Received 12 June 2024, accepted 4 July 2024, date of publication 10 July 2024, date of current version 23 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3425830

RESEARCH ARTICLE

How Do Crowd-Users Express Their Opinions Against Software Applications in Social Media? A Fine-Grained Classification Approach

NEK DIL KHAN[1], **JAVED ALI KHAN**[2], **JIANQIANG LI**[1], **(Senior Member, IEEE)**, **TAHIR ULLAH**[3], **AYED ALWADAIN**[4], **AFFAN YASIN**[5], **AND QING ZHAO**[1]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]Department of Computer Science, School of Physics, Engineering and Computer Science, University of Hertfordshire, AL10 9AB Hatfield, U.K.
[3]Department of Software Engineering, University of Science and Technology Bannu, Bannu 28100, Pakistan
[4]Computer Science Department, Community College, King Saud University, Riyadh 11543, Saudi Arabia
[5]School of Software, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

Corresponding author: Javed Ali Khan (j.a.khan@herts.ac.uk)

This work was supported by the Researchers Supporting Project, King Saud University, Riyadh, Saudi Arabia, under Project RSP2024R309.

ABSTRACT App stores allow users to search, download, and purchase software applications to accomplish daily tasks. Also, they enable crowd-users to submit textual feedback or star ratings to the downloaded software apps based on their satisfaction. Recently, crowd-user feedback contains critical information for software developers, including new features, issues, non-functional requirements, etc. Previously, identifying software bugs in low-star software applications was ignored in the literature. For this purpose, we proposed a natural language processing-based (NLP) approach to recover frequently occurring software issues in the Amazon Software App (ASA) store. The proposed approach identified prevalent issues using NLP part-of-speech (POS) analytics. Also, to better understand the implications of these issues on end-user satisfaction, different machine learning (ML) algorithms are used to identify crowd-user emotions such as anger, fear, sadness, and disgust with the identified issues. To this end, we shortlisted 45 software apps with comparatively low ratings from the ASA Store. We investigated how crowd-users reported their grudges and opinions against the software applications using the grounded theory & content analysis approaches and prepared a grounded truth for the ML experiments. ML algorithms, such as MNB, LR, RF, MLP, KNN, AdaBoost, and Voting Classifier, are used to identify the associated emotions with each captured issue by processing the annotated end-user data set. We obtained satisfactory classification results, with MLP and RF classifiers having 82% and 80% average accuracies, respectively. Furthermore, the ROC curves for better-performing ML classifiers are plotted to identify the best-performing under or oversampling classifier to be selected as the final best classifier. Based on our knowledge, the proposed approach is considered the first step in identifying frequently occurring issues and corresponding end-user emotions for low-ranked software applications. The software vendors can utilize the proposed approach to improve the performance of low-ranked software apps by incorporating it into the software evolution process promptly.

INDEX TERMS User reviews, app store analytics, software issues, bug reports, data-driven requirements.

I. INTRODUCTION

The software app market is highly competitive and constantly growing. Crowd-users can search for, download, and purchase software apps through various application stores, such as Amazon, Google Play, and Apple Stores, to fulfill their daily needs and tasks. As of June 2020, nearly 3 million software apps were available in the Google Play and Apple stores, with over 75 billion software app downloads per month [1]. This huge volume of downloads

The associate editor coordinating the review of this manuscript and approving it for publication was Xinyu Du.

and user interactions has created a fascinating phenomenon of app approval within the user community [2]. Furthermore, these app platforms enable crowd-users to express their feelings, opinions, or sentiments by providing feedback on the performance and functionalities of software applications.

End-user-generated reviews about software products and services are widely available on various social media platforms. Automated analysis of end-user feedback from large datasets is essential to extract valuable insights for software decision-making [3], [4], [5]. These reviews provide critical information that helps software engineers understand user requirements and design issues, thereby facilitating the evolution of software products [6], [7]. Additionally, end-users can request new features, report issues or bugs, share user experiences, and ask for non-functional requirements through these platforms [8]. Consequently, app vendors must regularly release new versions to address pressing issues or bugs and provide requested new features. High end-user satisfaction is achieved when application developers respond quickly to customer feedback regarding bugs and problems with software functionalities and performances [9], [10], [11]. The literature reports that dissatisfied end-users uninstall apps rapidly and are more likely to seek alternatives, damaging the software applications' reputation [12], [13]. Even renowned and highly rated apps can quickly succumb to user dissatisfaction if software vendors neglect timely listening to the user's voice [14]. Therefore, successful app evolution necessitates regularly monitoring and analyzing end-user feedback, particularly regarding software issues or bugs, to improve app quality and user satisfaction.

However, app vendors receive substantial end-user feedback regularly via various channels such as app stores, Twitter, or user forums. Manually analyzing, filtering, and evaluating such a large amount of feedback is challenging, complex, and time-consuming for software vendor organizations [15]. Recent advancements have made it possible to automatically identify and recover software-related information by processing end-user feedback [16], [17], such as software bugs or issues [18], [19], non-functional app usability issues [20], feature requests [15], [21], software methodologies insights [22], clustering similar comments [23] to discover common user problems [24]. However, most research focuses on app reviews for popular and highly rated software applications. Apps with higher rankings and star ratings are more likely to be downloaded [25]. As a result, software vendors might overlook vital information from crowd-users to improve the quality of lesser-known applications [26], [27]. Additionally, existing research approaches primarily classify end-user comments on social media into various requirements-related information but need more identification of specific software-related information that could be used as indicators to enhance the performance of discussed applications.

To fill this gap, We undertook a detailed quantitative and qualitative research study with the proposed approach. The proposed study focuses on how crowd-users report issues and bugs in low-ranked software applications in the ASA store. We propose a novel approach that involves identifying common issues or bugs reported in user comments using NLP and employing different ML algorithms to identify end-user emotions such as anger, fear, sadness, and disgust associated with these issues or bugs. This study aims to understand user opinions about software apps better and enhance their overall quality and performance by systematically identifying and addressing user-reported issues and their associated emotional responses.

The Key Contributions of the Proposed Approach Are as Follows:

- We propose a semi-automated approach leveraging NLP to identify and capture frequently occurring issues or bugs in the ASA store, involving developing an NLP-based algorithm that detects critical issues from end-user comments.
- Crowd-user comments on identified issues are collected to encapsulate end-user opinions or emotions.
- To better understand the frequently identified issues, an additional analysis of the crowd-user comments is performed to identify various types of end-user opinions associated with the captured issues, such as anger, fear, sadness, and disgust.
- A novel dataset is curated using a novel grounded theory and content analysis approach for identifying associated emotions. The dataset can be used by software vendors, researchers and educationists for understanding frequently occurring issues.
- Various pre-processing, feature engineering and data balancing approaches are used to fine-tune various ML algorithms for better issue classification results.
- The proposed approach can be used as a baseline to further improve the performance of issue detection and emotion classification approaches.

The manuscript structure is thoroughly organized to ensure a comprehensive understanding of the research. Section II, Related Work, reviews the existing literature and frames the study within the broader context of data-driven requirements engineering, focusing on analyzing app reviews and user feedback across various platforms, including Amazon. Section III, Methodology for the Proposed Research, elaborates on the research methodology, detailing the approach for data collection, analysis, and utilizing NLP and ML algorithms. Section IV, Data Gathering and Preprocessing discusses the initial steps of data collection and the preprocessing necessary for analysis. Section V, Issues Extraction using NLP, describes extracting significant issues from user feedback in the ASA store using NLP techniques. Section VI, Identify End-User Opinions with the Identified Issues, details how the study identifies and categorizes user opinions related to the extracted issues. Section VII, Automated Classification of End-User Emotion, covers the application of ML algorithms to classify the emotions associated with user feedback. Section VIII, Discussion, delves into the analysis of the findings, discussing their implications for software development and user satisfaction. Section IX, Conclusion and Future Work, concludes the manuscript, summarizing the study's contributions and outlining directions for future research.

II. RELATED WORK

Recently, research on data-driven requirements engineering has grown exponentially, with studies looking at app reviews [21], [28], [29], [30], tweets [24], [31], Developers forums (Stake Overflow) [22], product reviews, i-e, Amazon reviews [32], [33], or a combination of product descriptions and feedback [34]. They all have in common: i-e, a software product already exists, and customers review and write about their experiences with it [35]. User feedback and end-user participation are vital for software engineers and requirements analysts since they contain insightful information like issues, bugs, non-functional requirements, and feature requests [18], [23], [36]. Categorizing and analyzing end-user feedback [29] was an initial step in understanding user requirements. Also, research studies [33], [34], [37] examined the end-user reviews in the social media platforms to understand the rationale for requirements decision-making and identify conflict-free requirements-related information on a run. The release planning process is initiated by software vendors when they decide to incorporate, for example, an innovative feature request into a software product or improve a frequently reported issue or bug in the software application [38], [39]. Similarly, Khalid et al. [40] did a qualitative study on how app store users utilized app reviews to report difficulties. They detected 12 categories of complaints by critically analyzing end-user feedback in the app stores to detect and elicit latent requirements by analyzing variations in numerous patterns connected to app price, rating, and popularity. Additionally, Qazi et al. [41] proposed a generalized feature-optimized transfer learning approach to identify issues or bugs, which is evaluated with various benchmark datasets. However, their approach is not validated and tested on end-user reviews.

Furthermore, unlike Haering et al. [42], and Mezouar et al. [43], we consider app reviews from the Amazon platform. In contrast, they focus on end-user reviews from app stores and Twitter. Additionally, Haering et al. [42] proposed an automated approach using deep learning to match issues or bugs reported in app stores to the bug reports in the issue tracking system. Similarly, Mezouar et al. [43] conducted an empirical study to investigate whether the analysis and validation of end-user comments on the Twitter social network improve the issues identification and fixing process. Like, Twitter, the ASA store also allows for lengthy multi-level conversations with end-users [26], [44], which leads to in-depth insights into, i-e., the crowd-users context, such as software app version and steps to recreate. In comparison, app stores allow application developers to reply to crowd-user reviews and enable crowd-users to modify their feedback [45]. Also, Khan et al. [33] proposed an automated approach, which filters out crowd-user feedback in the user forum into end-user feedback containing rationale information and without rationale. In the second experiment, ML classifiers are employed on the end-user feedback containing rationale information and classified into supporting, attacking claims, new features, and issues. Alkadhi et al. [46] proposed a fine-grained ML approach that analyzes developers' chat messages to identify useful software and requirements-related information such as decisions, alternatives, issues, and positive & negative arguments. Additionally, khan et al. [37], [47] proposed an ML learning-based approach, which analyses end-user comments in the Reddit forum to identify conflict-free new features or issues using argumentation theory based on their supporting and attacking arguments.

Moreover, Sentiment analysis, often known as opinion mining, is the process of understanding user sentiments in app evaluations. It differentiates between positive, neutral, and negative sentiment polarities at a variety of discrete levels, including entire reviews, sentences, and phrases. Research conducted by Martens and Johann [48], Martens and Maalej [49], and Srisopha et al. [50] have investigated sentiment analysis at the review level, whereas Guzman and Maalej [21], Panichella et al. [51], and Panichella et al. [52] have concentrated on sentence-level analysis. Gu and Kim [53] and Dbrowski et al. [7] delved into phrase-level sentiment detection with more detail. App reviews are recognized as a valuable source of user comments, showcasing attitudes on various themes, features, and program attributes. This has been emphasized in studies conducted by Guzman and Maalej [21], Malik et al. [54], and Masrury et al. [55], along with other researchers. Analyzing these perspectives helps software developers comprehend user attitudes towards their applications, revealing user needs, preferences, and factors impacting app sales and downloads, as demonstrated in the studies conducted by Liang et al. [56] and Nicolai et al. [57].

In contrast, we first proposed NLP-based co-occurrence algorithms to identify frequently reported issues in the ASA store. Then, different ML algorithms are utilized to identify the end-user emotions with the identified issues, such as anger, disgust, fear, and sadness, to understand their importance and implications on the quality of software applications under discussion. Additionally, unlike Twitter's social media platform, issues reported in the crowd-user feedback in the ASA store specifically showcase the platform against which the software application is installed. On Twitter, it becomes challenging to identify the platform, i.e., Mac, Android, Windows, or Amazon, mentioned in the tweets.

Table 1 presents a comparative overview of existing research and highlights the distinctiveness of our study.

III. METHODOLOGY FOR THE PROPOSED RESEARCH

This section elaborates on the research methodology, which is comprised of two steps. Firstly, we describe the research questions aiming to answer using the proposed research

TABLE 1. Comparison of existing research and our study.

Research Work	Key Focus	Distinctiveness of Our Study
Guzman et al. [21]	Analysis of app reviews for feature requests	Our study focuses specifically on low-rated apps in the ASA store, using NLP and ML to identify issues and emotions.
Haering et al. [42]	Matching app store issues to bug reports using deep learning	We use NLP for issue extraction and ML for emotion detection, focusing on crowd-user feedback in the ASA store.
El Mezouar et al. [43]	Analysis of Twitter comments for issue identification	Our study analyzes ASA store reviews, offering a more structured and platform-specific approach.
Martens and Maalej [49]	Sentiment analysis at the review level	We perform sentiment analysis at a more granular level, identifying specific emotions (anger, fear, sadness, disgust) associated with issues.
Khan et al. [33]	Filtering user forum feedback into rationale information	Our work combines issue identification with emotion analysis, providing a comprehensive view of user feedback in the ASA store.
Nayebi et al. [38]	Innovative feature request incorporation in software products	We focus on low-rated apps and use NLP and ML to identify frequently reported issues and emotions, enhancing app quality and user satisfaction.
Khalid et al. [40]	Qualitative study on app store user difficulties	We provide a quantitative and qualitative analysis using NLP and ML to identify issues and emotions in low-rated ASA store apps.
Hassan et al. [45]	Developers' responses to crowd-user reviews in app stores	Our approach not only considers developers' responses but also systematically identifies issues and corresponding emotions.
Guzman and Maalej [21]	Sentiment analysis in app reviews	Our study provides a more detailed sentiment analysis by identifying specific emotions linked to issues in low-rated apps.
Panichella et al. [52]	Sentence-level sentiment analysis in app reviews	We extend this by applying sentiment analysis to low-rated ASA store reviews and focus on specific emotions related to issues.

approach. Secondly, we defined the approach that first identifies frequently occurring issues or bugs using NLP utilities. Then, using different ML algorithms, we identify end-user opinions with the associated issues in the ASA store.

A. RESEARCH QUESTIONS

In this research paper, our objective is to identify frequently reported issues and associated end-user opinions by focusing on low-ranked software applications in the ASA store to help software vendors or developers in timely incorporate the end-user feedback in the software evolution that helps to improve their performance and ultimately gets end-user satisfaction. For this purpose, we formulate the following research questions.

RQ1. How do end-users register issues against low-ranked software apps in the ASA store?

RQ2. Can frequent issues in end-user reviews on the ASA Store be automatically identified?

RQ3. What is the precision of ML classifiers in identifying end-users opinions against the issues captured in the ASA store?

In summary, RQ-1 emphasizes the detailed manual analysis of end-user feedback in the ASA store to recover different patterns on how crowd users report problems or bugs against the low-ranked software applications under discussion. Also, identify and retrieve various types of end-user opinions expressed against identified bugs using grounded theory and content analysis approaches that are mandatory to develop a ground truth for the ML classifiers. For RQ-2, we emphasize identifying frequently occurring issues or bugs using POS analytics, such as identifying objects cum nouns, verbs cum operations, adjectives, and their combinations. Furthermore, RQ-3 focuses on identifying the performance of various ML algorithms in automatically classifying and detecting the end user's opinions with the associated captured issues, such as anger, fear, sadness, and disgust.

B. RESEARCH METHODOLOGY

The proposed research approach, shown in Figure 1, comprises three main steps. In the first step, we curated a unique research data set containing crowd-user feedback against the various low-rating software applications in the ASA store (details are in Section IV). The selection criteria for these apps included those with ratings of three stars or below and at least 500 user reviews to ensure a significant amount of feedback data for analysis. The reviews were selected based on their recency and relevance to capture the most current issues faced by users. We selected low-rating software applications intending to identify and capture issues or bugs reported by the end-users to improve the ratings of the under-discussed software apps by developing a mechanism to incorporate the user feedback into software evolution in a timely manner. Next, we developed a mechanism using the NLP toolkit to recover frequently reported issues or bugs in the ASA store. This involved the use of part-of-speech tagging to identify key terms and patterns indicative of software issues. Specific patterns and rules were formulated to automate the identification process, based on the frequency

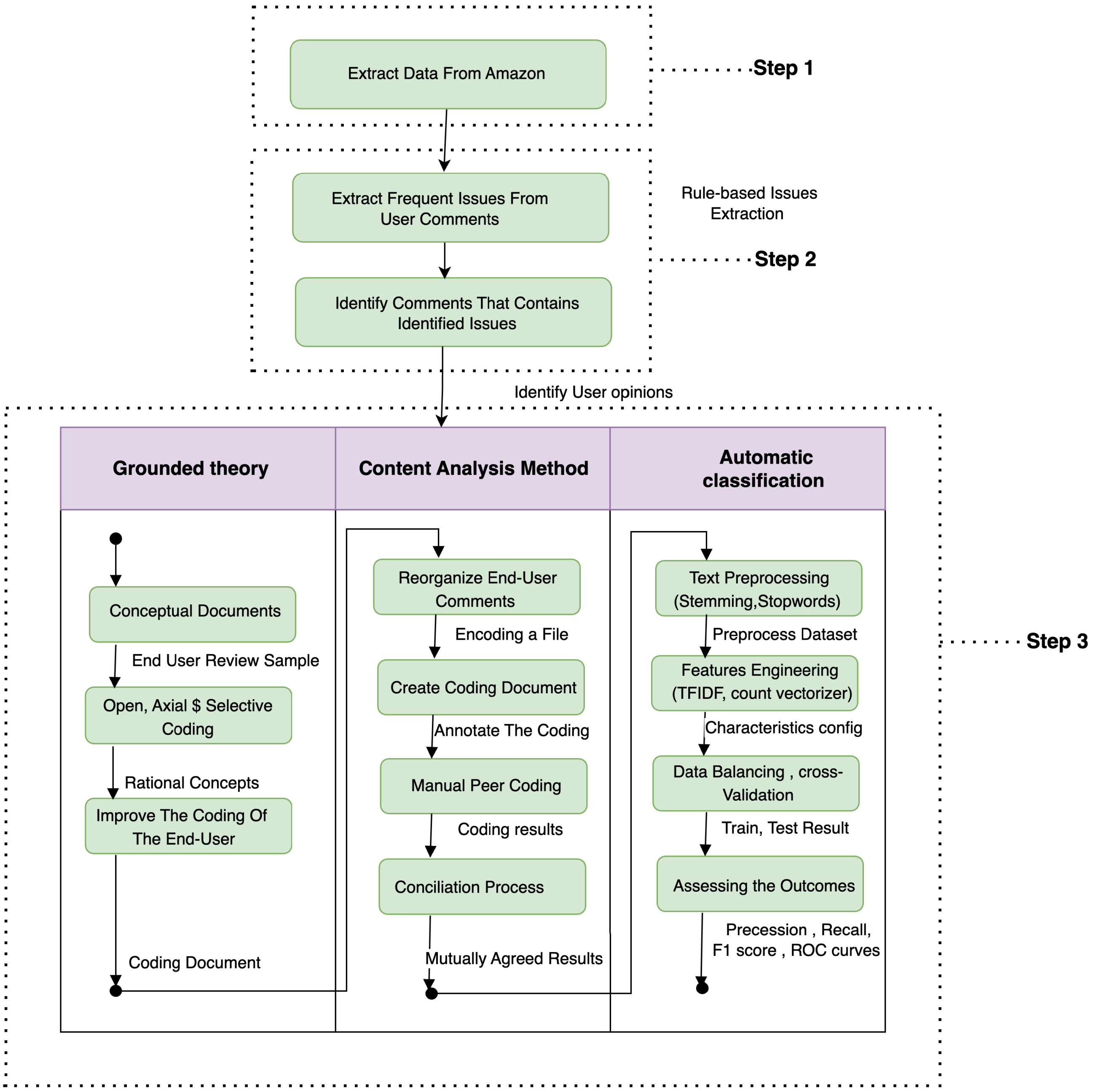


FIGURE 1. **The proposed research methodology.**

and context of the terms used. Next, in the proposed approach, we collected the end-user comments representing the captured issues (details in Section V). Its purpose is to record the end-users emotions about the problems or issues identified. Next, to run the ML experiment and automatically identify end-user opinions, we iteratively developed a coding guideline that would work as a baseline for the crowd-user annotation process. We developed an annotated research data set by processing and analyzing each crowd-user comment using the developed coding guideline[1] and content analysis approach. We assigned the captured label (anger, fear, sadness, and disgust) to each user comment in the data set. We removed the conflicts, if any, during the annotation of the user comments (explained in section VI). The labels were identified by manually analyzing the type of feedback collected from the ASA store. After manually identifying user emotions about reported issues, we aim to automate the proposed approach by employing different ML algorithms to identify the associated end-user opinions with the captured bugs. The ML algorithms used include MNB, LR, RF, MLP, KNN, AdaBoost, and Voting Classifier, chosen for their effectiveness in text classification tasks. For this purpose, we pre-process the input user comments to remove stop words, brackets, numeric values, special characters, etc.

[1]https://github.com/nekdil566/Issues_Extraction-_using-NLP_ML

We also used techniques like tokenization, lemmatization, and conversion to lower case to standardize the text data. For better results from the ML experiment, we used two baseline resampling methodologies, i.e., under-sampling and oversampling, to balance the number of instances (user comments) in each end-user opinion category (anger, fear, sadness, and disgust). Since ML algorithms operate on numerical data, we used various textual features (Bag of Words and TF-IDF) to convert textual data into a numerical form to train and validate the ML algorithms efficiently. We used the standard k-fold cross-validation approach to train and validate the ML algorithms for reliable results. Finally, we calculate each classifier's accuracy, recall, precision, and F-measure values to identify the most suitable ML algorithm for identifying end-users opinions or emotions associated with the user comments that possess the captured issues. Below, in each section, we elaborate in detail on the main steps of the proposed research approach.

IV. DATA GATHERING AND PREPROCESSING

To conduct our research effectively, we focused on collecting a dataset that encapsulates end-user feedback on software applications. Specifically, we targeted applications within the ASA store with three stars or below ratings. The ASA store, renowned for its extensive catalog of 824,220 applications provided by 174,237 app publishers, serves as a pivotal platform in the global mobile app ecosystem, with a significant portion of its 236,550 apps being actively rated by users [58]. A total of 45 applications across 10 distinct categories were selected for our study, aiming to cover a broad spectrum of software applications. This selection strategy was underpinned by the goal to delve into the reasons behind these applications' low ratings, dissect the emotions conveyed through user feedback, and categorize the various types of emotions to better understand user grievances with these software applications. Detailed information on each selected software application and its category is provided in Table 2.

For the collection of user reviews, the Instant Data Scraper[2] tool was employed, facilitating an efficient and systematic approach to data extraction from web pages. This tool enabled the aggregation of 71,853 end-user reviews across the selected 45 software applications from the 10 categories of the ASA store. The first two authors meticulously carried out the selection process and manually identified and analyzed the reported issues and user opinions for these low-ranked software applications. The diversity in application categories ensures the generalizability of our findings, capturing a wide array of end-user experiences and opinions. The categories were chosen based on a predefined criterion requiring a minimum of 500 reviews per application, aiming to collect a comprehensive dataset that reflects a wide range of user feedback and issues.

[2]https://chrome.google.com/webstore/detail/instant-data-scraper/ofaokhiedipichpaobibbnahnkdoiiah

To ensure the high quality of the collected data, we implemented several measures to mitigate noise and enhance data representativeness. First, we cleaned data to remove duplicate entries, irrelevant comments, and spam. We utilized automated and manual techniques to filter out non-informative reviews, such as single-word feedback or generic praise/complaints that lacked specific details about the software. To further ensure data representativeness, we stratified the data collection process to include reviews from different periods, capturing both recent and older feedback to account for changes in user sentiment or app performance over time. We also ensured that the selected reviews covered a range of app versions to reflect any updates or changes made by the developers. The rationale behind selecting these preprocessing techniques was to maximize the clarity and relevance of the feedback data, ensuring that the resulting dataset accurately represents user experiences and opinions. By focusing on detailed and specific user comments, we aimed to create a rich dataset that can provide meaningful insights into the issues and emotions expressed by users.

This manual selection process was particularly challenging, as finding applications with a significant number of reviews highlighting issues or missing functionalities is not straightforward [59]. Our study aims to fill this gap by focusing on low-rated software applications, a largely neglected area in requirements engineering (RE) literature which often concentrates on more popular software applications like AngryBird, Google Maps, Firefox, and Chrome [21], [37], [43], thereby overlooking the wealth of information present in user feedback on less popular applications. The comprehensive details of the applications selected for our study, including the category names and the total number of reviews per category and application, are showcased in Table 2. Alongside each review, we collected data on the feedback submitter's name, rating, review title, and the content of the feedback, placing a significant emphasis on end-user feedback as the cornerstone of our proposed approach.

V. ISSUES EXTRACTION USING NLP

By manually analyzing end-user comments in the ASA store, we learn that many crowd-users report issues, complaints, or bugs against low-rated software applications, mainly due to incomplete or missing features, the bombardment of advertisements, usability issues, performance issues, etc., affecting the overall quality of the application under discussion, which leads to user dissatisfaction and a low rating of the software application. For this purpose, the first two authors of the article analyze 1000 end-user comments randomly selected across the different apps, as shown in Table 2. Aiming to propose an automated approach that identifies frequently reported issues in the ASA store to improve the overall quality of software applications by satisfying end-user issues and emotions. Each author analyzes 500 end-user feedback to recover how crowd-users express their problems

TABLE 2. Summary of gathering data-set of crowd-user reviews.

No.	Applications Category	Applications Name	User Reviews
1	Business Apps	1: Hammer Print 2: Sketch Gur 3: Office Suite Free 4: PDF Max Pro - Read, Annotate	8626
2	Communication Apps	1: JusTalk - Free Video Calls and Fun Video Chat 2: TextMe - Free Text and Calls 3: Skype 4: AddMeSnaps 5: Free Text, Text anyone	8622
3	Education Apps	1: TED TV 2: Caspers Company 3: World Now 4: Amazon Silk: Duolingo.com	10299
4	Food and Drinks Apps	1: Food Network Kitche 2: Italinan Recipes by Fawesome.tv 3: ChefTap: Recipe Clipper, Planner	5425
5	Game Apps	1: Mobile Strike 2: Drive Car Spider Simulator 3: Chapters Interactive Stories 4: Crazy Animal Selfie Lenses 5: Cradle of Empires - Match 3 in a Row and Egyptian game	8923
6	Novelty Apps	1: Ghost Radar: CLASSIC2: Xray Scanner3: Cast Manager 4: Age Scanner Classic5: Clock in Motion	2219
7	Photos and videos Apps	1: AirBeamTV screen mirroring receiver 2: AirScreen 3: Snappy Photo Filter And Stickers 4: ScreenCast 5: RecMe Free Screen Recorder	6867
8	Productivity Apps	1: PrintBot 2: Floor Plan Creator 3: tinyCam Monitor FREE4: VPN for Fire TV (Fast, Secure, and Reliable) 5: Screen Stream Mirroring	2269
9	Sports and Exercise Apps	1: FOX Sports: Live NASCAR, Boxing 2: fuboTV: Watch Live Sports, TV Shows, Movies And News 3:NBC Sports 4: CBS Sports Stream And Watch Live	9616
10	Utilities Apps	1: PrintBot 2: Floor Plan Creator 3: tinyCam Monitor FREE 4: Tv Screen Mirroring 5: Optimizer and Trash Cleaner Tool for Kindle Fire Tablets	8987
Total Applications		45	71853

with the software apps. The task was completed in eight working hours. The finding from the analysis was grouped to remove the repetitions in how end-users report their grudges. The conflicts between the two authors were resolved with discussion and negotiations, if any.

During the manual analysis, we identified various linguistic patterns in user feedback that indicate common issues. These patterns typically involve using specific nouns, adjectives, and verbs to describe problems. For instance, in the feedback "Worst application ever for texts and calls," the nouns/objects and adjectives are used to express dissatisfaction. Similarly, the feedback "I have had for two months and all of a sudden I am unable to send texts at all. Contacted support, and they cannot seem to help either." uses adjectives and verbs to detail the issue. Another example is "This app does not work on my Kindle Fire HD. It always says that I am not connected to WiFi when I try to use it," which combines adjectives, nouns, and verbs in a specific sequence to report an issue. Based on these observations, we developed certain patterns and rules using the NLP toolkit to automate the identification of frequently reported issues in user comments. Our approach involves identifying nouns, adjectives, and verbs to detect common problems. We formulated these patterns by analyzing specific words' co-occurrence and syntactic relationships.

We developed patterns and collocations by determining the relationship between the captured nouns, adjectives, and verbs using the Natural Language ToolKit (NLTK).[3] A collocation is a group of words that appear together exceptionally frequently [60], [61]. Additionally, collocations only sometimes indicate that the words are next to one another. However, the number of words may be wedged between the words that make up the collocation. Collocations are often used to explain issues since they are typically used to express a particular meaning or concept. For example, in the sentence "This app frequently crashes on startup," the words "frequently" and "crashes" form a collocation that indicates a performance issue. Identifying such collocations can more accurately pinpoint specific problems users report.

Below, we describe the extraction of frequently reported bugs from the ASA store using POS analytics. We employed part-of-speech tagging to categorize words into nouns, verbs, adjectives, etc. and used dependency parsing to understand the grammatical structure of sentences. This enabled us to formulate rules such as "noun + verb + adjective" or "adjective + noun" combinations frequently appearing in bug reports. These rules were then used to automate the detection of similar patterns in a larger dataset. For instance, a rule might specify that any feedback containing the pattern "adjective + noun + verb" where the adjective is negative (e.g., 'poor,' 'bad,' 'unresponsive'), the noun is a feature or component (e.g., 'interface,' 'performance'), and the verb indicates a malfunction (e.g., 'crashes,' 'fails') should be flagged as an issue report. By implementing these NLP-based rules, we can systematically extract issues from a large volume of user feedback, thus providing valuable insights into the most common problems affecting software applications in the ASA store. This automated process significantly reduces the manual effort required and ensures a more comprehensive analysis of user-reported issues. The effectiveness of this approach was validated through a series of tests, where the automated extraction results were compared against manually identified issues to ensure accuracy and reliability. However, tool-chain support is still required to fully automate the issue extraction process for the software evolution that filters out spare data that does not exactly represent issue-related information, which in turn will further reduce the time taken to identify the frequently occurring issues only.

A. EXTRACTION OF ISSUES-RELATED INFORMATION FROM THE ASA STORE

We used POS tagging from the NLKT library to identify the domain objects, adjectives, and verbs (operations) in the research dataset to identify frequently reported issues,

[3]https://www.nltk.org/

TABLE 3. Top-ranked identified nouns cum objects.

Rank	Noun-Object	Frequency
1	app	24111
2	recipe	3834
3	tv	3846
4	skype	2391
5	video	1786
6	version	1765
7	printer	1558
8	quality	1510
8	amazon	1435
10	picture	1145
11	problem	892
12	problem	745
13	item	708
14	ads	568
15	garbage	443

TABLE 4. Frequently identified adjectives.

Rank	Adjectives	Frequency
1	little	1502
2	same	1408
3	bad	1091
4	difficult	1202
5	hard	676
6	slow	639
7	terrible	606
8	unable	513
9	horrible	482
10	disappointed	481
11	useless	473
12	constant	466
13	poor	448
14	false	64
15	invalid	64

bugs, or concerns in the ASA store. We have determined the commonly asked issues in the dataset by associating the captured objects, adjectives, and operations. Below, we elaborated on the process in detail:

1) CAPTURING NOUNS, OBJECTS AND ADJECTIVES

Previously, it is identified that crowd-users report issues or bugs using objects cum nouns and adjectives. For example, the end-user comments, "This is a terrible app. I prefer Facebook video chat rounds over this. It Constantly crashes. I hate this app," "terrible app - hated it - would not recommend to anyone," and "It does work, but very poor quality and response to the picture transmission is delayed. Not good stuff". We extracted the key software-related information, such as "terrible app" and "poor quality," using co-occurrence patterns by combining objects with adjectives. To automate the process, an NLTK library is employed by first identifying all the possible nouns, objects, and their frequencies in the dataset containing 71853 end-user comments. In total, 30689 unique objects have been identified in the end-user comment in the ASA store, along with their frequencies, among which the topmost ranking nouns are shown in Table 3. For example, the object "app" occurred in the dataset about 24111 times, and

TABLE 5. Frequently identified issues by combining nouns with adjectives.

Rank	Adjective-Object	Frequency
1	Terrible app	46
2	poor quality	46
3	false advertising	44
4	useless app	43
5	bad app	38
6	constant buffering	31
7	horrible app	26
8	same issue or same error	26, 14
9	little confusing	22
10	many glitches	21
11	low quality	20
12	invalid file	18
13	poor picture	18
14	absolute garbage	15
15	older-version	10
16	poor app	15
17	bad quality	14
18	poor video	11
19	disappointed with this app	13

the "picture" object appeared 1145 times in the research dataset, as shown in Table 3. By a similar method, all the adjectives in the crowd-user comments are identified, which represent issues. A total of 2922 unique adjectives were found, and the top-ranked ones are shown in Table 4. For example, the adjective "bad" has occurred in the dataset about 1091 times, and the "terrible" adjective occurred 606 times in the research dataset, as shown in Table 4. Furthermore, We developed a co-occurrence algorithm that identifies frequently occurring issues by associating the captured nouns and adjectives to provide an overview to the software developers and vendors that might help improve the performance of the low-ranked software apps by incorporating the frequently occurring issues in the next version of software application. Also, the objects or nouns do not always appear immediately after adjectives in the data set, but there could be a few words between them and vice versa. In total, 28230 unique objects and adjective pairs are identified in the dataset, among which the top-19 frequent pairs are shown in Table 5. The "Terrible app" and "Poor quality" noun-adjective association was reported 46 times, "false advertising" is reported 44 times in the dataset, and vice-versa. Some examples are "Poor quality app, lots of lag and distortion," "Says it's a free optimization app but when you go to use it, it immediately tries to charge you. False advertising and not worth the time it took to download," "Useless App. I deleted it", etc. Such requirement-related information is of pivotal importance for the software and requirements engineers that help improve the current version or a specific software application feature under discussion in the ASA store [33], [37].

2) VERB-ACTION

Furthermore, it is observed that capturing verbs in the end-users comments helps identify the potential issues or bugs by associating them with the previously identified

TABLE 6. Frequently identified verbs in the end-user feedback.

Rank	Verb-Action	Frequency
1	have	16212
2	want	2697
3	learn	2675
4	find	2242
5	play	2041
6	try	1583
7	worked	1580
8	give	1573
9	downloaded	1549
10	buy	1242
11	looking	1011
12	started	833
13	tell	553
14	view	347
15	moving	133

TABLE 7. Frequently identified bugs by combining verbs and adjectives in the user feedback.

Rank	Operation-Adjectives	Frequency
1	unable to get/use/watch	61/55/47
2	hard to find/understand/use	40/26/25
3	difficult to navigate/understand/find	28/25/20
4	unable to play/sign	22/22
5	expensive to play	22
6	hard to figure/navigate/tell	20/19/17
7	impossible to use	19
8	much buffering	17
9	unable to connect/make/find	16/15/15
10	slow loading	15
11	hard to watch	15
12	slow moving	14
13	unable to view	14
14	slow to load	14
15	couldn't figure	14
16	impossible to watch	14

objects and adjectives. For example, in the crowd-user feedback, "Unable to get it to work and no help available.", the adjective "unable" is associated with the verb "to get" representing a commonly occurring issue or bug with an application under discussion. Similarly, in the comment, "It says unable to connect to the internet when creating an account. Don't waste your time.", the adjective "unable" is combined with the verb "to connect", resulting in a commonly reported bug or issue. It represents meaningful information to software engineers, which needs attention from software vendors to improve software app features. Therefore, to automate the process, we first captured the verbs in the end-user feedback using NLTK. In total, 3040 unique verbs cum operations are identified, amongst which the top 15 most frequently occurring operations are shown in Table 6. For example, the operation "want" appears 2697 times in the data set, and the "find" operation appears 2242 times. Furthermore, the verbs alone do not represent helpful information for software engineers [6]. Therefore, by developing a co-occurrence algorithm, we associate the identified verbs and cumulation operations with previously captured adjectives to recover useful information on the issues or bugs. In total, 22319 unique adjective and verb pairs are identified in the data set, among which the top-16 frequently occurring pairs are shown in Table 7. The adjective "unable" appear quite often with the verb "to get," "to use," "to watch," "to play," etc., as shown in Table 7. Also, the adjectives do not always appear immediately after verbs in the data set, but there could be a few words between them and vice versa. For example, the possible issues "impossible to watch" and "hard to navigate" occurred 30 times, as shown in Table 7.

3) RELATIONS BETWEEN NOUN-OBJECT, ADJECTIVES (ISSUES) AND OPERATIONS

Additionally, by manually analyzing end-user comments in the ASA store, we learned that associating captured adjectives, verbs, and nouns together can give valuable and helpful information on frequently reported issues in the ASA store. Therefore, we developed the co-occurrence algorithm using the NLTK to find the association relationships between the identified adjectives, noun objects, and operations. Also, in the research dataset, we observed that an object or adjective does not always appear immediately after an operation or verb; there could be a few words between them. Therefore, we introduced the threshold range parameter λ to find its co-occurrence. The value is defined manually; we set it to 5, be inclusive, and observe its occurrence patterns in the dataset. For example, in the captured pair of entities, operations, and the adjective "disappointed with this app," where the adjective is "disappointed (expressing the end-user emotion with the app)," the noun is "app," and the verb is "with," demonstrating the relationship among the end-user emotion and the software app under discussion. The words in between them are "to" and "is." In this way, the proposed approach captured many pairs, among which the 14 most frequently occurring pairs are shown in Table 8. The pairs identified by the co-occurrence algorithm represent frequently occurring valuable information for the software engineers that can be utilized to improve the performance of low-ranked software applications by incorporating them in the software development and evolution processes in a timely manner. For example, the pair "long time to load," which is captured "17" times in the end-users feedback, demonstrates useful requirements-related information in which crowd-users complain about the current performance problem of the software application under discussion in the ASA store, such as, "horrible, it crashes too much and crashes during calls. then sometimes never loads messages when I'm on PC Skype. it bugs me so much, and loading messages takes a long time. I say this app is not recommended." Such information is necessary for the software development team if it is captured and incorporated into the decision-making process in a timely manner to earn end-user satisfaction and improve app ratings and quality. Algorithm 1 is developed to identify the adjectives-verb-nouns pairs. Algorithm 1 also identifies the pairs, such as "horrible customer service" or "bad sound quality," which does not explicitly include a

TABLE 8. **Frequently identified issues categorized by user feedback.**

Rank	Combination (Adj + Noun + Verb)	Frequency	Issue Category
1	Long time to load	17	Performance
2	Disappointed with this app	13	User Satisfaction
3	Poor picture quality	15	Media Quality
4	Poor video quality	11	Media Quality
5	Horrible customer service	8	Customer Service
6	Expensive for what you get	7	Value for Money
7	Unable to get this app	6	Accessibility
8	Bad sound quality	6	Media Quality
9	Unable to get this app to work	5	Accessibility
10	Disappointed in this app	5	User Satisfaction
11	Please fix this app	5	Technical Issues
12	Unhappy with this app	5	User Satisfaction
13	Poor streaming quality	5	Media Quality
14	Unable to play video	5	Media Quality

verb, but the implied verb can be "to have" and "is." For example, the end-user commented, "Whenever I am on a Skype call with my friend, it drops the call randomly. And when it doesn't drop the call, it is really bad sound quality. Please fix this, and I would give it 5 stars", where the "is" can be referred as a possible verb and the identified pair can be presented as "the sound quality is bad."

An algorithm is developed to extract frequently occurring issues by associating adjectives, verbs, and nouns by analyzing end-user comments in the ASA store. The algorithm accepts end-user comments as input and returns frequently occurring issues and their number of occurrences as output. An empty dictionary called "relations" is created for the algorithm, storing adjective, noun, and verb pairs. The algorithm loads stopwords from the NLTK corpus to capture the meaningful information for the software vendors between verbs, adjectives, and nouns and is stored in the "stop_words." The proposed algorithm then iterates over the end-user reviews stored in the "Base_Review." It tokenizes the end-user review into individual words and tags the tokens with the POS tags to determine the grammatical category of each word. Also, if a token is found in the "entity_chunks," which is a predetermined set of important words, the algorithm searches for the nearest adjective (JJ) 5 tokens before the current token. If a valid adjective is found, it is added to the "all_adjectives" set, and its count in the "verbs_relation" dictionary is increased. The algorithm also determines whether the adjective is already in the "co_occur" dictionary. If it does not, it inserts an empty dictionary entry for that adjective. The current token (noun) count in the "co_occur" dictionary under the associated adjective key is then increased. Finally, the algorithm identifies frequently occurring issues in the ASA store by combining verbs, adjectives, and nouns and their count in the "relations" dictionary. The algorithm iterates over the "relations" dictionary and lists the frequently identified issues and frequencies.

Algorithm 1 Extract Frequent Issues

```
1: Initialize an empty dictionary relations, verbs_relation, co_occur
2: Initialize an empty dictionary verbs_relation
3: Load stopwords from the nltk corpus and store them in the set stop_words
4: for each fea in data['Base_Review'] do
5:     Tokenize fea into words and store them in the list tokens
6:     Perform part-of-speech tagging on tokens and store the tagged tokens in the list tags
7:     for each index and (token, pos) pair in enumerate(tags) do
8:         if token is present in entity_chunks then
9:             Initialize a variable i with the value of index - 1
10:            while $i \geq 0$ and index - $i \leq 5$ and tags[i][1] $\neq$ 'JJ' do
11:                Decrease the value of i by 1
12:            end while
13:            if $i \geq 0$ and tags[i][1] == 'JJ' and tags[i][0] not in stop_words then
14:                Add tags[i][0] to the all_adjectives set
15:                if tags[i][0] is already present in the verbs_relation dictionary then
16:                    Increment the count of tags[i][0] in the verbs_relation dictionary
17:                else
18:                    Set the count of tags[i][0] in the verbs_relation dictionary as 1
19:                end if
20:                if tags[i][0] not in co_occur then
21:                    Add an empty dictionary for tags[i][0] in the co_occur dictionary
22:                end if
23:                if tags[index][0] not in stop_words and tags[index][0] not in co_occur[tags[i][0]] then
24:                    Increment the count of tags[index][0] in the co_occur[tags[i][0]] dictionary
25:                end if
26:                Initialize an empty string variable temp
27:                for each j in the range from i to index + 1 do
28:                    Append tags[j][0] to temp with a space separator
29:                end for
30:                if temp is already present in the relations dictionary then
31:                    Increment the count of temp in the relations dictionary
32:                else
33:                    Set the count of temp in the relations dictionary as 1
34:                end if
35:            end if
36:        end if
37:    end for
38: end for
39: Extract frequent issues using the Counter function on the relations dictionary and retrieve the most common items
40: Store the frequent issues in the frequent_issues list
```

VI. IDENTIFY END-USER OPINIONS WITH THE IDENTIFIED ISSUES

By manually analyzing end-user comments, we learn that crowd-users express their opinions, sentiments, or emotions when they report issues in the ASA store. Also, end-users submit many reviews in the ASA store against the software applications, making it challenging to identify crowd-user emotions manually. Therefore, various ML algorithms are employed to better understand the end-user emotions with the reported issues and automate the process. Martens and

Johann [48] reported that identifying sentiments or emotions with the requirements-related information can improve the developer's understanding of the current challenges with the software application and predict the end user's perception of the new release. For this purpose, first, we need to curate a research data set for identifying end-user emotions with the associated captured issues in the ASA store. We developed a research data set containing 7989 end-user comments. Each included user comment in the data set represents an identified issue in the ASA store. To make the data perusable for the ML algorithms, we need to develop a unique coding guideline to help the coders annotate the research data set using the content analysis approach. To enhance the rigor of our methodology, we integrated grounded theory to systematically develop a theoretical framework directly from the data, which guided the creation of our coding guidelines. This approach helped us understand and categorize the nuanced expressions of user emotions. Furthermore, our content analysis was structured around these guidelines, ensuring a systematic and quantifiable qualitative data analysis. Finally, to ensure the accuracy of our data annotation, several measures were implemented. We conducted a training session for all coders to calibrate their understanding of the coding guidelines and conducted a pilot annotation phase to refine them. Inter-rater reliability was assessed using Cohen's kappa [32], achieving a substantial agreement level, thus validating the consistency of our annotations. After establishing a reliable annotated dataset, we pre-process this data, re-sample it to address class imbalance, and then apply various ML algorithms to identify end-user opinions. Each step is elaborated on below:

A. GROUNDED THEORY FOR IDENTIFYING END-USERS' EMOTIONS

To curate an annotated sample for the ML experiment, it was imperative to develop a coding guideline by manually analyzing and evaluating end-user comments in the ASA store employing the grounded theory approach [62]. Through this systematic and qualitative analysis of frequently occurring user opinion information in the ASA store, a unique end-user emotion theory was formulated. This phase resulted in the creation of a consolidated coding guideline document, iteratively developed to outline the distinct coding concepts identified for implementing the proposed method, as depicted in Figure 1. This document serves as a baseline for establishing a truth set (annotated) for the proposed approach, facilitating consensus among annotators and preventing conflicts [63], [64]. Moreover, it includes definitions and examples of each identified end-user emotion concept within the ASA store. The development of these coding guidelines involved iterative discussions, refinements, and resolution of disagreements by the first two authors to further enhance and stabilize them.

The diversity or uniformity in the emotions identified from the reviews can be quantified using entropy. The entropy H for the distribution of emotions can be defined as:

$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{1}$$

where $p(x_i)$ is the proportion of the emotion x_i within the dataset and n is the number of distinct emotion categories identified.A higher entropy value suggests a more uniform distribution of emotions across the dataset, indicating a wide range of emotional responses, whereas a lower entropy value points to a skewed distribution, suggesting that certain emotions are more predominant

B. ANALYZING OF END-USER COMMENTS

In the following subsections, we elaborate on the various concepts identified in the ASA store using grounded theory and the annotation process to annotate a sample data set to automatically identify end-user emotions with the captured issues using different ML algorithms.

1) END-USER OPINIONS CONCEPTS

For the proposed classification approach, we collected end-user comments representing the captured issues in the ASA store. We are interested in identifying the end-user opinions or emotions of the type sadness, fear, disgust, and anger with the associated captured issue by aiming to improve the quality and overall user satisfaction of low-ranked software applications by helping developers better understand the reported issues in the ASA store. In comparison, preference will be given to the captured issues based on the identified end-user emotions in developing the next release of the software application to improve user satisfaction. To determine the corresponding end-user opinion, we solicit various concepts of user emotions by manually analyzing end-user comments in the ASA store. Furthermore, the related and similar end-user emotion codes were merged into concepts, which resulted in the theory of user emotions for software engineering. The codes developed and included in the coding guidelines were chosen based on their frequency in the crowd-user reviews and their relevance to the aim of the proposed research approach. As a result, if the end-user emotion code appeared only once or twice during the manual annotation process, it is then grouped with the appropriate user emotional concept. Similarly, if one coder identifies an end-user emotion code, it is rejected or eliminated if considered irrelevant to the proposed approach. Using grounded theory, we identified end-user emotion codes against low-ranked software applications in the ASA store, i.e., anger, disgust, sadness, and fear. The details of each recognized code are given below:

a: SADNESS

The code "sadness" is assigned to an end-user comment in the ASA store where crowd-users express grief or pain over a software feature, a bug or issue in the software application, or an issue reported in the software's quality or with the overall software. Generally, sadness occurs when a beloved one or an essential item is lost. Similarly, crowd-users express "sadness" when unsatisfied with a single feature or overall software application in their reported comment in the ASA store. Some examples are "For some time, the app shows invalid file when trying to open almost any PDF, I used to love the app and reinstalled hoping they fixed the problem, but it doesn't work" and "Fun way to connect with my granddaughter. And I was unable to figure out how to use. I am sad for it".

b: ANGER

The concept of "anger" is assigned to end-user feedback in the ASA store and refers to what stops crowd users from achieving the desired goals with either a single feature or overall software application. Also, a specific software feature or the overall application behaves unexpectedly or unfairly. In general, "anger" is one of the seven universal emotions that arise when we are not reaching our vision or are maltreated. At its most extreme, anger can be one of the most dangerous emotions due to its possible link to violence. Such information is of pivotal importance for software and requirements engineers, as it might result in the uninstallation of the software applications. Therefore, the proposed approach can be identified earlier to earn user satisfaction and improve the overall quality of the software application. Some examples of the "anger" concept in the ASA store are: "I downloaded and deleted because of the difficult layout." Also, I got a problem like this: "invalid file when trying to open." Can you please fix this? "You're bothering me, bro," and "They are big liars and stink." Every time I tried to open it, it would crash and then tell me an unknown error had occurred.

c: DISGUST

The concept "disgust" is assigned to end-user feedback in the ASA store where crowd-users strongly dislike a specific software feature, a bug or issue that has arrived in the software application, an issue reported in the software's quality, or the overall software application. In general, "disgust" is one of the seven universal emotions, defined by a strong dislike for something unpleasant. In requirements engineering, it is essential to identify the "disgust" emotions of crowd-users in the ASA store, as they mainly describe the software application's negative behavior, specifically the non-functional attributes. For example, "You can only make calls to users of this application." I couldn't even sign up; it just exited out and said the app had foreclosed three times, so I uninstalled it and got a better texting app that lets me create an account. And I'm on my Kindle Fire HD. "I don't like this app." and "I couldn't even sign up; it just exited out and said the app forbade it three times, so I uninstalled it and got a better texting app that lets me create an account." "Don't like it." Identifying such requirements-related information earlier can help improve software applications' quality and end-user satisfaction.

d: FEAR

The concept "fear" is assigned to end-user feedback in the ASA store where crowd-users sense or feel a possible threat with a specific software feature or the overall software application. In general, "fear" is induced by the threat of harm, which can be physiological, emotional, or mental in origin and can be real or imagined. In requirements and software engineering, "fear" is often considered a "negative" emotion. Identifying it earlier in the software development phase can give the crowd-users enough confidence in the software's functionalities and overall performance. Some examples of "fear" emotions in the ASA store are "useless." I was unable to make the simplest edit to my PDF. "I wish I could get my money back, but Amazon makes it impossible." and "I installed this app, and that's as far as I got." Every time I tried to open it, it would crash and then tell me an unknown error had occurred. I uninstalled and reinstalled it a few times but never opened it. "I can't say if I liked it or not because I never got to try it."

C. MANUAL CONTENT ANALYSIS

After capturing the most frequent end-user opinions concepts in the ASA store, we need to annotate the crowd-user comments by critically analyzing each comment using content analysis technique [32], as described by Neuendorf [64] and Maalej and Robillard [63]. The code for sadness, anger, disgust, or fear is assigned to each end-user comment based on the coding guidelines developed in the previous step of the proposed approach. We used the content analysis technique [32] to annotate the crowd-user comments in the ASA store (sadness, anger, disgust, and fear) by critically examining each end-user feedback to determine the relevant end-user emotion code. The objective of crowd-user feedback annotation is to determine the frequency of end-user emotion concepts and develop a truth set used to train and evaluate the various ML classifiers in an attempt to address RQ3. The manuscript's first two authors independently assessed each end-user feedback in the dataset to determine crowd-user emotion type. Furthermore, a unique coding guideline is developed to reduce misunderstandings and disagreements among the annotators, containing precise instructions, descriptions, and examples of end-user emotional types in the ASA store.

1) ANNOTATION OF CROWD-USER ISSUES COMMENTS

Each coder is provided with a coding guideline and a coding form, including a complete set of crowd-user feedback representing identified issues in the ASA store, to annotate the end-user comments in the curated data set. Each coder independently analyzes the title and the main contents of each end-user feedback to determine the user emotion associated with the feedback as either sadness, anger, disgust, or fear. A coding sample of the end-user emotions in the ASA store is depicted in Table 9. The columns "User Name," "Review Rating," "Review Title," and "Review Contents" represent

TABLE 9. Shows our labelled data-set.

Review Contents	Review Title	End-User Name	Review Ratings	Emotion Type
Downloaded and deleted because I faces problem in the difficult layout. And also I got problem like this "invalid file when trying to open". Can you please fix this? Bothering me bro.	Never Used it	Larry	1.0 out of 5	anger
Didn't work for me. Run out of calls quick and it's impossible to get more without paying	Two Stars	David W	2.0 out of 5	Disgust
I was unable to figure out how to use, I think I will not use this app	pure garbage	Kevin cox	1.0 out of 5	Fear
I am crying. Bought and paid the app. and don't work on my Kindle Fire HD 10.For some time the app shows an invalid file when trying to open almost any PDF, I used to love the app and reinstalled hoping they fixed the problem, but still, it doesn't work	don't work	jennet neville	1 out of 5	Sadness

TABLE 10. Distribution of end-user emotions for the identified issues in the ASA store.

Anger	Disgust	Fear	Sadness	Total Annotated Reviews
1746	881	3067	2295	7989

the end-user feedback information in the ASA store, while the column "Emotion Type" defines the possible emotion types that need to be recovered by the coder, including sadness, anger, disgust, and fear. The software applications in the issue coding form organize the crowd-user comments selected for the annotation process. For this purpose, a random sample of end-user feedback, denoted as $N = 7989$, is chosen from all the crowd-user comments in the data set, representing the captured issues in the ASA store. The sample size of end-user reviews collected for the annotation process is organized to represent each software category and application equally. The feedback for a particular software application appears first in the coding document, followed by the remaining user comments in a specific order. Additionally, links to each software application included in the annotation process are provided in the coding document to aid coders in case of confusion in understanding the user comments. Coders are allowed to pause and resume the issue annotation process at their convenience. The average time taken by coders to complete the annotation process is reported to be $T = 12$ working hours.

After completing the individual annotation tasks, all individual coding results are compiled, and a reconciliation method is applied, as illustrated in Figure2, to identify and resolve discrepancies. The inter-coder agreement, quantified by the percentage of agreement, is reported to be $\alpha = 87\%$. Concurrently, Cohen's kappa (κ), a measure of inter-rater reliability, is calculated as:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{2}$$

where P_o is the relative observed agreement among raters, and P_e is the hypothetical probability of chance agreement. For this study, Cohen's kappa is identified as $\kappa = 0.65$, indicating a substantial agreement between the annotators according to Cohen's kappa scale. Following the reconciliation process, a conflict-free labeled dataset is curated and used as input for various ML algorithms to identify and categorize end-user emotional types within the ASA store.

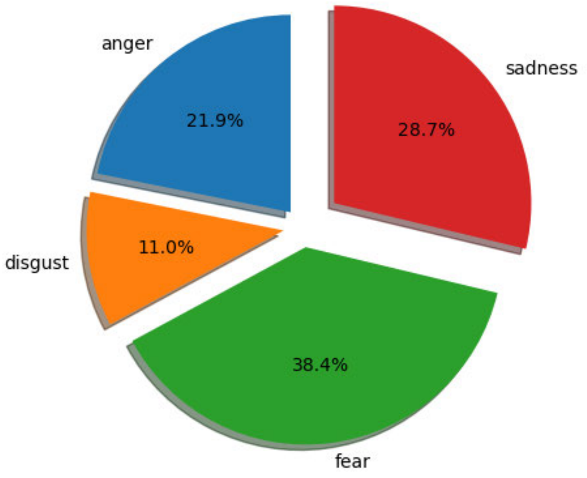


FIGURE 2. Shows issues sentiment distribution in the data-set.

2) FREQUENCY OF THE END-USERS EMOTIONS FOR THE SUBMITTED ISSUES

To better understand the curated data set and end-user feedback, we employed a content analysis approach to recover and identify the frequencies of the crowd-user emotion types and their relevancy to the associated issues. Also, studying the significance of such information for software developers will provide opportunities to enhance the performance of existing low-rated software apps in the ASA store. For this purpose, the proposed study reveals that crowd-user feedback in the ASA store is essential for identifying and capturing frequently reported issues and corresponding end-user emotions. Figure 2 shows the distribution of end-user emotions in the curated data set from the ASA store, and Table 10 describes the number of end-user feedback across each user's emotional type. We found that the "fear" end-user emotion type occurred most frequently in the curated research data set, with an overall percentage of 38.4% (3067 comments), as depicted in Figure 2 and Table 10, respectively. One possible reason might be that most crowd-users feel a potential threat when using a specific software feature or app. The second most prominent end-user emotion type captured in the end-user reviews is "sadness," which is identified as 28.7% (2295 end-user comments) of the total 7989 end-user comments. End-users show pain or grief and become sad when the software applications under discussion do not satisfy their desired needs and requirements. Next, the most common and visible end-user emotion type recovered in the ASA store is "anger," 21.9% (1746 comments). It is because end-users become frustrated and angry when they suddenly stop achieving the desired goals with either a single software feature or the overall software application. Finally, the "disgust" end-user emotion type is recovered as the least essential and most prominent sentiment type in the ASA store. The disgust emotion type is 11% (881 crowd-user comments) of the overall end-user comment selected for the annotation process because end-users do not frequently express their extreme dislike for a specific software feature, bug, or issue that has arisen in the software application. The analysis encourages software researchers, vendors, and developers to propose approaches by automating the issue identification process with the corresponding emotion type to better understand the issue's severity. Later, the process may be embedded in the existing software evolution phase to achieve higher user satisfaction by incorporating the identified issues in a timely manner.

VII. AUTOMATED CLASSIFICATION OF END-USER EMOTION

Social media comments are a rich source of publicly available text. Despite researchers' claims that crowd-users submit a large amount of information on these platforms, it becomes difficult, time-consuming, and challenging to capture issues and associated end-user emotions manually [8], [44], [45]. Also, previous research shows that extracting and identifying end-user emotions from crowd-user comments on social media is more complex than recognizing emotions through faces, voices, and gestures. Furthermore, the end-user's confusing context, the complexity of natural language, and its constant evolution (new expressions every day) are just a few challenges that make text-based end-user emotion identification challenging [65]. For this purpose, we are interested in employing different ML algorithms and recording their performance in automatically identifying end-user emotion types in the ASA store. For this, 71853 crowd-user feedback was collected from 45 various low-rating software applications, for which we first identified the frequently reported issues (explained in Section III). Next, we identified crowd-user feedback that represented at least one captured issue in the data set. In total, 49,700 such end-user comments were identified. We selected 7,898 end-user comments using a stratified random sample method to recover the end-user emotion type using distant ML classifiers and record their performance. To refine our approach to ML, we meticulously outlined our feature engineering process, which involved extracting n-grams, term frequency-inverse document frequency (TF-IDF) features, and using word embeddings to represent text data effectively. Each model was trained using a robust validation framework employing a 10-fold cross-validation methodology [66], ensuring that our results are reliable and generalizable. The ML classifiers selected for this task are Support Vector Machine (SVM), MNB, LR, KNN, MLP, Gradient Boosting (GB), Voting Classifier, Ensemble Methods, and RF. To address the challenge of imbalanced data, we implemented both oversampling and undersampling techniques. This ensured a balanced representation of classes, crucial for training unbiased machine learning models. Our evaluation measures included accuracy, precision, recall, and F1-score, providing a comprehensive view of model performance across multiple dimensions. Furthermore, we used the K-fold cross-validation methodology [66] to validate and train the various ML algorithms to identify end-user emotion types in the ASA store. Also, we are interested in adopting those ML features from the software engineering literature that lead to better classification results. The details of each ML experiment step are explained below:

A. EXPERIMENTAL SETUP

In the ML experiment, we used various text pre-processing and feature engineering approaches to examine the performance of the ML algorithms and demonstrate their efficacy in classifying end-user emotions in the ASA store. Before text pre-processing, text features selection, data resampling, and classifier training, we chose ML classifiers from the software engineering literature that did better on the textual data from social media [34], [67], [68]. To recap, the algorithms utilized in the experiment are SVM, RF, MNB, LR, Ensemble Methods, GB, MLP, Voting Classifier, and KNN. The voting algorithm is a classification method that generates predictions by integrating the classification results from multiple classifiers that perform better in a specific classification task. Similarly, the ensemble ML classifier builds a bunch of classifiers and then categorizes incoming text data points based on a (weighted) vote on their predictions. Moreover, each ML classifier is trained and validated with the crowd-user emotions to automatically identify and classify different emotions in the ASA store, i.e., anger, fear, disgust, and sadness. The ML experiment is conducted in Python, as elaborated below.

B. PREPROCESSING AND FEATURE ENGINEERING

Crowd-user submissions often contain raw feedback that includes special characters, symbols, HTML tags, links, etc. Directly utilizing such raw textual data for machine learning classifiers might impede their classification performance. Therefore, it is imperative to cleanse the input data through several preprocessing steps. These steps include converting all text to lowercase, and removing HTML tags, URLs, punctuation, special characters, and alphanumeric words from the end-user comments. Additionally, text lemmatization, a process of text normalization, is applied to further reduce the words in the dataset to their base or root form to enhance the classifiers' performance.

Formally, let D be the raw dataset containing end-user comments, where each comment $c_i \in D$ undergoes preprocessing to yield a cleaned dataset D'. The preprocessing function $\mathcal{P}$ can be represented as:

$$D' = \mathcal{P}(D) \tag{3}$$

where $\mathcal{P}$ encompasses the operations of case normalization, HTML tag removal, URL removal, punctuation removal, and lemmatization.

Subsequently, stop words are eliminated from the cleaned dataset. Stop words are the most frequently occurring words in the dataset that contribute minimally to the classification outcome, such as "is", "the", etc. Let $\mathcal{S}$ denote the stop word removal operation applied to D' to obtain the final preprocessed dataset D'':

$$D'' = \mathcal{S}(D') \tag{4}$$

Feature Engineering: For transforming the end-user comments into a numerical format comprehensible by ML classifiers, the Label Encoder function from the Python "sklearn.preprocessing" class is utilized. This operation converts the categorical labels of end-user emotions (e.g., anger, fear, disgust, sadness) into numerical form. Let $\mathcal{L}$ denote the label encoding operation with the emotional labels set E, and numerical labels set N:

$$N = \mathcal{L}(E) \tag{5}$$

Additionally, textual features are extracted using the Term Frequency-Inverse Document Frequency (TF-IDF) and CountVectorizer methodologies to generate a features-documents matrix. TF-IDF, for example, is calculated as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{6}$$

where $\text{TF}(t, d)$ is the term frequency of term t in document d, and $\text{IDF}(t)$ is the inverse document frequency of term t across the corpus.

C. DATA IMBALANCE

These days, a significant technical challenge in supervised ML is imbalanced data sets [69], elaborated as unequal instances of user comments across the identified labels in the data set. The research data set for this experiment is unbalanced, as depicted in Figure 3. The 38.4% of the end-user feedback in the dataset was recognized as "fear" user emotions, and 11% were recovered as "disgust" emotions. As a result, training a classifier on an imbalanced data set causes the ML classifier to skew towards the majority class and ignore the minority classes, i.e., the annotation classes containing a limited number of records in the data set. To overcome the imbalanced data issue, we adopted two popular rebalancing approaches in software engineering literature [69], i.e., undersampling and oversampling, to improve

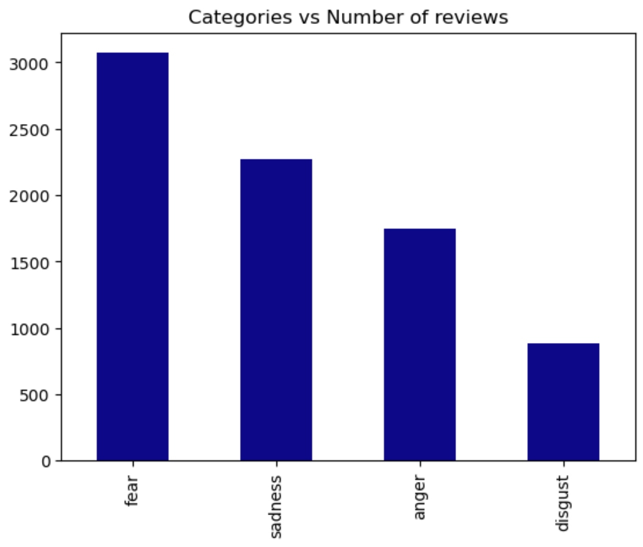


FIGURE 3. Distribution of end-user comments against emotions categories.

the performance of the classifiers in identifying end-user emotions.

Under-sampling is a non-heuristic methodology for balancing class distribution that randomly excludes or eliminates majority class samples [70]. The undersampling method can be mathematically represented as:

$$R_{\text{under}} = \frac{N_{\text{minority}}}{N_{\text{majority}}} \times 100 \tag{7}$$

where R_{under} is the percentage of majority class samples to be kept, N_{minority} is the number of minority class samples, and N_{majority} is the number of majority class samples.

While oversampling is a non-heuristic methodology for balancing class distribution that includes randomly repeating minority class examples [71]. The oversampling method can be mathematically represented as:

$$R_{\text{over}} = \frac{N_{\text{desired}} - N_{\text{minority}}}{N_{\text{minority}}} \tag{8}$$

where R_{over} is the number of times minority class samples need to be duplicated, N_{desired} is the desired number of minority class samples after oversampling, and N_{minority} is the original number of minority class samples.

Also, in the ML experiment, we used receiver operating characteristic (ROC) [72], and precision-recall [73] curves to assess and identify which data balancing approach, i.e., oversampling or undersampling, is appropriate for training the ML classifiers. To accomplish this, we identify and calculate the percentage of true positives against false negatives for each classifier employed in the ML experiment. Figure 4 shows ROC curves for RF and MLP classifiers that explore and evaluate under and oversampling methodologies to identify the best resampling methodology for capturing end-user emotions. These two ML classifiers were selected based on their better performance in identifying and classifying end-user emotion types in the ASA store. Furthermore, we identified from the classification experiment that ML algorithms perform better with the oversampling approach and yield better results than the undersampling approach. One possible reason for the degraded classification results is that valuable and critical information is discarded while employing an undersampling approach [74].

D. DATA ASSESSMENT & TRAINING

We employed a stratified 10-fold cross-validation methodology to train, validate, and test ML classifiers on end-users emotional feedback from the ASA store. The classifiers are trained on 9 folds of cross-validation and validated with 1 fold. The classifier training and validation are repeated 10 times by rotating the testing and training folds. The main advantage of cross-validation in training and validating the classifiers is to assess how well an ML model performs with the limited availability of textual data. The stratified K-fold approach has recently been widely adopted for training and validating ML algorithms. Each fold in the stratified K-fold approach has the same number of labels belonging to each class. To identify and test the efficacy of the ML classifiers in capturing the end-user emotion types, we computed and presented the average results obtained from the 10-fold cross-validation. We employed recall (R), precision (P), and F-measure scores to analyze and validate the machine learning classifiers. Below are the formulas for calculating P and R:

$$Pk = \frac{TPk}{TPk + FPk} \tag{9}$$

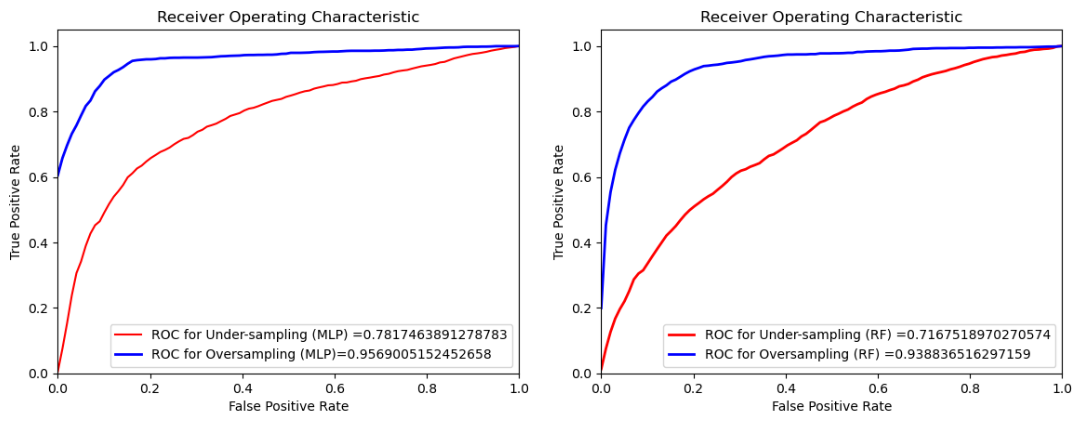
$$Rk = \frac{TPk}{TPk + FNk} \tag{10}$$

Pk is the proportion of true positives (correctly identified crowd-user emotion types) to the complete set of end-user feedback in the data set (both correctly and incorrectly classified user emotion types). Similarly, Rk measures and identifies the classifiers' reliability to capture end-user rationale types. TPk shows the number of crowd-users' emotions correctly identified as type k; FPk shows the number of end-user emotions wrongly identified as type k; and FNk indicates the number of end-user emotions incorrectly identified as not type k. Finally, F1 represents the harmonic mean between Pk and Rk.

E. END-USERS EMOTION IDENTIFICATION RESULTS

The optimized and improved results of various ML algorithms in identifying end-user emotion types in the ASA store are shown in Table 11. It can be concluded from Table 11 that the majority of the ML classifiers perform better in classifying end-user emotion types, but MLP, RF, and SVM perform comparatively better and outperform other ML classifiers. They achieved a higher classification accuracy of 82%, 80%, and 70%, respectively. The ML classifiers, i.e., RF-TFIDF, MLP-TFIDF, and MLP-CountVectorizer give the highest precision value of 82%, 79%, and 79%, respectively while MLP-CountVectorizer and MLP-TFIDF yield the highest recall and F-measure values of 82% and 81%, respectively, in identifying anger as an end-user emotional type in the ASA store. Next, MLP-TFIDF and MLP-CountVectorizer classifiers result in the highest precision, recall, and F-measure values of 77%, 79%, 78%, 74%, 82%, and 78%, respectively, in identifying and classifying disgust as an end-user emotion type in the user feedback on ASA store. Although RF-CountVectorizer results in the highest F-measure values, i.e., 81% in identifying end-user emotion type as disgust while its performance becomes degraded for identifying Precision and Recall values. Similarly, to classify the fear emotion type in the end-user feedback, the MLP-TFIDF and MLP-CountVectorizer classifiers yield the highest F-measure value of 85%, respectively. Also, when classifying end-user comments as fear emotion type, the RF-TFIDF and RF-CountVectorizer ML algorithms result in better F-measure scores of 83% and 81%, respectively. Likewise, the MLP-TFIDF and MLP-CountVectorizer classifiers yield the highest F-measure value of 82%, respectively, in classifying sadness as an end-user emotion type in the ASA store. These ML algorithm results encourage the identification of end-users emotional types to better understand the issues and their severity in the ASA application store. It provides opportunities for the software vendors to improve the current ranking of low-ranked software applications by emphasizing the identified issues during the software evolution process. To summarize, comparatively, most of the ML classifiers shortlisted for the proposed experiment perform comparatively better in identifying end-user emotion types in the ASA store. However, in the proposed automated approach, MLP, RF, and SVM classifiers outperform other ML algorithms in identifying different end-user emotion types, i.e., anger, disgust, fear, and sadness. Also, based on the classification result shown in Table 11, we conclude that the MLP or RF algorithm can be selected as the best classifier to identify and recover end-user opinions about the captured issues in the ASA store and improve the overall quality of low-ranked software applications by timely identifying frequently mentioned issues together with end-user emotion types to understand the issues better. This study can be considered a baby step toward understanding end-users emotions while reporting issues with software applications. It will help software developers prioritize the issues identified in social media based on emotional intensity based on our knowledge.

Further, to validate the baseline configuration of ML classifiers used to identify and classify end-user emotions with the identified issues of anger, disgust, fear, and sadness in the ASA store, we investigate the learning curves of the best-performing classifiers, such as MLP and RF. Learning curves are crucial as they demonstrate how the addition of training instances of end-user comments impacts classification accuracy and performance. Figure 5 (a) illustrates the learning curve of the RF classifier, showcasing its capability in identifying and classifying end-user emotions



(a) MLP (b) RF

FIGURE 4. Shows MLP and RF machine learning ROC curves showing the performance of oversampling and undersampling.

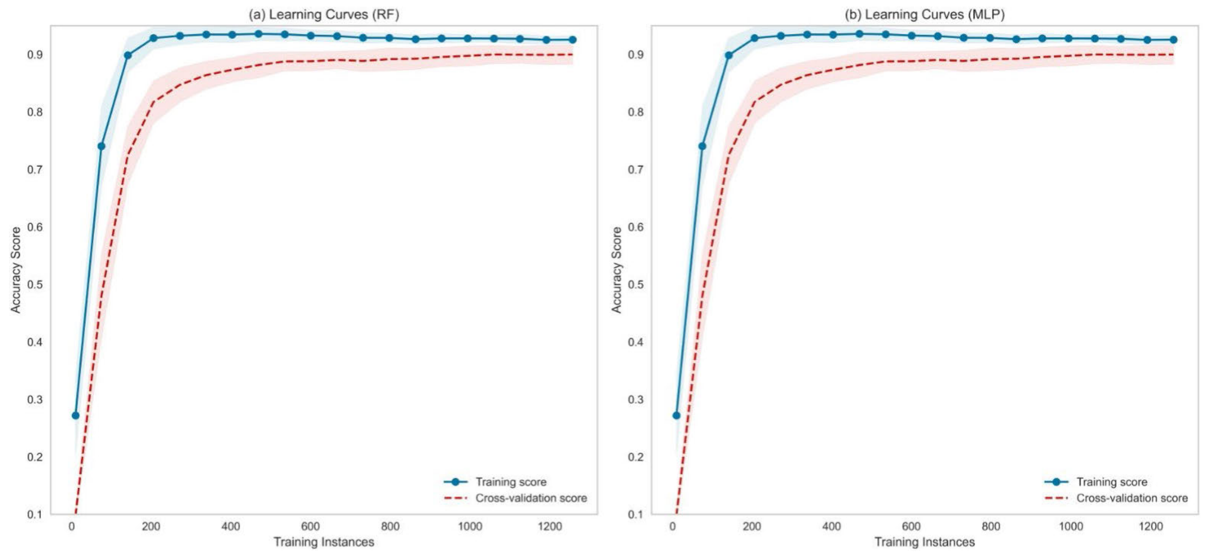


FIGURE 5. Learning curves of RF (a) and MLP(b) algorithms.

for the identified issues in the ASA store. In a similar vein, the MLP classifier, as depicted in Figure 5 (b), is highlighted for its efficacy in the classification task, thereby being recognized as the optimal classifier for discerning end-user emotions in the ASA store. The analysis of these learning curves plays a pivotal role in understanding the performance dynamics of the RF and MLP algorithms, with a particular emphasis on the superiority of the MLP algorithm for this specific classification challenge.

F. COMPARISON WITH THE STATE OF THE ART

In advancing the domain of app review analysis, the proposed study integrates NLP and ML to uniquely identify software issues and the corresponding user emotions from Amazon app store reviews, focusing specifically on low-rated software applications. The proposed approach is different from the state-of-the-art in a few perspectives: 1) it focuses on comparatively low-rating software applications, unlike previous approaches that analyze popular applications; 2) it approaches certain co-occurring algorithms that can identify frequently occurring issues using various POS rules; 3) the proposed approach identifies end-user emotions including Anger, Disgust, Fear, and Sadness with the associated issues to better understand the identified issues, as shown in Table 12. The proposed methodology is contrasted against other significant contributions in the field, emphasizing the unique aspects of the proposed work. Below is a discussion of how the proposed study compares with other seminal works in the domain, as shown in Table 12. Guzman and Maalej [21] primarily analyzed app reviews from the Apple App Store and Google Play Store. Their work focused on extracting fine-grained app features and associated sentiments using topic modelling and sentiment analysis approaches, respectively. While foundational, their study did not link the emotional aspect of user feedback to specific software issues as we have. Phong et al. [75] addressed the challenge of extracting relevant information from many reviews across various platforms. They developed advanced NLP techniques to facilitate information extraction, setting a precedent for handling large datasets without

TABLE 11. Results obtained by identifying end-user emotions with ML classifiers.

Labels	ML classifiers	Precision	Recall	F-Measure
Anger	MLP-TFIDF	79	82	81
	MLP-CountVectorizer	79	82	81
	RF-TFIDF	82	74	78
	RF-CountVectorizer	76	76	67
	SVM-TFIDF	61	64	62
	SVM-CountVectorizer	67	68	68
Disgust	MLP-TFIDF	77	79	78
	MLP-CountVectorizer	74	82	78
	RF-TFIDF	74	76	75
	RF-CountVectorizer	59	78	81
	SVM-TFIDF	51	73	60
	SVM-CountVectorizer	55	67	61
Fear	MLP-TFIDF	77	83	85
	MLP-CountVectorizer	88	82	85
	RF-TFIDF	83	81	83
	RF-CountVectorizer	84	78	81
	SVM-TFIDF	76	62	68
	SVM-CountVectorizer	78	69	74
Sadness	MLP-TFIDF	82	83	82
	MLP-CountVectorizer	81	83	82
	RF-TFIDF	75	85	80
	RF-CountVectorizer	80	78	79
	SVM-TFIDF	65	66	67
	SVM-CountVectorizer	69	73	71

ML Algorithms	Accuracy
SVM	70
RF	80
MLP	82

focusing on emotional analysis related to software bugs. Johann et al. [76] proposed an automated approach to extract frequently mentioned features from the app store and verify them with the features mentioned in the app description page. For this purpose, authors developed 18 POS-based patterns to identify frequently mentioned features. Their approach focuses on popular and high-rated apps aiming to identify features only unlike the proposed approach which foresees frequently occurring issues for low-rated software apps. However, the proposed approach is inspired by Johann et al. approach in terms of developing a POS-based pattern to identify frequently occurring issues for low-rating software applications. Williams and Mahmoud [24] explored how the vocabulary and length of reviews correlate with user ratings in major app stores. Their content analysis provided correlations but did not delve into the emotional nuances that might affect user perceptions and app ratings. Li et al. [14] combined machine learning and sentiment analysis to mine user opinions from reviews and social media platforms. Their work aimed to understand user requirements and preferences, offering a broad analysis of user sentiment without a specific focus on the linkage to software quality issues. Using sentiment analysis, Malik et al. [54] provided a detailed analysis of user opinions concerning specific app features. While informative, their research did not explore the emotional depth associated with user feedback on software issues. Masrury et al. [55] focused on detecting sentiment polarity related to specific app features from Android app store reviews. Their sentiment analysis helped identify positive, neutral, or negative sentiments but did not connect these sentiments to specific bugs or issues. Haering et al. [42] conducted a comprehensive sentiment extraction and analysis from app reviews across multiple platforms. To summarise, the approaches from the literature, mainly focused on high-rated and popular applications from various app stores that aim to identify software issues and associated sentiments. In contrast, the proposed approach focused on end-user feedback for low-rating software applications from the ASA store aiming to identify frequently occurring issues and associated end-user emotions. The proposed approach provides valuable insights into end-users grudges with the software applications and provides insight for software vendors and developers to improve app rating, quality and end-user experiences by enhancing the software evolution process. According to our knowledge, linking specific emotions with identified issues offers a novel perspective that currently needs to be explored in existing literature, as demonstrated in the comparison Table 12.

VIII. DISCUSSION

App and market-driven software developers must solve end-user issues, as their disappointment may lead to the loss of previously popular apps [13], [14]. One possible approach for reducing end-user dissatisfaction is to fix bothersome problems as soon as possible, which may lead crowd users to switch to competitor software and register negative reviews. On the other hand, bugs appear due to various factors identified while manually analyzing end-user reviews in the ASA store. For example, a specific software feature or overall software working incorrectly, an unexpected error while performing a task, a quality issue, etc. Some software issues or problems affect only a limited number of end-users with particular hardware or software versions, while others affect many people. Furthermore, not all issues are apparent to developers, especially non-bugs, which are difficult to detect in automated testing and quality assurance [26]. Additionally, the software issues and their associated end-user emotion types identified and captured from the end-user feedback in the ASA store prove pivotal for software developers to improve the quality of low-ranked software applications. Also, we identify different types of end-user emotions, i.e., anger, disgust, fear, and sadness, to understand better the intensity of frequently occurring issues in the ASA store. Using NLP and ML algorithms, this section summarizes and highlights the findings based on the types of end-user emotions and the issues that come up most often.

A. IMPLICATIONS FOR SOFTWARE QUALITY IMPROVEMENT

The comprehensive analysis of user emotions captured through the ML classifiers offers significant implications for improving software quality by understanding better the frequently occurring issues. By systematically identifying and classifying emotions such as anger, disgust, fear, and sadness, developers can obtain a nuanced understanding

TABLE 12. Detailed comparison of methodological approaches.

Study	Data Sources	Objective	Methodology	Key Contributions
Our Study	Amazon app store reviews	Identify software bugs/issues and corresponding user emotions	NLP with POS tagging, ML algorithms (MNB, LR, RF etc.)	Emotional analysis linked to specific bugs/issues, focusing on low-rated applications
Guzman and Maalej [21]	Apple App Store and Google Play Store reviews	Extract fine-grained app features and sentiments	Sentiment analysis, topic modeling	Pioneered fine-grained sentiment analysis on app features
Vu et al. [75]	App reviews across various platforms	Facilitate information extraction	Advanced NLP techniques	Developed methods for extracting specific information from large volumes of reviews
Johann et al. [76]	Reviews from multiple app stores	Trace features and sentiments	Information retrieval	Established traceability links between reviews and app features
Williams et al. [24]	Reviews on major app stores	Analyze relation between user ratings and review content	Content analysis	Analyzed how vocabulary and length of reviews correlate with user ratings
Li et al. [14]	User reviews and social media platforms	Understand user requirements and preferences	Machine learning, sentiment analysis	Mined user opinions to identify emerging user requirements and preferences
Malik et al. [54]	Online app reviews	Mine user opinions on app features	Sentiment analysis	Provided detailed analysis of user opinions concerning specific app features
Masrury and Alamsyah [55]	Android app store reviews	Detect sentiment polarity in reviews	Sentiment analysis	Focused on detecting sentiment polarity related to specific app features
Huebner et al. [42]	App reviews across multiple platforms	Extract and analyze user sentiment	Sentiment analysis	Conducted comprehensive sentiment extraction and analysis from a broad range of app reviews

of user reactions to software features and functionalities. This emotional feedback is vital for assessing the criticality of reported issues. For example, issues that elicit strong emotions like anger or disgust may signify fundamental flaws that could deter users from continued use of the software. Addressing these issues promptly not only rectifies usability concerns but also serves to enhance the overall reliability and user-friendliness of the software.

In addition, the emotional data extracted can guide developers in refining user interface design to better align with user expectations and emotional responses. For instance, knowing that a feature evokes confusion or fear can lead to targeted redesigns that make the software more accessible and less intimidating to new users. This proactive approach to software development ensures that emotional considerations are integral to the design process, potentially leading to higher user satisfaction and loyalty.

B. ENHANCING USER ENGAGEMENT THROUGH EMOTIONAL INSIGHTS

The study's findings can have profound implications for enhancing user engagement by associating their emotions with the issues reported in the ASA store. By leveraging emotional insights, software vendors can tailor their user interaction strategies to be more empathetic and responsive by adding the proposed approach to the software evolution process. Recognizing and addressing the emotions behind user feedback can transform standard customer service into a more engaging and supportive experience, thus deepening user trust and commitment. For example, responding to feedback denoted by 'sadness' with genuine understanding and prompt corrective actions mitigates the user's immediate frustrations. Moreover, integrating emotional analysis into software development strategies can help developers improve their products. Software developers who acknowledge common user frustrations and highlight specific improvements can effectively demonstrate the company's commitment to user satisfaction, enhancing the overall software applications.

C. STRATEGIC DECISION MAKING IN SOFTWARE DEVELOPMENT

Beyond immediate problem-solving, the classification of user emotions plays a critical role in strategic decision-making within software development projects. It provides a data-driven basis for prioritizing development tasks, allocating resources, and scheduling releases. Emotion-rich feedback helps management teams identify the most frequent and impactful issues that significantly affect user satisfaction and retention. For instance, features that consistently generate negative emotions might be deprioritized or redeveloped, whereas features that elicit positive responses can be leveraged to boost user engagement and attract new users. This strategic use of emotional data ensures that development efforts are closely aligned with actual user needs and emotions, optimizing the investment return in development resources. Furthermore, the ability to map emotions to specific aspects of software usability and functionality allows for more nuanced risk assessments and quality assurance strategies. It supports a more agile development environment where decisions are continually informed by up-to-date user feedback, enabling companies to remain competitive in fast-evolving markets.

D. THEORETICAL IMPLICATIONS OF THE FINDINGS

This study significantly improves the theoretical understanding of end-user feedback dynamics within social media analytics by connecting insights from human psychology with the expression of specific emotions in digital platforms. By focusing on app store reviews, our research examines how the emotions of anger, sadness, and disgust are linked to user-reported issues, offering a detailed perspective on emotional responses in user interactions. Additionally, our findings reveal different patterns of bugs and issues and their associated emotional expression. This compliance supports psychological theories that posit emotions as powerful influences of communication and user behavior in online environments. By mapping specific emotions to reported problems, our study highlights how emotional analysis can provide deeper insights into user dissatisfaction and engagement.

Moreover, this approach advances the concept of emotional sensitivity in digital communication, suggesting that different negative emotions are associated with unfavorable types of end-user feedback. For instance, anger may be more frequently associated with functionality problems, while sadness might relate to unmet expectations from the software. Understanding these nuances can help developers and companies tailor their responses more effectively, addressing the technical aspects of feedback and emotional undertones. This study also explores how emotional responses influence other users' perceptions and developers' decision-making processes. It opens up new avenues for research into the psychological impact of design and communication strategies in digital platforms.

E. ENHANCING LOW-RANKED SOFTWARE APPLICATIONS THROUGH ISSUES AND BUGS IDENTIFICATION

Based on this research study, software vendors have access to a lot of crowd-user feedback about frequently reported issues and their corresponding end-user opinions on social media platforms. Software vendors can utilize such important end-user feedback to improve the software's quality and achieve end-user satisfaction with low-ranked software applications by making essential requirements decisions. For this purpose, we manually analyzed and investigated the crowd-user feedback in the ASA software store using manual content analysis methodology and discovered that end-user reviews are quite significant for requirements and software engineers in recovering requirements-related information, such as issues and bugs and their corresponding user opinions that can be used as an alternative source to improve the quality of low-ranked software applications. However, it poses certain challenges for software vendors to detect, identify, and analyze requirement-related information (frequently reported issues or bugs) in the increasing number of users' feedback on social media platforms. For this, we develop co-occurrence algorithms that identify the frequently occurring issues on social media platforms by associating the captured nouns with adjectives, adjectives with verbs, and combining adjectives, nouns, and verbs using the NLP toolkit. Also, with a coding guideline and a content analysis approach, we identify the end-user emotions and the associated issues of anger, disgust, fear, and sadness that help understand end-user opinions, behaviours, and perceptions, improving the software quality and user satisfaction. Furthermore, the software decision-making process is improved by introducing different ML algorithms that automatically identify end-user opinions with associated captured issues and report them to the requirements and software teams to incorporate in the software application and improve the quality of low-ranked software applications.

F. AUTOMATED IDENTIFICATION OF FREQUENTLY REPORTED ISSUES IN THE ASA STORE

Thoroughly examining user feedback on the ASA store reveals that crowd-users often voice concerns and problems or identify bugs that negatively impact their experience and satisfaction with software applications. The flaws include insufficient or missing features, excessive ads, usability, and performance challenges, all contributing significantly to the software's diminished quality. Identifying common concerns, complaints, or faults from the numerous feedback submissions by customers on the ASA store is a challenging and time-consuming operation. We developed advanced co-occurrence algorithms to tackle this difficulty and simplify the process. The algorithms utilize NLP to identify prevalent topics on social media platforms by linking nouns with adjectives, adjectives with verbs, and combining adjectives, nouns, and verbs. This novel method decreases the amount of human work needed to analyze user feedback and improves the precision and effectiveness in identifying common concerns. Valuable critical insights are essential for requirements and software engineers seeking to enhance the quality of software programs and improve user pleasure. By integrating these insights into the decision-making processes about software requirements and development, software manufacturers can make well-informed decisions that directly cater to the demands and concerns of their consumers. This approach helps identify areas for improvement in advance, enabling a more user-focused approach to software development in today's fast-changing digital environment [33], [47].

G. EFFICACY OF ML CLASSIFIERS IN ANALYZING END-USER SENTIMENTS

Earlier, this research study highlighted that end-users submit a large number of reviews in the ASA store against the software applications daily. At the same time, this feedback is pivotal for requirements engineers to improve the quality of low-ranked software applications by identifying requirements-related information, such as issues, bugs, and complaints, and incorporating it into the ongoing software development process. However, manually analyzing and processing such a large number of requirements-related information in the social media platform becomes challenging and time-consuming [34], [37]. We propose a two-step automated process. First, we identify frequently reported issues and bugs using the NLP toolkit by developing co-occurrence algorithms. Secondly, to further understand the nature and intensity of the issues identified, we employed different machine learning algorithms to recover the end-user emotions associated with each identified issue on the social media platform. The end-user emotions identified in the ASA store are anger, disgust, fear, and sadness. Furthermore, to determine the performance of machine learning algorithms, we manually annotated 7,989 end-user comments collected from the ASA store against various software applications from different categories. We selected nine machine learning algorithms to identify their performance in detecting and classifying end-user emotion types. The classifiers selected are MNB, Support Vector Machine, LR, KNN, MLP, Gradient Boosting, Voting Classifier, RF, and Ensemble Methods. For this study, all the machine learning algorithms performed well in identifying end-user emotion types. MLP, RF, and Voting classifiers yield high accuracy, precision, recall, and F-measure when identifying and classifying different end-user emotion types by outperforming other machine learning algorithms. The MLP classifier yields high F-measure values for identifying and classifying end-user emotion types of 98%, 99%, 98%, and 98%, respectively, for anger, disgust, fear, and sadness. Such information helps requirements and software engineers understand end-user opinions and the identified issues and bugs to improve performance and user satisfaction with low-ranked software applications.

H. LEVERAGING USER FEEDBACK FOR ENHANCING THE SOFTWARE DEVELOPMENT LIFECYCLE

Integrating end-user feedback into the software development lifecycle signifies a shift towards a more user-centric approach in software engineering [59], [77]. This integration is essential for uncovering real-world usability and functionality concerns that users encounter, which may not be apparent in the early stages of software design and testing. As emphasized in this study, systematic gathering and examination of user input offer useful insights into user expectations, experiences, and issues faced when using software applications. The feedback loop allows developers to improve and prioritize feature development and issue fixes more efficiently, ensuring that software progresses according to user needs and preferences. This technique is crucial because of the growing complexity of user interactions with software programs on many platforms and devices, requiring a more detailed understanding of user experiences. Furthermore, including user feedback in the development process promotes the use of agile approaches, which emphasize iterative development with quick cycles of feedback and enhancement. This agile approach enables a dynamic and responsive software development process by incorporating user feedback to influence development objectives and decision-making. The difficulty is in effectively handling and examining the large amounts of feedback to get useful insights. Implementing this method successfully necessitates strong processes and tools for systematically collecting, categorizing, and analyzing feedback. Utilizing advanced NLP and ML technologies in our study has been crucial in automating and optimizing the process through the creation of complex analytical frameworks. Software teams can improve software programs by converting unprocessed customer feedback into organized, practical data to better meet user requirements and enhance overall quality and user happiness.

I. THREATS TO VALIDITY

The experiment and evaluation for the proposed approach were planned by the same authors who wrote the research article, annotated it and labelled the crowd-user comments in the data set. However, the authors annotated end-user feedback collected from the ASA store in a more professional, organized, and iterative manner. There is still a possibility that the annotation experts made a second guess without realizing it. Furthermore, it is difficult, if not impossible, to keep the researcher's bias out of these studies. End-user comments were intricately analyzed and evaluated to determine which characteristics reveal the most about why end-users think the way they do and how the performance of existing software features and applications can be improved. Although the proposed approach obtained valuable results utilizing different ML approaches, we did not test all possible combinations of textual features. Also, different parameters for the MLP, RF, and Voting-TFIDF algorithms, all of which use statistical feature selection techniques, could be optimized to improve the results. Using the proposed method, we showed that using user feedback on ASA to improve the performance of low-ranked software applications is a great way to do so. Furthermore, we only investigated 45 apps from 10 different software categories. It is a small proportion of ASA's total of 236,550 apps being actively rated by users [58]. However, our findings apply to the vast majority of ASA store apps. By studying end-user comments, we looked at low-ranked ASA Software App Store apps and considered ways to improve their quality. To broaden the scope of the proposed strategy, we will test it with high-, mid-, and low-ranked software. Also, by collecting many end-user comments and utilizing more categories, we can further improve the performance of automated experiments by using baseline deep-learning classifiers to identify the sentiments with the identified associated issues. Currently, we employ only traditional ML learning algorithms.

To address the potential for survivorship bias and manual label identification concerns, we acknowledge the limitations inherent in the proposed methodology and data set. The survivorship bias, arising from the focus on apps that have not been removed from the store despite low ratings, may limit the generalizability of the proposed findings. Furthermore, the manual process of label identification, while rigorous, may not fully capture the complexity and variability of human emotions expressed in user reviews. Additionally, we aim to further develop NLP-based rules to not only identify features and non-functional requirements but also to classify associated emotions, including positive sentiments such as happiness and joy. This extension of our approach will enable a more nuanced analysis of user feedback, capturing a broader spectrum of user emotions and their implications for software development and improvement. Moreover, the co-occurrence algorithms for identifying issues further need improvements to filter the frequently occurring issues from a large number of sparse data that do not present issues only. For this purpose, we need to introduce more POS-based patterns to better present issues-related information. In summary, while our current study provides valuable insights into the use of NLP and ML for analyzing user feedback in the ASA store, future work will seek to address the identified methodological limitations, expand the scope of the analysis to include additional app sources and emotional dimensions and refine the analytical techniques to enhance the generalizability and applicability of our findings. Furthermore, to thoroughly address and minimize threats to validity, we will implement ongoing revisions and enhancements based on real-time feedback from industry professionals. This continuous improvement process, guided by their expertise, will refine our methodologies and ensure their adaptability to various scenarios. By engaging directly with developers who use our approaches in their daily processes, we aim to uncover any latent biases or oversights and correct them promptly. This collaboration not only aids in validating our current findings but also helps identify potential differences that could impact the generalizability and reliability of our research outcomes.

IX. CONCLUSION AND FUTURE WORK

In this study, we proposed an automated approach to recover frequently occurring issues in the ASA store using NLP and ML algorithms. We focused on low-ranked software applications to improve their performances and end-user satisfaction by providing in-time remedies for frequently occurring bugs. For this purpose, we selected 45 low-rated software applications with comparatively low ratings from 10 software categories in the ASA App Store. Next, we investigated how crowd-users reported their grudges, sentiments, and opinions against the software applications in the ASA store using the grounded theory and content analysis approaches. Next, we identify the most frequent issues reported in the end-user comments by developing different co-occurrence algorithms using an NLP toolkit. Finally, to better understand the intensity of the captured issues, we employed different ML algorithms, such as MNP, LR, RF, MLP, KNN, AdaBoost, and Voting Classifier, to identify associated end-user emotions of the types of anger, disgust, fear, and sadness by utilizing the annotated end-user data set. For this, we first preprocess the input data to remove irrelevant data and then employ two standard resampling approaches (under-sampling and over-sampling) to balance the data sets, extract and recover different textual features (Bag of Words and TFIDF) from the end-user comments, and finally train and validate the various ML algorithms using a k-fold cross-validation approach. To enhance the robustness of our analysis and address the methodological flaws identified, we have incorporated a more detailed examination of the algorithmic performance, introducing additional metrics such as the area under the curve (AUC) for each ROC curve to understand the misclassifications better. In summary, each ML algorithm performs better and accurately identifies end-user emotions associated with the recovered issues. Notably, the MLP, RF, and voting classifiers perform better and yield higher precision, recall, and F-measure values for identifying end-user sentiments about the captured issues, such as sadness, fear, disgust, and anger. We also provide a theoretical framework that integrates our empirical findings with existing literature on sentiment analysis in social media analytics, highlighting the implications of our findings for software engineering practices. This theoretical integration helps position our work within the broader context of social media analytics, illustrating how automated tools can significantly enhance the understanding of end-user feedback and drive software development. From the classification experiment, we concluded that either MLP, RF or a voting classifier could be shortlisted as the best ML algorithm to identify crowd-user emotions with the recovered issues to understand better their implications on the quality of the software applications under discussion and suggest in time remedies to the development teams. Furthermore, our analysis delves deeper into the practical applications of our findings, discussing how software developers can utilize these insights in real-time to prioritize and address user-reported issues more effectively, thereby enhancing user satisfaction and engagement. Additionally, we achieved 96%, 98%, 97%, and 97% average F-measure values for anger, disgust, fear, and sadness when identifying end-user sentiments or opinions with the captured issues from the ASA store.

In the future, we are interested in collecting more crowd-user comments for software applications in the ASA store covering multiple app ratings, i.e., low, high, and medium, to explore the scalability and applicability of the proposed approach. Further, to improve the end-user satisfaction and quality of low-ranked software applications in the ASA store, a toolchain must be developed to integrate the remedies for the identified issues into the ongoing software development process. Another future direction is to restructure and redesign the current end-user feedback interfaces of the ASA store and other social media platforms to help software engineers efficiently and quickly identify requirements-related information by employing an ML algorithm that processes each end-user comment and determines whether it showcases any valuable information for requirements and software engineers of the type (new feature, issue, non-functional requirement, etc. In the future, we aim to detect and recover the key stakeholders [78] and prioritize [79] those who frequently contribute requirements-related information on social media platforms to timely report the valuable information to the software development team and improve end-user satisfaction and the quality of the software applications under discussion. Also, we are interested in exploring the existence of deceptive reviews [80] in the ASA store to validate the large volume of issues or bugs reported on social media platforms.

Another future direction is to redesign and restructure the crowd-user comments and discussion in the ASA store to quickly and efficiently extract prominent issues or bugs by introducing argumentation theory [19], [47]. Although we obtained encouraging results with the baseline machine learning algorithms, we still aim to employ cutting-edge deep learning and transfer learning classifiers, i.e., LSTM, RNN, BERT, etc., to improve the performance of the proposed approach in identifying requirements-related information and improve the quality of low-ranked software applications. Similarly, in the future, a bug prioritization [81] approach can be proposed to prioritize [82] the identified bugs in an agile software development setting.

AUTHORSHIP CONTRIBUTIONS

Nek Dil Khan and Javed Ali Khan developed the method, detailed investigation, and manuscript writing; Tahir Ullah curated the research data set; Javed Ali Khan, Jianqiang Li, Ayed Alwadain, and Qing Zhao revised the methodology, supervised and revised the manuscript writing (revised draft). Affan Yasin revised the experiments and edited the manuscript where necessary (Final draft). All authors have read and agreed to the published version of the manuscript.

DATA AVAILABILITY AND REPLICATION

Access to the data sets and replication information is available through this link.[4]

CONFLICT OF INTEREST

The authors would like to announce that there is no conflict of interest.

REFERENCES

[1] *Apple Store Downloads 2016 | Statista—Statista.com*. Accessed: Mar. 6, 2024. [Online]. Available: https://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/

[2] M. Pandey, R. Litoriya, and P. Pandey, "An ISM approach for modeling the issues and factors of mobile app development," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 7, pp. 937–953, Jul. 2018.

[3] P. Suresh and K. Gurumoorthy, "Mining of customer review feedback using sentiment analysis for smart phone product," in *Proc. Int. Conf. Comput., Commun., Electr. Biomed. Syst.* Cham, Switzerland: Springer, 2022, pp. 247–259.

[4] 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*, vol. 33, no. 3, p. e2316, Mar. 2021.

[5] Y. Liu, L. Liu, H. Liu, and X. Wang, "Analyzing reviews guided by app descriptions for the software development and evolution," *J. Softw., Evol. Process*, vol. 30, no. 12, p. e2112, Dec. 2018.

[6] J. Ali Khan, L. Liu, Y. Jia, and L. Wen, "Linguistic analysis of crowd requirements: An experimental study," in *Proc. IEEE 7th Int. Workshop Empirical Requirements Eng.*, Aug. 2018, pp. 24–31.

[7] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Softw. Eng.*, vol. 27, no. 2, pp. 1–63, Mar. 2022.

[8] J. A. Khan, L. Liu, L. Wen, and R. Ali, "Crowd intelligence in requirements engineering: Current status and future directions," in *Proc. Int. Working Conf. Requirement Eng., Found. Softw. Quality*, 2019, pp. 245–261.

[9] S. McIlroy, N. Ali, and A. E. Hassan, "Fresh apps: An empirical study of frequently-updated mobile apps in the Google play store," *Empirical Softw. Eng.*, vol. 21, no. 3, pp. 1346–1370, Jun. 2016.

[10] J. Song, Z. Wang, Z. Tu, and X. Xu, "Method for predicting mobile service evolution from user reviews and update logs," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 30, no. 10, pp. 1551–1586, Oct. 2020.

[11] V. Garousi, D. Cutting, and M. Felderer, "Mining user reviews of COVID contact-tracing apps: An exploratory analysis of nine European apps," *J. Syst. Softw.*, vol. 184, Feb. 2022, Art. no. 111136.

[12] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Investigating the relationship between price, rating, and popularity in the blackberry world app store," *Inf. Softw. Technol.*, vol. 87, pp. 119–139, Jul. 2017.

[13] G. Williams and A. Mahmoud, "Modeling user concerns in the app store: A case study on the rise and fall of Yik Yak," in *Proc. IEEE 26th Int. Requirements Eng. Conf.*, Aug. 2018, pp. 64–75.

[14] H. Li, L. Zhang, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," in *Proc. IEEE Int. Conf. Prog. Informat. Comput.*, vol. 2, Dec. 2010, pp. 1093–1097.

[15] J. A. Khan, A. Yasin, R. Fatima, D. Vasan, A. A. Khan, and A. W. Khan, "Valuating requirements arguments in the online user's forum for requirements decision-making: The CrowdRE-VArg framework," *Softw., Pract. Exper.*, vol. 52, no. 12, pp. 2537–2573, Dec. 2022.

[16] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf.*, Aug. 2015, pp. 116–125.

[17] J. A. Khan, "Mining requirements arguments from user forums," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 440–445.

[18] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *Proc. IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 220–226.

[19] J. A. Khan, A. Yasin, M. Assam, W. Khan, S. Y. Shah, and R. A. Khan, "Requirements decision-making as a process of argumentation: A Google maps case study with goal model," *Int. J. Innov. Sci. Technol.*, vol. 3, no. 5, pp. 15–33, Dec. 2021.

[20] A. Ali, Y. Xia, Q. Navid, Z. A. Khan, J. A. Khan, E. A. Aldakheel, and D. Khafaga, "Mobile-UI-repair: A deep learning based UI smell detection technique for mobile user interface," *PeerJ Comput. Sci.*, vol. 10, p. e2028, May 2024.

[21] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf.*, Aug. 2014, pp. 153–162.

[22] A. A. Khan, J. A. Khan, M. A. Akbar, P. Zhou, and M. Fahmideh, "Insights into software development approaches: Mining Q&A repositories," *Empirical Softw. Eng.*, vol. 29, no. 1, p. 8, 2024.

[23] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. D. Penta, "Release planning of mobile apps based on user reviews," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 14–24.

[24] G. Williams and A. Mahmoud, "Mining Twitter feeds for software user requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 1–10.

[25] E. Noei and K. Lyons, "A study of gender in user reviews on the Google play store," *Empirical Softw. Eng.*, vol. 27, no. 2, pp. 1–28, Mar. 2022.

[26] D. Martens and W. Maalej, "Extracting and analyzing context information in user-support conversations on Twitter," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 131–141.

[27] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Trans. Softw. Eng.*, vol. 36, no. 5, pp. 618–643, Sep. 2010.

[28] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," in *Proc. 9th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, Jun. 2012, pp. 108–111.

[29] W. Maalej, Z. Kurtanovic, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016.

[30] N. D. Khan, J. A. Khan, J. Li, T. Ullah, and Q. Zhao, "Mining software insights: Uncovering the frequently occurring issues in low-rating software applications," *PeerJ Comput. Sci.*, vol. 10, p. e2115, Sep. 2023.

[31] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do Twitter users say about software?" in *Proc. IEEE 24th Int. Requirements Eng. Conf.*, Sep. 2016, pp. 96–105.

[32] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychol. Bull.*, vol. 70, no. 4, pp. 213–220, 1968.

[4]https://github.com/nekdil566/Issues_Extraction-_using-NLP_ML

[33] J. Ali Khan, L. Liu, and L. Wen, "Requirements knowledge acquisition from online user forums," *IET Softw.*, vol. 14, no. 3, pp. 242–253, Jun. 2020.

[34] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Eng.*, vol. 23, no. 3, pp. 357–379, Sep. 2018.

[35] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Proc. 21st IEEE Int. Requirements Eng. Conf.*, Jul. 2013, pp. 125–134.

[36] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Softw.*, vol. 33, no. 1, pp. 48–54, Jan. 2016.

[37] J. A. Khan, Y. Xie, L. Liu, and L. Wen, "Analysis of requirements-related arguments in user forums," in *Proc. IEEE 27th Int. Requirements Eng. Conf.*, Sep. 2019, pp. 63–74.

[38] M. Nayebi, L. Dicke, R. Ittyipe, C. Carlson, and G. Ruhe, "ESSMArT way to manage user requests," 2018, *arXiv:1808.03796*.

[39] M. Nayebi and G. Ruhe, "Asymmetric release planning: Compromising satisfaction against dissatisfaction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 9, pp. 839–857, Sep. 2019.

[40] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE Softw.*, vol. 32, no. 3, pp. 70–77, May 2015.

[41] Q. M. U. Haq, F. Arif, K. Aurangzeb, N. U. Ain, J. A. Khan, S. Rubab, and M. S. Anwar, "Identification of software bugs by analyzing natural language-based requirements using optimized deep learning features," *Comput., Mater. Continua*, vol. 78, no. 3, pp. 4379–4397, 2024.

[42] M. Haering, C. Stanik, and W. Maalej, "Automatically matching bug reports with related app reviews," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng.*, May 2021, pp. 970–981.

[43] M. E. Mezouar, F. Zhang, and Y. Zou, "Are tweets useful in the bug fixing process? An empirical study on Firefox and chrome," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1704–1742, Jun. 2018.

[44] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *Proc. IEEE 25th Int. Requirements Eng. Conf.*, Sep. 2017, pp. 11–20.

[45] S. Hassan, C. Tantithamthavorn, C.-P. Bezemer, and A. E. Hassan, "Studying the dialogue between users and developers of free apps in the Google play store," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1275–1312, Jun. 2018.

[46] R. Alkadhi, M. Nonnenmacher, E. Guzman, and B. Bruegge, "How do developers discuss rationale?" in *Proc. IEEE 25th Int. Conf. Softw. Anal., Evol. Reengineering*, Mar. 2018, pp. 357–369.

[47] J. Ali Khan, L. Liu, L. Wen, and R. Ali, "Conceptualising, extracting and analysing requirements arguments in users' forums: The CrowdRE-Arg framework," *J. Softw., Evol. Process*, vol. 32, no. 12, p. e2309, Dec. 2020.

[48] D. Martens and T. Johann, "On the emotion of users in app reviews," in *Proc. IEEE/ACM 2nd Int. Workshop Emotion Awareness Softw. Eng.*, May 2017, pp. 8–14.

[49] D. Martens and W. Maalej, "Release early, release often, and watch your users' emotions: Lessons from emotional patterns," *IEEE Softw.*, vol. 36, no. 5, pp. 32–37, Sep. 2019.

[50] K. Srisopha, D. Swami, D. Link, and B. Boehm, "How features in iOS app store reviews can predict developer responses," in *Proc. Eval. Assessment Softw. Eng.*, Apr. 2020, pp. 336–341.

[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 *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2015, pp. 281–290.

[52] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "ARdoc: App reviews development oriented classifier," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2016, pp. 1023–1027.

[53] X. Gu and S. Kim, "'What parts of your apps are loved by users?' (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 760–770.

[54] H. Malik, E. M. Shakshuki, and W.-S. Yoo, "Comparing mobile apps by identifying 'Hot' features," *Future Gener. Comput. Syst.*, vol. 107, pp. 659–669, Jun. 2020.

[55] R. A. Masrury, Fannisa, and A. Alamsyah, "Analyzing tourism mobile applications perceived quality using sentiment analysis and topic modeling," in *Proc. 7th Int. Conf. Inf. Commun. Technol. (ICoICT)*, Jul. 2019, pp. 1–6.

[56] T.-P. Liang, X. Li, C.-T. Yang, and M. Wang, "What in consumer reviews affects the sales of mobile apps: A multifacet sentiment analysis approach," *Int. J. Electron. Commerce*, vol. 20, no. 2, pp. 236–260, Dec. 2015.

[57] M. Nicolai, L. Pascarella, F. Palomba, and A. Bacchelli, "Healthcare Android apps: A tale of the customers? Perspective," in *Proc. 3rd ACM SIGSOFT Int. Workshop App Market Analytics*, 2019, pp. 33–39.

[58] *Amazon Appstore Statistics and Trends 2023—42*. Accessed: Oct. 3, 2023. [Online]. Available: https://42matters.com/amazon-appstore-statistics-and-trends

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

[60] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009.

[61] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[62] A. Strauss and J. Corbin, *Basics of Qualitative Research Techniques*. Thousand Oaks, CA, USA: Sage, 1998.

[63] W. Maalej and M. P. Robillard, "Patterns of knowledge in API reference documentation," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1264–1282, Sep. 2013.

[64] K. A. Neuendorf, *The Content Analysis Guidebook*. Newbury Park, CA, USA: Sage, 2017.

[65] A. Kumar, A. Ekbal, D. Kawahra, and S. Kurohashi, "Emotion helps sentiment: A multi-task model for sentiment and emotion analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[66] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *Proc. IJCAI*, vol. 14, no. 2, pp. 1137–1145, 1995.

[67] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: Mining informative reviews for developers from mobile app marketplace," in *Proc. 36th Int. Conf. Softw. Eng.*, May 2014, pp. 767–778.

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

[69] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[70] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *Int. Trans. Comput. Sci. Eng.*, vol. 30, no. 1, pp. 25–36, 2006.

[71] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[72] J. Keilwagen, I. Grosse, and J. Grau, "Area under precision-recall curves for weighted and unweighted data," *PLoS One*, vol. 9, no. 3, Mar. 2014, Art. no. e92209.

[73] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, Apr. 1982.

[74] J. Tizard, H. Wang, L. Yohannes, and K. Blincoe, "Can a conversation paint a picture? Mining requirements in software forums," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 17–27.

[75] 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 *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 749–759.

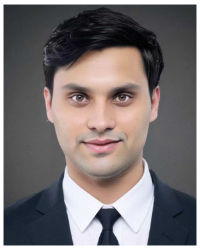
[76] T. Johann, C. Stanik, A. M. Alizadeh B., and W. Maalej, "SAFE: A simple approach for feature extraction from app descriptions and app reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 21–30.

[77] J. A. Khan, T. Ullah, A. A. Khan, A. Yasin, M. A. Akbar, and K. Aurangzeb, "Can end-user feedback in social media be trusted for software evolution: Exploring and analyzing fake reviews," *Concurrency Comput., Pract. Exper.*, vol. 36, no. 10, p. e7990, May 2024.

[78] F. M. Khan, J. A. Khan, M. Assam, A. S. Almasoud, A. Abdelmaboud, and M. A. M. Hamza, "A comparative systematic analysis of Stakeholder's identification methods in requirements elicitation," *IEEE Access*, vol. 10, pp. 30982–31011, 2022.

[79] H. Arshad, S. Shaheen, J. A. Khan, M. S. Anwar, K. Aurangzeb, and M. Alhussein, "A novel hybrid requirement's prioritization approach based on critical software project factors," *Cognition, Technol. Work*, vol. 5, no. 2, pp. 1–20, 2023.

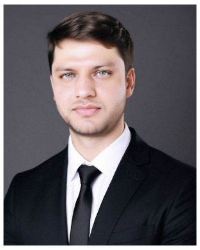
[80] M. I. Marwat, J. A. Khan, D. M. D. Alshehri, M. A. Ali, H. Ali, and M. Assam, "Sentiment analysis of product reviews to identify deceptive rating information in social media: A sentideceptive approach," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 3, pp. 830–860, 2022.
[81] J. Ali Khan, I. Ur Rehman, Y. Hayat Khan, I. Javed Khan, and S. Rashid, "Comparison of requirement prioritization techniques to find best prioritization technique," *Int. J. Modern Educ. Comput. Sci.*, vol. 7, no. 11, pp. 53–59, Nov. 2015.
[82] J. Khan, I. Rehman, L. Ali, S. Khan, and I. Khan, "Requirements prioritization using analytic network process (ANP)," *Int. J. Sci. Eng. Res.*, vol. 7, no. 11, pp. 1–7, 2016.



NEK DIL KHAN received the B.Sc. degree in software engineering from the University of Science and Technology Bannu, Khyber Pakhtunkhwa, Pakistan. He continued to pursue his passion and the master's degree in software engineering from Northeastern University, Shenyang, Liaoning, China. He is pursuing the doctorate (Ph.D.) degree in software engineering degree with the Beijing University of Technology, Beijing, China. He explores a broad range of subjects in the fields of software engineering and computer science. These include natural language processing, software requirements engineering, debugging, information extraction, big data, data mining, human–computer interaction, game theory, sarcasm detection, CrowdRE, argumentation and argument mining, mining software repositories, human values in software, quantum software engineering, feedback analysis, empirical software engineering, sentiment and opinion mining, requirements prioritization, mining fake reviews, and health analytics.

JAVED ALI KHAN received the Ph.D. degree in software engineering from Tsinghua University (QS ranked 12th), Beijing, China. He is currently working as a Senior Lecturer with the Foundation of Software Engineering (FSE) Group, Department of Computer Science, University of Hertfordshire, U.K. Previously, he worked as the Assistant Professor cum Chairperson with the Department of Software Engineering, University of Science and Technology Bannu, Pakistan. He regularly publishes papers in reputable software engineering journals and conferences. His areas of research interests include software engineering, requirements engineering, CrowdRE, argumentation and argument mining, mining software repositories, human values in Software, quantum software engineering, feedback analysis, empirical software engineering, sentiment and opinion mining, requirements prioritization, mining fake reviews, sarcasm detection, and health analytics.

JIANQIANG LI (Senior Member, IEEE) received the B.S. degree in mechatronics from Beijing Institute of Technology, Beijing, China, in 1996, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, in 2001 and 2004, respectively. He was a Researcher with the Digital Enterprise Research Institute, National University of Ireland, Galway, from 2004 to 2005. From 2005 to 2013, he was with NEC Laboratories China, as a Researcher. He was with the Department of Computer Science, Stanford University, as a Visiting Scholar, from 2009 to 2010. He joined Beijing University of Technology, Beijing, in 2013, as a Beijing Distinguished Professor. He has more than 200 publications, including one book, more than 100 journal articles, and 58 international patent applications (27 of them have been granted in China, USA, or Japan). His research interests include Petri nets, enterprise information systems, business processes, data mining, information retrieval, the semantic web, privacy protection, and big data. He was a PC member of multiple international conferences and organized the IEEE workshop on medical computing. He served as a guest editor to organize a special issue on information technology for enhanced healthcare service in computers and in industry.



TAHIR ULLAH received the bachelor's degree in software engineering from the University of Science and Technology Bannu, Khyber Pakhtunkhwa, Pakistan. His research interests include natural language processing, software requirements engineering, software bug identification (debugging), information extraction, big data, data mining, and human–computer interaction.

AYED ALWADAIN received the Ph.D. degree from Queensland University of Technology, Australia, in 2014. He is currently an Associate Professor with the Computer Science Department, Community College, King Saud University, Saudi Arabia. He has published his work at many international conferences and journals. In his research, he focuses on enterprise architecture, service management and engineering, business process management, requirement engineering, machine learning, and big data.

AFFAN YASIN received the Bachelor of Science in Computer Science (B.S.C.S.) degree from the National University of Computer and Emerging Sciences (NUCES-FAST), Lahore, Pakistan, the Master of Science in Software Engineering (M.S.S.E.) degree from Blekinge Tekniska Högskola (BTH), Karlskrona, Sweden, and the Ph.D. degree in software engineering from Tsinghua University, Beijing, China. Following the completion of the Ph.D. degree, he held the position of a Postdoctoral Fellow with Tsinghua University. Presently, he is actively engaged in research with Northwestern Polytechnical University, Xi'an, China. His primary research interests include empirical research and development within the realm of software engineering, with a focus on areas, such as game-based learning, social engineering, serious games, and requirements engineering.

QING ZHAO received the M.S. degree in software engineering and the Ph.D. degree in computer science and technology from Beijing University of Technology, China, in 2016 and 2021, respectively. She is currently a Lecturer with the Faculty of Information Technology, Beijing University of Technology. Her current research interests include natural language processing, semantic analysis, and medical data analysis.