LIME-Aided Automated Usability Issue Detection from User Reviews: Leveraging LLMs for Enhanced User Experience Analysis

Bassam Alsanousi
Computer Science and Engineering
University of North Texas
Denton, USA
email: bassamalsanousi@my.unt.edu

Stephanie Ludi
Computer Science and Engineering
University of North Texas
Denton, USA
email: stephanie.ludi@unt.edu

Hyunsook Do
Computer Science and Engineering
University of North Texas
Denton, USA
email: hyunsook.do@unt.edu

***Abstract*—Mobile applications have become essential in today's digital landscape so optimizing their User eXperiences (UX) is essential. Our study explored the application of Large Language Models (LLMs), including some Bidirectional Encoder Representations from Transformers (BERT) family architectures and advanced pre-trained models like Generative Pre-trained Transformer (GPT) by OpenAI GPT-3.5, GPT-4, and Llama 2 by Meta (zero and few-shot), for detecting usability issues in user reviews. The methodology encompassed data preprocessing, sentiment analysis, fine-tuning LLMs, and interpretability techniques, notably Local Interpretable Model-Agnostic Explanations (LIME). The findings indicated that the fine-tuned LLMs, particularly, Robustly Optimized BERT Approach (RoBERTa), XLNet, and DistilBERT were relatively successful in identifying the usability issues, achieving an accuracy rate of 96%. The study also assessed advanced pre-trained models Llama 2, GPT-3.5, and GPT-4, which generally fell short of the performance achieved by fine-tuned models. Finally, we also discovered the use of LIME that helped in understanding the decision-making processes of the fine-tuned models.**

Keywords-Human-Computer Interaction (HCI); Artificial Intelligence (AI); Usability; Large Language Models (LLMs); Local Interpretable Model-Agnostic Explanations (LIME).

I. INTRODUCTION

The rapid advancement of mobile technology over the past decade has revolutionized software creation and usage. According to [1], an average individual spends approximately 4.2 hours on apps daily. This interest has resulted in the expansion of mobile app stores such as Google Play and the Apple App Store. For instance, the Apple App Store has an impressive collection of close to two million apps [2]. These mobile app platforms allow users to provide feedback through ratings and reviews to provide invaluable insights into app updates and user preferences. Analyzing this feedback has become a significant and ongoing research area [3]. User reviews, while concise, offer insights into UX, bug reports, and suggested improvements [4].

Properly interpreting these user reviews for optimizing UX can assist developers in enhancing app quality [5] [6]. However, for popular apps that receive numerous reviews daily, manual analysis can become challenging [4]. A semantically-aware automated approach can effectively identify and categorize usability concerns while analyzing user reviews for usability issues. This approach can potentially reduce analysis time and provide developers with clear feedback to improve product usability. As we encountered in our previous research [7], handling numerous reviews was challenging without such a fully automated approach.

Recent advances in Natural Language Processing (NLP), specifically the development of LLMs, present promising opportunities for automated text analysis. These models have demonstrated remarkable proficiency in understanding and classifying textual data across various domains or tasks [8]–[11].

Chang et al. [8] provided an overview evaluation for LLMs in different domains and NLP tasks, including but not limited to sentiment analysis, text classification, question-answering, and summarization in medical, educational, social sciences, and other areas. [9] investigated the effectiveness of traditional app review classification models compared to LLMs models such as, BERT, XLNet, RoBERTa, and ALBERT. Their finding showed that these LLMs significantly outperformed conventional models, emphasizing the promise of domain-specific LLMs in app review classification. Another work [10] proposed an approach to classify multi-label user reviews into four Non-Functional Requirements (NFRs). Lastly, the authors in [11] introduced a BERT-based model that was fine-tuned to classify issues such as bugs, enhancements, or questions. Their proposed approach demonstrated noteworthy capabilities, particularly excelling in the classification of questions.

Despite their widespread application, the specific efficacy of LLMs in detecting usability issues in user reviews remains ripe for exploration. Our research focuses on this area, investigating the capability of LLMs to accurately identify usability concerns in user reviews. By doing this, we aim to illustrate how LLMs can be effectively used to improve the usability of mobile apps through enhancing UX analysis.

One of the most recognized and commonly referenced definitions of usability comes from ISO 9241, which states, "the extent to which specified users can use a system, product or service to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use " [12]. Our study aims to categorize user reviews into positive and negative through sentiment analysis. We employ sentiment analysis as a strategic tool to guide our focus toward reviews that are more

likely to offer actionable insights for usability enhancements. This classification is an initial step in identifying the usability issues mentioned in the reviews. Previous research suggests that negative reviews are more informative about potential problems [4]. By concentrating on these negative reviews, we aim to pinpoint specific usability aspects related to effectiveness, efficiency, and satisfaction that need improvement.

Several studies [7] [13]–[15] have analyzed user reviews of mobile apps to pinpoint usability challenges and improve software quality through manual or semi-automated approaches. These efforts underscore the growing need for more semantically-aware techniques, aiding developers in refining the quality of these apps [15]. As technology continuously reshapes how users interact and engage, a vast volume of user reviews needs to be analyzed automatically and semantically-aware to improve the usability of mobile apps.

In this context, we contribute by developing an automatic, semantically-aware technique leveraging LLMs to analyze user reviews. This approach will provide developers with a more efficient method to improve the usability of mobile applications and enhance the overall UX With these objectives in mind, our research desires to answer the following questions (RQs):

RQ1: How effectively can LLMs semantically detect usability issues related to effectiveness, efficiency, and satisfaction from user reviews?

RQ2: Which fine-tuned LLMs have the most accurate results in classifying usability issues from user reviews?

RQ3: How do the classifications from pre-trained models via APIs, such as GPT-3.5, GPT-4, and Llama 2, compare to fine-tuned LLMs?

RQ4: How does applying explanation techniques such as LIME enhance understanding model predictions for detecting usability issues?

This paper is structured as follows: Section II looks at recent work related to this study. Section III details the methodology. Section IV highlights the results, and Section V discusses the findings. Section VI addresses potential factors affecting the study's validity. In closing, Section VII concludes the paper and proposes suggestions for future work.

II. Related Work

This section presents some background by examining past approaches to usability evaluation, the recent rise of leveraging LLMs for text classifications, and the importance of LIME in evaluating the model's performance. Earlier studies have introduced historical approaches and recent advances, and we seek to link them to contribute to the amelioration of usability evaluation.

A. Previous Usability Evaluation Approaches

Over the years, researchers have explored different ways to understand users' experiences with software. A notable advancement in this field is the work of El-Halees [13]. This author proposed an innovative approach using opinion-mining techniques to evaluate the subjective usability of software systems. Their experiments showcased the potential of such methods. Drawing inspiration from this, our research delves into the automated analysis of user reviews, especially using LLMs for richer insights.

In a related study, Alhejji et al. [14] looked at problems in mobile banking apps by analyzing user reviews. Their manual analysis sheds light on the strengths and weaknesses of usability for these applications. The focus on sentiment analysis evoked the power of user feedback in improving app usability. Their results indicate that the possibilities of automated techniques imply the coming of an age with an enormous amount of feedback that can be efficiently analyzed to obtain usability insights.

Another study by Alsanousi et al. [7] discussed usability problems in AI-enabled mobile learning apps by analyzing user reviews. Their semi-automated methodology of user review analysis using sentiment analysis and keyword-based approaches gave insights into AI-related challenges that impacted the usability attributes related to user satisfaction, effectiveness, and efficiency. The research strained us toward considering an automatic method to refine usability and UX analysis.

Finally, Diniz et al. [15] looked at how user reviews might indicate specific usability issues. They found that many reviews mentioned common usability problems. However, they also found challenges in manually sorting these reviews. Diniz et al.'s work highlighted the potential benefit of automated tools, especially those that can understand the context and sentiment of user comments.

B. Leveraging LLMs for Text Classification

In recent developments, expanding LLMs have started a new period in text classification, bringing many new opportunities. Hadi and Fard [9] explored the effectiveness of traditional app review classification models compared to LLM models such as BERT, XLNet, RoBERTa, and ALBERT. They developed LLMs by including more domain-specific app reviews in the pre-training process. Their results showed that these LLMs significantly outperformed conventional models, emphasizing the promise of domain-specific LLMs in app review classification without requiring extensively labeled datasets.

Building on the theme of using BERT-based models, Kaur and Kaur [10] introduced MNoR-BERT model. This model was specifically fine-tuned to classify NFRs derived from mobile app user reviews. Their goal was ambitious: to refine the categorization of reviews into distinct NFR types such as, dependability and performance, shifting away from previous keyword-driven and machine-learning methodologies. MNoR-BERT marked a significant evolution in parsing NFRs from user reviews, pushing the boundaries beyond traditional approaches.

Similarly, Siddiq and Santos [11] introduced a BERT-based model, fine-tuned to classify issues such as, bugs, enhancements, or questions. This model was optimized using a vast

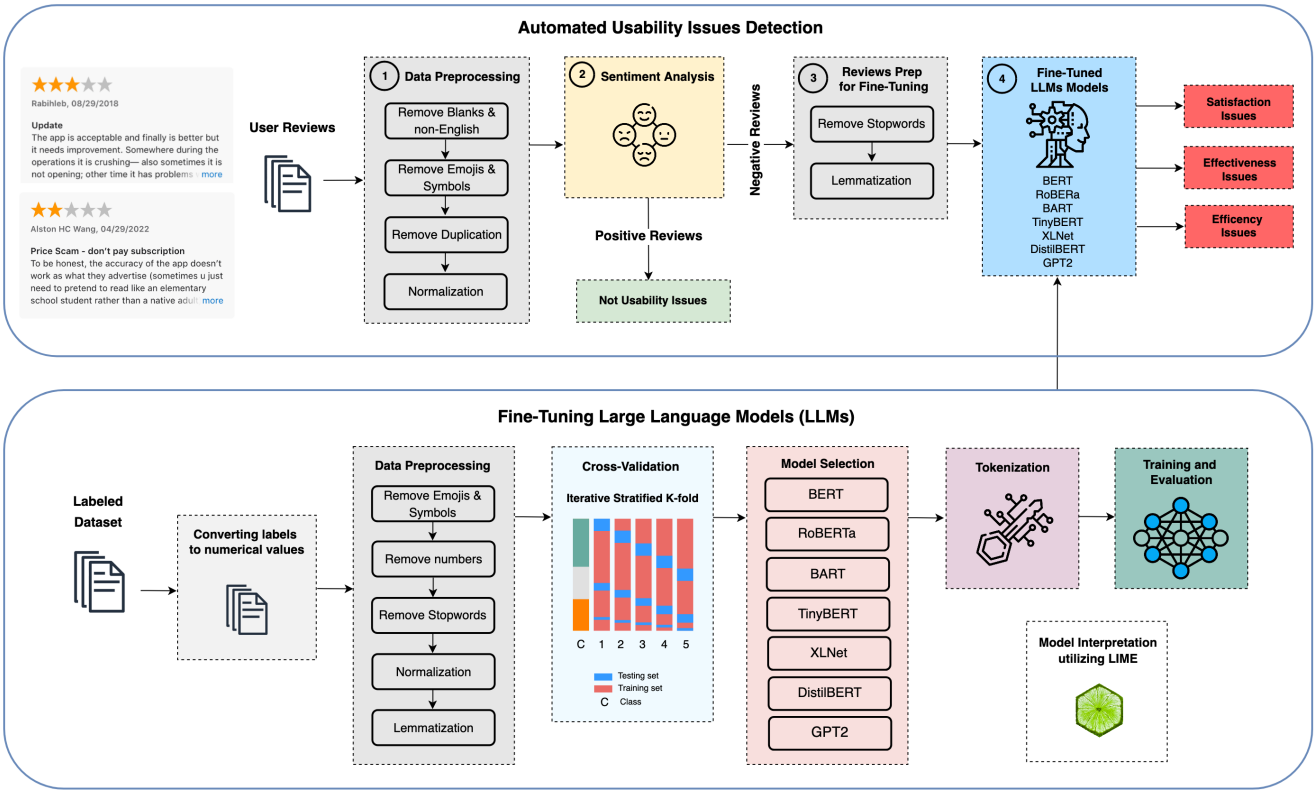

Figure 1. Proposed approach.

dataset from real GitHub projects, encompassing over 800,000 labeled issues. Remarkably, their model demonstrated noteworthy capabilities, particularly excelling in the classification of questions. Our research draws inspiration from their work as we leverage LLMs to explore the detection of usability issues, aiming to enhance the quality of mobile apps and UX.

Finally, BERT has also been used by Algamdi et al. [16] to classify mobile app reviews into five usability categories. They found that BERT was superior to Support Vector Machine (SVM) in multi-class classification. The specified usability factors achieved effective classification of app review issues. In contrast, our approach emphasizes different usability attributes and leverages various LLMs.

C. Model Understandability Through Utilizing LIME

Recent work by Rabby and Berka [17] utilized LIME for multi-class classification interpretation. Their study emphasized the importance of LIME in understanding complex models in classification, offering guidance for practitioners. Similarly, another work by Alharbi et al. [18] demonstrated the usefulness of LIME for evaluating fake news detection models. They established LIME's utility for providing localized interpretability to elucidate predictions and limitations of fake news detectors, improving model trustworthiness. In line with these studies, our research aims to investigate how the application of LIME aids in evaluating the performance of LLMs to detect usability issues from user reviews.

Our research builds on prior studies, acknowledging the importance of advanced technology in analyzing user reviews. These earlier studies mark significant milestones in the evolution of usability analysis and utilizing LLMs for various software engineering tasks. Our contribution lies in applying a semantically-aware sentiment analysis approach using LLMs to detect usability issues related to satisfaction, effectiveness, and efficiency to improve mobile app usability and UX. Additionally, we incorporate LIME to enhance model interpretability, ensuring transparency and trustworthiness in our usability evaluation process.

III. Methodology

In this study, we aim to develop an automated tool utilizing LLMs to detect usability issues from user reviews. As illustrated in Figure 1, our methodology consists of four key steps. This includes initially identifying usability issues in an automated manner, as indicated in the upper box of the figure. Further, the lower box in the figure illustrates the processes for fine-tuning various LLMs and applying LIME to detect and interpret multi-class usability issues.

A. Model Selection Rationale

In our study, we selected seven different LLMs, namely BERT [19], RoBERTa [20], Bidirectional and Auto-Regressive Transformer (BART) [21], the smallest open source pre-trained BERT model (TinyBERT) [22], XLNet [23], DistilBERT [24],

and GPT2 [25], for detecting and classifying user reviews into satisfaction, effectiveness, or efficiency usability issues. Each model has unique specifications, as detailed in Table III [26].

BERT and RoBERTa are based on words in context to improve text understanding, but they are computationally heavy [19] [23]. RoBERTa is an improved version of BERT transformer by longer training and more data, which leads to better performance [20]. Similarly, BART combines BERT and GPT and may therefore be used to generate new text and understand its contents at the same time. BART is trained by corrupting the input text using various techniques. The main limitations of BART are that it needs careful tuning for certain applications [21].

TinyBERT is an attempt to make BERT's capabilities available on a system with limited processing power and energy resources [22]. Likewise, DistilBERT is a small, fast, and cheap training version of BERT. It is the distilled version of BERT, designed in a way that most of its performance has been kept, but it is lighter and faster [24].

GPT-2 is an autoregressive LLM developed by OpenAI to generate coherent and contextually relevant text one word at a time within a sentence [25]. Meanwhile, XLNet utilizes the best from BERT and autoregressive models such as GPT by providing better long-range dependencies capturing multiple benchmarks. XLNet is also computationally heavy [23].

We also selected advanced pre-trained models that can be accessed via Application Programming Interfaces (APIs), that is, GPT-3.5 and GPT-4 by OpenAI and Llama 2 (llama-2-13b-chat) by Meta. GPT-3.5 and GPT-4 have much larger training corpora and model sizes. They generate human-like text, understand context much better, and can even do some specific tasks without additional fine-tuning. Despite their power, GPT-3.5 and GPT-4 are not fully trustworthy in some tasks due to hallucinations that make errors [8] [27], and computationally costly specifically, GPT-4 [28]. On the other hand, Llama 2 by Meta differs mainly in universal robustness across various tasks, including complex reasoning tasks, and low computational costs [8]. However, Llama 2 might give unpredictable, harmful, or biased content because it was trained on publicly available online datasets [29].

B. Usability Factors

Our research aims to identify usability issues within user reviews, considering factors related to satisfaction, effectiveness, and efficiency, aligning with the ISO 9241-11 standard, as outlined in Table I [13]. To identify usability issues, we employed a methodology similar to previous studies [7] [13] [14], focusing on these quality factors found within user reviews. These factors formed the basis of our approach's categories. Table II presents examples of each class and corresponding user reviews marked as usability issues.

C. Data Collection

The proposed approach was trained and evaluated using a manually labeled dataset from mobile banking apps [14]. This dataset included **8,376** reviews—which were categorized as positive, negative, or neutral—from both iOS and Android platforms. However, our objective was to detect usability issues, so we selected only the negative reviews **3,609** highlighting usability issues related to satisfaction, effectiveness, and efficiency. To boost the model's proficiency in differentiating between positive and negative feedback, essential for accurately identifying reviews related to usability issues, we added a randomly selected subset of **232** positive and neutral reviews. Consequently, the final dataset, as shown in Figure 2, comprised a total of **3,841** reviews.

TABLE I
USABILITY FACTORS.

Usability Factors	Definition
Effectiveness	Assesses the users' ability to achieve their goals accurately and completely. Focuses on the extent to which users can accomplish their objectives.
Efficiency	Evaluates the level of effectiveness relative to the resources expended. Helps determine how efficiently users can attain their goals.
Satisfaction	Measures users' overall comfort and attitudes toward the product's usage. Reflects how users find the product's usage enjoyable and satisfactory.

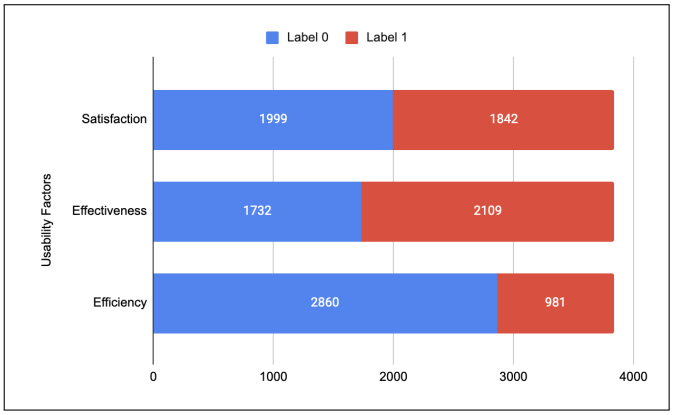

Figure 2. Dataset.

D. Automated Usability Issues Detection

This subsection details the four steps employed in our approach to automatically detect usability issues from user reviews, as illustrated in the upper box of Figure 1. Each step is designed to progressively analyze and extract relevant information from the reviews, ultimately identifying specific usability concerns.

Step 1: *Data preprocessing*

We employed the Natural Language Toolkit (NLTK) [30], a set of libraries for NLP, to enhance the accuracy of our approach by filtering out irrelevant and noisy data. The initial step in preparing our data involved the removal of empty reviews, which did not have enough significant information. Subsequently, we excluded non-English reviews to maintain a focus on a single language. We then eliminated emojis

TABLE II
EXAMPLES OF MULTI-CLASS CLASSIFICATION AND CORRESPONDING USER REVIEWS AS USABILITY ISSUES.

Multi Classes	User Reviews Examples
Effectiveness, Efficiency, and Satisfaction	"The new update is bad and the app is slow and sometimes gives errors"
Satisfaction	"The worst banking app in the world."
Efficiency	"The application is slow and takes a long time to open and navigation between menus is slow"
Effectiveness	"The application needs new maintenance and a new update. The amount does not appear in the account."
Satisfaction and Effectiveness	"The app keeps crashing. It's very frustrating."

and symbols such as, '([/.(] [— @],]', which can muddle the data. Additionally, we removed numbers, as they often do not contribute meaningfully to our text analysis. Another crucial step was the removal of duplicate entries to ensure the uniqueness and relevance of our data. Finally, we normalized the text, converting everything to a uniform style [31].

Step 2: *Sentiment Analysis*

In this phase, we utilized a fine-tuned DistilBERT model for sentiment analysis. Our choice of this model was influenced by its accuracy of 91.3%, as illustrated in Hugging Face [32]. This model categorizes textual data as positive or negative, assigning a confidence score ranging from 0 to 1 [33]. Our analysis focused primarily on negative reviews often linked to apps' issues [4]. These negative reviews were further processed through the fine-tuned LLMs for usability issue classification. Conversely, positive reviews were generally labeled as having no usability issues.

Step 3: *Reviews Prep for the Fine-Tuned Model*

In this step, we again prepared the reviews to align with the fine-tuned models using NLTK library. This involved two fundamental processes: removing stopwords and applying lemmatization. Stopwords commonly used, such as "is," "and," "the," and "are," were removed. We then performed lemmatization, reducing words to their base or root form. For example, "crashing" would be transformed to "crash." These steps helped enhancing the training approach used for the fine-tuned models [34].

Step 4: *Using the Best Fine-tuned LLMs*

During this phase, we used the best fine-tuned model among the seven selected LLMs, namely BERT, RoBERTa, BART, TinyBERT , XLNet, DistilBERT, and GPT2, in detecting and classifying usability issues into satisfaction, effectiveness or efficiency.

E. Fine-tuning LLMs

The lower box in Figure 1 in the proposed approach represents the methods for fine-tuning the seven picked LLMs to detect and classify usability issues from user reviews. First of all, all experiments were executed on an NVIDIA Tesla T4 GPU with 16GB of GDDR6 memory and FP32 CUDA cores. We then configured the batch size to 16 for this fine-tuning process and adjusted Adam's optimizer learning rate to 2e-5, conducting the training over four epochs. This process begins with data preparation, which involves loading user reviews and their associated labels (satisfaction, effectiveness, efficiency) from a CSV file. Then, we prepared our dataset by converting labels into numerical values. This step is essential for managing the encoded data effectively, particularly given the multi-labeled nature of our dataset. Following this, we utilized NLTK and implemented several steps to refine our data to achieve higher precision in subsequent stages. Our first step in this process was the removal of emojis and symbols. We then removed numbers and discarded stopwords, as these elements often introduce noise and do not add significant information [35]. Eventually, we normalized the text, converting all to the same style.

Thereafter, we used a cross-validation technique employing Iterative Stratification, which was especially appropriate for our unbalanced multilabel dataset to guarantee a balanced distribution of labels across training and test sets and reduce the risk of overfitting [36]. This technique provided a strong foundation for model validation and was more thorough than splitting a single dataset [9]. In particular, we carried out the k-fold iterative stratification cross-validation on the dataset five times. After this, we chose one of the seven models, and then we applied the suitable tokenization based on the selected model. Then, we started training and evaluating the chosen model. Next, the same thing has been done for all seven selected models. The training time and validation loss were computed, where every fold involved a specific training and evaluation cycle. We carefully monitored these important performance metrics for each model.

F. Using pre-trained models such as GPT-3.5, GPT-4 and Llama 2

In our comparative analysis, we additionally explored the performance of pre-trained models accessed via APIs using Python scripts. These models include GPT-3.5 and GPT-4 by OpenAI and Llama-2 by Meta (Llama-2-13b-chat few-shot and zero-shot). This was undertaken to assess how these models, without fine-tuning, fare against the fine-tuned LLMs in the previous steps. We utilized the same dataset and evaluation technique for this phase. Each pre-trained model was then applied to perform prediction tasks on this dataset.

G. Evaluation metrics

We employed several crucial metrics to evaluate the proposed approach's effectiveness: overall accuracy, precision,

TABLE III
DETAILS ABOUT THE FINE-TUNED LLMS EMPLOYED IN OUR RESEARCH

Model	Architecture	Parameters	Layers
BERT	bert-base-cased	110M	12
RoBERTa	roberta-base	125M	12
BART	bart-base	140M	6
TinyBERT	General 4L 312D	14M	4
XLNet	xlnet-base-cased	110M	12
DistilBERT	distilbert-base-cased	65M	6
GPT2	gpt2	117M	12

recall, and the F1-score. The concept of 'overall accuracy' pertains to the percentage of cases that have been correctly identified. Precision assesses the percentage of relevant cases determined among those retrieved, whereas Recall measures the percentage of relevant cases accurately retrieved from the entire pool of relevant cases.

For classification evaluations, accuracy evaluates the proportion of correct predictions (true positives and negatives) out of all predictions, as shown in **(1)**. Precision is determined by the percentage of accurately identified instances in a specific category out of all instances categorized into that category, as per **(2)**. Recall, in contrast, is calculated by the proportion of accurately identified instances in a specific category compared to the total number of actual instances in that category, as indicated in **(3)**. 'TP' (True Positive) in these formulas stands for the count of instances correctly classified into a particular category, 'FP' (False Positive) represents the count of cases wrongly classified into that category (thus TP + FP is the total number classified into that category), and 'FN' (False Negative) denotes the count of instances not classified into that category (therefore TP + FN is the total number of cases actually belonging to that category). The F1-score is an evaluative measure that integrates precision and recall by applying a harmonic mean, as outlined in **(4)**.

The formula to compute the **Accuracy** is:

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \quad (1)$$

The formula to compute the **Precision** is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad (2)$$

The formula to compute the **Recall** is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \quad (3)$$

The formula to compute the **F1-score** is:

$$F1 = 2 \cdot \frac{P \cdot R}{P+R} \quad (4)$$

H. LIME Interpretation

Within our methodology, we integrated LIME to interpret the predictions made by LLMs to detect usability issues related to satisfaction, effectiveness, and efficiency and evaluate their performance. LIME allowed us to generate localized explanations for the predictions, shedding light on how LLMs arrived at their decisions.

IV. RESULTS

In this results section, we answer four research questions focused on the capabilities of LLMs in detecting usability issues from user reviews. LLMs were assessed on their classification of usability problems related to satisfaction, effectiveness, and efficiency, considering various performance measures concerning their ability in semantic detection (RQ1), as well as identifying the most accurate models (RQ2). Next, we examined the effectiveness of using pre-trained models available via APIs such as, GPT-3.5 and GPT-4 and llama-2-13b-chat few-shot and zero-shot in contrast to fine-tuned LLMs (RQ3). Lastly, we focus on the use of LIME for the interpretation of model predictions (RQ4).

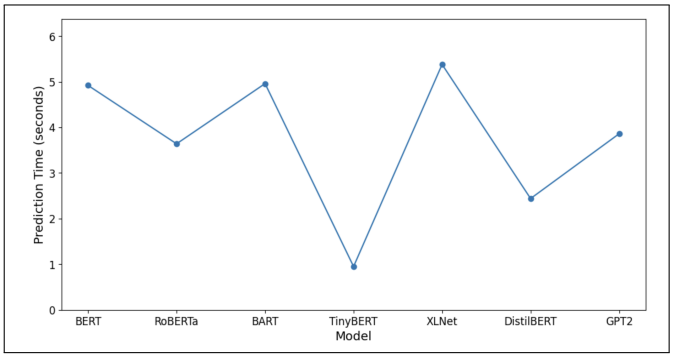

Figure 3. Prediction times of each model.

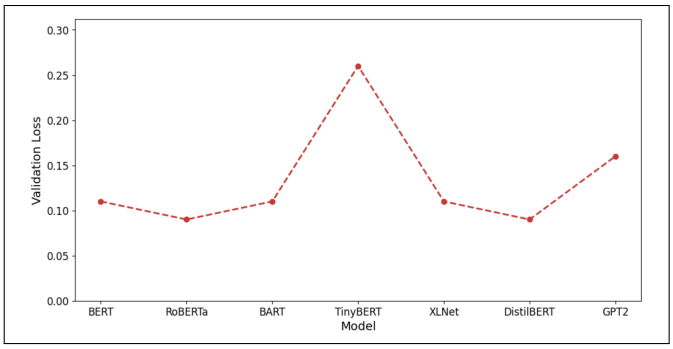

Figure 4. Validation loss of each model.

RQ1: How effectively can LLMs semantically detect usability issues related to effectiveness, efficiency, and satisfaction from user reviews?

LLMs have proven highly effective in semantically detecting usability issues. The performance data, including high accuracy as illustrated in Table IV, indicate that models such as RoBERTa, XLNet, and DistilBERT can accurately discern and categorize nuances in user reviews about effectiveness, efficiency, and satisfaction. The semantic detection is further validated by the LIME results, as shown in Figures 5,6, and 7

TABLE IV
PERFORMANCE RESULTS OF EACH LLM MODEL.

Model	Accuracy	Precision	Recall	F1-score	Training Time (s)
BERT	0.95	0.96	0.94	0.95	1645
RoBERTa	**0.96**	0.96	0.97	0.96	1336
BART	0.95	0.94	0.95	0.95	1619
TinyBERT	0.90	0.89	0.90	0.89	**173**
XLNet	**0.96**	0.95	0.96	0.95	1616
DistilBERT	**0.96**	0.96	0.96	0.96	**806**
GPT2	0.93	0.92	0.93	0.92	1526
Llama 2 - Zero-shot	0.41	0.86	0.71	0.74	-
Llama 2 - Few-shot	0.73	0.88	0.97	0.90	-
GPT-3.5	0.64	0.89	0.89	0.86	-
GPT-4	0.74	0.88	0.97	0.91	-

LIME plots for predicting classes, indicating that the models focused on key terms strongly associated with the users' expressions of usability concerns.

Summary for RQ1: RoBERTa, XLNet, and DistilBERT effectively identified usability issues in user reviews, as shown by their high accuracy and LIME analysis focusing on relevant key terms.

RQ2: Which LLMs have the most accurate results?

Our comprehensive evaluation of LLMs in detecting usability issues from user reviews involved an assessment based on accuracy, precision, recall, F1-score, validation loss, and training time, with detailed results in Table IV. Figure 3 highlights the prediction times of each model, showing notable variation. TinyBERT was the most efficient model, with a prediction time of 0.59 seconds, while XLNet was the least efficient at 5.38 seconds.

Figure 4 presents the validation loss of each model. The validation loss was the lowest for RoBERTa and DistilBERT, totaling 0.09. By comparison, TinyBERT had the highest validation loss of 0.26.

Regarding overall performance, BERT and RoBERTa performed better with an accuracy of 0.95 and 0.96 and an F1-score of 0.95 and 0.96, respectively. BERT's precision and recall were 0.96 and 0.94, respectively, after 1645 seconds of training. RoBERTa reached a lower training time of 1336 seconds after attaining precision and recall rates of 0.96 and 0.97 respectively.

BART and XLNet were also remarkable: 0.95 and 0.96 in accuracy. BART had 0.94 and 0.95 in precision and recall after 1619 seconds of training with an F1 score of 0.95. Similarly, after 1616 seconds of training, XLNet had precision and recall of 0.95 and 0.96 resulting in an F1 score 0.95.

Using the TinyBert architecture, the model's performance showed an accuracy of 0.90, with precision and recall of 0.89 and 0.90, resulting in an F1 score of 0.89 in just 173 seconds of training. GPT2, with training for 1526 seconds, achieved an accuracy of 0.93 and an F1 of 0.92, backed by precision and recall of 0.92 and 0.93.

Meanwhile, DistilBERT balanced high performance and reasonable training time, with an accuracy of 0.96, precision and recall of 0.96, and an F1-score of 0.96, all within 806 seconds of training.

Summary for RQ2: In our LLMs assessment for usability issue detection, RoBERTa, XLNet, and DistilBERT topped accuracy at 0.96 for each, with DistilBERT also excelling in training efficiency. TinyBERT, less accurate at 0.90, had the shortest prediction time of 0.59 seconds, highlighting its operational efficiency, whereas XLNet showed the longest prediction time of 5.38 seconds.

RQ3: How do the classifications from pre-trained models via APIs such as GPT-3.5, GPT-4, and Llama 2, compare to fine-tuned LLMs?

The analysis of advanced pre-trained models accessed via APIs such as Llama 2 and GPT versions, revealed mixed results. Llama 2, in a zero-shot, achieved modest results, notching a scoring accuracy of 0.41, paired with precision and recall scores of 0.86 and 0.71, culminating in an overall F1-score of 0.74. Meanwhile, Llama 2 in few shot demonstrated enhanced metrics, securing an accuracy score of 0.73 and precision and recall scores of 0.88 and 0.97, which gave rise to an F1-score of 0.90.

GPT-3.5 demonstrated moderate metrics, recording an accuracy score of 0.64 and matching precision and recall scores of 0.89, ultimately resulting in an F1-score of 0.86. Progressing further, GPT-4 exceeded anticipated benchmarks with an accuracy score of 0.74 and a precision score of 0.88 while securing an impressive recall of 0.97, collectively contributing to a final F1-score of 0.91.

While advanced pre-trained models like GPT-4 and Llama 2 few-shot demonstrated impressive abilities, it is evident that fine-tuned LLMs such as RoBERTa, XLNet, and DistilBERT still maintain an edge in performance. Their effectiveness in specific applications underscores their continued superiority over general-purpose, advanced pre-trained models.

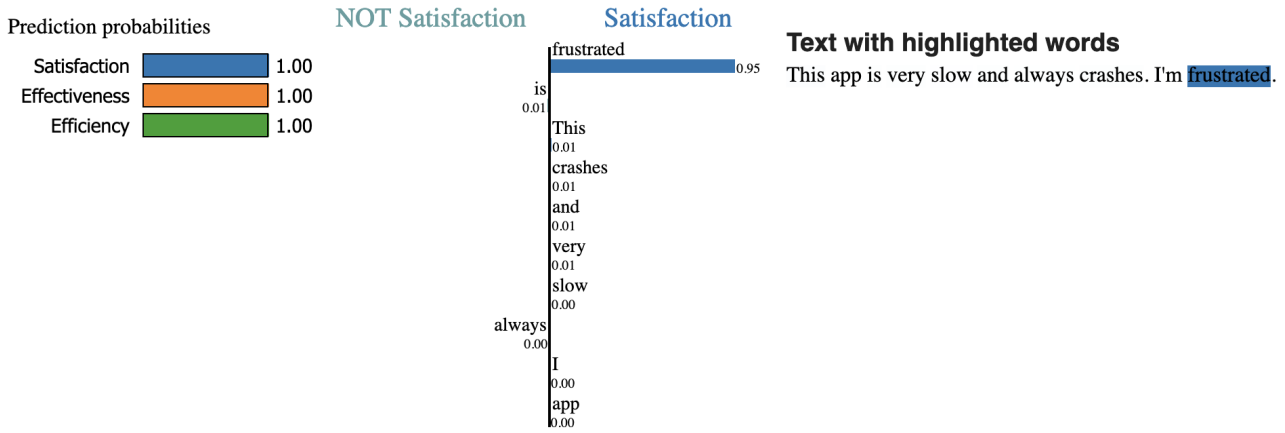

Figure 5. LIME plot for predicting satisfaction class.

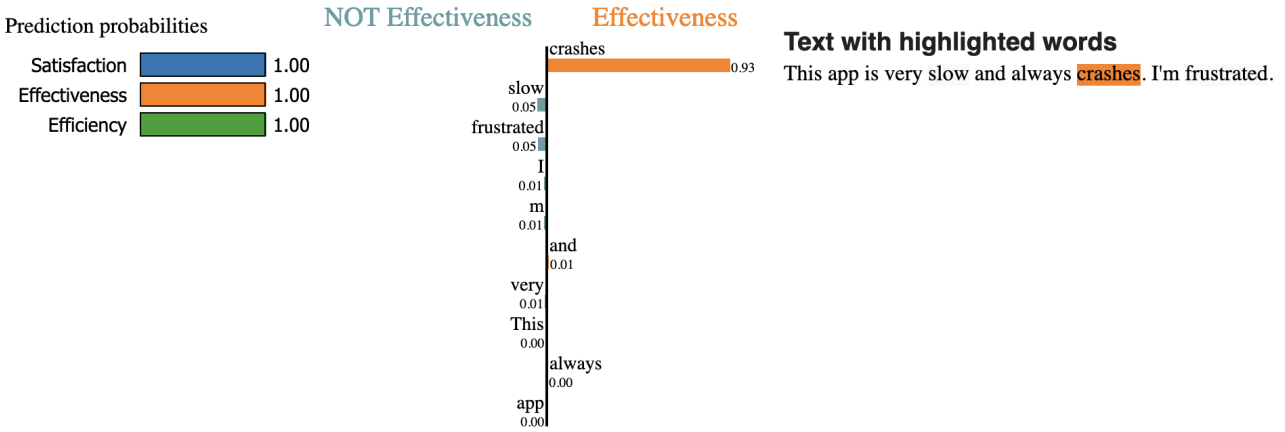

Figure 6. LIME plot for predicting effectiveness class.

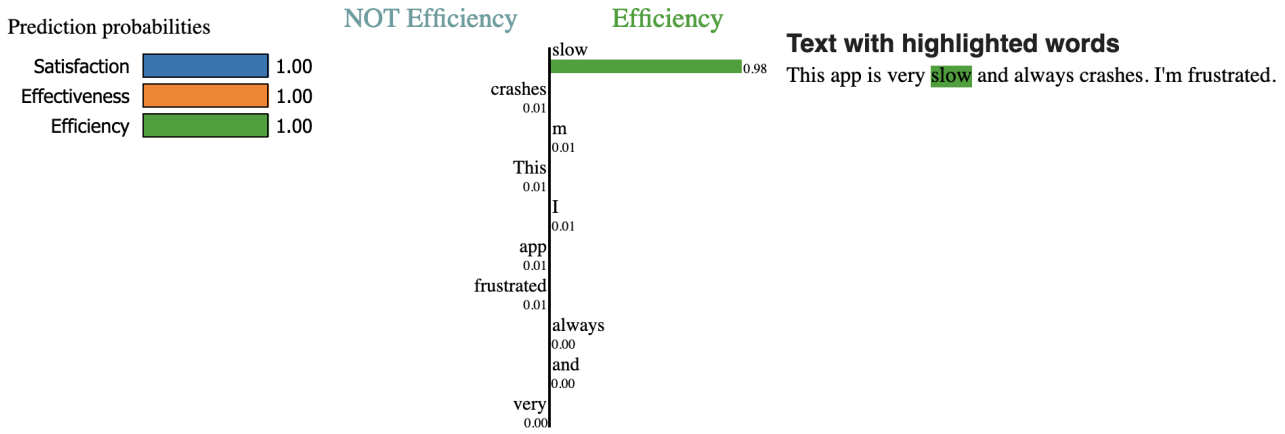

Figure 7. LIME plot for predicting efficiency class.

Summary for RQ3: Advanced pre-trained models accessed via APIs such as Llama 2 and GPT versions, showed mixed results compared to fine-tuned LLMs. Llama 2 had modest accuracy (0.41), GPT-3.5 improved (0.64 accuracy), and GPT-4 further excelled (0.74 accuracy). However, fine-tuned LLMs like RoBERTa, XLNet and DistilBERT still outperformed in specialized tasks, affirming their superiority in specific applications.

RQ4: How does applying explanation techniques such as LIME enhance understanding model predictions for detecting usability issues?

The LIME interpretations have offered significant insights into the decision-making mechanisms of our models, as shown

 ISBN: 978-1-68558-163-3

in Figures 5,6, and 7, which present the LIME plot for predicting satisfaction, effectiveness, and efficiency classes in a user review example ”This app is very slow and always crashes. I’m frustrated.” LIME plots were generated for each class, visually representing how different features influence the predictive outcomes. The bar’s length indicated the prediction probability of each class, with associated class color, blue satisfaction, orange effectiveness and green efficiency.

For instance, the satisfaction, as illustrated in Figure 5, the word ”frustrated” in the user review example has a significant impact, with the model associating it with a lack of satisfaction. In the effectiveness category, as shown in Figure 6, the term ”crashes” in orange color is a strong predictor for detecting effectiveness issues. The LIME plot explicated that when users mention crashes, it highly indicated an effectiveness concern. Similarly, for efficiency in Figure 7, the words ”slow” in green color is also significant feature for efficiency issues. The uniformity of prediction probabilities across all classes, with scores of 1.00, underlined the model’s confidence in its predictions. The high prediction probabilities, coupled with the model’s emphasis on specific keywords, corroborated the relevance of these terms in the context of usability issues.

Summary for RQ4: LIME effectively clarified model predictions for usability issues, highlighting how specific words like ”frustrated,” ”crashes,” and ”slow” impact predictions in areas of satisfaction, effectiveness, and efficiency. High prediction probabilities emphasized the model’s confidence and the relevance of these terms in usability analysis.

V. Results Discussion

This study revealed that LLMs effectively extract usability issues from user reviews. These findings were interpreted and inferred as follows in the discussion below. Our investigation yielded several important insights:

Efficacy in Semantic Analysis: LLMs showed a high performance in the semantic analysis of user reviews. It succeeded in identifying language features that indicate usability problems, thus demonstrating the potential of LLMs to automate and perfect the process of user review analysis.

Accuracy of Models: Among the above models, our findings showed that RoBERTa XLNet and DistilBERT are the best models for detecting usability problems in user reviews regarding their accuracy. This ability plays a significant role in product development and the customer satisfaction process, as an accurate understanding of the UX can result in significant improvements in mobile app analysis.

Enhancement of Model Interpretability: By introducing LIME to the models, the decision-making process has been improved to high levels of interpretability. It explained why specific reviews are flagged as highlighting usability issues, which is a true find for analysis that seeks clarity on model functioning and results.

Model Training Efficiency: DistilBERT and TinyBER models differed significantly in training time, which made them stand out due to the high speed. This implied that there is a tactical situation when frequent retraining is required or where speed of deployment is important.

Predictive Reliability: Validation loss metrics among various fine-tuned models demonstrated a satisfactory predictive ability, with lower loss scores implying greater reliability.

Performance of Advanced Pre-trained Models: The research also measured the performance of advanced pre-trained models accessible through APIs, including Llama 2, GPT-3.5, and GPT-4. Although Llama 2 zero shot and GPT-3.5 had lower accuracy, they continuously exhibited high precision, recall and F1-score. The GPT-4 and Llama 2 few shot variants showed promising performance with increased accuracy and F1 scores. These models could be useful for implementations in situations that do not require further fine-tuning. However, they may not be suitable for some tasks due to expensive computational costs and potential output errors.

Some Real-world Applications of These Models: Fine-tuned LLMs such as RoBERTa, XLNet, and DistilBERT can provide a more profound analysis of user feedback than commercial platforms, such as Appfigures [37], AppFollow [38], or Appbot [39], which might not offer AI capabilities. Additionally, fine-tuned LLMs are open source and many times less expensive than commercial tools, so these LLMs offer significant cost advantages for developers.

In summary, LLMs offer a powerful arsenal of methods for improving the UX through intelligent analysis of user reviews. The study highlights the broad scope of the capacities of both fine-tuned and advanced pre-trained models that provide different abilities for usability issue detection.

VI. Threats to Validity

In presenting the findings of our study, we acknowledge several threats to validity that may influence the interpretation and generalization of our results:

Internal Validity:

Model Overfitting: While efforts were made to prevent overfitting through validation techniques, the possibility that models may have overfitted to particularities in the training data cannot be entirely ruled out. *Parameter Tuning:* The hyperparameters chosen for each LLM were based on a combination of best practices and iterative testing, but

different configurations may yield varied results.

External Validity:

Dataset Specificity: The dataset used was specific to user reviews of mobile banking apps. The findings may not directly apply to other domains or types of textual data. *Language and Cultural Bias:* The study focused on English-language reviews. The models' performance in other languages or cultural contexts remains untested.

VII. Conclusion

Our study highlights the effectiveness of fine-tuned LLMs like BERT, RoBERTa, BART, TinyBERT, XLNet, and DistilBERT as well as advanced pre-trained models, namely, GPT-3.5 and GPT-4 by OpenAI and Llama 2 by Meta in detecting and classifying usability issues from user reviews. RoBERTa, XLNet, and DistilBERT outperformed other models based on the accuracy in evaluating user reviews, advanced pre-trained models also demonstrated promise, despite the significant divergence in accuracy levels. The use of interpretability methods, particularly LIME, played a significant role in improving the transparency and trustworthiness of fine-tuned models. In the next stage, we aim to apply our approach in a specific domain to deeply investigate usability and UX of mobile apps.

References

[1] "App annie," https://www.data.ai, retrieved: April, 2024.

[2] "Apple store," https://www.apple.com/app-store/, retrieved: April, 2024.

[3] M. Tushev, F. Ebrahimi, and A. Mahmoud, "Domain-specific analysis of mobile app reviews using keyword-assisted topic models," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 762–773.

[4] Y. Wang *et al.*, "Where is your app frustrating users?" in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 2427–2439.

[5] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 767–778.

[6] J. Sauro and P. Zarolia, "Supr-qm: a questionnaire to measure the mobile app user experience," *Journal of Usability Studies*, vol. 13, no. 1, pp. 17–37, 2017.

[7] B. Alsanousi, A. S. Albesher, H. Do, and S. Ludi, "Investigating the user experience and evaluating usability issues in ai-enabled learning mobile apps: An analysis of user reviews," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, 2023.

[8] Y. Chang *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.

[9] M. A. Hadi and F. H. Fard, "Evaluating pre-trained models for user feedback analysis in software engineering: A study on classification of app-reviews," *Empirical Software Engineering*, vol. 28, no. 4, p. 88, 2023.

[10] K. Kaur and P. Kaur, "Mnor-bert: multi-label classification of non-functional requirements using bert," *Neural Computing and Applications*, pp. 1–23, 2023.

[11] M. L. Siddiq and J. C. Santos, "Bert-based github issue report classification," in *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering*, 2022, pp. 33–36.

[12] C. Rusu, V. Rusu, S. Roncagliolo, and C. González, "Usability and user experience: What should we care about?" *International Journal of Information Technologies and Systems Approach (IJITSA)*, vol. 8, no. 2, pp. 1–12, 2015.

[13] A. M. El-Halees, "Software usability evaluation using opinion mining." *J. Softw.*, vol. 9, no. 2, pp. 343–349, 2014.

[14] S. Alhejji, A. Albesher, H. Wahsheh, and A. Albarrak, "Evaluating and comparing the usability of mobile banking applications in saudi arabia," *Information*, vol. 13, no. 12, p. 559, 2022.

[15] L. d. N. Diniz, J. C. de Souza Filho, and R. M. Carvalho, "Can user reviews indicate usability heuristic issues?" in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, pp. 1–6.

[16] S. Algamdi, A. Albanyan, and S. Ludi, "Investigating the usability issues in mobile applications reviews using a deep learning model," in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, 2023, pp. 0108–0113.

[17] G. Rabby and P. Berka, "Multi-class classification of covid-19 documents using machine learning algorithms," *Journal of Intelligent Information Systems*, vol. 60, no. 2, pp. 571–591, 2023.

[18] R. Alharbi, M. N. Vu, and M. T. Thai, "Evaluating fake news detection models from explainable machine learning perspectives," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[21] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[22] X. Jiao *et al.*, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[23] Z. Yang *et al.*, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[24] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arxiv 2019," *arXiv preprint arXiv:1910.01108*, 2019.

[25] OpenAI Community, "GPT2," https://huggingface.co/openai-community/gpt2, retrieved: April, 2024.

[26] Hugging Face, "Transformers Documentation (v4.39.3)," https://huggingface.co/docs/transformers/v4.39.3/en/index, retrieved: April, 2024.

[27] OpenAI, "GPT-4," https://openai.com/research/gpt-4, retrieved: April, 2024.

[28] S. Bubeck *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4. arxiv," *arXiv preprint arXiv:2303.12712*, 2023.

[29] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[30] "Nltk: Natural language toolkit," https://www.nltk.org/, retrieved: April, 2024.

[31] C. P. Chai, "Comparison of text preprocessing methods," *Natural Language Engineering*, vol. 29, no. 3, pp. 509–553, 2023.

[32] "Distilbert base uncased finetuned sst-2 english," https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english, retrieved: April, 2024.

[33] E. Chen, "Which factors predict the chat experience of a natural language generation dialogue service?" in *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–6.

[34] U. Naseem, I. Razzak, and P. W. Eklund, "A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter," *Multimedia Tools and Applications*, vol. 80, pp. 35 239–35 266, 2021.

[35] H. Kour and M. K. Gupta, "An hybrid deep learning approach for depression prediction from user tweets using feature-rich cnn and bi-directional lstm," *Multimedia Tools and Applications*, vol. 81, no. 17, pp. 23 649–23 685, 2022.

[36] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the stratification of multi-label data," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III 22*. Springer, 2011, pp. 145–158.

[37] Appfigures, "Overview of App Analytics and Market Data," https://appfigures.com/reports/overview, 2024, retrieved: April, 2024.

[38] AppFollow, "AppFollow: App Monitoring and Optimization," https://appfollow.io, retrieved: April, 2024.

[39] Appbot, "Appbot: Automated App Reviews and Ratings Analysis," https://appbot.co, retrieved: April, 2024.