Student ID: 10756505

CLASSIFICATION OF APP REVIEWS FOR REQUIREMENTS ENGINEERING USING PRETRAINED LANGUAGE MODELS

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF BACHELOR OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2024

Pavel Ghazaryan

Supervised by Dr Liping Zhao

Department of Computer Science

Contents

Word Count: 14798

List of Tables

List of Figures

Abstract

In the rapidly evolving landscape of mobile applications, developers face the challenging task of efficiently processing and categorizing vast amounts of user-generated reviews to enhance app functionality and user satisfaction. This research project targets the advancement of app review classification by comparing the performance of Pretrained Language Models (PLMs) such as RoBERTa, ELECTRA, ERNIE, and BART. This study evaluates these models across various scenarios, including balanced multi-class, unbalanced multi-class, and multi-label datasets.

The study's findings reveal that BART was the top performer in balanced datasets with an accuracy of 93.43%, while RoBERTa delivered the highest accuracy of 95.33% in the unbalanced dataset scenario. In multi-label scenario, BART also achieved the highest F1-score of 95.97%, indicating its effectiveness in complex classification tasks. Although all models demonstrated competitive performance, with minimal variance between them in each scenario, ELECTRA stood out for its efficiency. Notably, ELECTRA was approximately 28% faster in training time compared to BART, enhancing its appeal for commercial applications where time and resource optimization are crucial. When considering both performance metrics and operational efficiency, ELECTRA outperformed the other models, making it a compelling choice for commercial applications where resource optimization is crucial.

This research conclusively demonstrates that while all tested Pretrained Language Models (PLMs) perform competitively in application review classification, ELECTRA stands out for its balance of performance and efficiency, proving especially advantageous in resource-constrained environments.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Liping Zhao, for her continuous support and guidance throughout the duration of this project. Her insights and expertise have been invaluable in shaping both the direction and the execution of this research. Dr. Zhao's encouragement and constructive feedback have greatly assisted me in refining my ideas and methodologies. I am profoundly grateful for her unwavering commitment and mentorship, which have significantly enriched my learning experience and research skills.

Chapter 1

Introduction

1.1 Context and Motivation

With the fast technical advances, the spread of smartphones continues to ascend with each passing year, leading to a corresponding upwelling in the development of apps. According to a recent Forbes report[21], the smartphone era has seen the creation of over 8 million mobile applications, highlighting the dynamic growth of this domain. Application developers are persistently focused on enhancing the quality and functionality of their products, a process significantly aided by the integration of consumer feedback through app reviews. This feedback mechanism allows developers to identify the shortcomings of their applications and to incorporate ideas and user requirements for future updates and improvements[38].

Furthermore, the analysis of app reviews provides benefits across various dimensions of software engineering. In the context of requirements engineering, for instance, app reviews offer valuable insights, enabling software engineers to identify emerging user demands for new app features[18, 7]. In the testing domain, reviews are invaluable for uncovering bugs and collecting user feedback on beta versions of applications[28, 11]. Moreover, the evaluation of app reviews during the product evolution phase is crucial for discerning and prioritizing user requests for modifications[41]. This many-sided utility of app reviews not only aids in enhancing current functionalities but also informs the strategic direction for future development, ensuring that applications evolve in alignment with user expectations and needs.

However, the vast amount of user feedback presents a substantial challenge, as a significant portion of it remains unaddressed and unacknowledged due to its overwhelming quantity[14]. This gap in the feedback loop has encouraged the emergence of application review analysis as a crucial area of interest within the software engineering domain. The objective of this classification is to enable software engineers to efficiently categorize user reviews, thereby augmenting their productivity and enhancing their capacity to meet consumer demands.

To address this challenge, plenty of studies have been undertaken, employing a wide range of methodologies to analyze and categorize user feedback. The effort to optimize application

review classification not only speaks to the technical challenges inherent in managing large datasets but also highlights the industry's commitment to the development of a responsive and user-centered development ethos. By exploiting the insights collected from categorized user feedback, developers can more precisely target areas for improvement, leading to a more dynamic and user-responsive application ecosystem. This ongoing dialogue between users and developers, facilitated by advanced analytical tools, underscores the interdependent relationship between technological innovation and consumer satisfaction in the digital age.

Automating app review classification begins by classifying them based on the kind of feedback provided.

This research will implement the classification scheme incorporating the labels of **"Bug Report"**, **"Feature Request"**, **"User Experience"** and **"Rating"**[28].

"Bug report" is user feedback identifying specific malfunctions or errors in the application, critical for developers to resolve software defects. These reports often detail unexpected behaviors or errors encountered during the use of the application, providing crucial information for developers to isolate and address software defects. Categorizing feedback as "Bug report" enables the direct routing of these insights to developers and quality assurance teams, focusing on product refinement and issue resolution.

"Feature request" comprises suggestions from users for new functionalities or enhancements, guiding product development to meet user expectations. Analyzing these submissions helps guide product development priorities and strategies, ensuring alignment with user expectations and market trends. Identifying reviews as "Feature request" allows for their systematic assignment to requirements analysts, who are tasked with integrating user-demanded enhancements into future development plans.

"User experience" captures users' perspectives on usability, design, and overall interaction with the application, highlighting areas for improvement in user interface and satisfaction. Leveraging user experience feedback is key to designing intuitive and compelling applications. "User experience" feedback can be strategically delivered to business strategists, operators and UI/UX teams, equipping them with valuable data to guide strategic planning for application updates and improvements.

"Rating" reflects a quantifiable evaluation of the application, typically expressed through a numerical score or stars, summarizing user satisfaction and perceived quality. These ratings serve as a direct indicator of the application's overall appeal and performance from the user's perspective, influencing potential users' decisions and perceptions.

Accuracy measures the proportion of correctly identified cases out of the total instances, whereas precision assesses the fraction of true positive results within the total number of predicted positive outcomes. Recall evaluates the percentage of true positives identified from all genuine positive cases, and the F1-score is calculated as the harmonic mean between precision and recall, offering a balance between the two. Additionally, training time is another important metric in natural language processing, reflecting the duration required to train a model, which

directly impacts its efficiency and feasibility for deployment in various scenarios. Utilizing these metrics facilitates a comprehensive assessment of classification models, thus improving insight into their advantages and limitations.

1.2 Aims and Objectives

This research aims to advance the frontiers of this field by conducting an in-depth comparative analysis of novel, cutting-edge Pretrained Language Models (PLMs), with the goal of identifying and proposing a new classification model with different architecture that outperforms the current state-of-the-art model, RoBERTa[14], in terms of aforementioned performance metrics. Recognizing the complexity and diversity of textual data, this study not only explores the multi-class classification scenario but also investigates the less explored but critically important multi-label classification setting. Through dataset construction, model exploration, and in-depth analysis, this research seeks to discover insights that could potentially redefine model selection criteria and application strategies in the domain of app review classification. At the same time this study aims to leverage the classification capabilities of ChatGPT due to lack of time, resources and datasets available for the research. ChatGPT will assist and speed up the labeling process for multi-label setting and for datasets that are unlabeled in the required format from Hugging Face[9].

Table 1.1 outlines the key aims and objectives for this research alongside their success criteria.

Table 1.1: Aims and Objectives

Aims and Objectives	**Success Criteria**
Objective 1: To obtain and construct 6 datasets of Balanced multi-class; 4 datasets of Unbalanced multi-class; 4 datasets of multi-label.	- Clean, filtered and accurately labeled datasets in a consistent format. - Efficient organization of datasets in designated folders by type. -All multi-label datasets and some multi-class datasets will be constructed through prompt-tuning ChatGPT for labeling available large dataset from Hugging Face[9]. The other datasets are provided and are already in the required format[16]. - Validation of ChatGPT-labeled reviews through random-sampling human inspection.
Objective 2: To fine-tune 4 PLMs(ELECTRA, ERNIE, BART, RoBERTa) using constructed datasets.	- Clean, well-documented code for each model, adhering to best practices. - Evaluation and comparison of models based on test metrics, training evaluation metrics, validation metrics, and training times. - Optimization of each model's performance through consistent hyperparameter fine-tuning.
Objective 3: To analyze and compare the performance of selected PLMs, focusing on key computational metrics, and to investigate the impact of dataset balance and size on classification performance.	- In-depth discussion of results, significant performance influencers, and architectural impacts. - Detailed comparative analysis presented through graphs and tables. - Reflection on findings' implications, potential errors, limitations, and future research directions. - Proposal of a possible new state-of-the-art model.

1.3 Report Structure

This report contains six 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 of the project, providing a detailed description of key concepts and steps undertaken in the development process.
- **Chapter 4** provides the results of the experiments and an in-depth discussion.
- **Chapter 5** compares the obtained results of this project with that of related work, provides possible validity threats and summarizes key research findings.
- **Chapter 6** concludes the report by reflecting on the study's limitations, discussing the learning outcomes, and offering insights into potential future directions for the project.

Chapter 2

Background and Related Work

This chapter presents the foundational concepts and traces the historical development of app review classification. It reviews the progression of methodologies from basic machine learning techniques to the more sophisticated Pretrained Language Models (PLMs), providing an overview of significant related works in the field. Furthermore, it identifies existing research gaps and details the intended contributions of this research project, setting the stage for subsequent discussions and analyses.

2.1 Background to App Review Classification

The origin of app review classification as a focused area of research can be traced back to the late 2000s and early 2010s, coinciding with the launch and subsequent rise of major digital marketplaces such as the Apple App Store in 2008 and the Google Play Store[17]. The increasing volume of user-generated reviews in these platforms necessitated the development of automated tools for efficiently processing and analyzing feedback. Initial efforts in this domain were relatively basic, leveraging simple NLP techniques like keyword spotting and basic sentiment analysis to sift through user reviews.

As the field matured, the advent of machine learning and deep learning methodologies indicated a new era of sophistication in app review classification. Early machine learning models employed algorithms such as Naive Bayes, Support Vector Machines, and Random Forests to classify reviews based on extracted textual features[28]. The introduction of deep learning further enhanced classification accuracy through the adoption of neural network-based approaches, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks. These advanced techniques facilitated a deeper understanding of the context and sentiment embedded in user reviews, enabling more accurate and nuanced analysis.

A notable advancement within this research field is the development of aspect-based sentiment analysis. This refined approach not only identifies the overall sentiment of a review but

also distinguishes. Specific aspects or features of the app being discussed, alongside their associated sentiments. This granular level of analysis provides developers with detailed insights into various facets of user feedback, enhancing their ability to address specific concerns and improve app functionality. In conclusion, the classification of app reviews through NLP techniques has evolved from basic sentiment analysis to encompass sophisticated machine learning and deep learning approaches. This evolution reflects the field's response to the growing complexity and volume of user-generated content. This progress is essential for developers and businesses seeking to collect actionable insights from user feedback, ultimately leading to improved user satisfaction and app quality.

2.1.1 Transformers

Transformers are a pivotal architecture in Natural Language Processing (NLP), revolutionizing language understanding, translation, and generation since their introduction in 2017 by Vaswani et al[40]. Unlike traditional models, transformers use an "attention" mechanism to process all parts of the input data simultaneously, enhancing performance and training efficiency. The core of transformers is the attention mechanism, especially self-attention, which evaluates the significance of each word relative to others in a sentence, producing Query, Key, and Value vectors for each word from the same embeddings. This allows the model to dynamically focus on different sentence parts, capturing syntactic and semantic relationships effectively. Key components include:

- **Scaled Dot-Product Attention:** Calculates attention scores through Queries and Keys, scaling down by the square root of the Key vectors' dimension to moderate gradient size, and uses these to weight Value vectors.

- **Multi-Head Attention:** Enhances focus on various sequence parts by running multiple attention mechanisms in parallel, each with unique weights, and combining their outputs.

- **Architecture:** Comprises an encoder and a decoder stack (Figure 2.1), each with layers of multi-head attention and feed-forward networks. The encoder transforms input symbols to continuous representations, while the decoder produces the output sequence, aided by another attention mechanism targeting the encoder's output.

- **Positional Encoding**: Adds to input embeddings to provide sequence order information, using sine and cosine functions to uniquely encode positions.

This architecture allows for direct and efficient modeling of complex relationships within the data, marking a significant advancement in NLP technology.

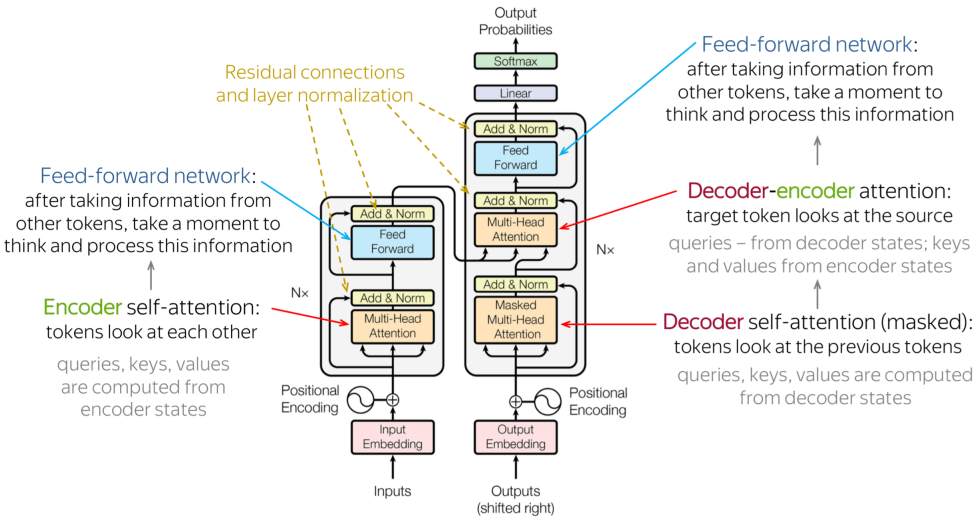

Figure 2.1: Transformer Architecture [16]

Impact on Pretrained Language Models (PLMs)

The transformer architecture has laid the groundwork for the development of advanced PLMs like BERT[8], GPT[33], RoBERTa[26], and others. By pretraining these models on large corpora with a self-supervised learning objective, such as predicting masked words (BERT), transformers learn a deep, contextualized representation of language. These representations can then be fine-tuned on a wide array of downstream NLP tasks, from text classification and sentiment analysis to question answering and summarization, achieving state-of-the-art performance across benchmarks.

In summary, the transformer model, with its innovative use of attention mechanisms, has revolutionized NLP, offering a flexible and powerful framework for understanding and generating human language. Its influence extends beyond NLP, inspiring new models in other domains of AI, signifying its foundational importance in the advancement of deep learning technologies.

2.1.2 Pretrained Language Models

Pretrained Language Models (PLMs) are advanced neural networks trained on large text corpora to learn language representations before being fine-tuned for specific NLP tasks. These models grasp the complexities of language, including syntax, semantics, and common-sense knowledge, enabling them to excel in tasks like text classification, question answering, and text generation. The evolution of PLMs began with word embeddings like Word2Vec[29], but transformative growth came with transformer-based models like BERT in 2018[8]. BERT introduced bidirectional context understanding, significantly improving over unidirectional models. Key Aspects of PLMs include:

- **Transformer Architecture:** Utilizing self-attention to assess the relevance of words

within sentences, capturing complex language patterns.

- **Large-scale Pretraining:** Training on extensive web-sourced datasets to learn language patterns without specific task orientation.
- **Fine-tuning for Tasks:** After pretraining, models are fine-tuned with task-specific data to specialize in particular NLP tasks.

Recent advancements in PLMs aim to improve efficiency (e.g., ALBERT[23]), incorporate adversarial training (e.g., ELECTRA[6]), and integrate external knowledge (e.g., ERNIE[44]). Generative models like GPT[33] focus on text generation by predicting the next word in a sequence. BART[24] combines the encoder-decoder structure with bidirectional training, excelling in text generation and comprehension tasks through innovative text corruption and reconstruction training methods. These developments have significantly enhanced PLMs' performance and efficiency across diverse NLP applications.

Pretrained Language Models have revolutionized NLP by providing a foundation upon which complex language understanding and generation tasks can be performed with unprecedented effectiveness. As research in this area continues to advance, we are likely to see PLMs become even more efficient, versatile, and capable of nuanced understanding and interaction with human language. The ongoing developments in this field promise to further bridge the gap between human and machine communication, opening up new possibilities for applications and technologies that leverage natural language.

2.2 Related Work

2.2.1 Traditional Machine Learning Approaches

Traditional machine learning techniques have long been the basis of automated user review classification, employing algorithms like Naive Bayes, Decision Trees, and Support Vector Machines (SVM) to categorize feedback into distinct themes such as feature requests, bug reports, and user experiences. These studies explore the effectiveness and limitations of various traditional approaches in extracting valuable insights from user-generated content.

Maalej et al.[28] in their paper implement multiple probabilistic approaches such as Naive Bayes Decision Trees, and MaxEnt, for classifying user reviews in four distinct labels, namely: feature request, bug report, user experience and rating. They have arrived to a conclusion that solely using metadata results in poor classification results while combining it with other NLP techniques such as BoW(Bag of Words approach), it produces an output of Precision of around 70-95% and Recall of around 80-90%. Their findings also indicate that multiple binary classifiers outperform a single multi-class classifier.

In another study conducted by Stanik et al.[38] they have classified app reviews into similar three categories: problem reports, inquiries and irrelevant comments. They have implemented

traditional Machine Learning algorithms like TF-IDF vectorization, and Deep Learning techniques such as Convolutional Neural Network with varying parameter selection. Their results show that traditional machine learning algorithms, in the presence of large enough datasets, produces results which are comparable to Deep Learning algorithms.

Moghaddam[30] conducted an analysis on 50,000 reviews from the eBay AppStore in 2015, applying Part of Speech (PoS) Tagging to detect patterns within user feedback. Following this, sentiment analysis, Support Vector Machine (SVM), and LDA feature reduction were utilized for the categorization of reviews into feature requests, bug reports, and user experience. This approach achieved an 88% precision rate in categorizing reviews.

Liang P et al.[27] introduce an automated classification approach that aligns user reviews with an NFR framework, categorizing them into four types of NFRs: reliability, usability, portability, and performance, alongside Functional Requirements (FRs) and miscellaneous feedback. By integrating four classification techniques—Bag of Words (BoW), TF-IDF, CHI2, and a novel approach, Augmented User Review-Based Bag of Words (AUR-BoW)—with three machine learning algorithms (Naive Bayes, J48, and Bagging), this study evaluates the effectiveness of various combinations in classifying user reviews. Their obtained results suggest that the AUR-BoW technique paired with the Bagging algorithm yielded the most favorable outcomes, achieving a precision of 71.4%, a recall of 72.3%, and an F-measure of 71.8%. They have concluded that augmented user reviews can lead to better classification results, and the machine learning algorithm Bagging is more suitable for NFRs classification from user reviews than Naive Bayes and J48.

2.2.2 Advancements in Review Classification Through Deep Learning

The introduction of deep learning has revolutionized the field of text analysis, offering significant advancements in the classification of user reviews. Through the utilization of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, researchers have achieved remarkable improvements in precision and recall, demonstrating the potential of deep learning models to understand and categorize complex user feedback more effectively than traditional methods.

Aslam et al.[2] analyzed 4,400 reviews, introducing a classification system based on Convolutional Neural Networks (CNN) and Deep Learning that achieved an F-measure of 94.7% in review classification. Qiao and Wang[32] developed a Deep Learning model with LSTM architecture for classifying Apple AppStore reviews, based on an analysis of 18,261,515 comments from 4,602 games. The reviews were categorized by 26 bachelor students into three classes: feature request, bug report, and miscellaneous, with the model achieving an F-measure of 76.9% and a Recall rate of 76.8%.

2.2.3 The Superiority of Pretrained Language Models in App Review Analysis

Pretrained Language Models (PLMs) such as BERT, RoBERTa and ALBERT represent a significant leap forward in natural language processing, providing robust frameworks for tackling a wide array of NLP challenges. This subsection delves into the application of PLMs in classifying application reviews, highlighting their superior performance across various experimental setups and their ability to outperform both traditional and deep learning models in understanding nuanced user feedback.

In the study conducted by Mohammad et al.[14], an evaluation of Pretrained Language Models (PLMs) for the classification of issues in application reviews across diverse scenarios is presented, comparing their performance with that of traditional models. This research explores PLMs such as RoBERTa, BERT and ALBERT in various experimental contexts like binary vs multi-class setting, zero-shot learning, multi-task learning, and multi-resource settings, providing a comprehensive overview of the applicability of PLMs in different NLP challenges. Their results have shown that PLMs perform better in terms of all performance metrics than previous state-of-the-art approaches in all mentioned settings except in multi-resource environment. They have obtained 98% and 92% micro- and macro-average F1-score for their largest dataset, 71% of F1-score in zero-shot setting, and 93% and 92% F1-score in multi-task and multi-resource settings, respectively.

2.2.4 Innovative Computational Frameworks for User Review Mining and Summarization

Ning Chen et al.[5] has designed a computational frame- work for App Review Mining: AR-Miner which performs comprehensive analytics from raw user reviews by extracting informative user reviews through filtering noisy and irrelevant ones, then grouping the informative reviews automatically using topic modeling, further prioritizing the informative reviews by an effective review ranking framework, and finally presenting the groups of most “informative” reviews via an intuitive visualization approach. They have concluded that AR-Miner is effective, efficient and promising for app developers.

Johann et al.[18] in their study introduced a tool named SAFE in 2017, aimed at feature extraction from Play Store reviews. Through the manual identification of 18 PoS and 5 sentence patterns, and after analyzing feedback from 10 apps, they reported a Precision of 55% and a Recall of 43%. Scalabrino et al.[36] applied a machine learning strategy along with a tool called CLAP for the classification and prioritization of reviews, which are essential for future development efforts, identifying categories like operational bug reports, new feature suggestions, and nonfunctional requirements. They achieved a precision of 87% by leveraging the random tree and DBScan algorithms, along with the AR-Miner database.

2.2.5 Leveraging Sentiment Analysis for Comprehensive Review Classification and Summarization

Another interesting study by Panichella et al.[31] presents a taxonomy to classify app reviews into categories of “feature request”, “opinion asking”, “problem discovery”, “solution proposal””, “information seeking”, “information giving”. They have proposed an approach that merges three techniques: Natural Language Processing, Text Analysis and Sentiment Analysis to automatically classify app reviews into the proposed categories. They have shown that the combined use of these techniques allows to achieve better results (a precision of 75% and a recall of 74%) than results obtained using each technique individually (precision of 70% and a recall of 67%).

Gu and Kim[13] in their study have presented a review summarization framework called SUR-Miner which instead of a bags-of-words assumption, classifies reviews into five categories and extracts aspects in sentences which include evaluation of aspect using a pattern-based parser. Their evaluation on 17 popular apps shows that SUR-Miner summarizes reviews with decent accuracy and clarity, scoring an average F1-score of 81%.

2.3 Research Gaps and Intended Project Contributions

2.3.1 Research Gaps

Multi-label Classification of App Reviews

Existing research primarily focuses on binary or multi-class classification of app reviews, overlooking the complexity of multi-label scenarios where reviews often touch on multiple topics simultaneously. This gap highlights the limitations of current models, including PLMs and traditional machine learning approaches, to fully capture the nature of user feedback. Addressing this requires the development of datasets specifically designed for multi-label classification to enhance the analysis precision of user sentiments and demands. For example, a single review might simultaneously offer praise for app functionality, suggest a new feature, and report a minor bug.

The current methodologies, including both cutting-edge PLMs and traditional machine learning algorithms, fall short in capturing this complexity. To bridge this gap, there’s a crucial need for developing and leveraging datasets tailored for multi-label classification. Such datasets, ideally formatted in a one-hot-encoding scheme, would significantly refine the granularity and accuracy of review analysis, offering a more comprehensive understanding of user sentiments and preferences.

Scarcity of Labeled Review Datasets

The field of app review analysis lacks of publicly available, labeled review datasets. These datasets are vital for training, testing, and validating machine learning models. The current scarcity of such data, as highlighted by the work of Mohammad et al., coupled with the time-consuming and potentially inconsistent manual annotation process, limits the development of new models and the reproducibility of research findings. Addressing this critical gap by creating, curating, and circulating comprehensive, labeled review datasets would constitute a substantial leap forward for the field, facilitating more effective model development and a richer understanding of user feedback dynamics.

2.3.2 Intended Research Contributions

Advancing Multi-Label Classification in App Review Analysis

The project proposes the development of a novel dataset, structured to support multi-label classification. This dataset will serve as a foundation for training and evaluating advanced machine learning models, including both novel PLMs and refined versions of traditional algorithms. By acknowledging that app reviews can simultaneously belong to multiple categories, this approach seeks to provide deeper insights into user feedback.

Leveraging Novel Pretrained Language Models for Enhanced Classification Performance

Exploring and applying recent advancements in PLMs, such as ELECTRA, ERNIE, and BART, this project challenges the current state-of-the-art, represented by the RoBERTa model. These novel PLMs, with their unique architectural innovations, are hypothesized to offer superior performance in app review classification tasks. By setting new benchmarks, this contribution aims to improve the precision and efficiency of user feedback analysis.

These contributions will offer both theoretical and practical benefits, enhancing the ability of developers to leverage user feedback for continuous improvement of their applications.

Chapter 3

Research Methodology

This chapter describes the research methodology adopted in this project. The key steps of this methodology are described in detail in the following sections.

- Data Collection and Processing
- Pretrained Language Models
- Hyperparameter Tuning
- Evaluation Metrics
- Programming Libraries and Environment
- General Design

3.1 Data Collection and Processing

This section outlines the methodologies employed for data collection in this research project. It details the sources of datasets, the rationale behind their selection, and the processes involved in preparing these datasets for analysis.

3.1.1 Dataset Overview

The research utilizes two principal datasets:

1. **Kaggle Dataset via Tianpeng Huang:** This dataset was curated by Tianpeng Huang, a previous student at the University of Manchester, who manually labeled 8,000 reviews from Kaggle for a similar research project using the same labeling scheme (“Bug Report”, “Feature Request”, “Rating“, “User Experience”). The reviews are formatted in a multi-class structure to facilitate analysis[16].

2. **Hugging Face Dataset:** Sourced from the Hugging Face platform, this dataset comprises 280,000 unlabeled reviews from various apps available on Google Play Store, including popular apps like Facebook and Twitter[9].

3.1.2 Data Labeling and ChatGPT

Given the existing labeling of the Kaggle dataset (provided by Tianpeng Huang), the focus shifted toward preparing the Hugging Face dataset for analysis. Due to constraints in resources and time, the Hugging Face dataset underwent a labeling process using ChatGPT's capabilities to assign multi-class and multi-label (one-hot-encoding) formats to the reviews. The application of ChatGPT as an automated labeling tool has been explored, focusing on prompt engineering and the development of solutions to improve labeling accuracy within the Hugging Face dataset context.

Initially, employing ChatGPT for labeling presented a substantial challenge. A basic prompt led to ChatGPT generating classification code implementing a simple lexicon-based approach (Figure 3.1), revealing inconsistencies and underscoring the necessity for a more sophisticated prompt strategy to effectively utilize ChatGPT's labeling potential.

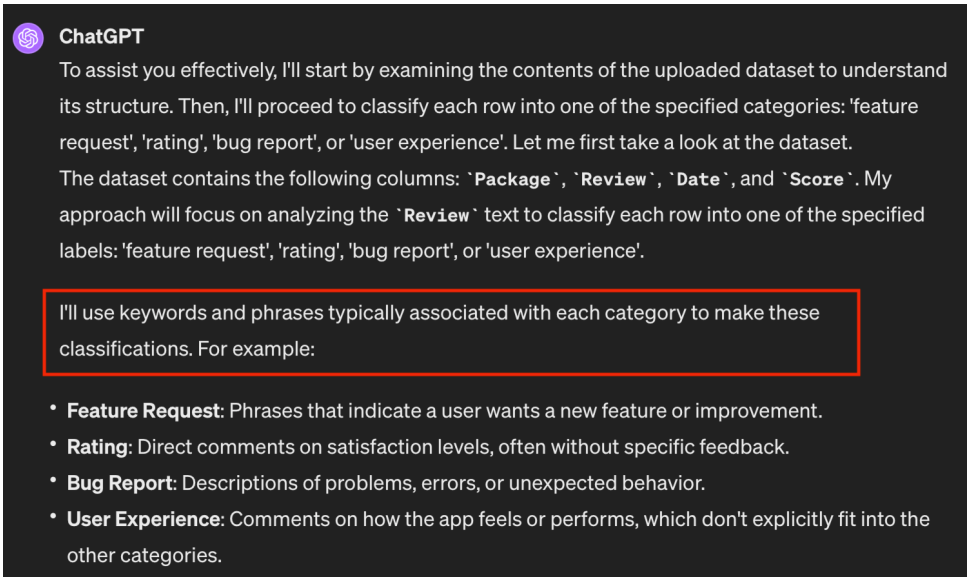

Figure 3.1: ChatGPT using heuristics/lexicon based approach

To address this, a more refined prompt was constructed, directly requesting ChatGPT to classify each review using its own decision-making rather than writing a simple heuristic code and executing it (Figure 3.2). This strategy was used for multi-class format as well as for multi-label format.

In the end 280,000 reviews labeled in multi-class and multi-label formats were obtained. From this, 32,000 of the longest reviews were selected to ensure a focus on quality and detailed feedback. A validation step was incorporated where 3% of the ChatGPT-labeled dataset was

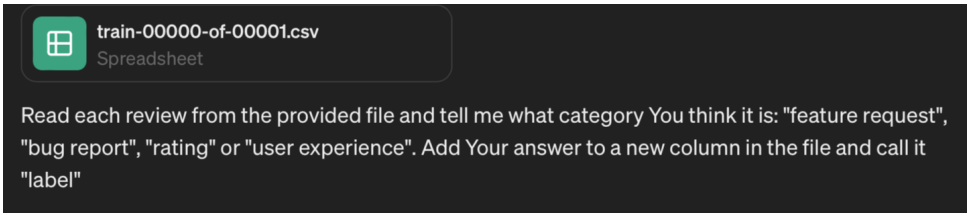

Figure 3.2: ChatGPT using heuristics/lexicon based approach

randomly reviewed to ensure the reliability of the automated process. This validation was essential to confirm ChatGPT's labeling reliability, permitting adjustments to the prompt strategy to enhance dataset labeling quality further.

It is vital to understand that the implementation of ChatGPT allows to experiment with larger dataset sizes and multi-label classification settings which would not be possible otherwise due to limited time and dataset availability.

3.1.3 Comparative Dataset Construction

The datasets were further segmented to facilitate comparative analysis:

- From Tianpeng Huang's Dataset: Three subsets were created:
  - 4,000 Balanced[1] reviews
  - 4,000 Unbalanced
  - 8,000 Balanced
- From the Hugging Face Dataset (labeled by ChatGPT): Ten subsets were created:
  - 4,000**;** 8,000**;** 20,000**;** 32,000 Balanced
  - 4,000**;** 8,000**;** 16,000 Unbalanced
  - 4,000**;** 8,000**;** 16,000 Multi-Label (no balancing possible)

These new datasets allowed to test the models in various settings, thus assisting to conduct a thorough comparative analysis of the models.

An overview of the datasets is shown in Table 3.1, showcasing the higher quality of the Hugging Face datasets[9] as the average word count for those is higher. This means that the Hugging Face dataset doesn't suffer from short and vague reviews which have no detailed content and would not be beneficial for the developers.

[1]Each category has equal amount of reviews

Table 3.1: Datasets Summary

Dataset	Bug Report	Feature Request	Rating	User Experience	Average Word Count
ds_balanced_8000	2000	2000	2000	2000	24.33625
ds_balanced_4000	1000	1000	1000	1000	22.945
ds_gpt_balanced_4000	1000	1000	1000	1000	97.5879
ds_gpt_balanced_32000	8000	8000	8000	8000	47.6698
ds_gpt_balanced_20000	5000	5000	5000	5000	59.2125
ds_gpt_balanced_8000	2000	2000	2000	2000	81.4575
ds_gpt_unbalanced_16000	2000	3000	5000	6000	49.7694
ds_gpt_unbalanced_4000	500	750	1250	1500	50.3848
ds_unbalanced_4000	1257	1410	740	593	25.4218
ds_gpt_unbalanced_8000	1000	1500	2500	3000	50.093
ds_gpt_multi_label_16000	4524	5275	3915	10223	65.6498
ds_gpt_multi_label_8000	2633	3048	2319	5437	81.781
ds_gpt_multi_label_4000	1429	1711	1330	2855	97.5368

3.1.4 Data Preprocessing

In a vast amount of NLP projects, text preprocessing is an essential step that follows data collection. This step typically includes methods like stemming, lemmatization, case-folding, and stop-words removal. However, in this project, it was decided not to apply any preprocessing, as previous research papers, such as Tianpeng Huang's project[16], have demonstrated that the performance of PLMs decreases with preprocessing.

Pretrained Language Models are designed to grasp the nuances of language by training on large, unaltered text corpora. Traditional preprocessing techniques can weaken their performance by stripping away contextual clues and changing sentence structure. These models rely on the full context of sentences, including the form and interplay of all words, to generate accurate word embeddings and understand the text. Since PLMs are adept at handling the natural "noise" in language, including variations in word forms and the presence of common words, preprocessing that modifies the original text structure can lead to a mismatch with the model's training data, thereby reducing its ability to accurately interpret and analyze the text. Essentially, preserving the richness and complexity of natural language input is key to leveraging the full capabilities of PLMs.

Instead of applying text preprocessing, data cleaning was performed, ensuring the removal of non-ASCII characters, that could make the text unreadable for the models.

3.2 Pretrained Language Models

This sections explains in-depth the PLMs used in this research project, covering their architectures and unique properties.

3.2.1 RoBERTa: A Robustly Optimized BERT Pretraining Approach

RoBERTa (Robustly optimized BERT approach)[26] represents a significant advancement in the domain of Natural Language Processing (NLP) through the enhancement of the BERT (Bidirectional Encoder Representations from Transformers)[8] model. Developed by researchers at Facebook AI, RoBERTa redefines the pretraining process of BERT by optimizing its key hyperparameters and training data size, which results in improved performance across a wide range of NLP tasks. BERT revolutionized the NLP domain by introducing a powerful pretrained model capable of understanding complex language nuances. However, its training regime left room for optimization. RoBERTa emerges as an enhanced variant of BERT, considering the effects of hyperparameters and training data size. By optimizing these aspects, RoBERTa substantially advances the state-of-the-art on several NLP benchmarks, demonstrating the importance of novel pre-training methodologies.

RoBERTa Architecture

RoBERTa follows the transformer architecture[40], maintaining the foundational design of BERT. It leverages a multi-layer bidirectional transformer encoder, consisting of multiple attention heads for processing sequences of text. The model is distinguished not by architectural modifications but by optimizations in its pretraining procedure.

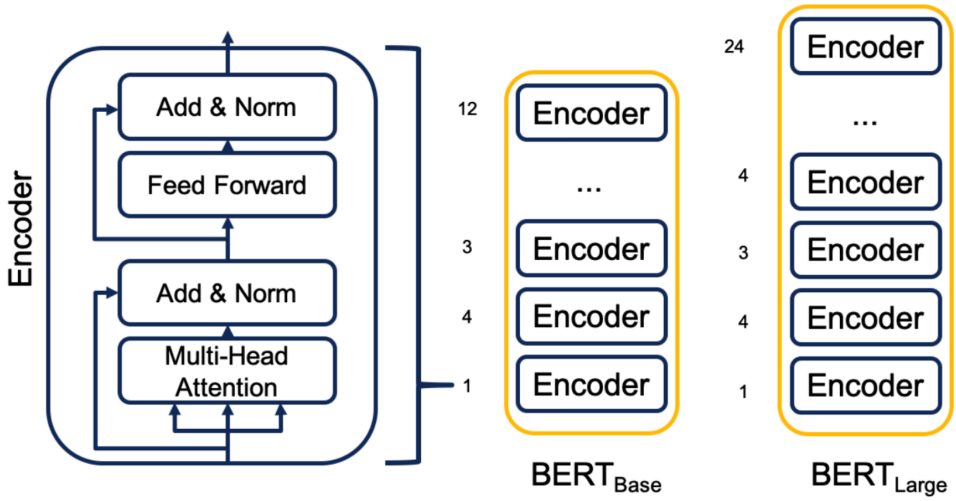

Figure 3.3: BERT Architecture[16]

In order to understand the innovations which RoBERTa brings, it is essential to learn about

two key concepts persistent in BERT: **Masked Language Modeling(MLM)** and **Next Sentence Prediction(NSP)**. Both were introduced in the original BERT paper[8].

Masked Language Modeling (MLM) is a training technique used in PLMs where some of the words in the input sentences are randomly masked and the model is trained to predict these masked words. This approach enables the model to learn a deep understanding of language context and structure, as it forces the model to infer the masked word based solely on its surrounding words. In the context of BERT, MLM is pivotal as it allows the model to leverage bidirectional context, unlike traditional left-to-right or right-to-left language models. By predicting words in this masked fashion, BERT learns comprehensive representations for words based on the full context of sentences, which significantly enhances its ability to understand and generate language.

Next Sentence Prediction (NSP) is another training methodology utilized in the initial design of BERT to further enhance its understanding of sentence relationships. In NSP, the model is presented with pairs of sentences and is tasked with predicting whether the second sentence in a pair is the logical and consecutive continuation of the first one. This binary classification task helps the model grasp the coherence and flow of discourse, teaching it to understand how ideas are connected across sentences. The inclusion of NSP in BERT's training process was aimed at improving its performance on tasks that require an understanding of the relationship between sentences, such as question answering and natural language inference.

The innovations of RoBERTa lie in its pre-training modifications rather than architectural alterations:

Dynamic Masking

Unlike BERT, which applies a static mask to the training data before the pretraining process begins, RoBERTa employs dynamic masking. This approach generates the masking pattern on the fly during training, allowing the model to learn from a more varied set of masked tokens. In simpler terms, the selection of tokens for masking is made on a random basis for each epoch, instead of being predetermined at the start of the pre-training phase. By adopting this flexible masking strategy, the model is encouraged to develop a broader and more resilient understanding of the input data, since it encounters a more varied array of contexts and combinations of tokens.

Removal of the Next Sentence Prediction (NSP) Task

RoBERTa eliminates the NSP task, a pretraining objective used in BERT, after identifying it as unnecessary for achieving high performance on downstream tasks. By discarding NSP, RoBERTa reallocates computational resources towards more impactful pre-training tasks, like improved masked language modeling. This shift enables a more efficient learning process, focusing on aspects that more directly contribute to the model's understanding of language.

Scaling Batch Sizes

RoBERTa employs larger mini-batch sizes during training compared to BERT. This approach is crucial for stabilizing the gradient updates and achieving more efficient optimization, especially given the increased data volume and model complexity. Larger mini-batches allow for more data to be processed in parallel, reducing the training time and improving the model's ability to generalize from the training data. This results in enhanced performance on downstream tasks, as the model benefits from more stable and effective learning dynamics.

Training Data Volume

RoBERTa significantly increases the size of the dataset used for pretraining. It leverages larger and more diverse datasets, including BOOKCORPUS[45], English Wikipedia[10], CC-News[43], OpenWebText[12], and Stories[26], culminating in a dataset that is substantially larger than that used by BERT.

Performance and Benchmarks

RoBERTa achieves remarkable success on multiple NLP benchmarks, surpassing its predecessors, including BERT and other models. Its performance on GLUE[42], SQuaD[34], RACE[22], and other benchmarks illustrates the model's ability to comprehend and analyze complex textual information effectively. RoBERTa demonstrates the continuous quest for improvement in the field of NLP. By refining the pretraining process of BERT, it achieves superior performance across a broad spectrum of language tasks[26].

3.2.2 ELECTRA

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is a novel pre-trained language model introduced by researchers at Google in 2020[6]. ELECTRA challenges the conventional pre-training approach of models like BERT[8] and GPT[33] by introducing a more sample-efficient pre-training method. Instead of predicting missing words or the next word in a sequence, ELECTRA focuses on distinguishing between "real" and "fake" words within the input text. This approach is grounded in the idea that learning to detect subtle differences between genuine and artificially replaced words can lead to a more nuanced understanding of language.

ELECTRA Architecture

ELECTRA's architecture is built upon the Transformer model, similar to other PLMs. However, the core distinction lies in its pre-training objective. Unlike BERT (and other BERT based PLMs), which randomly masks tokens and predicts them, ELECTRA introduces a sample-efficient generator-discriminator framework:

1. **Generator (G)**

   A smaller transformer model tasked with masked language modeling (MLM). For a given input sentence where some tokens are masked, the generator predicts what the masked tokens should be. However, unlike BERT, these predictions are used to create "fake" versions of the original tokens in the input sentence, not directly for learning representations.

2. **Discriminator (D)**

   A larger transformer model that takes the output from the Generator (where some tokens have been replaced with the generator's predictions) and tries to classify each token as "real" (the original token) or "fake" (a token replaced by the generator). The discriminator is the main model that learns the rich language representations.

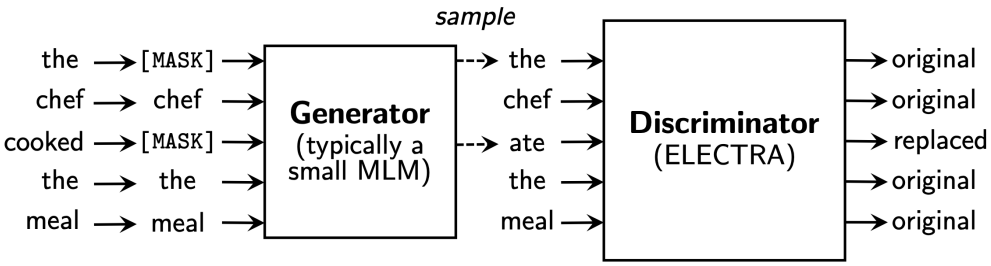

Figure 3.4: ELECTRA Architecture[6]

Training Objective and Methodology

The key innovation of ELECTRA is the training objective: "replaced token detection" task. Instead of predicting the original words for masked positions (MLM) or predicting the next word in a sequence (like in GPT[33]), ELECTRA trains the discriminator to detect whether each word in the text was replaced by a generated word by the generator. This approach effectively turns the pre-training process into a binary classification problem for each token, thereby utilizing the entire input sequence for training rather than just the masked portions. The generator and discriminator are trained together in a tandem process. As the generator improves at producing plausible token replacements, the discriminator simultaneously improves its ability to distinguish "real" from "fake" tokens, becoming better at understanding the nuances of language.

Efficiency and Effectiveness

ELECTRA's efficiency comes from its ability to learn from the entire input sequence and its innovative use of a smaller generator model. This allows for faster training times and reduced computational resource requirements compared to traditional MLM-based approaches.

- **Sample Efficiency**

  ELECTRA is more sample-efficient compared to models like BERT. BERT learns from only the 15% of tokens that are masked and predicted, while ELECTRA learns from the entire input sequence, as the discriminator predicts every token. This means ELECTRA can learn more from the same amount of data, making it faster and cheaper to train while achieving competitive or superior performance[6].

- **Scalability and Adaptability**

  Despite its efficiency, ELECTRA scales well with model size and data, maintaining its advantages over traditional pre-training methods as the computational budget increases. This adaptability makes it suitable for a wide range of tasks and languages, including those with limited data[6].

Performance and Benchmarks

ELECTRA demonstrates state-of-the-art performance on a variety of NLP benchmarks, including GLUE[42], SQuaD[34], and RACE[22]. Its efficiency in pre-training enables it to outperform models like BERT and GPT on these tasks, especially when considering the smaller model size and faster training time. Due to its efficiency and effectiveness, ELECTRA has broad applications in NLP, including but not limited to text classification, question answering, and language understanding tasks. Its ability to leverage smaller compute resources without compromising on performance makes it particularly appealing for deployment in resource-constrained environments.

ELECTRA presents a significant advancement in the field of pre-trained language models by introducing an efficient and effective training methodology. Its generator-discriminator framework and the replaced token detection task offer a novel approach to leveraging the full potential of the input data, setting new benchmarks in NLP tasks. As such, ELECTRA not only represents a leap forward in model efficiency but also provides a blueprint for future innovations in the development of pre-trained language models.

3.2.3 Baidu's ERNIE: Enhanced Representation through Knowledge Integration

ERNIE (Enhanced Representation through kNowledge Integration) is a pre-trained language model developed by Baidu[44], which introduces another novel approach to understanding complex language features by integrating structured knowledge from external sources into the pre-training process. By leveraging entity-level information and semantic relationships from knowledge graphs, ERNIE aims to enhance the model's ability to comprehend and process natural language beyond the capabilities of traditional context-based pre-training methods. Other

PLMs like RoBERTa, primarily rely on contextual information without explicitly modeling world knowledge or semantic relationships between entities. ERNIE addresses this gap by integrating knowledge graph information into the pre-training process, enabling the model to capture and utilize structured world knowledge in addition to contextual representations. This approach enhances the model's understanding of language, particularly in tasks requiring a deep understanding of entity relationships and factual knowledge.

ERNIE Architecture

ERNIE builds upon the Transformer architecture, similar to other state-of-the-art pre-trained models. The distinctive feature of ERNIE lies in its dual-channel architecture, which processes textual and knowledge information simultaneously:

- **Textual Channel**: This processes the input text in a manner similar to conventional Transformer-based models, capturing contextual information from the input sequence.
- **Knowledge Channel:** Concurrently, the knowledge channel processes entity information derived from a pre-constructed knowledge graph. This channel is designed to capture and encode the relationships between entities present in the text.

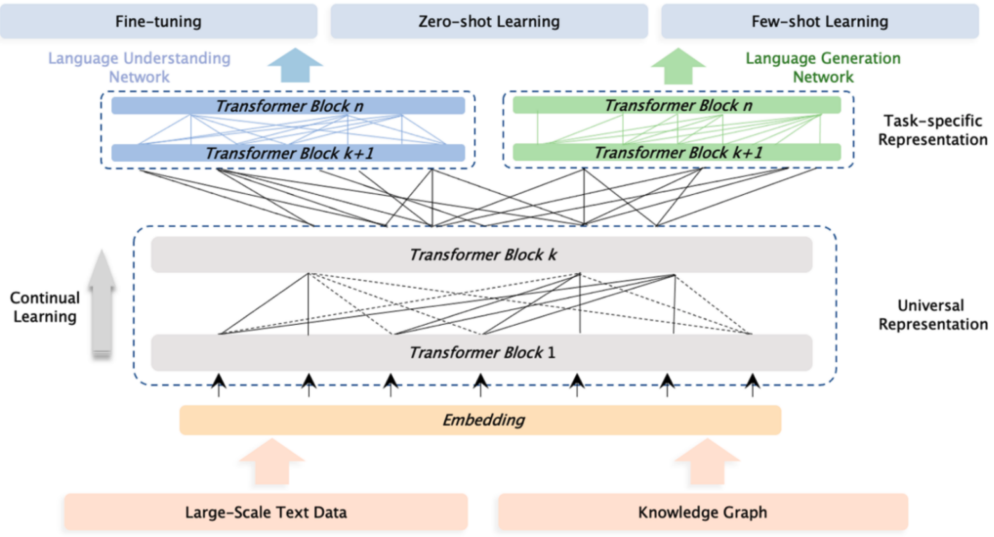

Figure 3.5: ERNIE Architecture
[35]

The outputs of both channels are merged to produce a comprehensive representation that includes both contextual and world knowledge.

Training Methodology

ERNIE is pre-trained using a combination of traditional language modeling objectives and knowledge-driven tasks. These include:

- **Masked Language Modeling (MLM):** Similar to BERT, ERNIE uses MLM but extends it to include entity-level masking, where entities recognized in the input text are masked and predicted as wholes, encouraging the model to understand entity semantics.

- **Knowledge Integration Task:** ERNIE introduces a novel pre-training task designed to align the embeddings of textual mentions with their corresponding entities in the knowledge graph, enhancing the model's ability to leverage structured knowledge.

A key component of ERNIE's training involves integrating information from knowledge graphs. This process includes identifying entities in the text, linking them to their counterparts in the knowledge graph, and encoding the semantic relationships between these entities into the model's representations.

Performance and Benchmarks

ERNIE demonstrates superior performance on a variety of NLP benchmarks, particularly in tasks that benefit from enhanced understanding of entity relationships and world knowledge, such as question answering, named entity recognition, and semantic similarity. Its innovative integration of knowledge graph information enables ERNIE to achieve notable improvements over its predecessors. ERNIE represents a significant advancement in pre-trained language model technology by effectively incorporating structured world knowledge into language representations. This approach addresses a critical limitation of prior models, offering deeper linguistic and semantic comprehension.

3.2.4 BART: Bidirectional and Auto-Regressive Transformers

BART (Bidirectional and Auto-Regressive Transformers) is a pre-trained language model developed with the intent to enhance performance across a variety of natural language processing (NLP) tasks through a novel combination of bidirectional encoding and autoregressive decoding. Proposed by Lewis et al.[24], BART integrates the strengths of both encoder-based models like BERT and autoregressive models like GPT[33]. This dual approach allows BART to effectively handle tasks requiring both understanding and generation of text, making it a powerful tool for a broad spectrum of NLP applications.

BART Architecture

BART adopts a standard Transformer architecture with a notable twist in its operational paradigm. It consists of two main components:

- **Encoder:** The encoder processes input texts bidirectionally, capturing the context and relationships within the input data. This part of the model is similar to BERT(RoBERTa),

which uses a masked language model (MLM) approach to pre-train on a vast corpus of text.

- **Decoder:** The decoder generates text auto-regressively, predicting the next token in a sequence given the previous tokens. This component mirrors the functionality of models like GPT[33], focusing on generating coherent and contextually relevant text.

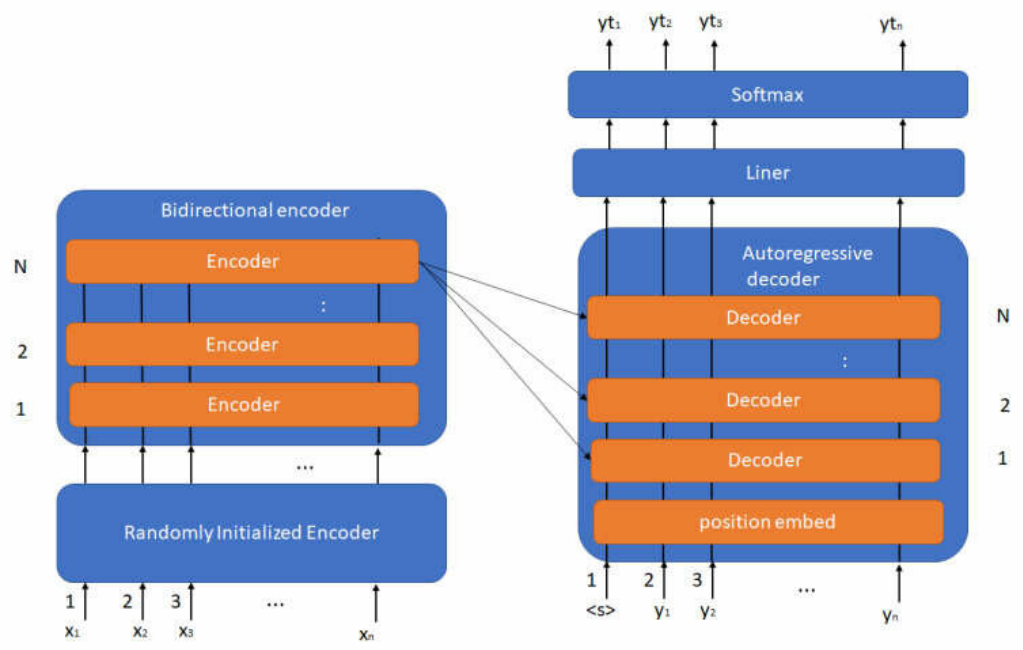

Figure 3.6: BART Architecture[1]

This encoder-decoder structure enables BART to be inherently flexible, and adept at both understanding input text and generating new text based on the learned representations.

Training Methodology

BART is pre-trained using an objective that involves corrupting text with an arbitrary noising function and then learning to reconstruct the original text. The corruption methods include token masking, deletion, text infilling, sentence permutation, and document rotation, encouraging the model to learn a robust understanding of language structure and content. Post pre-training, BART is fine-tuned on specific tasks by adapting its encoder-decoder architecture to suit the requirements of each task, such as sequence classification, question answering, or text summarization.

Performance and Benchmarks

BART achieves high performance in a range of NLP tasks, notably in text summarization, machine translation, and comprehension tasks. Its versatile pre-training and fine-tuning approach enables it to adapt effectively to different task requirements, showcasing significant improvements over previous models. BART represents a significant advancement in the field of pre-trained language models, bridging the gap between understanding and generating text. By innovatively combining bidirectional encoding with autoregressive decoding, BART provides a flexible and powerful framework that excels in various tasks.

3.2.5 PLMs' summary

Table 3.2 demonstrates the key differences of each model while at the same points out their advantages and disadvantages. This table summarizes the key aspects of the models when considering them for app review classification.

Table 3.2: Summary of PLMs

Model	Key Differences	Advantages	Disadvantages
RoBERTa	Removes the Next Sentence Prediction (NSP) task, uses dynamic masking, and is trained on an extended dataset and for longer periods.	Achieves superior performance by leveraging larger datasets and more thorough training processes.	Requires significant computational resources for training due to larger datasets and longer training times.
ELECTRA	Uses a generator-discriminator approach for pre-training, focusing on distinguishing between real and fake tokens.	Efficient use of training data by learning from all tokens, leading to faster training and smaller models.	The complexity of training two models (generator and discriminator) can be a barrier to some applications.
ERNIE	Integrates structured knowledge from external sources, focusing on entity-level information and semantic relationships.	Enhances model's understanding of language by incorporating world knowledge and semantic relationships.	Integration of external knowledge sources can complicate the training process and increase computational demands.
BART	Combines bidirectional encoding and autoregressive decoding in a unified framework, using a novel text noising and reconstruction pre-training task.	Flexible model capable of both understanding and generating text, making it well-suited for a wide range of tasks.	The complexity of the model and its pre-training tasks might lead to longer training times and increased computational needs.

3.3 Hyperparameter Tuning

After clearly understanding the nature of the PLMs used in this research, the next step is the tuning of their hyperparameters. The fine-tuning of hyperparameters is a critical step in NLP,

directly impacting the performance of the models on given tasks. There are various popular methods for hyperparameter tuning.

1. **Grid Search:** Exhaustively searches through a manually specified subset of the hyperparameter space, evaluating and comparing the performance of each set of hyperparameters. Straightforward but potentially time-consuming for large search spaces[4].

2. **Random Search:** Samples hyperparameter settings randomly within predefined ranges for each hyperparameter and evaluates their performance[3].

3. **Bayesian Optimization:** Utilizes a probabilistic model to predict the performance of hyperparameter configurations and sequentially chooses new hyperparameters to evaluate by optimizing the expected improvement over the current best. Effective for high-dimensional spaces and costly evaluations[37].

4. **Gradient-based Optimization:** Adjusts hyperparameters using gradient descent, where the gradient is estimated based on the model's performance with respect to the hyperparameters[19].

5. **Hyperband:** A bandit-based approach that dynamically allocates resources to a set of hyperparameter configurations and efficiently prunes less promising ones, focusing on more promising configurations[25].

In this project, it was decided to utilize Bayesian Optimization for tuning hyperparameters of the models as it is one of the most efficient ones in terms of time and resource utilization.

3.3.1 Bayesian Optimization

Traditional optimization methods often fall short either due to the high dimensionality of the hyperparameter space or the computationally intensive nature of model evaluation. Bayesian Optimization addresses these challenges by constructing a probabilistic model of the objective function and iteratively updating this model based on the outcomes of previous evaluations to guide the selection of new hyperparameter configurations[37].

The core of Bayesian Optimization is a probabilistic model that estimates the distribution over functions consistent with the observed evaluations of the objective function. Gaussian Processes (GPs) are commonly used for this purpose due to their flexibility in modeling diverse function shapes and their capability to provide uncertainty estimates alongside predictions. To decide which points in the hyperparameter space to evaluate next, Bayesian Optimization relies on an acquisition function. This function is derived from the probabilistic model and is designed to balance the exploration of under-explored regions against the exploitation of regions known to perform well. Common choices for acquisition functions include Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB).

Optimization Loop

The process follows an iterative loop:

- The probabilistic model is used to estimate the performance of unseen hyperparameter configurations.
- The acquisition function identifies the next configuration to evaluate based on these estimates.
- The selected configuration is evaluated by training the model and computing the objective function (e.g., validation accuracy).
- The probabilistic model is updated with the new observation. This loop continues until a stopping criterion is met, such as a maximum number of iterations or a satisfactory performance level.

Advantages of Bayesian Optimization

- Sample Efficiency: It typically requires fewer function evaluations to identify optimal or near-optimal hyperparameters compared to grid search or random search, which is advantageous for expensive evaluation functions.
- Incorporation of Prior Knowledge: Bayesian Optimization can incorporate prior beliefs about the function's behavior, potentially accelerating the optimization process.
- Quantification of Uncertainty: The probabilistic model provides uncertainty estimates, which can be used to navigate the trade-off between exploration and exploitation effectively.

Limitations

- Dimensionality: The efficiency of Bayesian Optimization can degrade in very high-dimensional spaces, although dimensionality reduction techniques and the development of more sophisticated acquisition functions can mitigate this issue.
- Choice of Kernel and Acquisition Function: The performance of Bayesian Optimization is sensitive to the choice of the kernel in the Gaussian Process and the acquisition function, requiring careful tuning and domain knowledge.

Bayesian Optimization provides a powerful framework for the optimization of hyperparameters in machine learning, including complex NLP models. By intelligently navigating the hyperparameter space and leveraging information from previous evaluations, it offers an efficient alternative to traditional optimization methods.

After clarifying the hyperparameter optimization method, it is necessary to decide which hyperparameters for the models should be tuned. Due to time and resource constraints, it was decided to optimize 5 hyperparameters: **learning rate, weight decay, num train epochs, training, and evaluation batch size**. The same interval for all the hyperparameters was ensured to be provided for each model to make the comparison fair.

3.3.2 Learning Rate

The learning rate is a hyperparameter that determines the step size at each iteration while moving toward a minimum of the loss function. In the context of PLMs, it influences how drastically the model's weights are updated during training. A higher learning rate may lead to faster convergence but can overshoot the minimum, while a too low learning rate may result in a prolonged training process, potentially getting stuck in local minima. The optimal learning rate balances fast convergence with the risk of overshooting. Additionally, learning rate schedules, such as warm-up and cool-down phases, can significantly enhance model performance.

3.3.3 Weight Decay

Weight decay is a regularization technique used to prevent overfitting by penalizing large weights. It works by adding a portion of the weights themselves to the loss function, effectively encouraging the model to maintain smaller weights. Incorporating weight decay helps in improving the generalization of PLMs by discouraging complex models that fit the training data too closely. This can be especially important in PLMs due to their large parameter spaces. The choice of weight decay factor requires balancing between too much regularization, which can hinder the model's ability to learn from the data, and too little, which may lead to overfitting.

3.3.4 Number of Training Epochs

An epoch in machine learning is one complete pass through the entire training dataset. The number of training epochs for PLMs dictates how many times the model is exposed to the training data. Too few epochs can result in underfitting, where the model fails to capture the underlying data patterns, while too many epochs can lead to overfitting, where the model learns noise from the training data as if it were a signal. Early stopping is a strategy used to prevent overfitting by terminating training when the model's performance on a validation set ceases to improve. This approach allows for dynamically determining the optimal number of epochs based on actual learning progress.

Table 3.3: Search Space of Hyperparameters

Hyperparameter	**Search Space**
Learning Rate	1e-5 to 5e-5(Real)
Weight Decay	0.01 to 0.2(Real)
Number of Training Epochs	3 to 5(Integer)
Per Device Training Batch Size	8 to 32(Integer)
Per Device Evaluation Batch Size	8 to 64(Integer)

3.3.5 Per-Device Training Batch Size

Per-device training batch size refers to the number of training examples utilized in one iteration for each computing device (such as a GPU or CPU). It directly affects memory utilization and the stochastic gradient descent process. Larger batch sizes provide a more accurate estimate of the gradient but require more memory and can lead to a less effective exploration of the parameter space. Smaller batch sizes, while more computationally intensive, often lead to better generalization. The choice of batch size is constrained by the available memory on the device but can be optimized for training efficiency and model performance. Techniques such as gradient accumulation allow for effectively simulating larger batch sizes on limited hardware.

3.3.6 Per-Device Evaluation Batch Size

Similar to training batch size, per-device evaluation batch size determines the number of examples processed in one iteration during model evaluation. It influences the speed and memory requirements of the evaluation phase. While it does not affect the training directly, an optimal evaluation batch size ensures efficient use of computational resources during model validation and testing, allowing for more frequent evaluations without sacrificing performance. Maximizing the evaluation batch size within the constraints of device memory can reduce evaluation times and facilitate more agile development cycles. However, it must be chosen to ensure accurate performance estimation without overloading the device.

3.3.7 Search Space

For the optimization process, it was decided to go with the negated accuracy property to be minimized as it is necessary to maximize the accuracy of the model. All models were optimized on the balanced 4,000 review dataset provided by Tianpeng Huang, in multi-class setting and in the same intervals for each hyperparameter as shown in the table (3.3).

The range for each search space was chosen based on popular norms and resource constraints. Bigger intervals would require much more computational resources and execution time. The range for Learning Rate is popular in models trained with the Adam optimizer, such

as BERT and its variants(RoBERTa). Weight Decay's range is broad enough to explore the effect of regularization without overly constraining the model's capacity. The range of 8 to 32 for Per Device Training Batch Size is a practical compromise for many NLP tasks, fitting within the memory limits of common GPUs while allowing for effective learning. A range for Per Device Evaluation Batch Size from 8 to 64 allows for faster evaluation by making efficient use of the available computational resources. It enables quick feedback on the model's performance adjustments and faster iterations during the development process.

3.4 Evaluation Metrics

Evaluation metrics are critical for assessing the performance of machine learning models, guiding the selection of models, and tuning their hyperparameters for optimal performance on specific tasks. In the domain of classification, commonly used metrics include accuracy, precision, recall, and the F1-score. Additionally, training time is a crucial factor in evaluating the efficiency of model training processes. This section provides a detailed overview of these metrics, including their definitions, formulas, and implications for model evaluation.

3.4.1 Accuracy

Accuracy is the simplest and most intuitive performance metric. It measures the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. While accuracy is straightforward, it may not provide a comprehensive view of a model's performance, especially in imbalanced datasets where the majority class dominates[15].

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

Where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

3.4.2 Precision

Precision measures the accuracy of the positive predictions made by the model. It indicates the proportion of positive identifications that were actually correct. High precision indicates a low rate of false positives. This metric is particularly important in applications where the cost

of false positives is high. However, precision alone does not consider false negatives; hence, recall is also a crucial metric[15].

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3.4.3 Recall

Recall measures the proportion of actual positives that were identified correctly. It is especially critical in scenarios where missing a positive instance (a false negative) has severe implications. High recall indicates a low rate of false negatives. Recall is vital in situations where it is crucial to capture as many positives as possible, even at the expense of increasing false positives[15].

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

3.4.4 F1-Score

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the two. It is particularly useful when there's a need to find an equilibrium between precision and recall. The F1-score is valuable in situations with imbalanced classes or when false positives and false negatives carry different costs. It ensures that models are not biased towards either precision or recall[15].

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.4.5 Training Time

Training time measures the duration required to train a model. In practice, this metric affects the feasibility of model selection, hyperparameter tuning, and retraining. Shorter training times are generally preferred for iterative development and when deploying models that require frequent updates. However, the training time should not compromise the model's performance; thus, it is often considered alongside the aforementioned metrics to evaluate overall model efficiency and effectiveness[15].

3.5 Programming Libraries and Environment

Various libraries to train and evaluate the PLMs on the datasets were used. **Pandas** was used for data manipulation and analysis, particularly for reading Excel files and processing the datasets. **Scikit-learn (sklearn)** was used for data splitting, model evaluation, and encoding labels, facilitating the creation of training and test datasets, as well as evaluating model performance. **Transformers** library from Hugging Face was utilized for accessing pre-trained models and

tokenizers of the necessary PLMs, for sequence classification and providing training infrastructure. **PyTorch (torch)** served as the underlying framework for tensors and neural network operations, crucial for defining custom dataset classes and tensor manipulations. **NumPy** provided support for numerical operations, handling arrays, and mathematical computations used throughout the code. **Scikit-optimize (skopt)** was employed for hyperparameter optimization using Bayesian optimization to find optimal training parameters for the model. **OS, shutil, re, time, and pathlib** are standard Python libraries that were used for accessing, storing, and deleting the output files containing the performance metrics for each dataset. **Google Collab Pro** platform was utilized for running the experiments using the **V-100 GPU** with **High-RAM** property.

3.6 General Design

Figure 3.7 depicts hyperparameter optimization design showing the general step by step process.

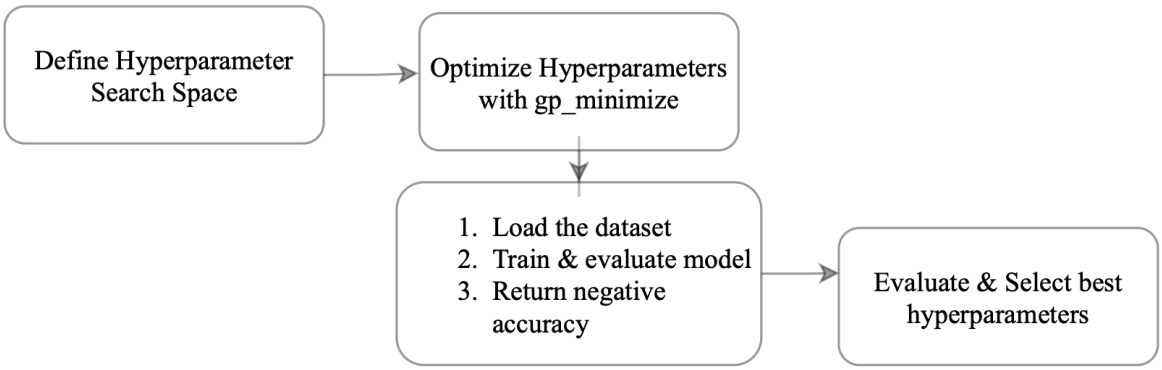

Figure 3.7: Hyperparameter Optimization Design

The search space for each hyperparameter is defined first, and then the model undergoes fine-tuning using a dataset of 4,000 Balanced Multi-Class format[16]. The hyperparameters specified are applied during this process. The function "gp_minimize" from the "skopt" library, which employs Bayesian optimization, is used to optimize these hyperparameters by aiming to achieve the highest possible accuracy, returning the negated accuracy as the output.

Figure 3.8 outlines the general design for conducting the experiment. Central to this design (Figure 3.8) is the splitting of the dataset and the application of Stratified K-Fold Cross-Validation. Initially, the dataset is divided into training and test sets, with an 80-20 split ratio. This separation ensures that a portion of the data remains untouched during the model training process, allowing for an unbiased evaluation of the model's performance on unseen data.

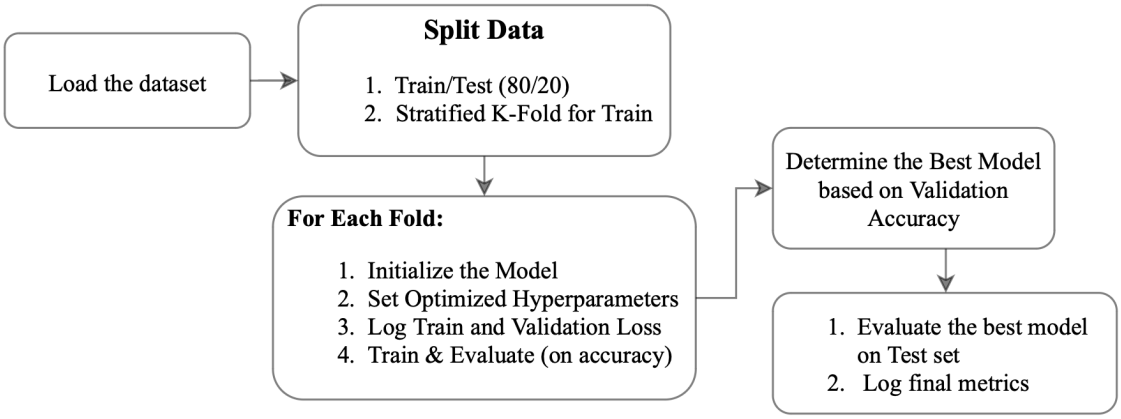

Figure 3.8: Model Experiment Design

3.6.1 K-Fold Cross-Validation

The K-fold cross-validation is implemented on the train set with K = 5. K-Fold Cross-Validation involves partitioning the original dataset into a specified number of equal segments, or folds[39]. During the cross-validation process, each fold is sequentially used as a test set, while the remaining folds collectively form the training set. This procedure is repeated such that each fold serves as the test set exactly once. The primary advantage of K-Fold Cross-Validation lies in its ability to utilize all available data for both training and testing, thereby providing a comprehensive evaluation of the model's performance. This method helps mitigate the risk of model overfitting by ensuring that the model's effectiveness is not overly dependent on a particular subset of the data. The Stratified K-Fold Cross-Validation is particularly beneficial for ensuring that each fold retains the same proportion of categorical labels as the original dataset, addressing potential imbalances that could skew the model's learning and generalization capabilities. Stratification, thus, plays a critical role in maintaining the integrity and representativeness of each training and validation cycle.

Choosing K= 5 for K-fold cross-validation is widely regarded as a balanced choice that provides both robust validation results and computational efficiency. This value is sufficiently large to ensure that each fold is a reasonable approximation of the overall dataset, minimizing the variance in the model evaluation metrics across different folds. Moreover, it avoids the higher computational cost associated with larger values, while still providing more reliability and stability in the validation results than smaller K values.

After the K-fold cross-validation, the best model based on evaluation accuracy of the fold was picked and evaluated on the test set. Afterwards, the aforementioned performance metrics were recorded for the model.

3.6.2 Overfitting and Underfitting

Another factor that is important to note is the **overfitting** and the **underfitting** of the model. Overfitting occurs when a model learns the training data to such an extent that it begins to incorporate noise and random fluctuations as if they were significant patterns[20]. This results in a model that has limited applicability beyond the training dataset, characterized by high variance and diminished performance on new, unseen data. Within the framework of classifying user reviews, overfitting may cause the model to misclassify reviews based on anomalies in the training set that are not representative of broader patterns, making it overly sensitive to specific terms or expressions.

In contrast, underfitting is observed when a model does not adequately learn the essential patterns present in the training data, often due to its simplicity[20]. This scenario is marked by high bias, where the model underperforms across both the training and testing datasets.

3.6.3 Training and Validation loss

In order to capture such scenarios, during every fold the training and validation loss were recorded. Training loss quantifies the model's error when evaluated against the training dataset, indicating how well the model predicts the known outcomes within this data. Validation loss, on the other hand, measures the model's error on the validation dataset, representing its accuracy on data not seen during training. In deep learning contexts, where models typically have a vast number of parameters, there is a heightened risk of overfitting. This situation implies that the model may perform exceptionally well on the training data but poorly on new, unseen data. By regularly observing the trends in training and validation loss, early signs of overfitting can be detected, allowing for timely intervention to refine and enhance the model's performance. More specifically, the behavior of training loss and validation loss mean the following:

- **Train loss down, Validation loss down:** Indicates effective learning and generalization. This is the ideal scenario, suggesting that the model is improving on both seen and unseen data.
- **Train loss down, Validation loss increased or unchanged:** Suggests the model is starting to overfit to the training data.
- **Train loss stable, Validation loss down:** May indicate issues with the data quality or distribution.
- **Train loss stable, Validation loss stable:** This situation occurs when the learning process hits a plateau. Options to overcome this include reducing the learning rate or adjusting the batch size.
- **Increase in train loss, increase in Validation loss:** Reflects a fundamental issue with the model or data, such as poor network architecture, suboptimal parameters, or the need

for data cleaning. This is a critical situation requiring a thorough review of the model design and data preprocessing steps.

Chapter 4

Results Analysis

This chapter provides an in-depth discussion of obtained results focusing on five main perspectives:

- **Balanced Datasets:** Models' performance with Balanced Datasets.
- **Unbalanced Datasets:** Models' performance with Unbalanced Datasets.
- **Multi-label Datasets:** Models' performance in multi-label setting.
- **Training Time:** Models' training time in various scenarios.
- **Training and Evaluation Loss:** Impact of the Datasets on models' overfitting and underfitting.

4.1 Balanced Datasets

Figure 4.1 illustrates the accuracy of each model across all balanced datasets. Accuracy is a suitable metric for assessing multi-class classification as it provides a straightforward evaluation of overall model performance by measuring the proportion of total correct predictions out of all predictions made. It may be noticed that as the size of the datasets increases, so does the accuracy of all models. This improvement is attributed to the larger volumes of training data, which enable the models to more effectively capture the semantic nuances present in the reviews. Larger datasets offer numerous benefits for training. They provide a greater number of samples and features, enabling the models to identify a wider and more diverse range of patterns, which enhances their ability to generalize to new data. Additionally, they diminish the likelihood of overfitting, a scenario in which a model learns the noise in the data rather than the actual patterns it should generalize. Furthermore, larger datasets contribute to model stability by minimizing the effects of random variations and noise on the performance of the model.

Another critical aspect to consider is the variation in accuracy across datasets labeled by ChatGPT and manually labeled ones, particularly between those of the same size. For all

models, the datasets labeled by ChatGPT score significantly higher accuracy. This variation can be partly explained by the quality of the datasets. The datasets from Hugging Face[9] labeled by ChatGPT contain, on average, more than twice the word count of those labeled manually by Tianpeng Huang[16]. This increased word count likely leads to a better understanding of the review content and a more accurate capture of significant meanings.

Furthermore, the consistency in labeling by ChatGPT contributes to its higher accuracy. ChatGPT's deterministic response generation, based on its training data and underlying algorithms, ensures uniform labeling quality. This consistency contrasts with the variability often observed with human annotators, who may interpret classification criteria differently, leading to discrepancies in labeling quality across the dataset.

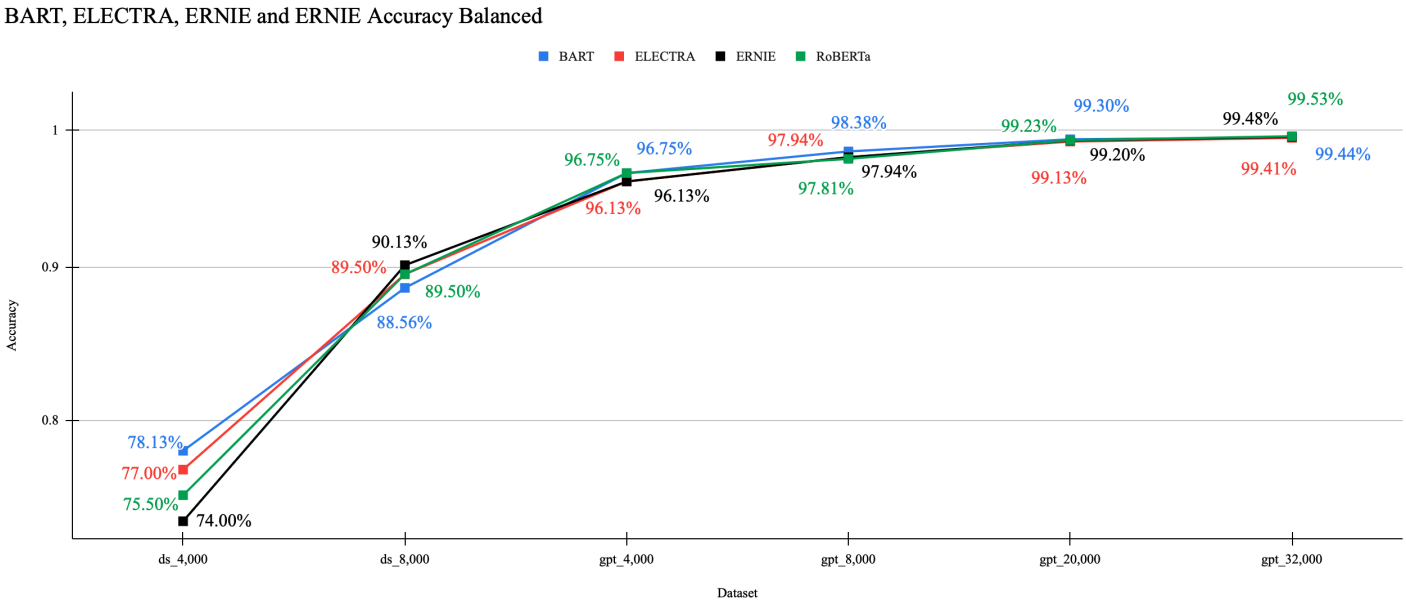

Figure 4.1: Each Model's Accuracy on Balanced Datasets

Additionally, the implemented models, being transformer-based PLMs, are highly sensitive to the nuances of data they are trained on. ChatGPT's labeling is characterized by high consistency and adherence to its trained patterns, potentially leading to a dataset that is more homogeneous in language use and style. This homogeneity can artificially enhance model performance when the training data (ChatGPT-labeled) closely aligns with the linguistic patterns these models are optimized to detect. In contrast, human-labeled datasets likely introduce a wider variety of linguistic expressions and a broader interpretation of classification labels, reflecting natural language diversity but potentially increasing noise and variability in the training data. This variance can challenge the model's ability to learn generalized patterns, thus reducing accuracy. Moreover, since models like ELECTRA and BART are particularly adept at picking up on subtle language cues and contextual nuances, they may perform better with the more uniformly styled ChatGPT data, which mimics the structured training environment these models were originally developed with. Therefore, the higher accuracy on ChatGPT-labeled

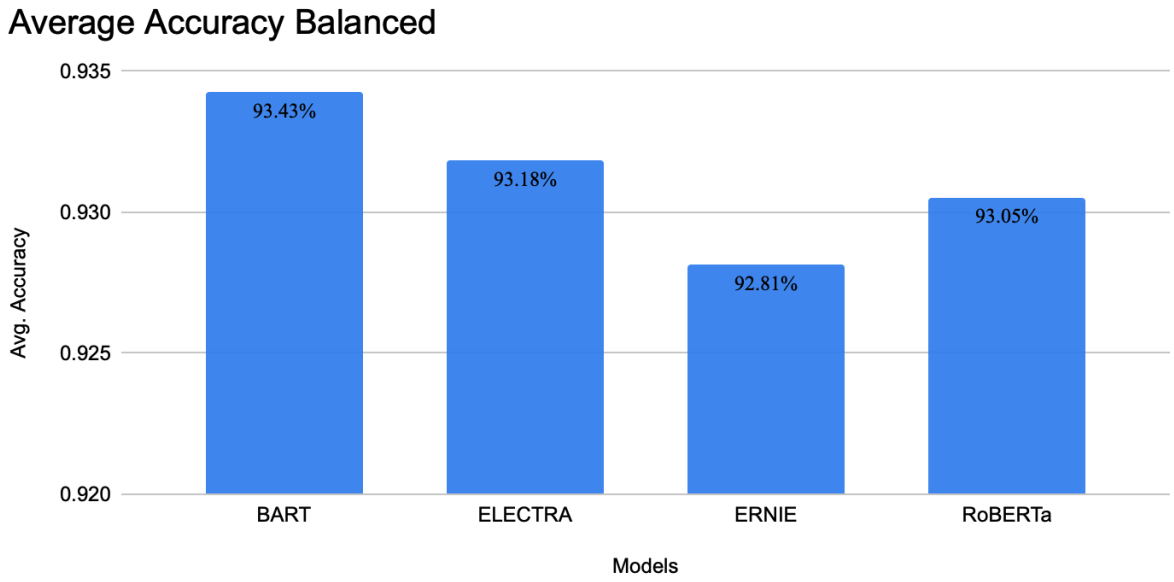

Figure 4.2: Each Model's Combined Average Accuracy on Balanced Datasets

datasets might indicate an alignment between the models' training on PLM-generated text and their evaluation on similarly generated text, rather than a true superiority in handling diverse real-world data.

On human-labeled datasets, BART and ERNIE have had the highest accuracy scores, surpassing the state-of-the-art RoBERTa model, as shown in Figure 4.1. In contrast, on ChatGPT-labeled datasets, while all models demonstrated similar performance levels, BART generally achieved marginally higher accuracy.

Furthermore, an analysis of the combined average accuracy across all balanced datasets indicates that BART was the highest achiever, followed by ELECTRA (4.2). It is important to note, however, that the accuracy metrics across all models were relatively comparable. BART's slight outperformance over other PLMs can be attributed to its unique architectural features and training methodology. BART is a denoising autoencoder for pretraining sequence-to-sequence models, which combines bidirectional and auto-regressive transformers. The model's strength lies in its ability to reconstruct text from corrupted versions, which closely aligns with the task of interpreting and classifying nuanced text data in app reviews. Unlike RoBERTa and ELECTRA, which focus more on understanding text inputs via masked language modeling or adversarial training, BART's training on both the encoder and decoder enables a deeper understanding of context and intent.

4.2 Unbalanced Datasets

The interesting insight about the performance on unbalanced datasets is that all models have achieved higher accuracy compared to balanced datasets scenario (4.4). PLMs are generally

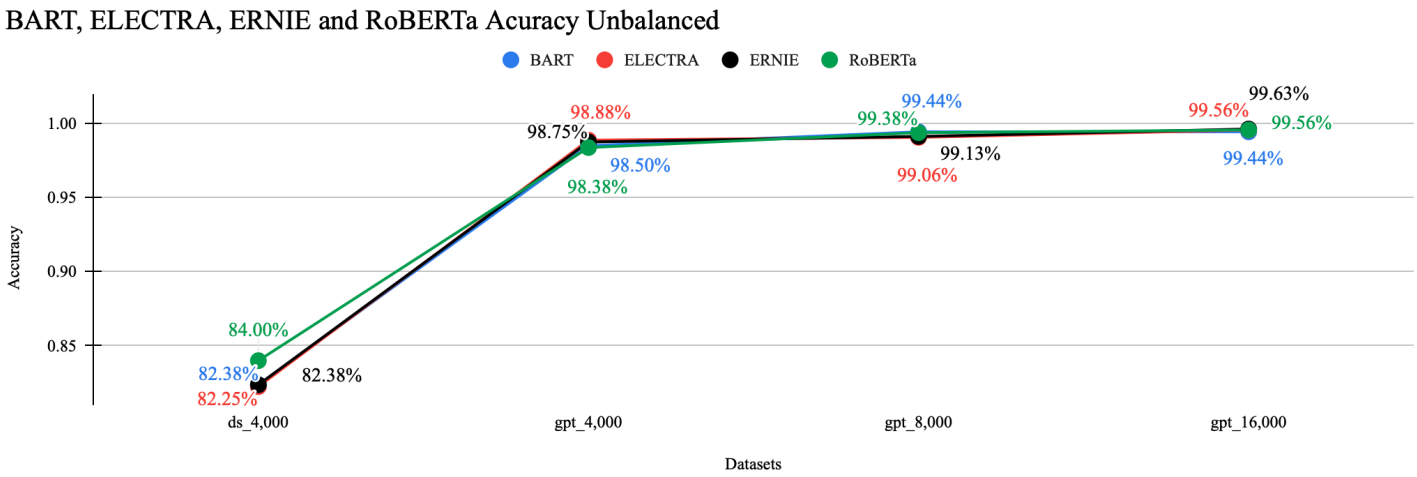

Figure 4.3: Each Model's Accuracy on Unbalanced Datasets

adept at leveraging large amounts of data, and unbalanced datasets, by their nature, often contain a majority of examples from more frequent classes. This abundance allows PLMs to develop a more robust understanding of the common classes, optimizing their predictive accuracy for these labels. These models are engineered to extract and generalize key features from the dominant data during training, which can disproportionately influence their learning and decision-making processes. Consequently, the models might exhibit an enhanced ability to accurately classify reviews into these more frequently represented categories, thereby increasing overall accuracy metrics. This effect is further boosted by the fact that the statistical impact of misclassifying less represented classes is diminished in unbalanced settings, leading to apparently higher performance values.

On human-labeled dataset, RoBERTa has slightly outperformed all other models, as shown in Figure 4.3. On ChatGPT-labeled datasets, all models performed relatively similar.

Figure 4.4 depicts that RoBERTa has achieved highest average overall accuracy with less than 1% difference.

RoBERTa's intensive focus on understanding contextual relationships within the text makes RoBERTa particularly adept at extracting meaningful patterns from dominant classes, which are typically overrepresented in unbalanced datasets. Moreover, RoBERTa's ability to integrate deeper contextual nuances aids in better generalization across the frequent classes, maximizing predictive accuracy where data is plentiful.

4.3 Multi-Label Datasets

In multi-label scenario, ELECTRA was the lowest performer while BART and ERNIE have scored comparably the highest F1-Score (Figure 4.5). The F1-score is highly effective for assessing multi-label classification because it harmonizes the precision and recall of the model, offering a balanced measure of its performance in scenarios where class imbalance may affect

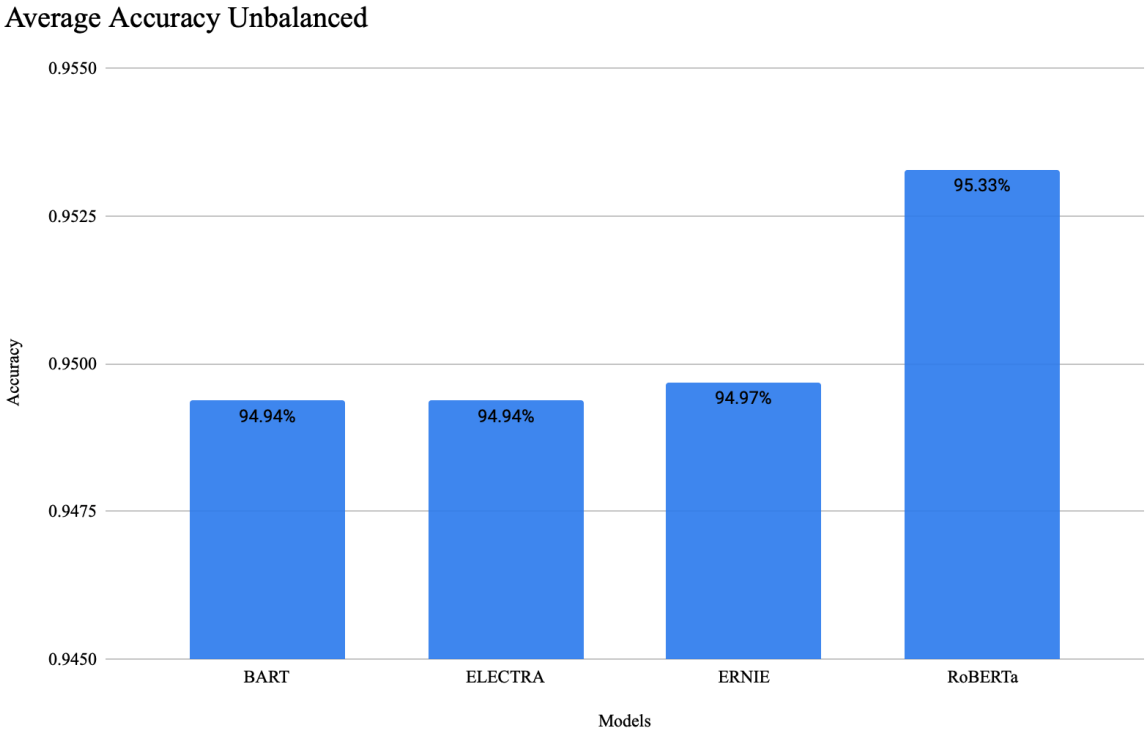

Figure 4.4: Each Model's Combined Average Accuracy on Unbalanced Datasets

the accuracy metric. This metric is crucial for understanding how well the model identifies relevant instances across multiple labels, ensuring that both the completeness of the model's predictions and their accuracy are accounted for. Unlike other models that focus on generating or predicting text based on contextual clues, ELECTRA operates on a discriminative task where it distinguishes between "real" and artificially generated tokens within the input text. This method, while effective for certain types of tasks, may not align optimally with the demands of multi-label classification, which requires the simultaneous recognition and categorization of multiple relevant labels within a single instance. Moreover, ELECTRA's training is heavily dependent on the quality and characteristics of the replaced tokens used during its pre-training, which may not provide sufficient diversity or representational depth needed for effectively handling the complexities associated with multi-label data. Consequently, this can lead to a reduced capability in accurately identifying and assigning multiple labels, thus impacting the overall F1-Score.

ERNIE enhances its performance by integrating knowledge graph embeddings with transformer architectures, which allows it to capture semantic relationships between entities more effectively. This capability is crucial in multi-label classification, where understanding interdependencies between labels can significantly enhance predictive accuracy.

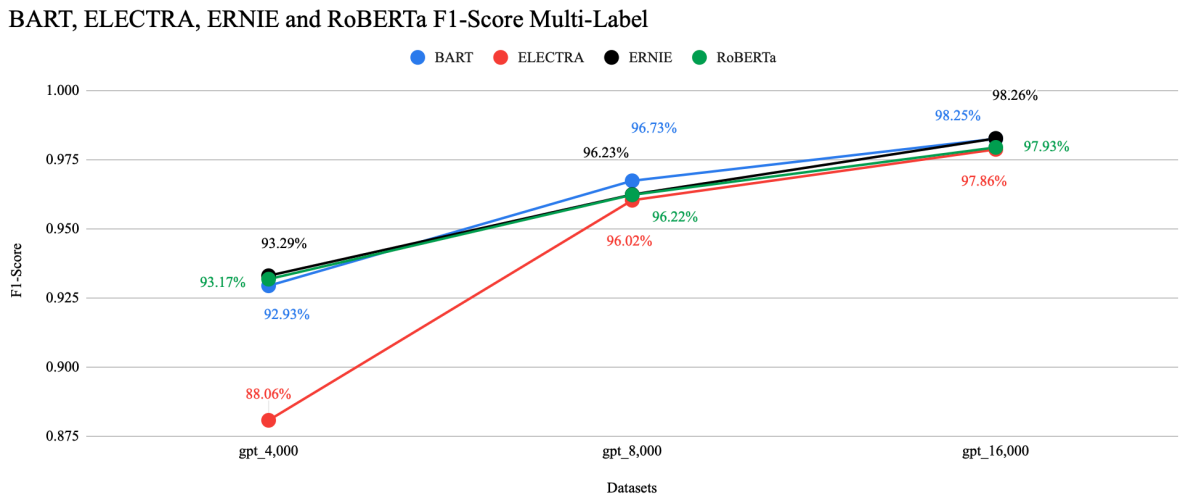

Figure 4.5: Each Model's Combined Average F1-Score on Multi-Label Datasets

4.4 Training Time

The Figure 4.7 depicts the training time, in seconds, for each model on Balanced Datasets. It can be noted that on all datasets ELECTRA demonstrated the shortest training times.

As discussed in Chapter 3, ELECTRA employs a unique approach where it trains a smaller generator network alongside a larger discriminator network. This generator modifies input tokens to create corrupted versions, which the discriminator then classifies as real or fake, effectively making it a more efficient adversarial training process. Unlike other language models that require predicting every token in the masked language modeling task, ELECTRA focuses only on distinguishing between genuine and replaced tokens, reducing the complexity of the tasks it needs to solve during training. This approach minimizes the computational load and accelerates the training process. Additionally, the discriminator in ELECTRA, which is the primary component used during inference, is trained to perform this binary classification rather than reconstructing entire sequences, leading to less computational overhead and faster overall training times.

On the other hand, BART showed the longest training times among the models tested, which can primarily be due to its sequence-to-sequence architecture and the nature of its training tasks. BART, designed as a denoising autoencoder, reconstructs original text from corrupted versions, requiring simultaneous operation of both its encoder and decoder components during training. This two-part mechanism involves first encoding a corrupted input and then generating the correct output, a process that inherently demands more computational resources and time compared to single-component models. Additionally, BART's training involves more complex operations such as cross-attention between the encoder and decoder and generation of complete text outputs, rather than just classifying or predicting tokens, further extending its training duration. This sequence-to-sequence learning, although demonstrating high accuracy and F1-scores, results in longer training time for achieving that performance.

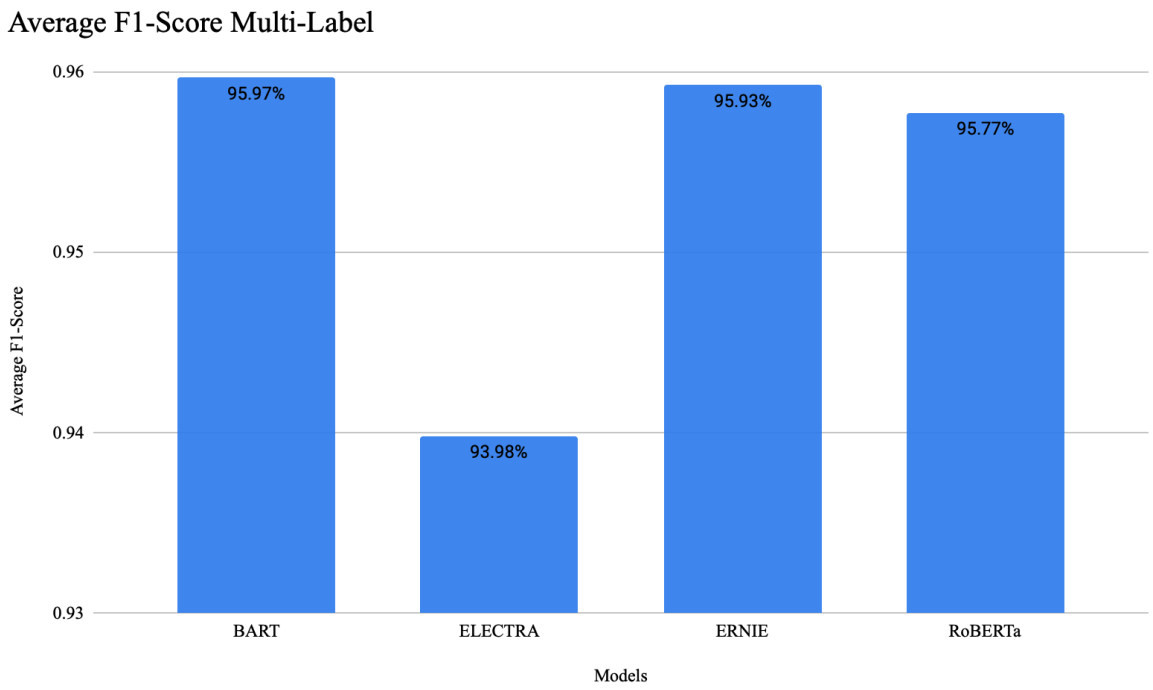

Figure 4.6: Combined Average F1-Score on Multi-Label Datasets

Further analysis of the total training times of the models across all scenarios reveals that ELECTRA demonstrated the highest efficiency, as indicated in Figure 4.1. Specifically, ELECTRA's total training duration was more than 28% shorter than BART's, translating to a time savings of over two hours, and over 9% shorter than RoBERTa's, equating to a reduction of more than half an hour. In commercial applications, where time and resource efficiency are crucial for reducing operational costs and accelerating development, these improvements in training efficiency could prove to be highly significant.

4.5 Training Loss and Validation Loss

It is also important to discuss the training and validation loss across both ChatGPT-labeled and the manually labeled datasets.

In Figure 4.8, it is visible that during training, BART's training loss consistently decreases while its validation loss remains relatively stable and even shows an increase when applied to

Table 4.1: Training time (in minutes) of the models in all scenarios.

Training Time (min)	**BART**	**ELECTRA**	**ERNIE**	**RoBERTa**
Balanced	255.17	183.74	190.17	199.42
Unbalanced	110.55	79.57	83.75	88.00
Multi-Label	98.42	70.16	72.70	77.44
Total	464	333	347	365

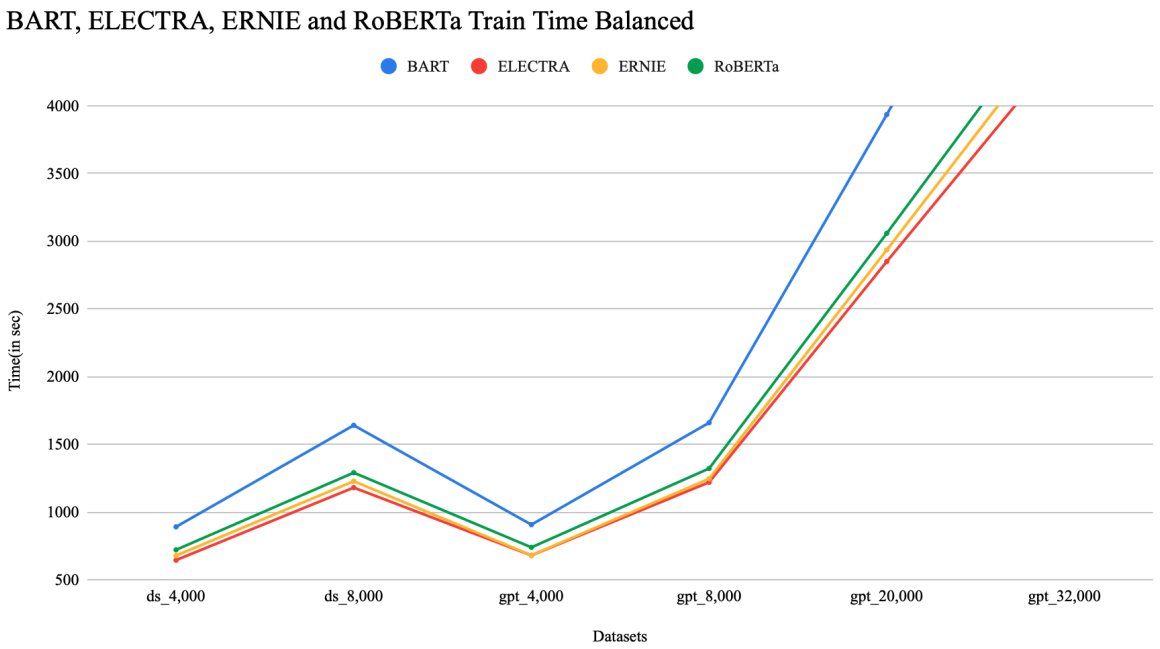

Figure 4.7: Each Model's Training Time on Balanced Datasets

the 8,000 balanced, manually labeled dataset. As discussed in Chapter 3, these trends indicate that BART is overfitting to the training data, which suggests that the model may not perform well on unseen data. On the other hand, Figure 4.9 demonstrates a more desirable outcome for 8,000 balanced ChatGPT-labeled dataset; both training and validation losses decrease, indicating effective learning and generalization, which is the optimal scenario.

The overfitting caused with the manually labeled dataset can be attributed to it being of lower overall quality. As previously mentioned, the average word count in these datasets is lower than in those labeled by ChatGPT, implying that each review contains less informative content. This limitation likely weakens the model's ability to learn effectively, as there is less relevant data from which to generalize, likely leading to a poorer adaptation to new, unseen data.

It is important to note that such behavior is present in all other models for manually labeled datasets which further strengthens the aforementioned explanation.

4.6 Summary

To summarize the performance of the models across various scenarios, it has been observed that all models displayed comparable levels of performance. Notably, the state-of-the-art model RoBERTa only outperformed other models in scenarios involving unbalanced datasets. Table 4.2 shows the total average accuracy, F1-Score and Training time of all the models in all settings. It can seen that BART emerges as the top performer in terms of accuracy and F1-Score.

However, an essential factor in comparing these models is their efficiency. ELECTRA

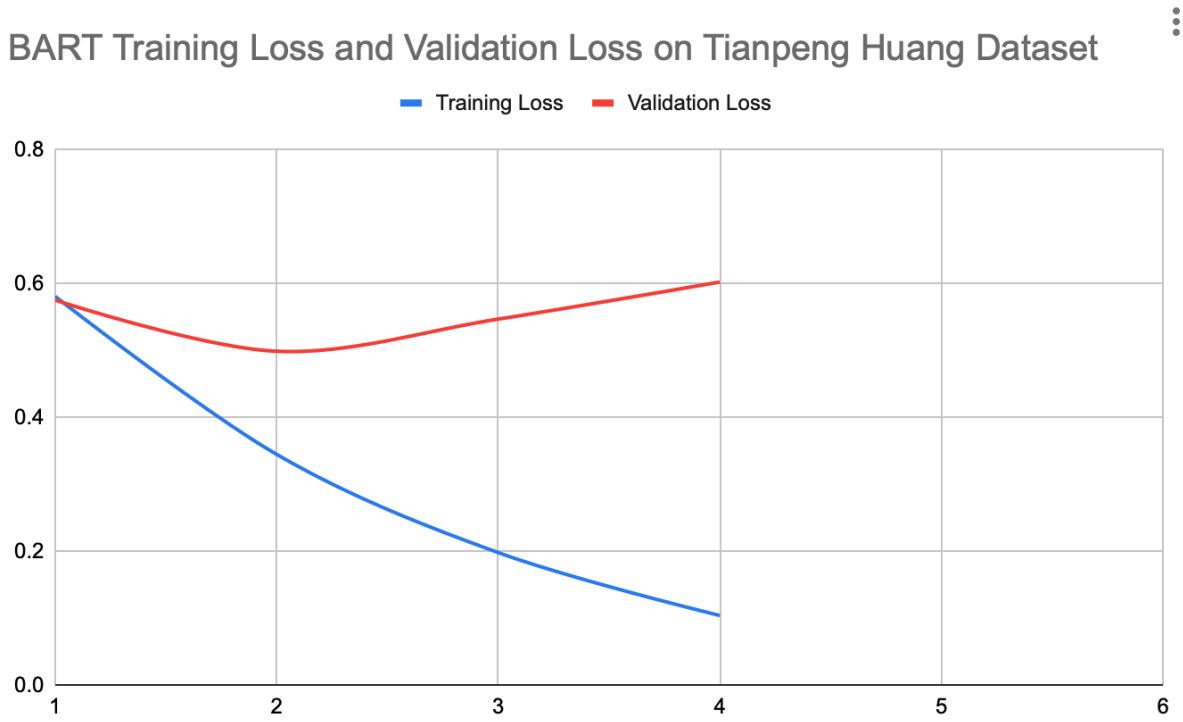

Figure 4.8: BART Training and Validation Loss on 8,000 balanced manually labeled dataset[16]

has been identified as the most efficient in terms of training time, significantly outpacing both RoBERTa and BART. This efficiency is particularly advantageous in commercial applications where reducing operational costs is a priority. For instance, in app review classification, deploying ELECTRA-based automated classification schemes could be beneficial for companies seeking similar performance outcomes but with less resource and time expenditure.

Overall, considering the efficiency alongside performance, ELECTRA can be regarded as surpassing the current state-of-the-art model RoBERTa when evaluating the balance between time and performance.

Table 4.2: All Overall Average metrics of the Models across all scenarios

	BART	ELECTRA	ERNIE	RoBERTa
Accuracy	95.18125	94.05989584	93.89036459	94.190625
F1-Score	94.98923774	93.89441662	94.76432109	94.89050069
Time(min)	464	333	347	365

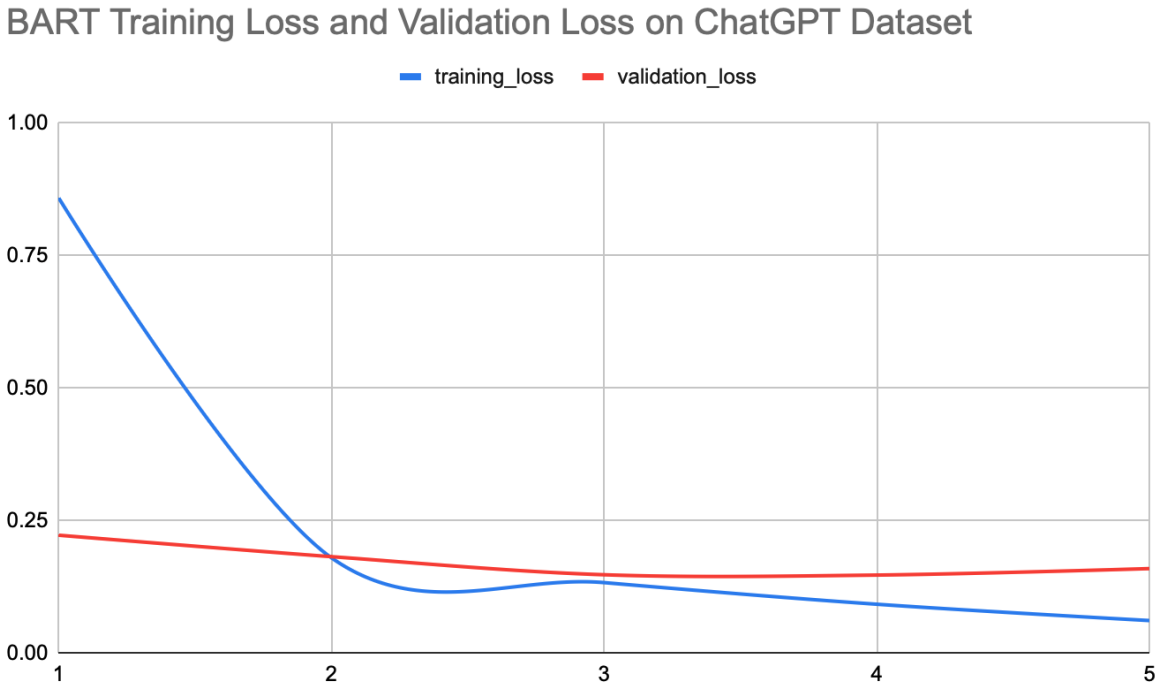

Figure 4.9: BART Training and Validation Loss on 8,000 balanced ChatGPT-labeled dataset[9]

Chapter 5

Research Evaluation

This chapter gives an in-depth evaluation of this research project, comparing the obtained results with appropriate related work, outlining possible validity threats and key research findings.

5.1 Comparison with Related Work

The adoption of Pretrained Language Models (PLMs) in the field of app review classification is a relatively recent development, and the literature in this area is still evolving. Among the existing studies, the research conducted by Mohammad et al.[14] is noteworthy for its comprehensive use of multiple BERT-based language models, specifically ALBERT and RoBERTa, in both tuned and untuned configurations. As detailed in Chapter 2, their experimental design incorporated datasets from prior studies, aiming to surpass previous performance benchmarks. The pinnacle of their findings was the achievement of a new state-of-the-art using a fine-tuned RoBERTa base model, although their reporting focused exclusively on F1-Scores without discussing accuracy metrics.

In parallel, the work of Tianpeng Huang[16] from the previous year also utilized RoBERTa across various datasets but chose to report solely on accuracy, omitting total F1-Scores. This divergence in reported metrics between studies presents a challenge in directly comparing their outcomes comprehensively.

In this context, Table 5.1 facilitates a comparative analysis of PLM performances from this study against the RoBERTa implementations by Mohammad et al.[14] and Tianpeng Huang[16], focusing on both F1-Score and Accuracy. The performance metrics from this research are averaged across different scenarios—Balanced, Unbalanced, and Multi-Label—yielding F1-Scores closely aligned with those reported by Mohammad et al. for their state-of-the-art RoBERTa model. However, the absence of accuracy metrics in Mohammad et al.'s report limits the depth of this comparison, making it difficult to draw definitive conclusions based solely on F1-Scores. Furthermore, differences in dataset characteristics, sizes, and the application of

Table 5.1: Overall Accuracy, F1-Score and Train Time compared to Related Work

	BART	**ELECTRA**	**ERNIE**	**RoBERTa**	**RoBERTa (Mohammad et al.[14])**	**RoBERTa (T.Huang[16])**
Accuracy	0.9518	0.9406	0.9309	0.9419	N/A	0.93
F1-Score	0.9498	0.9389	0.9476	0.9489	0.9836	N/A

fine-tuning methodologies may account for the slight variations observed. Notably, Mohammad et al. focused on binary classification, which can impact performance differently compared to the multi-class approaches evaluated in this project.

In contrast, this project's PLMs demonstrated superior accuracy compared to Huang's reported figures for RoBERTa, potentially attributed to more extensive datasets and refined hyperparameter tuning. To enable a more holistic comparison and robust conclusions, additional performance metrics such as training time and a comprehensive suite of F1-Scores would be beneficial. This would provide a fuller picture of how different approaches and configurations influence the efficacy of PLMs in app review classification.

In the realm of app review classification, this research has demonstrated significant advancements over existing models, particularly with the implementation of BART. BART has notably excelled in accuracy compared to other models evaluated in this study, establishing itself as a new potential benchmark in this domain. Its superior performance underscores the effectiveness of its architecture and training methodology in handling the complexities associated with app review texts.

Conversely, while ELECTRA has exhibited slight underperformance in comparison to the leading models, it presents a compelling case for efficiency and operational advantage. Despite marginally lower performance metrics, ELECTRA's quicker processing time highlights its potential as a benchmark model, especially in scenarios where time efficiency is paramount. This characteristic makes ELECTRA particularly valuable in practical applications where speed and resource optimization are critical.

These findings suggest a nuanced understanding of model selection in app review classification, where both accuracy and efficiency play crucial roles. BART's emergence as a high-accuracy model and ELECTRA's appeal in terms of efficiency offer new perspectives and choices for future research and implementation in this evolving field.

Additionally, this project marks the first implementation of a multi-label classification scenario in this field, revealing that Pretrained Language Models (PLMs) are highly capable of handling complex, multi-label scenarios. The results indicate that PLMs achieve commendable performance in this new and challenging classification task, thereby expanding the utility and applicability of PLMs in nuanced natural language processing tasks. This advancement is critical given that app reviews typically encompass multiple topics, underscoring the importance of multi-label classification in this context. Multi-label classification will allow developers to

not miss key factors from different categories mixed in single review, further enhancing their productivity and satisfaction of app users.

5.2 Validity Threats

In this research project, the primary validity threat arises from the use of ChatGPT for labeling the app review dataset obtained from Hugging Face[9]. While a portion of the dataset was manually validated to ensure accuracy, a significant majority of the data remains unchecked. Consequently, the reliability of the ChatGPT-labeled data cannot be assumed to be at par with that provided by human annotators, whose judgments are generally considered the gold standard in dataset labeling.

Initial observations suggest that models trained on the ChatGPT-labeled dataset exhibit enhanced performance compared to those trained on datasets labeled manually. This enhancement could be attributed to the consistent labeling style of ChatGPT, albeit potentially at the expense of capturing the nuanced understanding of complex reviews that human annotators typically provide.

However, it is also important to acknowledge the inherent challenges associated with manual labeling, particularly the issue of inter-annotator disagreement. Differences in interpretation among human annotators can introduce variability into the labeled data, which, while offering a rich spectrum of human understanding, may also lead to inconsistencies.

Addressing these validity threats requires a balanced approach. One potential strategy could be the implementation of a hybrid labeling methodology, combining the efficiency of automated tools like ChatGPT with the nuanced comprehension of human reviewers. Furthermore, enhancing the rigor of the validation process—perhaps by increasing the proportion of manually reviewed labels—could help in aligning the automated labels more closely with human judgment. Additionally, quantifying inter-annotator agreement and incorporating these metrics into the model training process can provide insights into the variability of human-labeled data and help refine the evaluation of model performance.

5.3 Summary of Key Research Findings

This research has made substantial advancements in the domain of app review classification by exploring multiple aspects of how Pretrained Language Models (PLMs) can be optimized and implemented effectively. Firstly, the study reaffirms the robust capability of PLMs in app review classification by documenting high accuracy scores across various models. This underscores the adaptability and efficiency of PLMs in extracting and processing complex user feedback.

Secondly, our findings demonstrate that BART not only meets but exceeds the performance

metrics of the current state-of-the-art model, RoBERTa, in terms of both overall accuracy and F1-scores. This indicates BART's potential to set a new benchmark in the field, offering a compelling alternative for future implementations.

Furthermore, the research does not overlook the practical implications of deploying these models in real-world scenarios, where efficiency is almost as critical as accuracy. Here, ELECTRA stands out by striking an optimal balance between performance and computational efficiency. Its ability to deliver strong results more rapidly than its counterparts makes it particularly suitable for business applications, where time and resource constraints are crucial considerations.

Additionally, this project introduces the application of multi-label classification in app review analysis by leveraging ChatGPT to annotate an unlabeled dataset in a multi-label format. This innovative approach has shown that all PLMs, particularly BART and ERNIE, excel in this complex classification task, achieving high F1-scores. This success demonstrates that PLMs can be confidently applied to multi-label scenarios, ensuring high performance across a range of evaluative metrics.

In summary, the research presents a nuanced exploration of PLMs in app review classification, highlighting BART and ELECTRA for their respective strengths in performance and efficiency. Moreover, by introducing multi-label classification to the industry, it expands the potential applications of PLMs, paving the way for more versatile and comprehensive analytics in app review processing.

Chapter 6

Conclusion and Future Work

In conclusion, this research project has made significant contributions to the field of app review classification by exploring the efficacy and efficiency of Pretrained Language Models (PLMs). Through thorough experimentation, we have demonstrated that BART surpasses the current state-of-the-art, RoBERTa, in terms of accuracy and F1-scores, thereby establishing a new benchmark for future research and practical applications. Simultaneously, ELECTRA has been highlighted for its optimal balance between performance and computational efficiency, making it an attractive option for enterprises that prioritize rapid processing along with reliable results. Moreover, this study has extended the application of PLMs to multi-label classification scenarios, a first in the industry, using ChatGPT to effectively annotate multi-label datasets. The high performance metrics achieved in this new domain confirm the robustness and adaptability of PLMs to diverse and complex classification tasks. Collectively, these advancements not only enhance our understanding of the capabilities and potential of PLMs in natural language processing but also open up new avenues for their application in analyzing and interpreting user-generated content in the app industry.

6.1 Aims and Objectives: Reflection

This research project outlined three primary objectives in Table 1.1, each conducted to explore specific parts of app review classification using Pretrained Language Models (PLMs). Reflecting on these objectives:

Objective 1 aimed to assemble and refine a series of datasets for the study, involving both multi-class and multi-label scenarios. This objective was successfully met. The datasets were accurately constructed and organized, employing ChatGPT for labeling under a controlled setup to ensure consistency across the datasets. These were further validated through random sampling by human inspection, affirming the reliability of the automated labeling process. The datasets were efficiently organized in designated folders, allowing easy access and usability.

Objective 2 focused on the detailed exploration and fine-tuning of four selected PLMs:

ELECTRA, ERNIE, BART, and RoBERTa. Each model was fine-tuned on the obtained datasets, and hyperparameter optimization was uniformly conducted across the same search space to ensure a fair comparison. The code was developed following best software engineering practices, which not only enhanced the reproducibility of the results but also ensured an organized structure for recording and retrieving evaluation metrics.

Objective 3 involved a comparative analysis of the performance of the PLMs across different computational metrics and dataset scenarios. This objective was achieved through careful evaluation, where models were analyzed under various conditions to provide a comprehensive view of their performance. The analysis was supported by graphs and tables that aided in visualizing performance differences and understanding the outcomes in depth. This objective yielded the proposal of potential new benchmarks of ELECTRA and BART, which indicated superior performance compared to the existing state-of-the-art model RoBERTa.

In summary, the project successfully met all set objectives, substantiating the effectiveness and adaptability of PLMs in the task of app review classification. The findings not only affirm the capabilities of these models but also highlight potential avenues for future research, particularly in refining model efficiency and exploring further applications of multi-label classification.

6.2 Learning Outcomes

This research project provided an invaluable opportunity to explore the architecture of Pre-trained Language Models (PLMs). Theoretical engagement with these models enhanced my understanding of the core architectural patterns that define the evolving landscape of PLMs. Specifically, this project allowed me to explore the foundational principles of BART, ELECTRA, ERNIE, and RoBERTa. Prior to this research, my exposure to PLMs was limited to a basic acquaintance with Transformers and a superficial engagement with BERT during other course units. This project enabled me to explore the current advancements in the field and strengthen my understanding of the background theories underpinning the PLMs used, which was crucial for analyzing results and understanding the underlying mechanics of the models.

From a technical standpoint, the project facilitated my learning in implementing PLMs using Python and the Hugging Face libraries. I dedicated extensive hours to learning the process of model execution, fine-tuning on datasets, and navigating the various training arguments and hyperparameters available. Additionally, I gained practical experience with the Google Collab Pro environment, leveraging its enhanced computational resources, which were essential given the computational demands of the project. For instance, tasks executable on Google Collab's V100 GPU in 15 minutes would take 2-3 hours on my personal computer. However, this also introduced challenges such as managing large model sizes that quickly filled Google Drive storage, necessitating manual management of file deletions.

Moreover, the project significantly advanced my skills in conducting literature reviews. A

substantial portion of my time was allocated to reviewing related work, learning about various implemented approaches, their methodologies, and outcomes. This not only informed my research direction but also enriched my understanding of the field's current state and emerging trends.

Overall, this project was not just an academic exercise but a comprehensive learning journey that enhanced both my theoretical knowledge and technical skills in the realm of PLMs, equipping me with the tools and insights necessary to contribute to and navigate the rapidly evolving domain of machine learning.

6.3 Limitations

This section outlines several potential limitations of this research project that may impact the interpretation of its findings and the generalizability of the results:

1. **Reliability of ChatGPT-Labeled Data:** The dataset used for multi-label classification, labeled using ChatGPT, introduces a degree of uncertainty. Given that automated labeling via ChatGPT is not universally recognized as a fully reliable method, the results obtained from these datasets should be approached with caution. The potential inaccuracies in labeling could affect the accuracy of the models' performance assessments.

2. **Model Selection via K-Fold Cross-Validation:** This project employs K-Fold Cross-Validation for training and then selects the best-performing model from the folds based on accuracy. This method assumes that the fold with the highest accuracy will generalize best on the unseen test set, which might not necessarily always hold true. The variation in a single fold's performance may not be indicative of overall model's capabilities. Including additional metrics such as validation and training loss, precision, recall, and F1 scores (macro or micro) could provide a more in-depth view of model performance and support in selecting the best model for testing.

3. **Limitations in Hyperparameter Tuning:** The hyperparameters selected for fine-tuning were constrained by available resources, which may not have been optimal. Numerous other significant hyperparameters, like warmup steps, could potentially enhance model performance but were not explored due to these constraints. Thus, the models may not have been tuned to their fullest potential, affecting their efficacy and the soundness of the research conclusions.

4. **Constrained Hyperparameter Search Space:** Due to limited computational resources, the hyperparameter search space was restricted, potentially preventing the models from reaching their optimal performance. Expanding the search space with more abundant resources could allow for a broader exploration of hyperparameter values, potentially leading to improved outcomes and insights.

5. **Exclusion of Prediction Time Analysis:** While the research focused on training time to evaluate model efficiency, incorporating prediction time could have offered additional insights. Understanding how long each model takes to make predictions is crucial for practical applications, especially in environments where response time is critical.

These limitations highlight the need for cautious interpretation of the study's findings and suggest areas for further research to enhance the robustness and applicability of the results. Future studies could address these limitations by expanding the resource allocation, exploring a wider range of hyperparameters, and including more comprehensive performance metrics.

6.4 Future Work

The findings and limitations of this research project illuminate several avenues for future work, which could further enhance the robustness and applicability of Pretrained Language Models (PLMs) in app review classification. Here are key areas identified for potential future research:

- **Enhanced Reliability of Automated Labeling:** Future research should focus on improving the reliability of datasets labeled using automated tools like ChatGPT. Implementing hybrid labeling techniques that combine human annotation with automated processes could enhance label accuracy. Additionally, increasing the scale of manual verification within these datasets could validate the reliability of automated labeling, leading to more robust model training and evaluation.

- **Advanced Cross-Validation Techniques:** Expanding the cross-validation methodology to incorporate a broader range of performance metrics would provide a more comprehensive assessment of model generalizability. Future studies could explore ensemble techniques that consider multiple metrics for model selection, such as precision, recall, F1 scores, and loss metrics, to better predict model performance on unseen data.

- **Comprehensive Hyperparameter Optimization:** Addressing the limitations in hyperparameter tuning identified in this study, future work could explore a wider range of hyperparameters and extend the search space. This could be facilitated by the availability of more substantial computational resources, allowing for more exhaustive searches that may uncover optimal model configurations.

- **Resource Allocation for Enhanced Computational Capabilities:** Future projects could benefit from securing access to more powerful computing resources. This would enable a broader hyperparameter search space and potentially shorter training times, making the exploration of more complex models feasible. Additionally, with enhanced resources, more detailed simulations and experiments could be conducted to refine models to their theoretical best performance.

- **Inclusion of Prediction Time in Model Evaluation:** Including prediction time as a key metric in future studies would provide a more holistic view of model efficiency. This is particularly relevant for real-world applications where not only model accuracy but also the speed of prediction is crucial. Future work could explore optimization techniques that balance accuracy with computational efficiency.

- **Exploring ChatGPT for Review Classification:** An exciting direction for future research would be to experiment with using ChatGPT, via its API, as a model for classifying reviews. This involves assessing the feasibility and accuracy of ChatGPT in real-time review classification, exploring its limitations, and comparing its performance against other PLMs in similar tasks. This exploration could potentially leverage the continuous improvements in language model capabilities and provide insights into the practical applications of generative models in business analytics.

By addressing these areas, future research can build on the groundwork laid by this project, moving towards more reliable, efficient, and broadly applicable machine learning solutions in natural language processing.

Word Count: 14798

Bibliography

[1] A. Alokla, W. Gad, W. Nazih, M. Aref, and A.-B. M.Salem. Pseudocode generation from source code using the bart model. 10:3967, 10 2022.

[2] N. Aslam, A. RAMAY, X. KEWEN, and N. Sarwar. Convolutional neural network-based classification of app reviews. *IEEE Access*, 8:1–11, 10 2020.

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, feb 2012.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[5] N. Chen, J. Lin, S. Hoi, X. Xiao, and B. Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. 05 2014.

[6] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[7] J. Daąbrowski, E. Letier, A. Perini, and A. Susi. Mining user opinions to support requirement engineering: An empirical study. In S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, editors, *Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings*, volume 12127 of *Lecture Notes in Computer Science*, pages 401–416. Springer, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[9] H. Face. App reviews dataset. Hugging Face Dataset Repository, 2023.

[10] W. Foundation. English wikipedia dump, Year of Dump.

[11] C. Gao, W. Zheng, Y. Deng, D. Lo, J. Zeng, M. R. Lyu, and I. King. Emerging app issue identification from user feedback: Experience on wechat. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '19*, pages 279–288. IEEE Press, 2019.

[12] A. Gokaslan and V. Cohen. Openwebtext corpus, 2019.

[13] X. Gu and S. Kim. "what parts of your apps are loved by users?" (t). pages 760–770, 11 2015.

[14] M. Hadi and F. Fard. Evaluating pre-trained models for user feedback analysis in software engineering: a study on classification of app-reviews. *Empirical Software Engineering*, 28, 05 2023.

[15] G. Handelma, H. Kok, R. Chandra, A. Razavi, S. Huang, M. Brooks, M. Lee, and H. Asadi. Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods. *American Journal of Roentgenology*, 212:1–6, 10 2018.

[16] T. Huang. Classification of app reviews for requirements engineering using machine learning models. Master's thesis, University of Manchester, April 2023. Supervisor: Dr. Liping Zhao.

[17] A. Insider. Apple's app store launches with more than 500 apps, July 2008.

[18] T. Johann, C. Stanik, A. MollaAlizadeh Bahnemiri, and W. Maalej. Safe: A simple approach for feature extraction from app descriptions and app reviews. pages 21–30, 09 2017.

[19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[20] W. Koehrsen. Overfitting vs. underfitting – a complete example, March 2018. Accessed: 2024-04-12.

[21] J. Koetsier. There are now 8.9 million mobile apps, and china is 40% of mobile app spending. https://www.forbes.com/sites/johnkoetsier/2020/02/28/there-are-now-89-million-mobile-apps-and-china-is-40-of-mobile-app-spending, 2020. Accessed: 2024-03-01.

[22] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, 2017.

[23] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[24] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[25] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[27] M. Lu and P. Liang. Automatic classification of non-functional requirements from augmented app user reviews. 06 2017.

[28] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. pages 116–125, 08 2015.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[30] S. Moghaddam. Beyond sentiment analysis: Mining defects and improvements from customer feedback. pages 400–410, 03 2015.

[31] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. 09 2015.

[32] Z. Qiao, A. Wang, A. Abrahams, and W. Fan. Deep learning-based user feedback classification in mobile app reviews. 2020.

[33] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language*$_u$*nderstanding*$_p$*aper.pdf*, 2018.

[34] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[35] B. Research. Title of the webpage. http://research.baidu.com/Blog/index-view?id=160, June 2017. Accessed: 2024-04-12.

[36] S. Scalabrino, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering*, PP:1–1, 10 2017.

[37] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[38] C. Stanik, M. Haering, and W. Maalej. Classifying multilingual user feedback using traditional machine learning and deep learning, 09 2019.

[39] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Roy. Stat. Soc.*, 36:111–147, 1974.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[41] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. Release planning of mobile apps based on user reviews. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 14–24. IEEE, 2016.

[42] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[43] G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave. Cc-news, 2019.

[44] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. Ernie: Enhanced language representation with informative entities, 2019.

[45] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Bookcorpus, 2015.