International Journal of Computing and Digital Systems
2025, VOL. 17, NO.1, 1-19

http://dx.doi.org/10.12785/ijcds/1571023806

Extracting Features from App Store Reviews to Improve Requirements Analysis: Natural Language Processing and Machine Learning Approach

Ishaya Gambo[1], Christopher Agbonkhese[2], Theresa Omodunbi[1], Michael Peter[3], Rhodes Massenon[1] and Israel Odetola[1]

[1]*Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria*
[2]*Department of Digital and Computational Studies, Bates College, Lewiston, ME 04240, USA*
[3]*Industry Ecosystems (INDECO), Interswitch Group, Lagos, Nigeria.*

Received 23 April 2024, Revised 30 January 2025, Accepted 31 January 2025

Abstract: User reviews of mobile apps on platforms such as the Google Play and Apple App Store are a rich and valuable source of information for requirements engineering and software evolution. They reveal the users' needs, preferences, and opinions about the apps and their features. However, extracting and classifying the non-functional requirements (NFRs) from these reviews is a challenging task that requires sophisticated methods and techniques. In this research, we propose a novel approach that uses data mining, natural language processing (NLP), and machine learning (ML) to automatically identify and prioritize NFRs from user reviews of 99 top-rated games in four categories. Sport, Racing, Puzzle, Action and Casual. We collected 271,656 reviews from both platforms and used feature extraction techniques to select and extract the most important NFRs from the reviews. We then used four ML algorithms: Naïve Bayes, Support Vector Model (SVM), Decision Tree J48, and Logistic Regression (LR) to perform sentiment analysis and rank the NFRs based on their importance and relevance. We focused on three types of NFRs: security, flexibility, and maintainability. Our findings show that user reviews can help improve the outcomes of these NFRs and that our approach can help developers understand their users and meet their needs from an NFR perspective, thus increasing user satisfaction and retention.

Keywords: User reviews, requirement engineering, software features, non-functional requirements, functional requirements, machine learning

1. INTRODUCTION

Software requirements have evolved from a fundamental need in software development to a specialized field of research known as requirements engineering (RE). As a critical activity in software engineering, RE establishes the necessary foundations for successful system development [1], [2], [3]. Throughout the software lifecycle, from initial communication to final modeling stages, RE plays a crucial role in the design of robust, fail-free systems [2]. In this context, requirements are expressed as software features, typically categorized as either functional requirements (FRs) or non-functional requirements (NFRs) [4]. While FRs focus on what the system should do to meet end-user needs, NFRs constrain the types of solutions that will satisfy these functional demands.

NFRs encompass crucial aspects such as security, reliability, accessibility, maintainability, reusability, flexibility, performance, usability, and efficiency [5]. The complementary nature of FRs and NFRs is vital for any software project, particularly in mobile applications. Without adequate consideration of both types of requirements, a software system may fall short of stakeholder expectations [6]. Identifying and classifying NFRs presents unique challenges in the field of RE. Unlike their functional counterparts, NFRs are often implicit, subjective, and difficult to quantify [7]. This complexity is further compounded by the diverse nature of NFRs, which can span multiple quality attributes and sometimes conflict with one another. Traditional methods of requirement elicitation, such as interviews and workshops, may not adequately capture the full spectrum of NFRs, especially in the context of widely distributed user bases [8].

In this research, stakeholders are users, and mobile applications are the software in focus. Users provide feed-

E-mail address: ipgambo@oauife.edu.ng, cagbonkhese@bates.edu, tessydunbi@oauife.edu.ng, michael.peter@interswitchgroup.com, ramassenon@pg-student.oauife.edu.ng, odetolaisrael@gmail.com

back on their interactions with mobile apps by leaving ratings and/or comments expressing their views about the application [9]. These reviews provide direct feedback from users on their interactions with mobile applications, offering insights into features that need alteration, addition, improvement, or fixing [10]. The advent of app marketplaces has streamlined the process of crowdsourcing requirements, making it easier for developers to access a wealth of user feedback. However, the unstructured nature of user reviews presents its own set of challenges. Reviews often contain diverse information from users with varying backgrounds, including text abbreviations, spelling errors, and unlabeled data [11], [12], [13]. This complexity makes it difficult for developers to manually process and extract relevant data for requirement improvement [11], [14], [15].

Existing methods for app review analysis have shown promising results in extracting valuable information from user feedback. Groen et al. [16] investigated the presence of NFRs in user reviews of mobile apps, focusing on usability and reliability concerns. Ciurumelea et al. [17] proposed a taxonomy to analyze reviews and codes of mobile apps for better release planning, using machine learning (ML) and information retrieval techniques. Guzman and Maalej [18] introduced an automated approach to filter, aggregate, and analyze app reviews using collocation finding algorithms and topic modeling.

Although some studies have applied ML for app review classification and sentiment analysis [19], [20], none have specifically focused on adapting natural language processing (NLP) techniques to handle large volume of reviews. Chen et al. [21] highlighted the potential of sentiment analysis in understanding user satisfaction, which can be closely tied to NFRs. Luiz et al. [22], Messaoud et al. [23], Raharjana et al. [24], and Ossai et al. [25], applied topic modeling, collocation finding, frequency techniques and sentiment analysis on the reviews to determine their positivity or negativity. This approach enables the identification of key features and sentiment scores, providing valuable insight for the evolution of software requirements in mobile applications.

To address the challenges in NFR identification and classification, researchers have turned to NLP and ML techniques [26], [27], [28]. Jha and Mahmoud [27] utilized NLP techniques to extract functional requirements from user reviews, while Maalej et al. [29] demonstrated the effectiveness of ML algorithms in classifying user feedback into actionable insights. However, these approaches often face limitations in terms of precision and recall, especially when dealing with the multilabeled nature of user reviews, as highlighted by Mcilroy et al. [11].

Despite these advances, several research gaps remain. The accuracy of NFR identification and classification still needs improvement, especially in handling the diverse and often ambiguous language used in app reviews. Furthermore, more robust methods are needed to prioritize NFRs based on their significance and impact on user satisfaction. In this work, our aim is to address these gaps by combining four classification techniques: Term Frequency - Inverse Document Frequency (TF-IDF), Chi Squared (CHI2), and Augmented User Reviews - Bag-of-Words (AUR-BoW) with four ML algorithms: Naive Bayes, support vector machine (SVM), Decision Tree J48, and Logistic Regression. Our approach aims to automatically classify user reviews into four types of NFRs (security, flexibility, and maintainability), FR, and Others. We conducted experiments on the top 99 free apps on Google Play and iTunes, comparing the classification results through all combinations of techniques and algorithms.

This research seeks to know how to get the right features included in a software system to meet its expectations from users' reviews of mobile applications. Secondly, the research seeks to know how to handle various user preferences in a software system based on the user's feedback in the extracted review. Thirdly, the research seeks to know how to extract, classify, and prioritize these extracted features to improve the RE process and the entire system during software maintenance.

In general, this paper aims to fill a gap in the field of RE by proposing a novel model that can extract and classify software features from user reviews of mobile apps. The model can also prioritize features based on their importance and relevance for the RE process. The features can be FR or NFR, which are often overlooked or neglected by existing models. By using user reviews as a source of information, the model can enhance software development practices, particularly in the mobile app domain, by helping practitioners identify and address key features that users want and need in new applications [30].

2. BACKGROUND AND MOTIVATION

In this section, we explain the importance of NFRs in software quality (SQ), provide an overview of the ML and NLP techniques. We analyze the strengths and limitations of each approach discussed. We also present the gaps in current research, using evidence from this literature review, and explain how our proposed approach addresses these gaps. Understanding these methodologies is essential for comprehending the technical approaches and advancements discussed throughout this paper.

A. NRFs in the Context of Mobile App Ecosystems

Software RE is a critical discipline in software development, focusing on identifying, documenting, and managing requirements to ensure project success [31], [32]. The field has evolved from primarily addressing functional requirements to encompassing NFRs, which are quality attributes such as performance, security, and usability [33], [34]. NFRs play a crucial role in determining SQ and user satisfaction, yet they present significant challenges in identification and management due to their often implicit nature [35], [36]. The mobile app ecosystem has transformed

the software landscape, with app stores serving as vital platforms for distribution and user feedback [21], [26]. User reviews play a pivotal role in the evolution of mobile applications, offering direct feedback on product strengths and weaknesses. By analyzing user reviews, developers can identify common themes related to NFRs, such as performance issues or usability concerns, guiding future development efforts and prioritizing enhancements [37]. However, the subjective nature of reviews can complicate interpretation and decision-making.

Despite significant advancements in RE, several limitations persist. The lack of standardized methods for documenting and prioritizing NFRs hampers effective management throughout the development lifecycle [38]. Additionally, the rapid pace of mobile app development often conflicts with traditional RE practices, necessitating more agile approaches [39], [40].

The integration of user feedback from app stores with formal RE processes represents a promising direction for research. Using NLP and ML techniques, developers can potentially automate the extraction and classification of NFRs from user reviews, improving the responsiveness of software development to user needs [27].

B. NLP and ML Techniques

NLP has become a crucial tool in software engineering, particularly for analyzing user-generated content such as reviews of mobile apps. NLP techniques enable the extraction of valuable insights from unstructured text data, facilitating requirements engineering and feature identification.

Several NLP approaches are commonly used in software engineering (SE). Stemming algorithms, such as the Porter Stemmer, simplify words by removing common suffixes, grouping terms with similar meanings [41], [42]. This technique is particularly useful for reducing the dimensionality of text data [17]. Sentiment analysis, exemplified by tools like Sentiwordnet, SentiStrength. SentiStrength is applied to assign sentiment scores, the overall sentiment polarity and strength of a text [43], [29], [44]. This approach helps in understanding user satisfaction and identifying areas for improvement in software applications. Sentiwordnet assigns sentiment scores (positivity, negativity, objectivity) to WordNet synsets, aiding in the computation of term counts for positive and negative sentiments [45], [46].

N-gram extraction captures contextual information by grouping continuous sequences of tokens [47]. This method is valuable for understanding the nuanced structures within textual data. The Bag-of-Words (BoW) model represents text as an unordered collection of words, allowing for the evaluation of term presence and frequency [48]. While simple, BoW can be effective for basic text classification tasks. TF-IDF measures the importance of a word in a document relative to a collection of documents. CHI2 evaluates the independence between terms and classes, identifying features that are most likely to be informative for classification. TF-IDF builds upon BoW by adding a weighting mechanism to account for term distribution across the corpus [49]. This approach helps in identifying important terms within documents. An advanced version of BoW called AUR-BoW [27], [28] can enhance this by utilizing word similarity to augment the user reviews. This means that AUR-BoW not only looks at the individual words but also considers how words relate to each other in terms of meaning, which can provide a richer context for classification.

ML complements NLP in classifying features extracted from reviews. Traditional classification methods include SVM, which excel in handling high-dimensional data and sparse datasets. Naive Bayes classifiers, grounded in Bayes' theorem, offer simplicity and robustness to noise, making them effective for noisy datasets like Twitter data [50].

Decision Trees provide an easily explainable classification method, making them accessible for various applications [48]. Random Forests build upon Decision Trees by introducing randomness and building collections of trees, demonstrating promising results in text classification while mitigating overfitting [47].

The integration of NLP and ML in SE has revolutionized the analysis of user feedback. While automated approaches offer scalability and efficiency, they may miss subtle nuances captured by manual analysis. This paper focuses on hybrid approaches that combine the strengths of both manual and automated methods to enhance feature extraction and classification in mobile app development.

C. Related Work

The exploration of extracting and classifying NFRs from app store reviews through NLP and ML has gained significant attention in recent years. Various studies have showcased diverse methodologies, each with unique strengths and limitations.

The work of Guzman and Maalej [18] introduced a sophisticated method for filtering and aggregating app reviews, successfully leveraging topic modeling for feature grouping. However, relying on the location can present challenges in capturing the diverse linguistic nuances present in user feedback. A crucial aspect addressed in Chen et al. [21] involves integrating sentiment analysis to grasp user satisfaction, which is deeply correlated with NFRs.

Guzman and Maalej [18] developed an automated system for filtering and analyzing reviews using collocation finding and topic modeling. Their results demonstrated successful feature aggregation; however, reliance on collocation could lead to missing nuanced meanings conveyed in reviews. Mcllroy et al. [11] investigated multi-labeled user reviews, emphasizing the capacity of SVM to effectively manage complex feedback. However, the requirement for extensive feature engineering can lead to computational inefficiencies.

Panichella et al. [51] identified linguistic features in reviews to classify technical feedback, demonstrating that decision trees produce effective outcomes. However, this method's dependency on linguistic features raises questions about its adaptability to varied user feedback across different app categories. The study by Maalej and Nabil [43] highlighted the accuracy of binary classifiers but pointed to the limited impact of feature extraction techniques on classification performance.

Jha and Mahmoud [52], have demonstrated the efficacy of NLP techniques in automatically extracting FRs from reviews, emphasizing their potential in requirements analysis. However, while their approach effectively identifies functional aspects, it falls short in addressing the context-specific nuances that characterize NFRs, such as usability and performance factors. Similarly, Maalej et al. [29] showcased the application of ML algorithms in transforming user feedback into actionable insights, yet the generalizability of these models remains a substantial limitation as they often lack adaptability across diverse application ecosystems.

Groen et al. [16] specifically examined NFRs in mobile app reviews, identifying usability and reliability as primary user concerns. Their proposed linguistic patterns facilitated the precision of capturing usability-related issues; nevertheless, a low recall highlighted the limitations of their method in encompassing the breadth of user feedback.

Ciurumelea et al. [17] proposed a taxonomy aimed at better review analysis, with findings indicating a commendable average precision of 80% and recall of 94%. However, identification of code modifications related to user comments yielded a lower precision of 51%. This discrepancy suggests the need for further refinement in connecting user feedback to actual software changes.

Rustam et al. [53] investigated the application of ML techniques to classify user reviews for Shopify apps. The study categorized reviews into positive and negative groups, pre-processed data and used feature engineering techniques, including BoW and TF-IDF, to extract meaningful information. The extracted features were then used to train ML models, specifically the AdaBoost classifier (AC), logistic regression (LR) and random forest (RF), to classify reviews as happy or unhappy. By incorporating linguistic context and semantics with AUR-BoW, can improve the model accuracy.

Di Sorbo et al. [54] investigated the impact of user feedback on the quality and ratings of Android apps by analyzing user reviews. They used a feature extraction technique to identify key issues reported by users that significantly affect app ratings. The authors utilized the ARDOC tool [55] to automatically detect intentions in app reviews and the SURF summarizer tool [56] to extract problem discovery (PD) and feature requests categories from text data. By quantifying the impact of different types of user feedback on app ratings, the researchers revealed valuable insights for app developers to prioritize maintenance and improvement efforts effectively.

Tao et al. [57] introduced SRR-Miner, a novel framework to summarize security-related user reviews and identify security issues and user sentiments in mobile applications. The framework employs a keyword-based approach that leverages BOW, part-of-speech tagging, and triples to extract security-related review sentences. By analyzing the structure and semantics of these sentences, SRR-Miner distinguishes between verbs indicating app misbehaviors, nouns representing aspects, and sentiment words reflecting user opinions. This fine-grained approach provides users with a detailed understanding of security concerns and other users' opinions, enhancing their comprehension of app security issues.

Gao et al. [46] designed the SOLAR tool to automatically summarize helpful user reviews for developers. The tool utilized a trained review helpfulness prediction model to filter non-informative reviews, group topics with corresponding sentiments, and prioritize reviews for each topic. Feature extraction techniques included POS tagging, readability, lexicon, sentiment score (SentiWordNet), polarity, and content dimension (unigram tf-idf).

Zhang et al. [58] introduced a semi-automatic framework for detecting privacy features in App reviews. This framework employed a dependency parsing method to extract relevant features from privacy-related reviews, which were then matched with manually annotated features in App descriptions using phrase similarity. The approach has significant implications for developers, who can use it to identify key topics to prioritize and summarize future changes.

Ossai and Wickramasinghe [25] investigated user concerns regarding diabetes mobile apps by analyzing user comments and developing a sentiment classification model. The study employed a combination of non-negative matrix factorization (NMF) and BoW techniques to extract relevant features from user comments. NMF decomposed comments into thematic topics, revealing underlying sentiments, while BoW represented comments as word collections, focusing on overall sentiment. This integrated approach enabled the model to effectively classify user sentiments and understand concerns about diabetes mobile apps.

In more recent studies, tools such as SAFE [59] and CLAP [47] have advanced the methodologies surrounding feature extraction from user reviews. While SAFE focused on linguistic patterns without extensive computational requirements, its repeatability concerns drawn from evaluation practices necessitate ongoing scrutiny. CLAP's emphasis on clustering user feedback demonstrated applicability in real-world release planning contexts but also revealed the challenges of maintaining classification accuracy amid high variability in user expectations.

Lu et al. [27] consolidated various feature extraction methods, notably TF-IDF, CHI2, and AUR-BoW, for classifying user reviews into NFRs. Their findings underscored the collaborative power of multiple methods, yet they did not explore the role of contextual variance sufficiently, which may limit the robustness of the system in different domains. Furthermore, Johann et al. [59] and Triantafyllou et al. [60] expanded on these methodologies by proposing sophisticated frameworks for feature extraction and categorization, but the intricacies involved in training various ML models can lead to overhead costs in rapid development cycles.

Emerging frameworks such as the MAPP-Reviews method [61] utilized contextual word embeddings to improve feature extraction from reviews, showcasing advancements in processing nuances in language favorably. Efforts to analyze the dynamics of user feedback have underscored the importance of contextual learning, yet their computational complexity raises questions about resource allocation for development teams.

The studies by Tao et al. [57] and Nadeem et al. [62] emphasize the unique challenges associated with extracting NFRs, focusing particularly on security and change requests. These challenges reveal that while sentiment analysis enhances understanding of user needs, it may inadvertently overshadow or obscure key NFRs critical for the development process.

Recent work has sought to integrate advanced ML techniques, such as deep learning (BERT, LISTM, RoBERTa, S-BERT) and transformer models [13], [63], [64], demonstrating a nuanced understanding of user feedback and feature extraction that could substantially streamline requirements evolution. While these methods significantly enhance performance, they often require substantial computational resources and a robust understanding of model behaviors to interpret results effectively.

To synthesize these findings, Table I outlines the strengths and limitations of the various methodologies and approaches discussed. These foundational works underscore the potential of NLP for transforming unstructured data into actionable insights, although they do not directly address NFRs. This gap in the literature indicates the need for more focused explorations that target NFRs in user reviews.

D. Research Gaps

Current research in extracting NFRs from app store reviews faces several limitations. Existing methods often struggle with the diversity and unstructured nature of user feedback, leading to inaccuracies in NFR identification and classification [1] [2]. Moreover, the dynamic nature of mobile app ecosystems presents challenges in prioritizing NFRs based on their significance and impact on user satisfaction [3].

To address these gaps, we propose a new approach that combines ML and NLP techniques to extract and classify NFRs from user reviews. Our approach uses AUR-BoW to enhance the traditional BoW approach by incorporating semantically similar terms. We also propose using four ML algorithms (Naive Bayes, SVM, Decision Tree J48, and Logistic Regression) to perform sentiment analysis and rank NFRs based on their importance and relevance. In this paper, we address the following research questions (RQs) that guide our investigation and analysis:

- **RQ1:** How to identify and prioritize Non-Functional Requirements in user reviews for effective consideration?
- **RQ2:** How can we effectively prioritize Non-Functional Requirements based on their significance and impact?
- **RQ3:** What tools, techniques, and methodologies are appropriate for addressing RQ1 and RQ2 effectively?
- **RQ4:** How can the effectiveness and validity of the methods and tools applied in addressing RQ3 be reliably validated?

These questions are designed to systematically address identified research gaps and contribute to the advancement of RE practices in the development of mobile applications.

Our proposed approach leverages state-of-the-art NLP techniques for feature extraction and sentiment analysis, combined with advanced ML algorithms for classification and prioritization. This integration offers potential advantages over existing methods, including improved accuracy in NFR identification and more nuanced prioritization based on user sentiment and app context.

The timeliness and importance of this research are underscored by the rapid growth of the mobile app market and the increasing emphasis on user-centric development practices. By improving the extraction and prioritization of NFRs from user reviews, our work has the potential to significantly impact the quality of the application, user satisfaction, and the efficiency of the software development process.

In the following sections, we will detail our methodology, demonstrating how it addresses each research question and contributes to the field of RE in mobile app development.

3. METHODOLOGY

In our paper, we used qualitative and quantitative methods, following the case study research approach [66], [67]. In this section, we explain how we designed and implemented a novel model that integrates two methods that have shown promising results in previous studies. We also describe the data analysis techniques and algorithms we used to ensure the validity and reliability of our findings.

TABLE I. Comparative Overview of Key Studies in NFR Extraction

Study	Methodology	Strengths	Limitations
Jha & Mahmoud [52]	NLP for functional requirements	Established foundational techniques	Limited focus on NFRs
Groen et al. [16]	Linguistic patterns for NFRs	High precision in usability feedback capture	Low recall rates
Ciurumelea et al. [17]	Taxonomy of user reviews	High precision and recall in classifying reviews	Performance varies with app types
Guzman & Maalej [18]	Topic modeling + collocation finding	Effective grouping of features	Generalizability limited by dataset size
McIlroy et al. [11]	Multi-label classification	Captured nuanced user feedback	Complex implementation challenges
Panichella et al. [51]	Linguistic features with classifiers	Competitively efficient classifiers	Classification variance based on linguistic complexity
Maalej & Nabil [43]	Probabilistic classification	Superior binary classification performance	Need for tailored classifier selection
Wang et al. [65]	BERT + Attr-CRF	High contextual understanding	High computational resource requirement
McIlroy et al. [11]	Multi-label classification techniques	Captures multiple issues in reviews	Complexity in classifying overlapping issues
Johann et al. [59]	Linguistic patterns	Reduces need for large training dataset	Possible lack of depth in analysis
Lu et al. [27]	TF-IDF, BoW, AUR-BoW	Comprehensive feature extraction	Potential complexity in implementation
Tao et al. [57]	Keyword-based extraction	Simple and effective for security review extraction	Contextual nuances may be overlooked
Gambo et al. [63]	LLMs for feature extraction	Advanced precision and contextual relevance	Complexity of model interpretation
Motger et al. [64]	Large language models	Enhanced performance metrics	Dependence on extensive training datasets

Fig. 1 illustrates our novel method for analyzing user reviews of mobile apps. We collected user reviews from Google Play and iTunes, the leading platforms for Android and Apple apps. We then cleaned and transformed the reviews into a suitable format for feature extraction. We used three techniques to extract the features and preferences of the users: Augmented User Reviews – Bag of Words (AUR-BoW) proposed by Lu et al. [27], TF-IDF, and chi-square (Chi2). We split the dataset into two subsets: 70% for training and 30% for testing. We applied the feature extraction techniques to both subsets to select and extract the most important features from the user reviews. Fig. 2 reflects the flow chart of the conceptual method. It illustrates a comprehensive workflow for analyzing game app reviews to extract and prioritize and rank the identified NFRs.

Additionally, we used a classification model to assign the extracted features and their associated sentiment scores to five types of NFR: Flexibility, Security, and Maintainability. We used four ML algorithms to perform the classification: SVM, Naive Bayes, J48, and LR. We used Python and the scikit-learn package to implement the text mining and ML methods. The following sub-sections detailed the different stages:

A. Data Collection and Analysis

In this study, we collected and analyzed the reviews of the top 99 free apps on Google Play and iTunes, the leading platforms for Android and Apple apps. We used NLP and ML techniques to extract and classify the features and preferences of users. Fig. 3 shows the overview of our analysis method.

For data collection, we built a web crawler that used selenium and Appcomments, two web automation and testing tools, to gather user reviews from Google Play and iTunes. The web crawler visited every page that had an iOS or Android review for one of the 99 top-rated game apps. It extracted metadata from each app, such as its name, title, description, category, device, and star rating. The web crawler also opened a new browser window for each app and clicked on its review pages. Table II reflects the categories and number of Game Apps selected.

In addition, we obtained 200,733 user reviews from both platforms. Each review had a timestamp, a rating, and a comment. The comments revealed the users' issues and opinions about the apps and their feelings towards them. Table III lists the variables in our dataset. We focused on two variables: text reviews and ratings. We filtered the

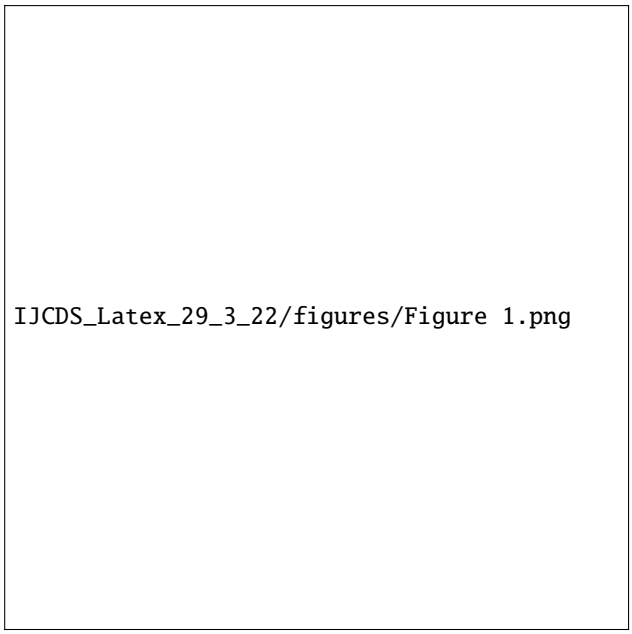

Figure 1. Conceptual view of the Novel Method

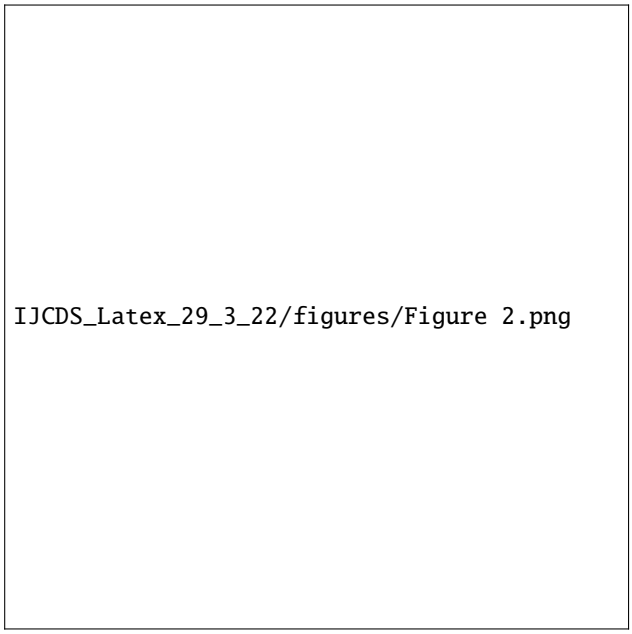

Figure 2. Flow chart of the Proposed Model

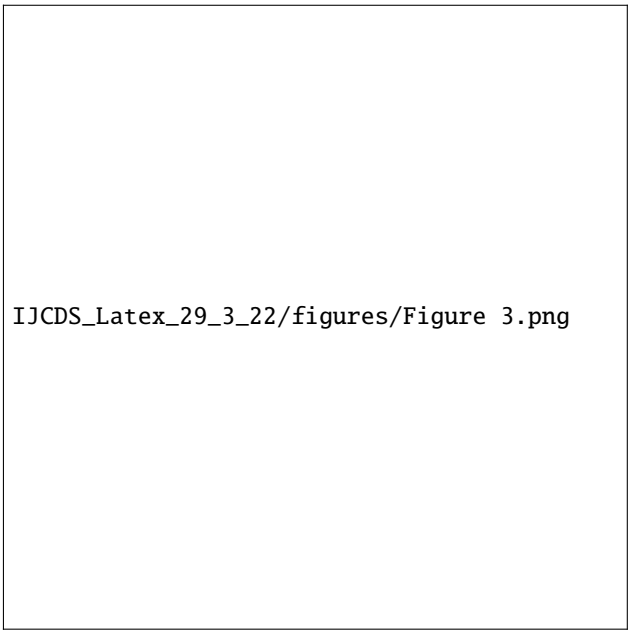

Figure 3. Overview of the data collection process

TABLE II. Categories and number of game apps selected

Categories	**Number of Apps users from Google Play store**	**Number of Apps users from the Apple store**
Sport games	12150 reviews	6292 reviews
Racing games	7821 reviews	6041 reviews
Puzzle games	12854 reviews	26978 reviews
Action games	20300 reviews	1580 reviews
Casual games	9632 reviews	17852 reviews

dataset to include only the most recent reviews from 2020 to 2021 and the highest ratings from 3 to 5. This reduced the dataset to 121,500 user reviews. Table IV shows the sample of reviews we analyzed.

B. Data Preprocessing

We collected user reviews from Google Play and iTunes, the leading platforms for Android and Apple apps. However, these reviews were not ready for ML analysis. They had missing, inconsistent, or irrelevant information that could affect the accuracy and reliability of our results. Therefore, we used NLP techniques to clean and transform the reviews into a suitable format for ML. These techniques included: - Splitting the reviews into sentences - Converting all words to lowercase - Removing punctuation and non-standard words - Removing stop words and short sentences - Lemmatizing the words - Measuring the similarity of sentences. Fig. 4 reflects the order of our preprocessing activities and are all executed in Python, as there are NILTK modules that can perform these tasks. By removing the noise and restoring the meaning of the user reviews, we created a solid foundation for the next step: extracting and

TABLE III. Description of dataset variables

Variables	Description
app_url	App URL
AppName	App name
url	URL of the web page where the review was taken from
author	Name of the author
review	Text review
rating	The number of stars that the author assigned to the app
helpful_count	Number of times the review was considered as helpful
time	Date when the review was written

TABLE IV. Sample reviews collected

App	Category	Platform	Reviews	Rating
Join clash 3D	Action	Google Play	152	4.0
Garena Free Fire	Action	Google Play	260	4.2
Subway Surfers	Action	Apple store	3589	4.4
High Heels	Action	Apple store	781	4.0
Among Us!	Action	Apple store	924	3.6
Water sort Puzzle	Puzzle	Apple store	528	3.9
Candy Crush saga	Puzzle	Google Play	3979	4.6
Call for Duty	Action	Apple store	630	4.4
Temple Run 2	Action	Apple store	1021	4.2
Fruit Ninja	Action	Google Play	612	4.3
Hill Climb Racing	Racing	Apple store	796	4.2
Sonic Dash	Action	Google Play	252	4.6
Fun Race 3D	Racing	Apple store	160	4.2
My Talking Tom	Casual	Google Play	98	4.0
Basketball stars	Sport	Google Play	369	4.5

classifying the features and preferences of the users.

Punctuations, non-standard characters, abbreviations,

IJCDS_Latex_29_3_22/figures/Figure 4.png

Figure 4. Preprocessing Phase of user reviews collected.

acronyms, characterize user reviews and do not contribute to the study. They can distort the outcomes of the model and make the dataset larger than it should. Noise removal and word restoration are the two broad categories of all data preprocessing operations on user reviews before modelling. Sentence tokenization reduces sentences to tokens (words), lowercasing ensures that the model does not interpret capital letters as small letters of the same letter as different, duplicate words removal eliminates redundancy in the dataset.

C. NFR Feature Extraction

The inability of ML algorithms to process user reviews directly requires that user reviews be extracted and transformed into features that can be handled by ML classifiers. This process involves the conversion of textual into numerical representations that ML algorithms can interpret. Three techniques, TF-IDF, CHI2, and AUR-BoW are used in feature extraction. The TF-IDF captures the importance of words within documents while considering their rarity across the entire corpus. This method is particularly useful for identifying distinctive terms in app reviews. The Chi-Squared assesses the independence between a term and a class, which is particularly useful for identifying terms that are strongly associated with specific NFR categories. The AUR-BoW enhances the traditional Bag-of-Words approach by incorporating semantically similar terms, which is particularly beneficial for capturing the diverse ways users might express similar concepts in app reviews. These three feature extraction techniques offer complementary strengths in processing app review data.

To provide more technical depth, this paper provides the algorithms 1, 2 and 3 for the three feature extraction techniques: TF-IDF, Chi-Squared, and AUR-BoW. By

employing these algorithms in conjunction, we can create a robust feature set that captures various aspects of user feedback, ultimately leading to more accurate identification and prioritization of non-functional requirements in mobile app development.

Algorithm 1 TF-IDF Feature Extraction

Require: Corpus of reviews D, vocabulary V
Ensure: TF-IDF matrix M
Initialize TF-IDF matrix M of size $|D| \times |V|$
for each review d in D **do**
  for each term t in V **do**
    $TF \leftarrow \frac{frequencyoftind}{totalwordsind}$
    $IDF \leftarrow \log\left(\frac{|D|}{numberofreviewscontainingt}\right)$
    $M[d,t] \leftarrow TF \times IDF$
  end for
end for
return M

Algorithm 2 Chi-Squared Feature Extraction

Require: Corpus of reviews D, vocabulary V, class labels C
Ensure: Chi-squared scores for each term χ^2
Initialize Chi-squared scores χ^2 of size $|V|$
for each term t in V **do**
  Create contingency table: A: docs with t in class, B: docs without t in class, C: docs with t not in class, D: docs without t not in class
  $N \leftarrow A + B + C + D$
  $\chi^2[t] \leftarrow \frac{N(AD-BC)^2}{(A+C)(B+D)(A+B)(C+D)}$
end for
return χ^2

Algorithm 3 AUR-BoW Feature Extraction

Require: Corpus of reviews D, vocabulary V, augmentation threshold θ
Ensure: Augmented BoW representation B
Initialize BoW matrix B of size $|D| \times |V|$
for each review d in D **do**
  for each term t in d **do**
    $B[d,t] \leftarrow frequencyoftind$
  end for
end for
for each term t in V **do**
  if frequency of t across D ¿ θ **then**
    Find semantically similar terms S to t
    for each review d in D **do**
      $B[d,t] \leftarrow B[d,t] + \sum(frequencyofsind)$ for s in S
    end for
  end if
end for
return B

D. Sentiment Analysis

Extracted words can be related to both FRs and NFRs having positive, neutral, and negative feeling, sentiment analysis is applied to combine data mining and NLP to assign polarity. Subjectivity is not of concern here as emotions are obvious when playing games [68]. The SentiWordNet lexicon contains labeled English words that can be used to determine the opinion polarity in reviews. First, the system identifies each word in a review and looks up its corresponding sentiment scores in SentiWordNet. These scores are categorized into three dimensions: positivity (pos_score), negativity (neg_score), and neutrality (neu_score). For instance, consider the example #1 from Table V, words like *"bad"* and *"error"* would contribute to the negative score, while *"update"* might be neutral or slightly positive depending on context. Next, we aggregate these individual word scores to calculate an overall sentiment score for the entire review. This is done using the following Eq. (1), (2) and (3):

$$s^+ = \sum_i t_{pos\_score_i} 1 \tag{1}$$

$$s^- = \sum_i t_{neg\_score_i} 2 \tag{2}$$

$$polarity\_score = s^+ + s^- + \sum_i t_{word\_neutral} 3 \tag{3}$$

Where s+ represents the sum of all positive scores, s- the sum of negative scores, and the final term accounts for neutral words.

Subsequently, this polarity_score is normalized to fall within a standardized range, typically -1 to 1, where -1 indicates extremely negative sentiment, 0 is neutral, and 1 is extremely positive.

The integration of these sentiment scores into the classification process is where our approach becomes particularly innovative. Rather than treating sentiment as a separate analysis, we incorporate it directly into our feature vectors for ML classification.

Specifically, for each identified feature or keyword related to our NFRs (security, flexibility, maintainability), we augment the feature vector with its associated sentiment score. This creates a richer representation that captures not just the presence or absence of a feature, but also the user's emotional response to it. This enriched feature representation is then fed into our ML classifiers.

E. NFR Classification

Four supervised classifiers are used in this study: SVM, Logistic Regression (LR), Decision Tree (DT-J48), and Naïve Bayes (NB).

The NB algorithm is anchored on the Bayes conditional probability rule with assumptions of independence between characteristics. To form an R reviews group, the classifier computes the probability that a review belongs to a category Ci. This relationship is defined as below, making use of the conditional probability distribution as shown in Eq. (4) and (5):

$$P(c_i \mid R) = \frac{P(c_i)P(R \mid c_i)}{P(R)}4 \tag{4}$$

$$\Omega \quad P(R \mid c_i) = \prod_{j=1}^{n} P(d_j \mid c_i) \tag{5}$$

$$P(c_i) = \frac{N_i}{N} \quad and \quad P(d_j \mid c_i) = \frac{1 + N_{ji}}{M + \sum_{k=1}^{M} N_{ki}}5 \tag{6}$$

where R is the review instance, n is the review length, and P(dj—ci) is the probability/chance of a term dj in a review instance. Naïve Bayes will use the frequency of occurrence of words to define their category.

LR is an algorithm that assigns observations in a sample to discrete classes. In this research, the LR algorithm used for the multiple classification tasks is called multinomial LR. Pandas, Numpy, scikit learn that Python libraries are used to build multinomial LR. The formation of the multinomial logistic regression model requires the corresponding characteristics and targets obtained using the softmax function Thus, the linear regression equation and the softmax function can be given in Eq. (6) and (7):

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n 6 \tag{7}$$

$$F(X_n) = \frac{\exp(X_n)}{\sum_{j=0}^{k} \exp(X_j)}7 \tag{8}$$

F. NFR Prioritization

To classify and prioritize reviews, regression-based ranking is applied. This way, software features having significant relationships with app ratings and user feedback are identified. Keywords are ranked to aid selection of most important keywords. The entries here are the sentences describing the characteristic NFRs and their corresponding comments from the user.

G. Evaluation

While Python programming language and its relevant libraries are used in developing the model, the evaluation techniques to review the system's performance of the system are accuracy, precision, recall and f measure. These metrics are standard measures used in evaluating ML models. The classification matrix is also used to evaluate performance of algorithms via a visualization. The next section describes the results of the model, an assessment of the outcomes and the limitations of the model.

4. RESULTS AND DISCUSSION

The study was limited to five categories of gaming applications (sports games, racing games, puzzle games, action games, casual games) and only the 99 top-rated gaming applications from Google Play and Apple Store were selected. Each crawled exam contains a title, a long description of the exam content, the number of exams, the creation time, the reviewer ID, and the associated rating. Finally, 271,656 user reviews for all 99 gaming applications were accumulated, with an average of 2,744 reviews per application.

Fig. 5 shows that 88% of the feedback came from Google Play store and 12% from iOS apps reviews users. Fig. 6 shows that action games and casual games were the most popular categories on both platforms, with 62,302 and 27,677 feedback respectively. We also examined the ratings of the feedback, ranging from 2 to 4.9 stars. Table V shows the distribution of ratings for each category.

IJCDS_Latex_29_3_22/figures/Figure 5.png

Figure 5. Number of Apps users.

TABLE V. Rating score and total number of user reviews

Rating score	Total number of user reviews
[2, 2.9]	20
[3.0, 3.9]	45,697
[4.0, 4.9]	225,939

The four research questions *(RQs)* contribute to the results presented. This way, we can assess the model for its ability to answer the *RQs* presented. As seen in Fig. 5, there is an overwhelming majority of Google play store apps over the Apple App store. This is unsurprising as there are more mobile apps in Play Store than the App store. While this should mean a higher number of apps from Play store across all app categories, Fig. 6 points to casual games from App Store significantly exceeding those from the Play Store.

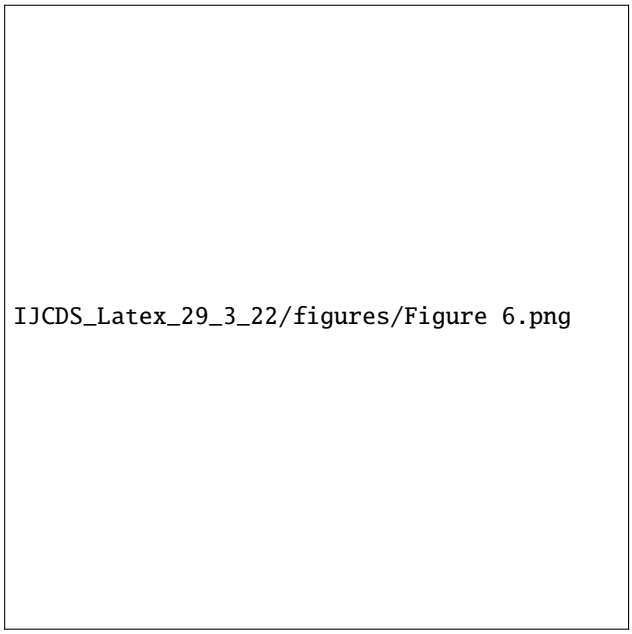

Figure 6. Reviews per category.

A. *RQ.1: How to identify and prioritize Non-Functional Requirements in user reviews for effective consideration?*

Three feature extraction techniques; TF-IDF, CHI2, and AUR-BoW were used to extract high-level features from the cleaned user reviews. The extraction provided unigram words that refer to NFR vocabularies and their frequency. The user reviews underwent clustering to categorize them by sentiments by using SentiWordnet to assign a polarity score to the reviews before eight classification techniques, which were a combination of extraction and ML techniques, were done to give relevant classified features in the three NFRs considered: security, flexibility, and maintainability.

B. *RQ.2: How can we effectively prioritize Non-Functional Requirements based on their significance and impact?*

Frequency, rating, positive and negative reviews were the prioritized input attributes used to rank the selected features based on NFR. Prioritization was done in order of ranking score, where a high score from negative reviews should be met with a low number of positive reviews.

C. *RQ.3: What tools, techniques, and methodologies are appropriate for addressing RQ1 and RQ2 effectively?*

RQ1 was achieved by using TF-IDF, CHI2, and AUR-BoW for feature extraction and four ML algorithms; NB, Naive Bayes, DT-J48, and LR performed the classification before evaluation metrics were applied.

RQ2 was achieved by using regression-based ranking to get prioritized list. The performance of the rank prediction was measured using the ROC curve and the mean square error (MSE) measured regressor performance.

D. *RQ4: How can the effectiveness and validity of the methods and tools applied in addressing RQ3 be reliably validated?*

The use of the confusion matrix to assess the classification results ensured it could be compared to the approach taken by other researchers.

E. *Feature Extraction Sentiment Analysis Results*

Table VI showed sample user reviews that AUR-BoW and TF-IDF techniques are applied to. In the sample in Table VI, there are 10 unigrams extracted from the sample review sentences which refer to the features which are “Update,” “Good,” “Favourite,” “Download,” “Uninstall,” “New,” “Account” “Bad,” “Fix,” and “Error.”. The values of features in Table VII point to the frequencies of unigrams of each feature for the sample data used between the sample reviews #1 and #2. It noticed that TF-IDF of common words (“Update”) was zero, which shows they are not significant. On the other hand, the TF-IDF of “bad”, “new”, “error”, are non-zero. These words have more significance. Table VIII displays the weight of each feature in the sample data. Also, AUR-BoW refers to two-word pair have been considered. Bigrams such as “enjoy game,” “new update,” and “is good” are positive. On the other hand, bigrams like “very bad,” “no play,”,” “fix bug,” and “wont download” have negative orientation. Based on the features extracted from the reviews, a vocabulary bag of words were built by checking them on the Word2Vec.

These words were used to understand specific complaints about features found in the user’s reviews. NFR types included security, availability, operational, portability, maintainability, performance, reliability, scalability, and usability.

In this research, we considered only three types such as Security, Flexibility, and Maintainability aspects. Table IX depicts top 15 topic terms mined from each type of NFR while Table X listed for the vocabulary list. Based on the topics as presented as example in Table IX extracted from the reviews, the senti_score was calculated with the feature types, the associated words and the frequency where these features appeared. Table X shows three examples of sentences scored by SentiWordnet.

These words were used to understand specific praising, complaints about features found in the apps. Table XI showed samples of topics and the sentiments associated with them. The feature extraction output and sentiment are given as input to the classifiers used in this research SVM, NB, DT, and LR algorithms to mine app reviews. The classification and prioritization results will be presented in the next step.

F. *Classification and Ranking Results*

In this section, the results from the features extraction done using TF-IDF, Chi2, and AUR-BoW after the application of ML algorithms SVM, NB, DT, LR, are presented. Accuracy, Precision, Recall and F-measure were selected

TABLE VI. Sample of Reviews

Name App	Target Class	Sample of reviews before preprocessing	Sample of reviews after preprocessing
Pubg Mobile (Android app)	Critical	#1: Very bad, I tried updating the game, It updated and kept saying error failed.	very bad try update game keep say error fail
	Critical	#2: The new update is the worst update I've seen yet. I just want my game to work smoothly, and the server lag to be fixed.	new update is worst want game work smoothly server lag fix
	Positive	#3: This is my favourite mode to play. The graphics are pretty good, the controls are good and gameplay is good.	favourite mode to play graphics pretty controls gameplay good
Subway Surfers	Critical	#4: Please fix this if it is a bug, I really want to play this again but all my boards are gone	fix bug want to play
	Critical	#5: Worst game I have ever played. There is no legal cause to make you fall suddenly on the train.	worst game ever play no legal cause fall suddenly
	Critical	#6: Why this game still not automatically connected to google account. I lost my data	game no connect google account
	Critical	#7: There's no online save option, neither I can login to my previous records nor I can save my current progress. There should be option for google or facebook.	no online save login previous records save current progress google facebook
	Critical	#8: Lost progress. Logged into my accounts and still no items.	lost progress log account no item
8 Ball Pool (iOs app)	Critical	#9: STOP ASKING FOR ACCESS TO MY FACEBOOK FRIENDS!!! The many pop-up ads are irritating enough.	stop ask for access Facebook friends popup ads irritate
	Critical	#10: DO NOT DOWNLOAD! This is set up to cheat you out of your in-game funds to force you to pay.	do not download cheat force to pay
	Positive	#11: It is nice game, easy to log in, very addictive and challenging. Little of considerable advert	nice game easy login addictive challenge advert

TABLE VII. Results of TF-IDF technique on pre-processed data for the two first preprocessed sample reviews

	TF			TF*IDF	
Sentences	**Sample reviews #1**	**Sample reviews #2**	**IDF**	**Sample reviews #1**	**Sample reviews #2**
(very bad try update game keep say error fail)					
(new update is worst want game work smoothly server lag fix)					
Update	$\frac{1}{9}$	$\frac{1}{11}$	$\log\left(\frac{2}{2}\right) = 0$	0	0
New	0	$\frac{1}{11}$	$\log\left(\frac{2}{1}\right) = 0.3$	0	0.27
Bad	$\frac{1}{9}$	0	$\log\left(\frac{2}{1}\right) = 0.3$	0.33	0
Fix	0	$\frac{1}{11}$	$\log\left(\frac{2}{1}\right) = 0.3$	0	0.27
Error	$\frac{1}{9}$	0	$\log\left(\frac{2}{1}\right) = 0.3$	0.33	0

as evaluation metrics for performance of features retrieval, weighted average, and classification results of user reviews. A confusion matrix is to describe the performance of each classifier composed of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) NFRs features.

For each feature extraction, features that are correctly classified are labelled as TP, positive reviews for a feature are labelled FP, and bad reviews labelled as referring to another feature as FN. Confusion matrix is obtained from training data; and comparison with the ML is presented. Fig. 7(a) presents the confusion matrix of the classification results from combining TF-IDF with all ML algorithms to classify NFRs features while Fig. 7(b) presented the

TABLE VIII. Results of AUR-BoW Technique on Pre-Processed Data

	No	Update	Good	Favourite	Download	Uninstall	New	Account	Bad	Fix	Error
#1	1	1	0	0	0	0	0	0	0	0	1
#2	1	1	0	0	0	0	0	0	1	0	1
#3	1	1	1	0	0	0	0	0	0	0	0
#4	0	0	0	0	0	0	0	0	0	0	0
#5	0	0	0	0	0	0	0	0	0	0	0
#6	0	0	0	0	0	0	0	0	0	0	0
#7	0	0	0	0	0	0	0	0	0	0	0
#8	0	0	0	0	0	0	0	0	0	0	0
#9	0	0	0	0	0	0	0	0	0	0	0
#10	0	0	0	0	0	0	0	0	0	0	0
#11	1	1	0	0	1	1	0	1	0	0	0
#12	2	0	0	0	1	1	1	0	0	0	0

TABLE IX. Sample indicator terms 'mined' from the training set

No	Security	Flexibility	Maintainability
1	online	adjust	update
2	access	next	release
3	authorize	multiplayer	new
4	login	match	additional
5	data	level	season
6	account	graphics	change
7	incorrect	time	integrate
8	authenticate	soundtrack	upgrade
9	system	characters	fix
10	verify	unlock	restart
11	lost	offline	load
12	sign in	credits	reset
13	attack	rewards	uninstall
14	sync	interface	download
15	secure	appear	crash

TABLE X. Examples of SentiWordNet Scores in the User Reviews

Topic	Pos_Score	Neg_Score	Neu_Score	Synset	Sentence
update	0.625	0	0.375		#1: Very bad, I tried updating the game, It updated and kept saying error failed.
update	0.575	0	0		#2: The new update is the worst update I've seen yet. I just want my game to work smoothly, and the server lag to be fixed.
update	0.115	0	0.785		#3: This is my favourite mode to play. The graphics are pretty good, the controls are good and gameplay is good.

confusion matrix of the classification results from the combination between AUR-BoW with all ML algorithms to classify NFRs features. A comparison of the results from the ML algorithms was performed for each classification techniques.

Furthermore, three attributes (frequency, rating, positive and negative reviews) were identified as they provide base constructs for priority ranking. Regression-based ranking was used to get prioritized lists. Table XII showed the top 15 significant features from regression analysis. The coefficients with higher value have higher impact on the dependent variable. From the Table XII, update ranks as the highest feature. Exploring the reviews containing this

TABLE XI. Most Common Topics Extracted from the User Reviews with Their Sentiment Scores

Topic	Senti score	Topic	Senti score
online	(+0.25)	adjust	(-0.181818)
update	(+0.874333)	access	(+0.25)
next	(-0.1875)	release	(+0.5225)
authorize	(-0.222222)	multiplayer	(+0.4)
new	(+0.25)	login	(+0.1)
match	(+0.1174)	additional	(0)
data	(-0.086)	level	(+0.6522)
season	(-0.44)	account	(-0.363636)
graphics	(+0.46363)	change	(+0.1159)
incorrect	(-0.14)	time	(+0.3754)
integrate	(-0.1875)	authenticate	(-0.30)
soundtrack	(-0.111111)	upgrade	(+0.8181)
system	(+0.44)	characters	(+0.2778)
fix	(+0.56522)	verify	(+0.4166)
unlock	(+0.5714333)	restart	(+0.76923)
lost	(-0.214268)	offline	(-0.34444)
load	(+0.6363)	sign in	(-0.27777)
credits	(-0.115)	reset	(+0.7277)
attack	(+0.214285)	rewards	(-+0.41666)
uninstall	(+0.37)	sync	(0)
interface	(-0.088888)	download	(+0.75)
secure	(+0.333333)	appear	(-0.0597777)
crash	(+0.67)		

IJCDS_Latex_29_3_22/figures/Figure 7a.png

Figure 7. Confusion matrices of (a) SVM, DT, LR, and NB using the TF-IDF technique; (b) SVM, DT, LR, and NB are shown using the AUR-BoW technique.

feature reveals that the app had become unstable. The explanation revealed that most reviews that contained the world update were complaining that the app had indeed become unusable.

TABLE XII. Number of User's Reviews Corresponding to Each Rating Score

Rank	Feature	Coefficient
1	update	9.17*
2	release	8.89*
4	new	6.12*
5	additional	6.07*
6	change	5.95*
7	integrate	5.70*
8	upgrade	5.23*
9	fix	4.89*
10	restart	4.55*
11	load	3.48*
12	reset	3.91*
13	uninstall	3.58*
14	download	3.02*
15	crash	2.87*

The confusion matrix in Fig. 7(a) illustrates that LR, when combined with TF-IDF, achieved the most accurate results, demonstrating the highest true positive (TP) and true negative (TN) rates. LR successfully predicted 2,781 NFRs, comprising 1,490 positive features and 1,291 negative features, with an error rate of 717 incorrect predictions

(366 positive features and 351 negative features). As shown in Fig. 7(b), LR provided 2,730 correct NFR predictions (1,520 positive features and 1,210 negative features) and made 750 incorrect predictions (320 positive features and 430 negative features). In comparison, AUR-BoW predicted 3,498 NFRs (2,252 positive features and 1,569 negative features), outperforming other classifier models, including SVM, NB, and DT.

Fig. 8 and Fig. 9 illustrated performance measure indices and results for the classification model with TF-IDF, AUR-BoW, respectively. For TF-IDF, the results for classified models in Table XIII showed that across the evaluation metrics, LR had the highest values respectively. Evidently, LR is better suited in classifying NFRs in user reviews than the other classification algorithms.

TABLE XIII. Classification Results Combining ML with TF-IDF

Classifiers Models	**Accuracy**	**Precision**	**Recall**	**F-measure**
SVM	0.70	0.78	0.70	0.74
NB	0.74	0.78	0.74	0.76
LR	0.79	0.80	0.81	0.80
DT	0.74	0.79	0.74	0.76

Table XIV and Fig. 10 present a comparative analysis of F-measure, accuracy, precision, and recall across different classifiers using various feature engineering techniques. The results indicate that LR outperformed SVM, DT, and NB when utilizing AUR-BoW, achieving an 80% performance across all metrics. By integrating AUR-BoW with Chi-squared (Chi2) feature selection, the LR model further refined its predictive capabilities by focusing on more valuable features. Notably, the LR model achieved an 80% classification accuracy for security and maintainability features, although it showed a lower accuracy of 66% for flexibility-related features based on user reviews. Table XV illustrates the classification results of NFRs types—Security, Flexibility, and Maintainability—obtained using the LR classifier. Among these, the Security type achieved the highest F-measure of 82% when employing AUR-BoW, Chi2, and ranking, as depicted in Fig. 11. This demonstrates the effectiveness of the LR model in accurately classifying NFRs, particularly when enhanced with advanced feature engineering techniques.

TABLE XIV. Classification Results Combining Machine Learning with AUR-BoW

Classifiers Models	**Accuracy**	**Precision**	**Recall**	**F-measure**
SVM	0.78	0.82	0.78	0.80
NB	0.80	0.86	0.78	0.82
LR	0.80	0.82	0.80	0.82
DT	0.66	0.72	0.63	0.66

IJCDS_Latex_29_3_22/figures/Figure 8.png

Figure 8. Classification results combining machine learning (SVM, NB, LR, DT) with TF-IDF technique.

IJCDS_Latex_29_3_22/figures/Figure 9.png

Figure 9. Classification results combining machine learning (SVM, NB, LR, DT) with AUR-BoW technique.

G. Implications of Our Research Based on the Classification Results

The implications of our research is four-fold: First, the consistent outperformance of LR over other classifiers like SVM, DT, and NB indicates that LR is a more reliable model for classifying NFRs, especially when combined with effective feature engineering techniques. This suggests

TABLE XV. Classification Results of Three NFR Types Predicted by Rank Prediction with LR Classifier

Type	Proportion of Reviews	Accuracy	Precision	Recall
Security	1437	0.80	0.82	0.80
Flexibility	1350	0.66	0.72	0.63
Maintainability	1234	0.80	0.86	0.79

Note: The proportions of reviews reflect the total number of reviews classified into each type.

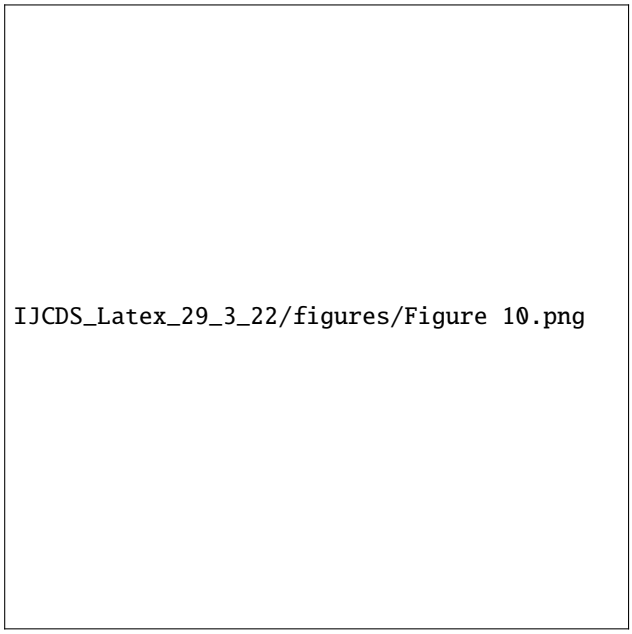

Figure 10. Comparison between Classification results combining machine learning (SVM, NB, LR, DT) with TF-IDF and AUR-BoW technique.

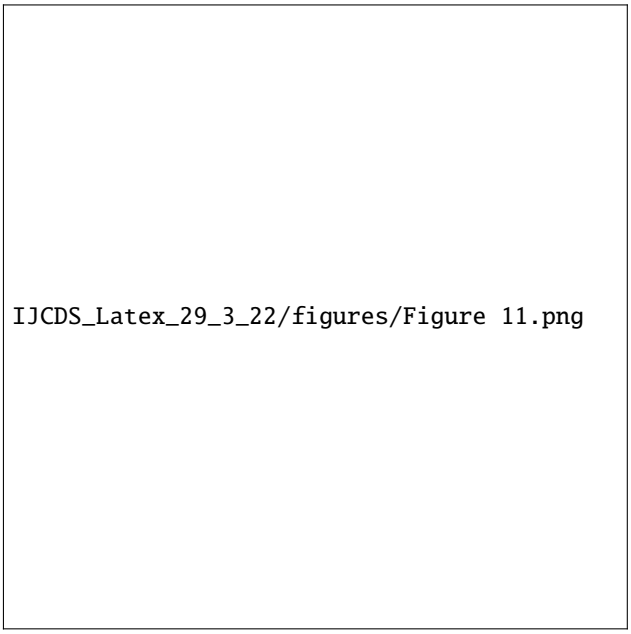

Figure 11. Comparison between the F-measure and proportion of each type..

that for similar tasks, LR should be the model of choice, particularly when accuracy, precision, and recall are critical.

Secondly, the combination of AUR-BoW with Chi-squared feature selection substantially improved the classification performance of the LR model. This underscores the importance of feature engineering in enhancing model performance. It implies that carefully selecting and engineering features can lead to more accurate and meaningful classifications, particularly in complex tasks like NFR classification.

Thridly, the varying accuracy rates across different NFR types (e.g., 80% for security and maintainability vs. 66% for flexibility) highlight that certain NFRs may be inherently easier to classify than others. This suggests that more refined or additional feature engineering may be required to improve classification in domains like flexibility, where accuracy was lower.

Lastly, the high F-measure and overall accuracy achieved by the LR model, particularly in security and maintainability, suggest that this approach can be effectively applied in real-world scenarios. For software development teams, this means that integrating such models could lead to more accurate and automated classification of NFRs, improving the efficiency of requirements analysis and ultimately contributing to the development of higher-quality software.

H. Research Limitations

Owing to the restriction placed by Google Play store's API when accessing reviews, only the latest 2021 could be reviewed. As android apps formed the bulk of the dataset, the possibility of bias in the reviews and an unfair representation of user sentiment is likely. As this research considered ratings below 4, any reviews containing features that could be considered as NFRs in reviews with rating 4 or 5 would not have been accounted for.

5. CONCLUSION AND FUTURE WORK

This research validates existing research on the need for feedback in the design of efficient systems. By extracting and classifying a pool of user reviews, developers can identify NFRs in apps that are already in use and

make improvements to meet user's expectations. Classifying NFRs into maintainability, security and flexibility provides clarity on which NFRs would require the most attention by developers.

App stores unlock a new repository of data for research and as observed in this work, NFRs can be extracted from this data. There are other insights that could be derived from user reviews by utilizing ML and NLP.

Other researchers can consider expanding the number of apps used in analysis and enlarge the scope to cover user response to NFRs in mobile apps. As the Apple store includes both mobile and non-mobile apps, comparing results from separate analysis of each would be an area worth exploring in the future.

Also, the research findings provide a strong case for further exploration of LR models in NFRs classification and the potential for combining them with other advanced feature selection methods. Future research could investigate ways to enhance the flexibility classification or explore the application of these techniques in different domains.

There are many domain-specific apps in the marketplaces. Conducting future evaluation on domains such as health and social media is intended, as domain-specific insights can be gleaned to know features that are to be prioritized to improve user experience and app functionality.

Further, it will be necessary to consider conflict management, especially the identification and resolution of conflicts when classifying the FRs and NFRs. The first step to achieve this can be to leverage on the framework for resolving conflicts, as postulated in [69], [70], [71], [72].

ACKNOWLEDGMENT

The authors acknowledge the support of TETFund and Centre of Excellence Obafemi Awolowo University, Ile-Ife in carrying out the research.

REFERENCES

[1] H. U. Khan, M. Niazi, M. El-Attar, N. Ikram, S. U. Khan, and A. Q. Gill, "Empirical investigation of critical requirements engineering practices for global software development," *IEEE Access*, vol. 9, pp. 93 593–93 613, 2021.

[2] I. Gambo, R. Ikono, P. Achimugu, and A. Soriyan, "An integrated framework for prioritizing software specifications in requirements engineering," *Requir. Eng*, vol. 12, no. 1, pp. 33–46, 2018.

[3] I. Gambo, H. Odukoy, A. Oke, and E. Adagunodo, "Analysis and classification of requirements specification for web application development: A case study approach," *Journal of Computer Science and Its Application*, vol. 27, no. 1, 2020.

[4] S. Keertipati, B. T. R. Savarimuthu, and S. A. Licorish, "Approaches for prioritizing feature improvements extracted from app reviews," in *Proceedings of the 20th international conference on evaluation and assessment in software engineering*, 2016, pp. 1–6.

[5] A. Aurum and C. Wohlin, "Requirements engineering: setting the context," *Engineering and managing software requirements*, pp. 1–15, 2005.

[6] R. S. Wahono, "Analyzing requirements engineering problems," in *IECI Japan Workshop*, vol. 2003, 2003.

[7] J. Dabrowski, E. Letier, A. Perini, and A. Susi, "Mining user feedback for software engineering: Use cases and reference architecture," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 2022, pp. 114–126.

[8] J. "Dabrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: a systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, p. 43, 2022.

[9] B. Lin, N. Cassee, A. Serebrenik, G. Bavota, N. Novielli, and M. Lanza, "Opinion mining for software development: a systematic literature review," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–41, 2022.

[10] T. Iqbal, M. Khan, K. Taveter, and N. Seyff, "Mining reddit as a new source for software requirements," in *2021 IEEE 29th international requirements engineering conference (RE)*. IEEE, 2021, pp. 128–138.

[11] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Software Engineering*, vol. 21, pp. 1067–1106, 2016.

[12] N. Ali, J.-E. Hong, and L. Chung, "Social network sites and requirements engineering: A systematic literature review," *Journal of Software: Evolution and Process*, vol. 33, no. 4, p. e2332, 2021.

[13] I. Gambo, R. Massenon, C.-C. Lin, R. O. Ogundokun, S. Agarwal, and W. Pak, "Enhancing user trust and interpretability in ai-driven feature request detection for mobile app reviews: an explainable approach," *IEEE Access*, 2024.

[14] N. Al Kilani, R. Tailakh, and A. Hanani, "Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications," in *2019 sixth international conference on social networks analysis, management and security (SNAMS)*. IEEE, 2019, pp. 541–548.

[15] T. Ullah, J. A. Khan, N. D. Khan, A. Yasin, and H. Arshad, "Exploring and mining rationale information for low-rating software applications," *Soft Computing*, pp. 1–26, 2023.

[16] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users—the hidden software product quality experts?: A study on how app users report quality aspects in online reviews," in *2017 IEEE 25th international requirements engineering conference (RE)*. IEEE, 2017, pp. 80–89.

[17] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H. C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017, pp. 91–102.

[18] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *2014 IEEE 22nd international requirements engineering conference (RE)*. Ieee, 2014, pp. 153–162.

[19] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE transactions on software engineering*, vol. 43, no. 9, pp. 817–847, 2016.

[20] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen, "Phrase-based extraction of user opinions in mobile app reviews," in *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, 2016, pp. 726–731.

[21] 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.

[22] W. Luiz, F. Viegas, R. Alencar, F. Mourão, T. Salles, D. Carvalho, M. A. Gonçalves, and L. Rocha, "A feature-oriented sentiment rating for mobile app reviews," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1909–1918.

[23] M. B. Messaoud, I. Jenhani, N. B. Jemaa, and M. W. Mkaouer, "A multi-label active learning approach for mobile app user review classification," in *Knowledge Science, Engineering and Management: 12th International Conference, KSEM 2019, Athens, Greece, August 28–30, 2019, Proceedings, Part I 12*. Springer, 2019, pp. 805–816.

[24] I. K. Raharjana, V. Aprillya, B. Zaman, A. Justitia, and S. S. M. Fauzi, "Enhancing software feature extraction results using sentiment analysis to aid requirements reuse," *Computers*, vol. 10, no. 3, p. 36, 2021.

[25] C. I. Ossai and N. Wickramasinghe, "Automatic user sentiments extraction from diabetes mobile apps–an evaluation of reviews with machine learning," *Informatics for Health and Social Care*, vol. 48, no. 3, pp. 211–230, 2023.

[26] N. Jha and A. Mahmoud, "Mining non-functional requirements from app store reviews," *Empirical Software Engineering*, vol. 24, pp. 3659–3695, 2019.

[27] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *Proceedings of the 21st international conference on evaluation and assessment in software engineering*, 2017, pp. 344–353.

[28] R. Santos, E. C. Groen, and K. Villela, "An overview of user feedback classification approaches." in *REFSQ workshops*, vol. 3, 2019, pp. 357–369.

[29] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, pp. 311–331, 2016.

[30] R. Massenon, I. Gambo, R. O. Ogundokun, E. A. Ogundepo, S. Srivastava, S. Agarwal, and W. Pak, "Mobile app review analysis for crowdsourcing of software requirements: a mapping study of automated and semi-automated tools," *PeerJ Computer Science*, vol. 10, p. e2401, 2024.

[31] I. Sommerville, "Software engineering (ed.)," *America: Pearson Education Inc*, 2011.

[32] K. E. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

[33] A. M. Davis, *Software requirements: objects, functions, and states*. Prentice-Hall, Inc., 1993.

[34] D. Leffingwell and D. Widrig, *Managing software requirements: a unified approach*. Addison-Wesley Professional, 2000.

[35] M. Glinz, "On non-functional requirements," in *15th IEEE international requirements engineering conference (RE 2007)*. IEEE, 2007, pp. 21–26.

[36] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Springer Science & Business Media, 2012, vol. 5.

[37] D. Dave and V. Anu, "Identifying functional and non-functional software requirements from user app reviews," in *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, 2022, pp. 1–6.

[38] D. Kumar, A. Kumar, and L. Singh, "Non-functional requirements elicitation in agile base models," *Webology*, vol. 19, no. 1, pp. 1992–2018, 2022.

[39] R. Jindal, R. Malhotra, A. Jain, and A. Bansal, "Mining non-functional requirements using machine learning techniques," *e-Informatica Software Engineering Journal*, vol. 15, no. 1, 2021.

[40] A. E. Yahya, A. Gharbi, W. M. Yafooz, and A. Al-Dhaqm, "A novel hybrid deep learning model for detecting and classifying non-functional requirements of mobile apps issues," *Electronics*, vol. 12, no. 5, p. 1258, 2023.

[41] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 40, no. 3, pp. 211–218, 2006.

[42] M. F. "Porter, "Snowball: A language for stemming algorithms," 2001.

[43] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015, pp. 116–125.

[44] P. R. Henao, J. Fischbach, D. Spies, J. Frattini, and A. Vogelsang, "Transfer learning for mining feature requests and bug reports from tweets and app store reviews," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021, pp. 80–86.

[45] N. M. Rizk, A. Ebada, and E. S. Nasr, "Investigating mobile applications' requirements evolution through sentiment analysis of users' reviews," in *2015 11th International Computer Engineering Conference (ICENCO)*. IEEE, 2015, pp. 123–130.

[46] C. Gao, Y. Li, S. Qi, Y. Liu, X. Wang, Z. Zheng, and Q. Liao, "Listening to users' voice: Automatic summarization of helpful app reviews," *IEEE Transactions on Reliability*, vol. 72, no. 4, pp. 1619–1631, 2022.

[47] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 14–24.

[48] M. V. Phong, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 749–759.

[49] E. Guzman, O. Aly, and B. Bruegge, "Retrieving diverse opinions

from app reviews," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2015, pp. 1–10.

[50] G. Williams and A. Mahmoud, "Mining twitter feeds for software user requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 1–10.

[51] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 281–290.

[52] N. Jha and A. Mahmoud, "Mining user requirements from application store reviews using frame semantics," in *Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27–March 2, 2017, Proceedings 23*. Springer, 2017, pp. 273–287.

[53] F. Rustam, A. Mehmood, M. Ahmad, S. Ullah, D. M. Khan, and G. S. Choi, "Classification of shopify app user reviews using novel multi text features," *IEEE Access*, vol. 8, pp. 30 234–30 244, 2020.

[54] A. Di Sorbo, G. Grano, C. Aaron Visaggio, and S. Panichella, "Investigating the criticality of user-reported issues through their relations with app rating. j softw evol process 33 (3): e2316," 2020.

[55] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "Ardoc: App reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, 2016, pp. 1023–1027.

[56] A. Di Sorbo, S. Panichella, C. V. Alexandru, C. A. Visaggio, and G. Canfora, "Surf: Summarizer of user reviews feedback," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017, pp. 55–58.

[57] C. Tao, H. Guo, and Z. Huang, "Identifying security issues for mobile applications based on user review summarization," *Information and Software Technology*, vol. 122, p. 106290, 2020.

[58] J. Zhang, J. Hua, Y. Chen, N. Niu, and C. Liu, "Mining user privacy concern topics from app reviews," *arXiv preprint arXiv:2212.09289*, 2022.

[59] T. Johann, C. Stanik, W. Maalej *et al.*, "Safe: A simple approach for feature extraction from app descriptions and app reviews," in *2017 IEEE 25th international requirements engineering conference (RE)*. IEEE, 2017, pp. 21–30.

[60] I. Triantafyllou, I. C. Drivas, and G. Giannakopoulos, "How to utilize my app reviews? a novel topics extraction machine learning schema for strategic business purposes," *Entropy*, vol. 22, no. 11, p. 1310, 2020.

[61] V. M. A. de Lima, J. R. Barbosa, and R. M. Marcacini, "Learning risk factors from app reviews: A large language model approach for risk matrix construction," 2023.

[62] M. Nadeem, K. Shahzad, and N. Majeed, "Extracting software change requests from mobile app reviews," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. IEEE, 2021, pp. 198–203.

[63] I. Gambo, R. Massenon, R. O. Ogundokun, S. Agarwal, and W. Pak, "Identifying and resolving conflict in mobile application features through contradictory feedback analysis," *Heliyon*, vol. 10, no. 17, 2024.

[64] Q. Motger, A. Miaschi, F. Dell'Orletta, X. Franch, and J. Marco, "T-frex: A transformer-based feature extraction method from mobile app reviews," in *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024, pp. 227–238.

[65] Y. Wang, J. Wang, H. Zhang, X. Ming, L. Shi, and Q. Wang, "Where is your app frustrating users?" in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 2427–2439.

[66] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[67] R. K. Yin, *Case study research and applications*. SAGE Publications US., 2017.

[68] Z. Gao, Y. Li, Y. Yang, X. Wang, N. Dong, and H.-D. Chiang, "A gpso-optimized convolutional neural networks for eeg-based emotion recognition," *Neurocomputing*, vol. 380, pp. 225–235, 2020.

[69] I. P. Gambo and K. Taveter, "Identifying and resolving conflicts in requirements by stakeholders: A clustering approach." in *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, April 26 – 27, 2021, pp. 158–169.

[70] I. Gambo and K. Taveter, "A pragmatic view on resolving conflicts in goal-oriented requirements engineering for socio-technical systems." in *Proceedings of the 16th International Conference on Software Technologies (ICSOFT)*, 2021, pp. 333–341.

[71] I. P. Gambo and K. Taveter, "Stakeholder-centric clustering methods for conflict resolution in the requirements engineering process," in *Evaluation of Novel Approaches to Software Engineering*, R. Ali, H. Kaindl, and L. A. Maciaszek, Eds. Cham: Springer International Publishing, 2022, pp. 183–210.

[72] S. Shafiq, I. Gambo, M. A. Asghar, J. Ibrahim, and F. Safdar, "A suggestive framework for handling conflicts in stakeholder's emotional goals using machine learning approach," in *2024 International Conference on Engineering Computing Technologies (ICECT)*, 2024, pp. 1–7.