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[3], and Rhodes Massenon[4]

[1,3,4] Department of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
[2] Department of Digital and Computational Studies, Bates College, Lewiston, ME 04240, USA …
[N]Department,Name of the Organization, City, Country
[5]Department of Computer Science, Joseph Sarwuan Tarka University, Makurdi, Nigeria

E-mail address: ipgambo@oauife.edu.ng, cagbonkhese@bates.edu, tessydunbi@oauife.edu.ng, ramassenon@pg-student.oauife.edu.ng

Abstract: User reviews of mobile apps on platforms like Google Play and Apple App Stores 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, and machine learning to automatically identify and prioritize the NFRs from user reviews of 99 top-rated games across 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 machine learning 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 need in software development to a field of research called Requirement Engineering (RE). RE is an activity in software engineering that establishes the necessary foundations for a successful system [1] [2] [3]. From the communication stage to the modelling stage, RE is necessary for the design of failure-free systems, and requirements in this context are software features [4] expressed as functional requirements (FR) or non-functional requirements (NFR) by stakeholders.

In this research, stakeholders are users, and mobile applications are the software in focus. Users provide feedback on their interactions with mobile apps by leaving ratings and/or comments expressing their views about the application. These reviews are a quality source of feedback for software teams on the features to alter, add, improve, or fix [5]. Crowd-sourcing requirements have become necessary in an age where an app can be available to millions of users worldwide. App marketplaces provide all these reviews in one place, making the process of crowdsourcing requirements streamlined and easier to accomplish. As observed in Iqbal et al. [5] the app stores provide rich and valuable source of information for RE and software evolution.

Though user reviews have useful information, it is difficult for developers to process that information manually and get relevant data for requirement improvement [6] [7] [8]. User reviews contain diverse information from different backgrounds with different complaints or reviews comprising text abbreviations, spelling errors, and unlabeled data [6] [9]. This unstructured nature makes user reviews complex as a primary data source for identifying user requirements, particularly NFRs.

On the one hand, FR focuses on what the system should do and how to meet the end user's needs. On the other hand, the NFR constrains the type of solutions that will meet the FR, such as security, reliability, accessibility, maintainability, reusability, flexibility, performance, usability, and efficiency [10]. FR and NFR must complement a software project, especially for mobile applications. The absence of either FR or NFR means the software system might not meet a hundred percent of the stakeholders' expectations [11].

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 review extracted. 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.

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 the features based on their importance and relevance for the RE process. The features can be either FR or NFR, which are often overlooked or neglected by existing models. By using user reviews as a source of information, the model can help practitioners (such as software developers) to identify the key features that users want and need in new applications.

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?

We conducted both analytical and empirical investigations to address the RQs that guide our study. You can find the details of our methods and results in Sections III and IV.

2. RELATED WORK

The need for requirement engineering to characterize systems and manage their entire development lifecycle is highlighted by Wieringa et al. [12]. The requirements, which can be functional or non-functional, define the functions and constraints in the software, and are defined before software development and improved upon after deployment.

Harman et al. [13] developed an Appstore repository mining strategy, extracting, and grouping app feature from app descriptions. Their strategy was to use apply collocations and a greedy algorithm to apps in the Blackberry app store, showing the correlation between app rating and download rank. Their work is limited the non-consideration of other app stores, mining algorithms and requirements.

Villarroel et al. [14] presented a review clustering and prioritization technique to provide app release planning using user reviews. Their approach has been called CLAP (Crowd Listener for releAse Planning). They categorized 1,763 user comments into three categories: bug reports, new feature suggestions, and other categories using the Random Forest machine learning algorithm. They used the Stanford parser to identify negation terms in comments and remove them to process negation terms. DBSCAN was used as a clustering algorithm based on their similarity to identify groups of related notices. The reviews grouped have been prioritized to recommend which group to focus on the next time the app is released. The results showed high accuracy with 86% categorization of user reviews and outperformed AR-Miner. However, it is impossible to take all the reviews in a group to prioritize them because not all reviews have the same importance or the same issue. In addition, assigning an average rating to clusters is not justified because the clusters with the highest rated reviews will have higher priority than the lowest rated. This is why the proposed method considers the calculation of the weight of each review using the rating associated with it.

Khalid et al. [15] assessed 6,390 reviews from iOS apps with low ratings: 1- and 2-star ratings, selecting the 20 most popular apps among them. They identified the most frequent compliments in reviews, discovering that the functional errors have a high number of conformers, seconded by the demand for functionality. Their work is limited by the use of a single mobile platform and the number of apps considered.

Stanik et al. [16] applied traditional ML and deep learning to classify user feedback. The tried to understand the possibility of using deep learning to deliver better outcomes in classifying user feedback. Using inquires, problem reports and irrelevant to group the feedbacks, they gathered 5 million English tweets and 1.3 million Italian tweets directed as support accounts of telecommunication companies. They applied both ML techniques and deep learning to determine which technique delivers the best result and used only 10,000 English tweets and 15,000 Italian tweets. From their study, there was no significant difference in the results given by ML and deep learning.

Grano et al. [17] [18] combined NLP, sentiment analysis, and text analysis to classify android apps and user feedback. An F-droid repository was used to conduct the study. About 280,000 user reviews from about 288 different mobile apps were extracted from the repository through a web crawler developed. The results showed useful information found in the extracted and analysed data that help understand the possible correlations between the data metrics collected in an individual app and terms of integration of a set of multiple apps.

Additionally, Guzman et al. [19] developed a solution called ALERTme, which uses the ML approach to classify, group, and rank tweets related to the requirement. The data set consists of 68,108 tweets from three popular software: Shopify, Slack and Dropbox. Their results highlighted the potential of Twitter data in gathering feedback from users as well as the applicability of techniques used for app store data in mining feedback from Twitter.

Zhao et al. [20] used intelligent mining techniques to extract software development information from reviews in mobile app marketplaces. The aim was to assess the quality in the development of techniques for mining user reviews. From their findings, quality control measure being applied to topics in user reviews is not sufficient in ensuring feedback serves as a source for RE process improvement.

Zhao et al. [20] performed a comparative analysis of publications and discovered that recent papers focus on extraction techniques and earlier papers focused on finding discussed topics. While the central aim of the study was achieved, it also provides rich insights on areas that could be explored in future research, including automating the transformation of extracted user feedback into RE specifications and the use of modelling alongside existing extraction tools.

Malgaonkar et al. [21] used extraction approaches to mine user feedback to identify and prioritize feature improvement from app reviews. The study extracted data from Google Play Store using Google API. The data mined were prioritized in considering the following parameters: bug report and frequency of a reported enhancement request.

To detect potentially malicious applications, Gorla et al. [22] used K-Means clustering to group app descriptions. The applications' API details contained in their manifesto was used in the development of a one-class SVM. Every group had an was used to train SVM at one class, allowing for the detection of apps with abnormal API usage—¬an indicator of possible malware.

Chen et al. [23] designed a DiffCom, a malware detection system that requires no prior knowledge of malware. Retrieving over 1 million apps from Google Play store and third-party Android stores, only a sample of 50,000 was used. DiffCom had a false positive rate of 0.04 and when the entire data was applied, it could detect 127,429 cases of malware.

Batyuk et al. [24] developed an APK analyzer and tested it using 1,856 apps from the Play Store. This static analyzer was able to detect apps accessing private credentials (167) and those that could contain spyware (114). This work has evolved into the Androlyzer, a static analysis tool.

Chia et al. [25] in their study of privacy risks assessed app ratings from three app marketplaces: Chrome, Facebook, and Google Play. While they discovered a strong correlation between review count and popularity, there was none when permissions requested and risk to privacy were considered. Their work shows the non-effectiveness of ratings as a privacy indicator. Since new apps usually have fewer ratings, suspicious apps fly under the radar as ratings are not sufficient to alert users.

Zhu et al. [26] extracted data from Apple store and used hypothesis testing to find apps that could be dangerous by formulating an algorithm for Automatic Detection of Security Levels. They used app ratings and user reviews of 15,045 apps.

Tao et al. [27] developed a methodology for summarizing issues relating to security from sentiments in user reviews. This approach called the SRR-Miner considered only neutral and negative sentiments in user reviews extracted using Vader, an NLTK model. POS tags of words were used to identify verbs and nouns related to security grouped in 19 keywords based on security. The results showed a good accuracy performance as the F1 scores were between 0.83 and 0.85. The SRR-Miner was only applied to Google Play store reviews of 17 apps and how long the app has existed is not considered.

Mukherjee et al. [28] analysed compatibility issues of the NFRs by analysing app commits and app reviews. They used four classification algorithms used in Machine Learning (Naive Bayes, SVM, LR, and Random Forest) to see how each performs in analysing compatibility. They collected 258,056 commits and 205,847 reviews from GitHub and Google Play Store. Applying a keyword search and building a list of words, they identified 48,262 messages having at least one word from the list of words. The results showed that 3.16% of app features are dedicated to compatibility issues, and 4.30% are compatibility related. They just focused only on the Google Play store as an experiment area.

While existing research have performed extraction and classification of characteristics from user reviews, there are uncovered areas. These include handling large number of mobile apps reviews, classifying FRs and NFRs in reviews, leveraging ML and prioritizing NFRs in user reviews, particularly at the elicitation stage. This research addresses these and other areas of concern. The next section would outline the methodology used in approaching the research problem, from data collection to design a model for the system.

A. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

3. METHODOLOGY

We employed both qualitative and quantitative methods in our paper, following the case study research approach [29] [30]. 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.

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. [31], 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.

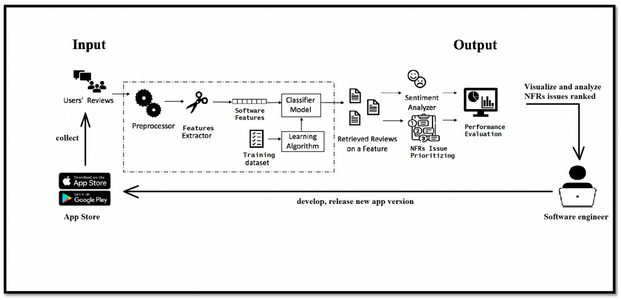


Fig.1. Conceptual view of the Novel Method

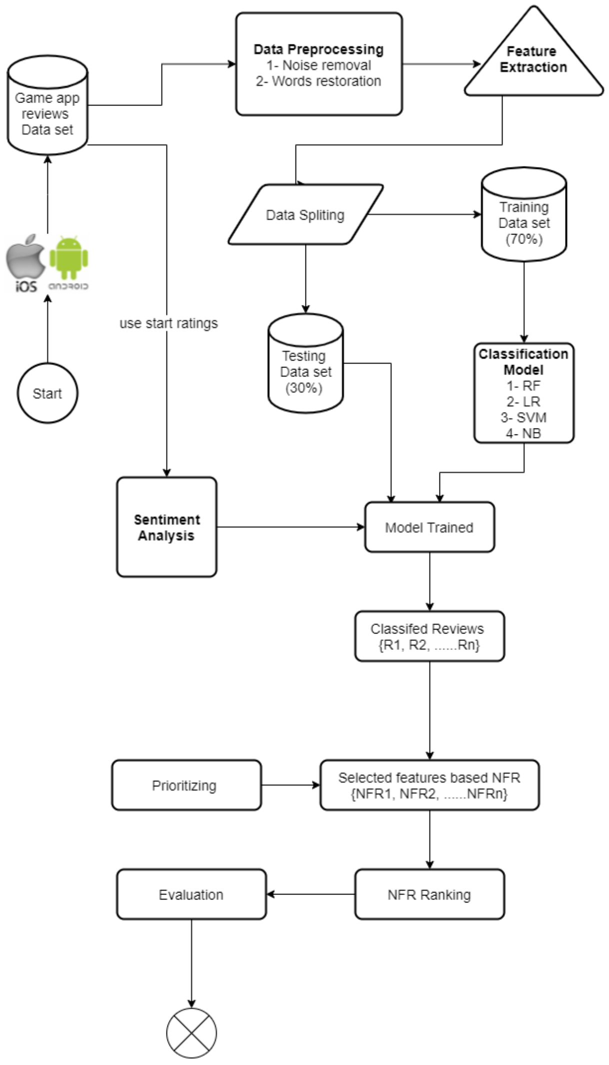
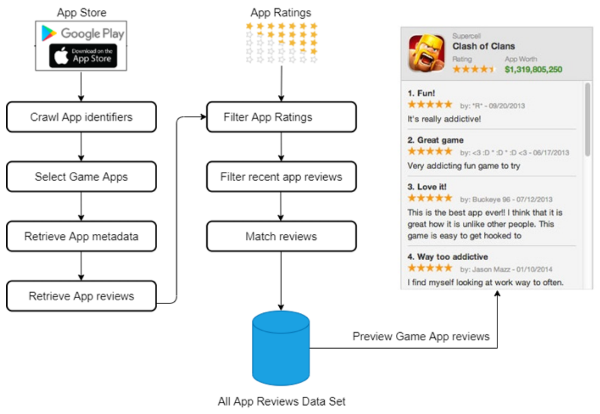


Fig. 2. Flow chart of the Proposed Model

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 machine learning 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 machine learning 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 natural language processing and machine learning techniques to extract and classify the features and preferences of the users. Fig. 3 shows the overview of our analysis method.



1. Fun! by: *R* - 09/20/2013
It's really addictive!
2. Great game by: <3 :D * :D * :D <3 - 06/17/2013
Very addicting fun game to try
3. Love it! by: Buckeye 96 - 07/12/2013
This is the best app ever!! I think that it is great how it is unlike other people. This game is easy to get hooked to
4. Way too addictive by: Jason Mazz - 01/10/2014
I find myself looking at work way to often.

Fig. 3. Overview of the data collection process

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 I reflects the categories and number of Game Apps selected.

TABLE I. 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

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 II lists the variables in our dataset. We focused on two variables: text reviews and ratings. We filtered the 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 III shows the sample of reviews we analyzed.

TABLE II. 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 III. 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 Clumb 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	Causal	Google Play	98	4.0
Basketball stars	Sport	Google Play	369	4.5

B. *Data Preprocessiong*

We collected user reviews from Google Play and iTunes, the leading platforms for Android and Apple apps. However, these reviews were not ready for machine learning analysis. They had missing, inconsistent, or irrelevant information that could affect the accuracy and reliability of our results. Therefore, we used natural language processing techniques to clean and transform the reviews into a suitable format for machine learning. 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 classifying the features and preferences of the users.

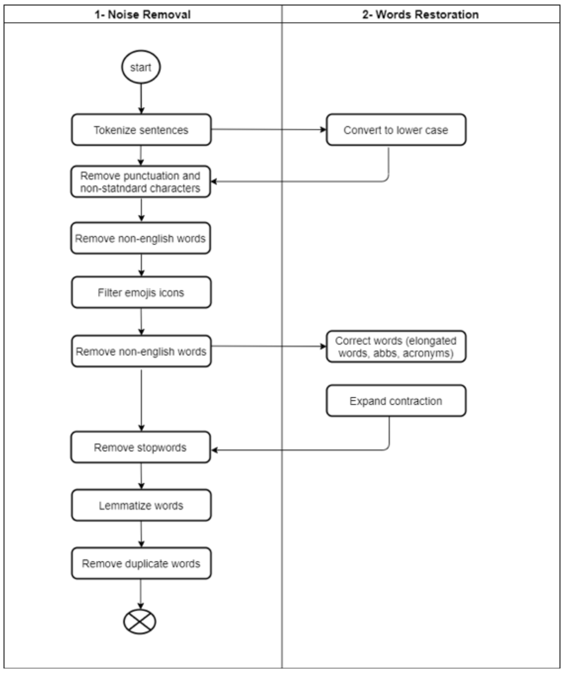


Fig. 4. Preprocessing Phase of user reviews collected.

Punctuations, non-standard characters, abbreviations, acronyms, characterize user reviews and do not contribute to the study. They can distort the outcomes from 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. Feature Extraction

The inability of ML algorithms to process user reviews directly, user reviews need to be extracted and transformed into features that can be handled by ML classifiers. This process involves conversion of textual into numerical representations that ML algorithms can interpret. Three techniques, TF-IDF (Term Frequency - Inverse Document Frequency), CHI^2 (Chi Squared), and AUR-BoW (Augmented User Reviews - Bag-of-Words) are used in feature extraction.

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 [32]. The SentiWordNet lexicon contains labeled English words that can be used to determine the opinion polarity in reviews. Eq. (1), (2) and (3) describe how the search in SentiWordNet is performed:

$$\boxed{s_+ = \sum_{i \in t} pos\_score_i} \quad (1)$$

$$\boxed{s_- = \sum_{i \in t} neg\_score_i} \quad (2)$$

$$\boxed{polarity\_score = s_+ + s_- + \sum_{i \in t} word\_neutral} \quad (3)$$

E. Data Classification

Four supervised classifiers are used in this study: Support Vector Machine (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):

$$\boxed{P(c_i|R) = \frac{P(c_i)P(R|c_i)}{P(R)}} \quad (4)$$

Ωιτη $\boxed{P(R|c_i) = \prod_{j=1}^{n} P(d_j|c_i)}$, $\boxed{P(c_i) = \frac{N_i}{N}}$ αν δ $\boxed{P(d_j|c_i) = \frac{1+N_{ji}}{M+\sum_{k=1}^{M} N_{ki}}}$ (5)

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 wat 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):

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta n X n \quad (6)$$

$$\boxed{F(X_n) = \frac{Exp\ (X_n)}{\sum_{j=0}^{k} Exp(X_j)}} \quad (7)$$

F. Data 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 machine learning 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 IV shows the distribution of ratings for each category.

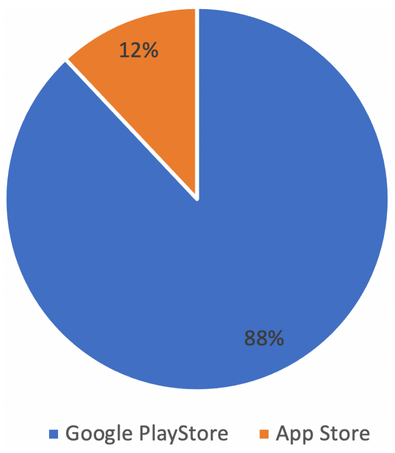


Fig. 5. Number of Apps users

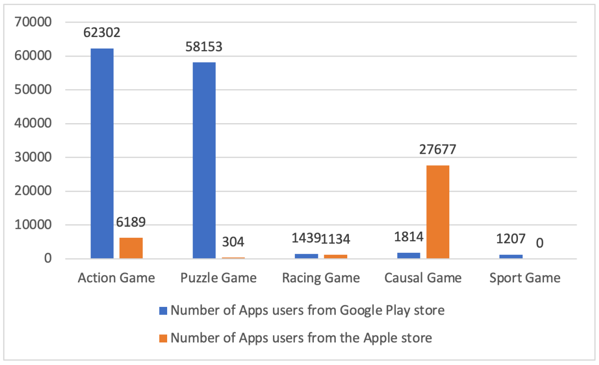


Fig. 6. Reviews per category

TABLE IV. NUMBER OF USER'S REVIEWS CORRESPONDING TO EACH RATING SCORE

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

The four research questions contribute to the results presented. This way, we can assess the model for its ability to answer the research questions 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.

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

Three feature extraction techniques; TF-IDF (Term Frequency - Inverse Document Frequency), CHI^2 (Chi Squared), and AUR-BoW (Augmented User Reviews - Bag-of-Words) 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 machine learning 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, CHI^2, and AUR-BoW for feature extraction and four machine learning 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 V showed sample user reviews that AUR-BoW and TF-IDF techniques are applied to. In the sample in Table V, 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 VI 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 VII 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.

TABLE V. 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 VI. RESULTS OF TF-IDF TECHNIQUE ON PRE-PROCESSED DATA FOR THE TWO FIRST PREPOSSESSED SAMPLE REVIEWS

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

TABLE VII. RESULTS OF AUR-BOW TECHNIQUE ON PRE-PROCESSED DATA

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

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. This research considered only three types such as Security, Flexibility, and Maintainability aspects. Table VIII depicts top 15 topic terms mined from each type of NFR while Table IX listed for the vocabulary list. Based on the topics as presented as example in Table VIII extracted from the reviews, the senti_score was calculated with the feature types, the associated words and the frequency where these features appeared. Table IX shows three examples of sentences scored by SentiWordnet. These words were used to understand specific praising, complaints about features found in the apps. Table X 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, Chi^2, and AUR-BoW after the application of ML algorithms SVM, NB, DT, LR, are presented. Accuracy, Precision, Recall and F-measure were selected 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 machine learning is presented. Fig. 7(a) presents the confusion matrix of the classification results from combining TF-IDF with all machine learning algorithms to classify NFRs features while Fig. 7(b) presented the confusion matrix of the classification results from the combination between AUR-BoW with all machine learning algorithms to classify NFRs features.

TABLE VIII. 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 IX. 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.

TABLE X. MOST COMMON TOPICS EXTRACTED FROM THE USER REVIEWS WITH THEIR SENTIMENTS SCORE

Topic	Senti score	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)

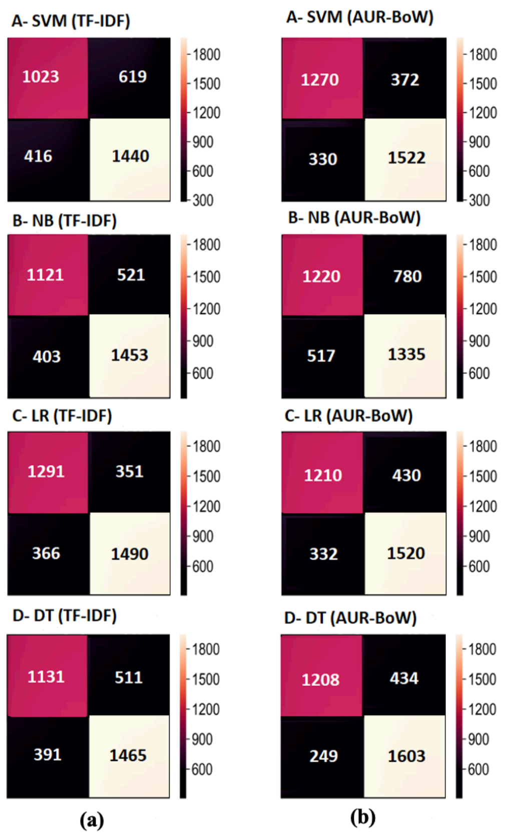


Fig. 7. Confusion matrices of (a) SVM, DT, LR, and NB using the TF-IDF technique; (b) SVM, DT, LR, and NB are shown using the AUR-BoW technique.

A comparison of the results from the machine learning 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 XI showed the top 15 significant features from regression analysis. The coefficients with higher value have higher impact on the dependent variable. From the Table XI, update ranks as the highest feature. Exploring the reviews containing this 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.

According to the confusion matrix as shown in Fig. 7(a), Logistic Regression combined with TF-IDF had the best result along with the highest TP and TN rates. LR gives 2,781 (1490 positive features & 1,291 bad features) correct NFRs predictions and 717 (366 positive features & 351 bad features) wrong NFRs predictions. According to Fig. 7(b), LR gives 2730 (1520 positive features & 1210 bad features) correct NFRs predictions and 750 (320 positive features & 430 bad features) wrong NFRs predictions against 3498 (2252 positive features & 1569 bad features) NFRs predictions with AUR-BoW, which is higher than the other classifiers models (SVM, NB, 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 XII 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 XI. 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*

TABLE XII. CLASSIFICATION RESULTS COMBINING MACHINE LEARNING WITH TF-IDF

Classifiers Models	Feature Extraction Technique: TF-IDF			
	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 XIII. CLASSIFICATION RESULTS COMBINING MACHINE LEARNING WITH AUR-BOW

Classifiers Models	**Feature Extraction Technique: AUR-BoW**			
	Accuracy	**Precison**	**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

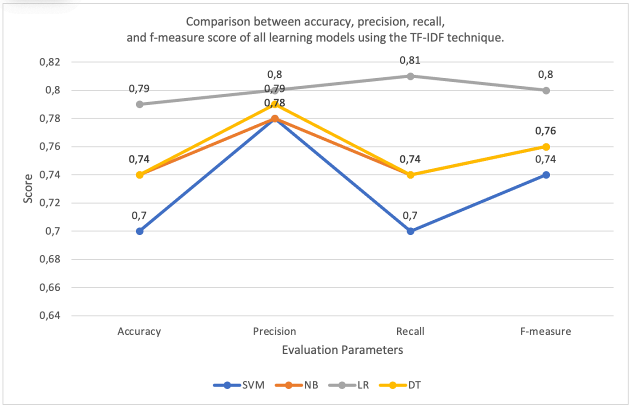


Fig. 8. Classification results combining machine learning (SVM, NB, LR, DT) with TF-IDF technique.

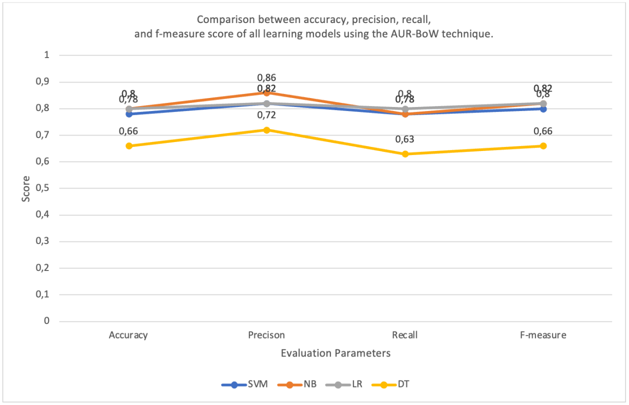


Fig. 9. Classification results combining machine learning (SVM, NB, LR, DT) with AUR-BoW technique.

According to Fig. 10, F-measure, accuracy, precision, recall compared to the performance of classifiers models with all feature engineering technique. According to the comparison, LR performed better that SVM, DT and NB in the case of using AUR-BoW (80%) for all the performance metrics. LR model was used as the predictive model and combining AUR-BoW with Chi^2, which gives more valuable features. In this case, it found that the LR model achieved the classification accuracy of about 80 percent using security features and maintainability other hand 66 percent for flexibility type form the user reviews. So, Table XIV showed the results of classification of NFRs types (Security, Flexibility, Maintainability) obtained through the rank prediction with LR classifier. One type provided the highest f-measure for LR with 82% using AUR-Bow, Chi^2 and Ranking as shown in Fig. 11.

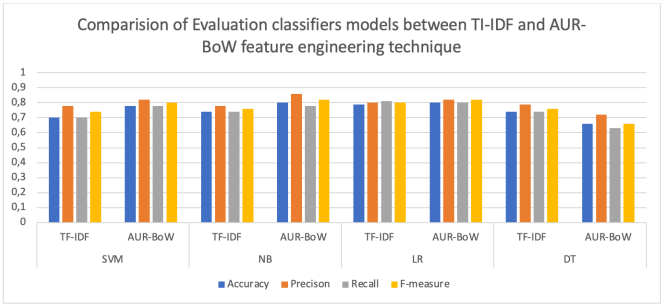


Fig. 10. Comparison between Classification results combining machine learning (SVM, NB, LR, DT) with TF-IDF and AUR-BoW technique.

TABLE XIV. CLASSIFICATION RESULTS OF THREE NFRS TYPES PREDICTED BY RANK PREDICTION WITH LR CLASSIFIER

Type	Proportion Reviews	AUR-BoW + Chi^2 + LR + Ranking			
		Accuracy	Precision	Recall	F-measure
Security	1437	0.80	0.82	0.80	0.81
Flexibility	1350	0.66	0.72	0.63	0.67
Maintainability	1234	0.80	0.86	0.79	0.82

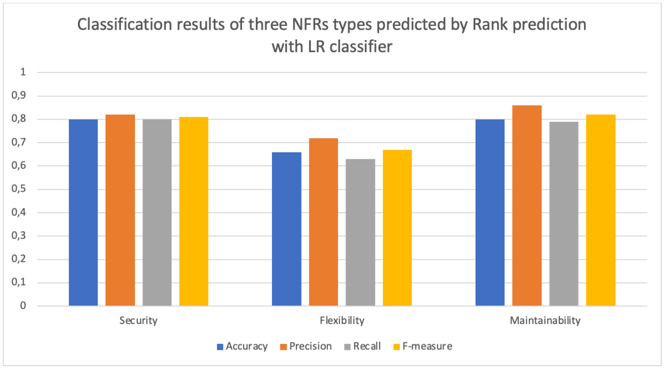


Fig. 11. Comparison between the F-measure and proportion of each type.

G. *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 machine learning and natural language processing.

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.

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 [33] [34] and [35].

ACKNOWLEDGMENT

The authors would like to thank the Africa Centre of Excellence OAK-Park for the support in executing 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, 2021, pp. 93593-93613.

[2] I. Gambo, R. Ikono, P. Achimugu, and A. Soriyan, "An Integrated Framework for Prioritizing Software Specifications in Requirements Engineering", *International Journal of Software Engineering and its Applications (IJSEIA)*, vol. 12, no. 1, 2018, pp. 33-46.

[3] I. P. Gambo, H. O. Odukoy, A. A. Oke, and E. R. 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, pp. 144-160.

[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] 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)*, 2021, pp. 128-138.

[6] 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*, 21, 2016, pp. 1067-1106.

[7] 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)*, 2019, pp. 541-548). IEEE.

[8] 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*, 2023, pp. 1-26.

[9] 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, 2021, e2332.

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

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

[12] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements engineering*, vol. 11, 2006, pp. 102-107.

[13] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," In *2012 9th IEEE working conference on mining software repositories (MSR),* 2012, pp. 108-111.

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

[15] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?," *IEEE software,* vol. 32, no. 3, 2014, pp. 70-77.

[16] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," In *2019 IEEE 27th international requirements engineering conference workshops (REW)*, 2019, pp. 220-226.

[17] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella, "Android apps and user feedback: a dataset for software evolution and quality improvement," In *Proceedings of the 2nd ACM SIGSOFT international workshop on app market analytics*, 2017, pp. 8-11.

[18] G. Grano, A. Ciurumelea, S. Panichella, F. Palomba, and H. C. Gall, "Exploring the integration of user feedback in automated testing of android applications," In *2018 IEEE 25Th international conference on software analysis, evolution and reengineering (SANER)*, 2018. pp. 72-83.

[19] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," In *2017 IEEE 25th International requirements engineering conference (RE)*, 2017, pp. 11-20.

[20] L. Zhao, M. Tavakoli, A. Heydari, and G. Nenadić, "Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools," *Expert Systems with Applications*, vol. 113, 2018, pp. 186-199.

[21] S. Malgaonkar, S. A. Licorish, and B. T. R. Savarimuthu, "Prioritizing user concerns in app reviews–A study of requests for new features, enhancements and bug fixes," *Information and Software Technology*, vol. 144, 2022, pp. 106798.

[22] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," In *Proceedings of the 36th international conference on software engineering*, 2014, pp. 1025-1035.

[23] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, and P. Liu, "Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale," In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 659-674.

[24] L. Batyuk, M. Herpich, S. A. Camtepe, K. Raddatz, A. D. Schmidt, and S. Albayrak, "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications," In *2011 6th International Conference on Malicious and Unwanted Software*, 2011, pp. 66-72.

[25] P. H. Chia, Y. Yamamoto, and N. Asokan, "Is this app safe? A large scale study on application permissions and risk signals," In *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 311-320.

[26] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Mobile app recommendations with security and privacy awareness," In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 951-960.

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

[28] D. Mukherjee, and G. Ruhe, "Analysis of Compatibility in Open Source Android Mobile Apps," In *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE),* 2020, pp. 70-78.

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

[30] R. K. Yin, "*Case study research and applications*," vol. 6, 2018. Thousand Oaks, CA: Sage.

[31] 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.

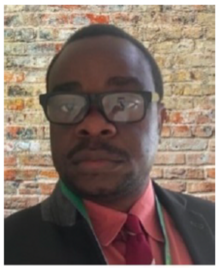
[32] Z. Gao, Y. Li, Y.Yang, X. Wang, N. Dong, and H. D. Chiang, "A GPSO-optimized convolutional neural networks for EEG-based emotion recognition," *Neurocomputing*, vol. 380, 2020, pp. 225-235.

[33] I. 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, 2021, pp. 158–169.

[34] I. Gambo, 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, July 6-8, 2021, 2021, pp. 333–341. doi:10.5220/0010605703330341.

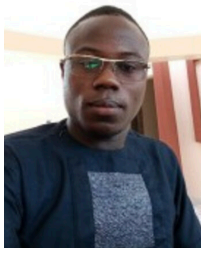
[35] I. Gambo and K. Taveter, "Stakeholder-centric clustering methods for conflict resolution in the requirements engineering process," ser. Communications in Computer and Information Science, 2022, vol. 1556 CCIS, pp. 183–210. [Online]. Available: www.scopus.com

Ishaya Gambo research is in the area of software engineering (SE), particularly in requirements engineering (RE), software testing and software architecture. He appreciates applying his research in the healthcare domain. The emphasis of his research is on user's and developers' perspectives of software systems. Ishaya has experience of applied projects and has a clear history of the full research life cycle, including academic publishing in journals and extensive presentation at conferences. He can be contacted at email: ipgambo@oauife.edu.ng.



Christopher Agbonkhese is a Visiting Assistant Professor in the Department of Digital and Computational Studies, Bates College, Lewiston, USA. His research encompasses software systems, health informatics and data analytics, where he employs techniques like machine learning and data mining to construct models for addressing critical challenges. He can be contacted at the email: cagbonkhese@bates.edu.

Theresa Omodunbi, has been a lecturer of Computer Science at Obafemi Awolowo University, Ile Ife, Nigeria in the last 12 years. Her research focus has been on computational linguistics, information retrieval, semantics and biomedical informatics, data mining, and summarization system in web applications. She has published different articles in peer-reviewed international journals and conference proceedings. She had been a visiting scholar at different times to renown Universities in the USA and Finland. She is a chartered member of Nigerian Institute of Management and Nigeria Computer Society. She has mentored and supervised the research of many college graduates in the course of her career. She has attended, facilitated and moderated many sessions at different international conferences and forum in Africa and USA.



Rhodes Massenon is currently a PhD Student in the field of Software Engineering at the Department of Computer Science and Engineering, Obafemi Awolowo University, Nigeria. His research interest is in the area of medical cyber-physical system, health informatics and privacy requirements engineering.