terry Anderson. Sentiment analysis of customer reviews: Balanced versus unbalanced datasets. In *Knowledge-Based and Intelligent Information and Engineering Systems: 15th International Conference, KES 2011, Kaiserslautern, Germany, September 12-14, 2011, Proceedings, Part I 15*, pages 161–170. Springer, 2011.

[47] Alhanoof Althnian, Duaa AlSaeed, Heyam Al-Baity, Amani Samha, Alanoud Bin Dris, Najla Alzakari, Afnan Abou Elwafa, and Heba Kurdi. Impact of dataset size on classification performance: an empirical evaluation in the medical domain. *Applied Sciences*, 11(2):796, 2021.