# Driving the Technology Value Stream by Analyzing App Reviews

Souvick Das, Novarun Deb, *Member, IEEE*, Nabendu Chaki, *Senior Member, IEEE*, and Agostino Cortesi

***Abstract*—An emerging feature of mobile application software is the need to quickly produce new versions to solve problems that emerged in previous versions. This helps adapt to changing user needs and preferences. In a continuous software development process, the user reviews collected by the apps themselves can play a crucial role to detect which components need to be reworked. This paper proposes a novel framework that enables software companies to drive their technology value stream based on the feedback (or reviews) provided by the end-users of an application. The proposed end-to-end framework exploits different Natural Language Processing (NLP) tasks to best understand the needs and goals of the end users. We also provide a thorough and in-depth analysis of the framework, the performance of each of the modules, and the overall contribution in driving the technology value stream. An analysis of reviews with sixteen popular Android Play Store applications from various genres over a long period of time provides encouraging evidence of the effectiveness of the proposed approach.**

***Index Terms*—Continuous software development, technology value stream, NLP, app reviews.**

## I. Introduction

Ward Cunningham describes "*technical debt*" [1] as decisions taken within the technology value stream that result in increasingly difficult problems with reduced available options over time. This, in turn, incurs more interests on the organization in managing the technology value stream efficiently. The broad motivation behind this research work is to facilitate an organization in taking the right decisions so that maximum customer satisfaction can be achieved. We intend to incorporate the insights contained within customer feedback (available as app reviews) while making such informed decisions.

App stores have evolved into digital distribution platforms for mobile applications (apps). They allow users to download and rate apps while providing the app owners with useful analytics such as the number of downloads, ratings, reviews and revenue generated by the app. Such analytics provide app developers with a rich source of information for driving the technology value stream [2]. The volume of customer feedback (or reviews), however, is just too enormous to be examined manually [3], [4]. An empirical study by Pagano et al. [5] found that mobile apps received 23 reviews per day (approximately) and that popular apps, such as Facebook, received 4,275 reviews per day (on average). This vast volume of textual data needs to be analysed in order to identify customers' opinions and demands.

Furthermore, users generally submit feedback in the form of unstructured text that is difficult to process and analyse. As a result, researchers have devised a number of approaches [6], [7], [8], [9] for automatically analyzing the texts contained in user reviews in order to extract the most valuable information they contain. Important insights from the app reviews such as current bugs (e.g., crashes) and undesirable app features can offer app developers with useful evidence towards better management of the technology value stream. Some approaches [10], [11] aim to categorize customers' reviews into different topics based on summarization. App developers must also analyse the impact of such reviews on specific features to ensure the success of their apps.

Prior studies have mostly focused on reducing manual intervention in extracting software aspects or user preferences [12], filtering out non-informative reviews [10], or categorizing reviews into predetermined topics [11]. Despite past explorations, there is the need for a systematic and efficient end-to-end framework that analyses customer reviews and reduces the technical debt within the technology value stream. This, in turn, requires us to address the following specific research challenges.

- **RQ1:** How can the end-user app reviews help the development team to precisely identify the app features which are of immediate concern?
- **RQ2:** How can the user sentiments be integrated into the recommendation system to identify the more relevant reviews that are significant for the technology value stream?
- **RQ3:** How can we integrate the entire framework within the software development pipeline to automate the continuous maintenance process?

Our research methodology for addressing the above mentioned research challenges has been carried out as follows:

- *Literature Study:* A focused study of existing works on the analysis of app reviews was done to understand the existing state-of-the-art. This is presented in detail in Section III.
- *Topic Analysis Techniques:* Topic analysis approaches were explored for effective identification of specific

Manuscript received 1 September 2022; revised 17 April 2023; accepted 20 April 2023. Date of publication 26 April 2023; date of current version 18 July 2023. This work was supported in part by the Next Generation EU under Grants ECS00000043 iNEST and CUP H43C22000540006 and in part by Ca' Foscari SPIN2021 project RESSA_ROB. Recommended for acceptance by R. Hoda. *(Corresponding author: Souvick Das.)*

Souvick Das and Agostino Cortesi are with the DAIS Department, Ca' Foscari University, 30123 Venice, Italy (e-mail: souvik.cmsa019@gmail.com; cortesi@unive.it).

Novarun Deb is with the Indian Institute of Information Technology (IIIT), Vadodara, Gujarat 382028, India (e-mail: novarun.db@gmail.com).

Nabendu Chaki is with the Department of Computer Science and Engineering, University of Calcutta, Kolkata, West Bengal 700073, India (e-mail: nabendu@ieee.org).

Digital Object Identifier 10.1109/TSE.2023.3270708

features addressed in review texts. Some of the methods that were explored include Text Classification techniques and Key Phrase Extraction techniques to get insights from the user reviews and extract information that is relevant for the future releases.

- *Sentiment Analysis:* Such techniques were explored to uncover the customers' intents within the submitted reviews that document specific topics. This work involves highly detailed analysis, utilizing advanced sentiment analysis techniques based on transformer models like BERT [13], and RoBERTa [14] that enable classification into five different sentiment classes. The use of such techniques adds an extra layer of credibility to the analysis of app reviews.
- *Experimental Evaluation:* To ensure an unbiased evaluation of our proposed framework, we evaluated it against sixteen well-known Android applications from various genres, rather than custom-made applications developed by us. Our evaluation process includes an extensive collection of 3400 reviews for each of the applications, all of which are sourced from the Android Play Store. This ensures an unbiased evaluation of our framework. The proposed framework also addresses potential areas of the Apps that require updates for further enhancement and provides evidence in terms of statistics and graphical representations.

The main contributions of this paper are as follows:

- A novel key phrase extraction model based on transformer architectures.
- A novel zero-shot classifier for topic recognition of app reviews.
- A comprehensive end-to-end recommendation framework that combines sentiment analysis, key phrase extraction, and topic recognition to identify the emerging concerns and issues for upcoming releases of the app.
- A tool prototype[1] that allows software development organisations to use our proposed framework.

The organization of the paper is as follows. We briefly recapitulate the NLP approaches used in our framework in Section III. Section IV presents the framework and elaborates all the techniques used in the different components of the framework. The methodology of the validation of the framework has been briefly highlighted in Section V. We elaborate on the implementation of each of the modules of the framework and the entire experimental evaluation in Section VI. Section VII presents a discussion of the end-to-end framework. Section VIII identifies the threats to validity of our proposed framework. Section II documents the related works existing in the literature and compares them with our proposed framework. Section IX concludes the paper and outlines the future research directions.

## II. Related Work

In this section, we highlight current state-of-the-art research works that are most closely aligned with the specific research problem being addressed in this paper. We consciously omit the huge amount of literature related to the NLP techniques used in our framework: the listed research works have been selected with respect to analyzing app reviews and capturing different user aspects.

### A. Information Extraction From App Reviews

App reviews contain a large amount of unstructured text, making it difficult to extract relevant information. Many studies [9], [29], [30] have proposed approaches to facilitate information extraction, which is the process of extracting specific, pre-specified information such as app features, qualities, problem reports, opinions, and user stories from the reviews. Guzman et al. [15] proposed an approach for extracting app features mentioned in user reviews and their association with the sentiments of the reviews. Iacob and Harrison [7] employed linguistic principles to extract feature requests from app reviews and then used Latent Dirichlet Allocation (LDA) [31] to group the feature requests. Harman et al. [32] introduced an approach for app store mining and then analyzed the relationship between technical, business, and users' perspectives by extracting app features from the official app descriptions. In the SAFE technique, developed by Johann et al. [19], the authors propose the extraction attributes from app descriptions and user reviews. A set of pre-defined language patterns are used in this process. The obtained attributes were then examined to check if each feature was mentioned in user reviews. In another work, Jiang et al. [8] also mentioned the feature extraction problem when they created a classifier to extract features from App descriptions and then identified missing features for similar Apps. Developers often need to search for code descriptions in external artifacts, such as bug reports and emails, as source code lacks comments to adequately describe behavior. Finally, [12] discusses how to automatically extract method descriptions from communications in bug-tracking systems and mailing lists.

### B. Classification of App Reviews

Classification is a technique used to separate informative reviews from those that are uninformative, spam, or fake. ARMiner [10] is a prime example of this method. It uses a pre-trained classifier to recognize useful reviews, then organizes these reviews into topics and prioritizes these topics. Other studies as well include the classification of informative and non-informative reviews [33], fake or spam [34]. The classification of app reviews is widely used in research and studies to understand user feedback and improve software products by identifying user intentions. [35] introduces several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and text ratings. The results of the study inspired the design of a review analytics tool that helps app vendors and developers to filter and assign critical reviews to the appropriate stakeholders. In another work [21], authors present a tool ARdoc that uses natural language parsing, text analysis, and sentiment analysis to automatically classify useful feedback in mobile app reviews for software maintenance and evolution tasks. Pagano and Maalej [5] presented the categories of user reviews that are possible and the derivation of the possible topic out of those categories. Khalid et al. [18] mined

[1] https://github.com/svk-cu-nlp/reviews-to-app-maintenance

the Apple iOS App Store for reviews with one or two stars in order to categorize different forms of customer complaints and assess how complaints impact ratings. Sorbo et al. [22] proposed NEON, a tool that uses NL parsing to automate the mining of rules for classifying software artifacts, reducing manual effort. It was found that NEON can efficiently automate the detection of rules useful for classifying user intents in the context of app reviews through a small study involving NL experts. In another work, Mcilroy et al. [17] contribute to automatically assigning multiple labels to each review. Another research [9] describes the development of an automated approach called User Request Referencer (URR) that uses Machine Learning and Information Retrieval techniques to help mobile app developers monitor and analyze user feedback received in the form of reviews. The URR prototype is able to group reviews according to high and low-level categories and recommend source code files that need to be modified to handle issues described in the user reviews.

### C. Clustering of App Reviews

Clustering is the process of organizing reviews, sentences, and snippets into groups, known as clusters, where members within the same group share some similarities. Clustering is commonly used as an exploratory analysis technique to discover topics commonly discussed by users, aggregate reviews containing semantically related information, and group reviews based on shared characteristics such as requested features, reported problems or discussed characteristics of the app. Clustering is widely used in research and studies to understand the user's feedback and opinion and improve the software product. Zhou et al. [23] presents an automated approach called RISING (Review Integration via classification, clustering, and linking) that groups user reviews into fine-grained clusters concerning similar user requests. Then, by combining textual information from both commit messages and source code, it automatically localizes potential change files to accommodate the users' requests. Another paper [27] presents DIVERSE, a feature and sentiment-centric retrieval approach that automatically provides developers with a diverse sample of user reviews that is representative of the different opinions and experiences mentioned in the whole set of reviews. Guzman et al. [36] presents an interactive user feedback visualization that displays app reviews from four different points of view: general, review-based, feature-based, and topic-feature based. Peng et al. [16] propose a semi-automated approach to extract feature requests from app reviews using machine learning approaches. The approach first identifies reviews on feature requests by defining suitable classification features and selecting appropriate classification approaches. Then, it clusters these reviews using topic models and extracts phrases as feature requests, which serve as the basis of feature modeling. In another paper [15], an automated approach has been proposed to help app developers filter, aggregate, and analyze user reviews. It utilizes natural language processing and topic modeling techniques to identify fine-grained app features from user reviews, extract user sentiments about these features, and group them into more meaningful high-level features.

### D. Sentiment Analysis in App Reviews

Sentiment analysis (also known as opinion mining) refers to the task of interpreting user emotions in app reviews. Guzman and Maalej [15] proposed an automated approach to help app developers filter, aggregate, and analyze user reviews. It utilizes natural language processing and topic modeling techniques to identify fine-grained app features from user reviews, extract user sentiments about these features, and group them into more meaningful high-level features. Another article [37] presents five release lessons to assist app vendors in maintaining positive emotions among their users and gaining competitive advantages in a highly competitive app store market. These lessons are based on emotional patterns identified using sentiment analysis tools. Another study [38] conducts an empirical evaluation of the opinion mining approaches and aims to answer two main research questions: What is the effectiveness of feature extraction approaches? and What is the effectiveness of feature-specific sentiment analysis approaches? Another research [39] examines the impact of design overhauls on user behavior and satisfaction. The study uses the example of Snapchat and finds that substantial changes in an app design can trigger a new adoption process and impact the perceived ease of use, leading to a decrease in app store ratings and negative responses among users. Malik et al. [24] proposed a methodology that can be used to understand a user's preference for a certain mobile app and could uncover the reasons why users prefer one app over another, by analyzing a large volume of reviews.

### E. Summarization for App Reviews

This paper [26] introduces SRR-Miner, a novel review summarization approach for mobile apps that automatically summarizes security issues and users' sentiments by extracting security-related review sentences and summarizing them with <misbehavior-aspect-opinion> triples. Sorbo et al. [25] proposed a tool called SURF that helps mobile developers analyze and classify information in app reviews to identify actionable change tasks for improving mobile applications. The tool performs a systematic summarization of user reviews, generating an interactive, structured, and condensed agenda of recommended software changes. A framework called Software User Review Miner (SUR-Miner) [3] aims to summarize users' sentiments and opinions towards corresponding software aspects, by making use of the structure and semantics of software user reviews. SUR-Miner parses aspect-opinion pairs from review sentences based on pre-defined sentence patterns, analyzes sentiments for each review sentence, associates sentiments with aspect-opinion pairs, and summarizes software aspects by clustering aspect-opinion pairs with the same aspects.

Table I offers a comprehensive and detailed analysis of the relevant state-of-the-art research works that are closely aligned with our research objectives. However, based on our current understanding, the research conducted by Sorbo et al. [25] is deemed to be the most closely related to our study. The work presented by Sorbo et al. on SURF (Summarizer of User Reviews Feedback) provides a valuable contribution to the field of app development by addressing the challenge of managing user

TABLE I
COMPARISON OF DIFFERENT APP REVIEW ANALYSIS TECHNIQUES USED IN EXISTING LITERATURE VERSUS OUR PROPOSED FRAMEWORK

| Research Work | Information Extraction | Review Classification | Review Clustering | Sentiment Analysis | Summarization |
|---|---|---|---|---|---|
| Chen et.al. [10] | ✓ | ✓ | ✓ | ✗ | ✗ |
| Pagano and Maalej [5] | ✗ | ✓ | ✓ | ✗ | ✗ |
| Guzman et.al. [15] | ✓ | ✓ | ✓ | ✓ | ✗ |
| Peng et.al. [16] | ✗ | ✓ | ✓ | ✗ | ✗ |
| Mcilroy et.al. [17] | ✗ | ✓ | ✓ | ✗ | ✗ |
| Scalabrino et.al. [6] | ✓ | ✓ | ✓ | ✗ | ✗ |
| Iacob and Harrison [7] | ✓ | ✗ | ✓ | ✗ | ✗ |
| Khalid et.al. [18] | ✓ | ✓ | ✗ | ✗ | ✗ |
| Johann et.al. [19] | ✓ | ✗ | ✗ | ✗ | ✗ |
| Gu et.al. [20] | ✓ | ✓ | ✗ | ✓ | ✓ |
| Panichella et.al. [21] | ✓ | ✓ | ✗ | ✓ | ✓ |
| ARdoc [22] | ✓ | ✓ | ✗ | ✓ | ✗ |
| Sorbo et.al. [23] | ✓ | ✓ | ✗ | ✗ | ✗ |
| Zhou et.al. [24] | ✓ | ✓ | ✓ | ✗ | ✗ |
| Malik et.al. [25] | ✓ | ✗ | ✗ | ✓ | ✗ |
| SURF [26] | ✗ | ✓ | ✗ | ✓ | ✓ |
| SRR-Miner [27] | ✓ | ✗ | ✗ | ✓ | ✓ |
| DIVERSE [28] | ✓ | ✗ | ✓ | ✓ | ✗ |
| Ciurumelea et.al. [29] | ✓ | ✓ | ✗ | ✗ | ✗ |
| **Our Approach** | ✗ | ✓ | ✗ | ✓ | ✓ |

feedback. The following observations of the SURF framework motivate us to invest a significant amount of research and development in these directions.

1) The creation of the training dataset for topic classification required a significant amount of manual intervention and expertise.
2) The intention classification was achieved through a combination of parsing, text analysis, and sentiment analysis. However, a detailed study regarding the effectiveness of the classifier's performance was out of the scope of this paper.
3) The framework presented in the study effectively addresses the challenge of managing user feedback, however, it does not prioritize recommendations for changes necessary for app maintenance or address which features should receive the most attention from the development team. Although this presents a gap in the work, the efforts made in the study are an area that holds potential for further exploration and improvement.
4) The study presents a well-organized hierarchical structure for summarizing user reviews, where the reviews are first clustered based on topics and intentions. While this approach effectively organizes the information, the effectiveness of the summarization technique is not discussed in the paper. Additionally, the framework does not highlight key points mentioned in the reviews nor incorporate any abstractive or extractive summarization techniques on the reviews.

The study on SURF has made a valuable impact on the field and has set a solid foundation for future research endeavors. Its innovative approach towards managing user feedback in app development has been a source of inspiration for our own research. As a result, our comprehensive framework specifically addresses the above-mentioned observations with respect to the SURF framework and provides solutions accordingly, such as prioritizing recommendations for maintenance changes, focusing on features that should receive the most attention from the development team, and an effective summarization technique.

## III. NLP APPROACHES

In this section, we provide preliminary insights on different NLP techniques that have been employed in different modules of our proposed framework. We take the help of a restricted case study to elaborate the concepts. The case study consists of user reviews presented in Table II.

### A. Sentiment Analysis

A fine-grained sentiment analysis approach involving five sentiment classes - *Strongly Positive, Positive, Neutral, Negative* and *Strongly Negative* - needs to be carried out for analysing the reviews and understanding the subtle intentions of the reviewer.

We use transformer based approaches (like BERT [13], RoBERTa [14], and DistilBERT [40]) for sentiment analysis task as they are better at analysing longer sentences and achieve state-of-the-art performance [13], [14]. However, other alternatives that could be explored include classical machine learning based approaches (like Naive Bayes, Logistic Regression, Support Vector Machines) or neural network based approaches (like LSTM, Bi-LSTM).

The sentiment analysis of reviews is shown in column 3 of Table II. Reviews with negative sentiments often talk about bugs, faults, feature issues and so on. On the other hand, reviews with neutral and positive sentiments talk about improvement, feature request and information enquiry.

### B. Topic Classification

Topic Classification plays a significant role by classifying a specific end-user review into a pre-defined set of topics such as - *feature information, bug, fault and shortcoming, feature*

TABLE II
REVIEWS WITH SENTIMENT AND POSSIBLE INFERENCES

| ID | Reviews | Sentiment | Topic Classification | Mapped Feature | Key Phrases |
|---|---|---|---|---|---|
| R1 | Lost virtual background on Pixel 6, worked on Pixel 4. Please fix. | Negative | Bug | Virtual background | virtual background on pixel, lost virtual background |
| R2 | Put more options in settings and virtual backgrounds. Improve zoom Please | Neutral | Improvement suggestion | Virtual background | settings and virtual backgrounds, options in settings, improve zoom |
| R3 | This app is not letting me put virtual background and I need that in mi because in my school I just cannot do it every time on laptops we need the virtual background in mobiles | Neutral | Feature request | Virtual background | virtual background in mobiles, need virtual background |
| R4 | Zoom is an amazing app for people who need to have meetings with people who are not in the same location. Every time I can login onto my account effortlessly and find all of my meetings available and ready at the click of a button. | Strongly Positive | Non-informative | NA | NA |
| R5 | I am unable to login to my zoom ID even when I have logged into my Gmail account and I am using my device to login to my zoom but I am getting some error. I can't even send a request that I have forgot my password. please tell me the solution to regain my account. | Negative | Information enquiry | Change Password | login to my zoom, forgot my password, login zoom id |
| R6 | no microphone!!! Other meeting apps work with the mic except ZOOM! | Strongly Negative | Non-informative | NA | NA |
| R7 | Zoom is a good video chat app for secure meetings, but unfortunately the user interface is awkward. I'm aware that some of that could be due to the app's secure nature, but at least it should remember permissions, settings and user preferences. Despite the flaws, I do recommend the Zoom app. | Positive | Improvement suggestion | Access settings | remember permissions settings, user preferences, awkward interface |
| R8 | What is happening with screen sharing?!!. Screen sharing freezes often | Negative | Bug | Screen sharing | screen sharing freezes, freezes often |
| R9 | Can we share a particular area of the screen? In the advanced option I could not find anything | Neutral | Information enquiry | Screen sharing | share particular area, advanced option |

*request, content request, improvement suggestion,* or *information enquiry*. Topic classification is a supervised machine learning activity that needs training before being able to automatically analyze texts and identify topics. However, there is a notable scarcity of datasets available for review topic classification, which poses a significant challenge when training a machine learning model for this purpose. To address this issue, we have implemented a zero-shot classification approach that allows for the successful execution of this activity even in the absence of a specific training dataset. Detailed documentation of how this classifier was built has been provided in a supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2023.3270708.[2]

In order to build the zero-shot model, we define our template using reviews along with sentiments and one unfilled slot. We choose BART-large-mnli [41] as our pre-trained model to predict the appropriate class for the unfilled slot and derive the potential class accordingly. Alternative zero-shot classification models could be built on top of transformer-based models such as BERT, RoBERTa [14] as well.

In Table II, we list the topics identified for the nine reviews of our case study. More specifically, column 4 of Table II shows the topics identified while considering the sentiments. Topic classification changes for some of the reviews when sentiments are considered. This is because topic classification, without sentiment analysis, fails to understand the language structure. Review $R6$ also best fits into the *non-informative* category but topic classification, without sentiment analysis, recognizes it as *Bug*. Also, review $R8$ is more about expressing some problem or *Bug*; rather than an *Information enquiry*.

### C. Semantic Search

We also need to identify which particular features (or components) should be targeted for addressing the individual app reviews. This helps the development team to focus on those specific app features and drive the technology value stream. The app documentation - such as user guide, FAQs - needs to be available for this purpose.

[2] 10.5281/zenodo.7836451

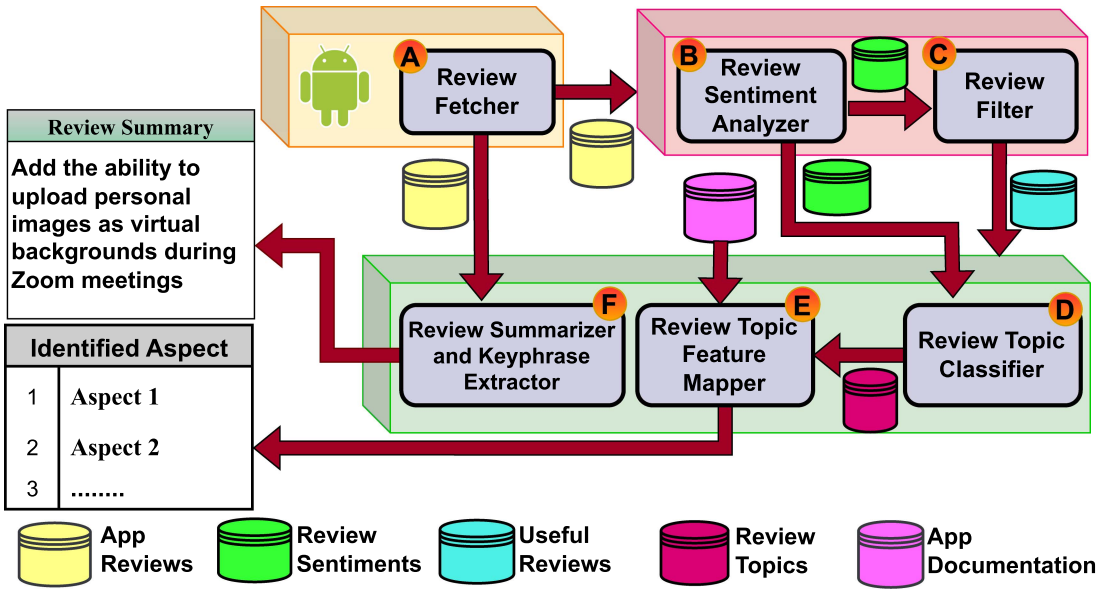



Fig. 1. Proposed framework.

The semantic search mechanism identifies the appropriate app feature(s) from the app documentation which are most closely related to the user's concern. Semantic search is achieved by training RoBERTa model on the MS MARCO [42] passage retrieval dataset.

Column 5 of Table II shows how different features have been identified for each of the nine reviews of our case study. It is worth mentioning here that, in this motivational case study, we assume that such app documentations or FAQs are available beforehand.

### D. Summarization and Key Phrase Extraction

The business value coming out of the technology value stream could be drastically improved if developers could be informed about the particular aspects of a feature that are of concern. In this phase, the mechanisms for text summarization and key phrase extraction are employed to extract key phrases and create summaries of user reviews by analyzing the semantic aspect of the text. Key phrase extraction is used for shorter reviews, while text summarization is used for longer ones, making it suitable for reviews of varying lengths. A 20-word threshold is set for the key phrase extraction process, with reviews containing fewer words than this subject to this process and those with more words subject to summarization.

We utilize the GPT-3 based text summarization technique with a pre-defined prompt to extract the key concerns from the reviews. On the other hand, we propose a Transformer-based key phrase extraction mechanism that combines Yake with semantic similarity measures. This proposed key phrase extraction mechanism works well for small size of texts and provides more diverse key phrases related to the app features. It is worth to be noted that, there are several keyword or key phrase extraction approaches available such as RAKE (Rapid Automatic Keyword Extraction) [43], Textrank [44], Yake [45], and KEA [46]. However, these models are not suitable as they either extract keywords from large paragraph of text (not suitable for short reviews) or provide a repetitive set of keywords.

Column 6 of Table II shows the key phrases extracted from the nine reviews of our case study. They highlight the specific aspects of the mapped app features that need to be considered by developers. Reviews $R4$ and $R6$ have no key phrases extracted as they are non-informative.

Our proposed end-to-end framework aims to perform all the above activities and help software development enterprises to drive their technology value stream more efficiently while reducing the burden of technical debt.

## IV. The Proposed Framework

In this section, we provide the details of our framework. Fig. 1 shows its components and how the information flows within them.

### A. Review Fetcher

This is a pre-processing component which fetches the most desirable set of end-user app reviews and dump them in the local repository. The process is parametric and developers can specify date ranges and even particular regions (e.g. US) from which reviews need to be fetched. The fetched dataset consists of the review content, the reviewer's name, the date of creation, and the rating. It is important to note that while writing the application reviews users may make grammatical errors or use unconventional language, which can make it difficult to understand the meaning behind their comments. Thus it is important to apply a grammar correction mechanism to these reviews in order to rectify the grammar mistakes. The grammar correction mechanism is achieved by implementing the concept of fine-tuning the T5 model on cleaned NAIST Lang-8 [47] dataset. The mechanism is originally proposed by Rothe et al. [47] in 2021 and achieves

state-of-the-art performance on grammar correction benchmark datasets such as CoNLL-14 [48]. The authors also provided a GitHub[3] repository where the entire process of the implementation has been documented. Experimental details about the implementation of this module is presented in Section VI-A.

### B. Review Sentiment Analyzer

This module helps to identify the intentions, emotions, and mood of the end-user by applying sentiment analysis on the review contents. The analysis is performed on all the reviews dumped in the local repository. It may appear that two reviews having the same topic(s) but one may be about a bug and another about adding some features.Let us consider the following two user reviews:

1) *"Don't like the awful button in the app screen is not working."*
2) *"A back button in the App screen will be helpful."*

Topic analysis alone would reveal that both these reviews discuss the same two topics: "*button*" and "*app screen*". However, review (1) is expressing a negative sentiment ("*awful button*") and pointing to a bug, whereas for review (2) the sentiment expressed is more neutral ("*will be helpful*") and can be considered as a feature request.

We identify two important tasks which are affected by the outcomes of this module:

1) Predicting the user's intention accurately.
2) Filtering out non-informative reviews.

The recommendation and maintenance system can be improved to identify the most important and relevant reviews more accurately, by incorporating user sentiments into the framework. Such reviews are expected to reflect the user's expressed intention. With this perspective, we can now consider it as a potential answer to Research Question 2 (**RQ2**).

The analyzer classifies reviews into five classes as mentioned in Section III-A. The identified sentiment class is tagged with the review and passed onto the next module. In this module, we leverage the capabilities of the transformer-based pre-trained language model, which has undergone fine-tuning on SemEval 2016 [49] and SemEval 2017 [50] sentiment analysis datasets. This powerful model enables us to accurately evaluate the sentiment expressed in various reviews. The detailed evaluation of these models is elaborated in Section VI-B.

### C. Review Filter

This module works on the reviews with tagged sentiments. It identifies reviews which are non-informative and removes them as they play no role towards maintenance of the app.

We consider some specific properties of non-informative reviews and use them as the criteria for eliminating them from our requirements set, namely

1) Reviews having extremely positive or negative sentiment. e.g. *"Good app, fantastic! I love it"*
2) Reviews having only emojis. e.g. "🔥🤩"
3) Reviews which are too short and generic. e.g. *"It's not installing", "Cannot update"*

[3]https://github.com/google-research-datasets/clang8

TABLE III
MAPPING OF TOPICS WITH STATE-OF-THE-ART TAXONOMY

| Topic | Relevant for Developers | Mapped Categories |
|---|---|---|
| Praise | No | Praise |
| Helpfulness | No | |
| Feature information | Yes | Feature Information |
| Shortcoming | Yes | Fault and shortcoming |
| Bug report | Yes | Bug |
| Feature request | Yes | Feature request |
| Other app | No | |
| Recommendation | No | |
| Noise | No | |
| Dissuasion | No | |
| Content request | Yes | Content request |
| Promise | Yes | Feature request |
| Question | Yes | Information enquiry |
| Improvement request | Yes | Improvement suggestion |
| Dispraise | No | Criticism |
| Other feedback | No | |
| Howto | Yes | Information enquiry |

The Review Sentiment Analyzer identifies the reviews with *Strongly Positive* and *Strongly Negative* sentiments. The BART-large-mnli model acts as a zero-shot classifier and identifies *non-informative* reviews. *Short* reviews having less than 10 words are also identified as potential elimination candidates. In the final step, we eliminate all the reviews that satisfy *all three criteria* from our local repository. It is worth noting that the current state-of-the-art recognizes the significance of extracting informative reviews as a critical aspect of enhancing the effectiveness of review analysis, as highlighted in previous works such as [10], [25]. As this process constitutes a pre-processing task in NLP, we have opted for a straightforward approach and developed our own review filter by following the techniques already existing in state-of-the-art approaches. The evaluation of the module is elaborated in Section VI-C.

### D. Review Topic Classifier

Different researches [7], [11], [20] leverage topic modelling to extract necessary information from the users' reviews. The sole purpose of topic classification is to know the abstract theme of the review or set of reviews. It assists developers to know about software issues like bugs, faults and shortcomings, or particular features that require improvement.

Pagano et al. [5] presented a set of possible topics (presented in the first column of Table III) that can be extracted from user reviews. They investigated that topics like shortcomings and bugs are mostly reported in reviews having negative sentiment. The feature requests and improvement suggestions are mostly on the neutral side, while information enquiry and feature information can be seen as positive reviews. It may be concluded that, apart from the review content, the sentiment of the review itself is another important feature to recognize the topic. Thus, we feed the reviews, along with their labelled sentiments, to the *Review Topic Extractor* module.

A study by Panichela et al. [12] aims to identify a taxonomy of user reviews that are relevant to software maintenance and

evolution. The authors identify six (6) categories of reviews that are crucial for software evolution, namely Feature Request, Opinion Asking, Problem Discovery, Solution Proposal, Information Seeking, and Information Giving. They also perform a systematic mapping with the taxonomy proposed by Pagano et al. [5] The results indicate that eight of the topics proposed by Pagano et al. [5] are relevant for developers performing maintenance tasks on apps. Similarly, Gu et. al. [3] also defined five review categories based on the taxonomy proposed by Pagano et al. [5]. In this research, we identified seven (7) categories of reviews that can be mapped to eleven (11) out of seventeen (17) categories of reviews proposed by Pagano et al. [5]. According to Pagano et al. these eleven categories are actually related to the app's improvement or maintenance tasks. Furthermore, we included two topics, *praise* and *criticism*, in our topic classification task to effectively categorize reviews with strong positive or negative sentiments. Table III displays the identified categories and aligns them with the current state-of-the-art review topic taxonomy. However, topic classification with these categories presents a challenge due to the lack of a training dataset to train a supervised model. In such a scenario, a zero-shot classifier, which can solve the problem of topic classification without any prior training, is an ideal solution.

Yin et. al. [51] proposed zero-shot classifiers for classifying emotions by training Word2Vec [52] and BinaryBERT [53] model on MNLI [54], FEVER [55], and GLUE-RTE [56] datasets. We train larger models like BART [41], BERT [13] and RoBERTa [14] on MNLI dataset. We introduce the required template for the classification task and create zero-shot classifiers on top of the pre-trained models. We compared these larger zero-shot models with the the models proposed by Yin et al. [51] on the Yahoo Answers [57] benchmark dataset. We found that the large variant of BART model performs better than all other models including the Binary-BERT model. The BART-large model achieves an F1-score of 54% whereas Yin et al. [51] report an F1-score of 37.9%, achieved by the BinaryBERT model (trained on MNLI dataset).

In our framework, we feed seven (7) labels for topic classification to the BART-large-mnli model (as listed in Section III-B). Additionally, we keep two topics *praise* and *criticism* in our experiments for mapping reviews with strongly positive or strongly negative sentiments, respectively. It is noteworthy that the zero-shot topic classifier classifies reviews and assigns scores against each of the topic, facilitating the categorization of reviews into various topic categories. The experimental evaluation of this module is presented in Section VI-D. We also provide a detailed experiment of constructing zero-shot classifier in Annexure II of the supplementary document, available online[4] available online.

### E. Topic Feature Mapper

Based on the topics identified for each of the reviews, this next module of the framework map these reviews to specific features of the application. The main intention of this task is to facilitate the development team to focus on the issues and the corresponding features that need to be worked upon for the next release of the app. The developer should provide concise documentation of app features in the form of text pairs $\langle feature\_name, description \rangle$. It is intuitive that, out of the seven topic categories, five are related to the improvement of existing features. These five topics include - {*feature information, bug, fault and shortcoming, improvement suggestion, information enquiry*}. The remaining two topics {*feature request, content request*} are not exactly relatable to existing features but rather suggest introducing new features.

We use the concept of asymmetric semantic search within the app documentation in order to find the features relatable to each particular review. Semantic search aims to increase search accuracy by comprehending the search query's content. Unlike standard search engines, which only discover items with lexical matches, semantic search can also find synonyms. The semantic search technique also enables us to identify multiple app features that may be associated with a particular review. This mechanism provides developers with the opportunity to identify and prioritize app features for software evolution and maintenance tasks, partially answering Research Question 1 (*RQ1*). A comprehensive answer to *RQ1* can be obtained from the outcome of the next module of the framework, the *Review Summarizer and Key Phrase Extractor*.

The detailed evaluation of this module is presented in Section VI-E.

### F. Review Key Phrase Summarizer

This module is designed to extract key phrases and provide a summary of user reviews by analyzing the semantic aspects of the text. The goal is to assist developers by highlighting the specific aspects of the app feature discussed in the review. Review content can be diverse in nature, and in some cases, it can be challenging to create a summary of short reviews. In such situations, key phrase extraction is essential. However, for longer reviews, summarization techniques can be used to identify the key points discussed by the reviewers. By combining text summarization and key phrase extraction, the module is able to effectively handle reviews of diverse lengths. In this module, a threshold of twenty (20) words is established. Reviews containing fewer words than the threshold are selected for key phrase extraction, while reviews containing more words are subject to the summarization process.

*1) Summarization Using GPT-3:* With the advent of advanced language models, such as GPT-3, text summarization has become more accurate and efficient as it can understand the context and generate coherent summaries, unlike traditional methods which rely on keyword extraction and frequency analysis. GPT-3 is a zero-shot text summarization model which means it doesn't require any training. The latest version of GPT-3 model, Text-DaVinci003, can process both long and short text of various types, making it a highly versatile tool for different use cases. Studies [58] have compared the performance of GPT-3 with other state-of-the-art models like T5 and BERT and found that GPT-3 is preferred for keyword-focused text summarization. Human annotators prefer the quality of summaries generated

[4] 10.5281/zenodo.7836451

TABLE IV
COMPARISON OF SUMMARIZATION MODELS BASED ON ROUGE SCORES FOR THE CNN DATASET

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| BART | 44.16 | 21.28 | 40.90 |
| T5 | 43.52 | 21.55 | 40.69 |
| GPT-3 | 38.68 | 14.24 | 28.08 |

by GPT-3 over other models, even though GPT-3 generated summaries may have low factuality scores. Furthermore, the study found that there were very few factual errors found in the qualitative analysis of the GPT-3 generated summaries. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [59] refers to a collection of metrics and accompanying software tools that have been tailor-made for the purpose of assessing the quality of automated summarization. A short description of each of the metric is given below.

1) *ROUGE-1:* measures the overlap of unigrams (single words) between the model-generated summary and the reference summary.
2) *ROUGE-2:* measures the overlap of bigrams (pairs of adjacent words) between the model-generated summary and the reference summary.
3) *ROUGE-L:* measures the longest common subsequence between the model-generated summary and the reference summary, giving credit to consecutive matches of words in addition to individual word matches.

Table IV shows a comparative analysis of the scores against ROUGE metrics for different text summarization models on CNN [60] benchmark dataset. Given that the GPT-3 model is a zero-shot model and does not require any training, yet it is preferred for keyword-focused text summarization, we decided to use it in this research to generate summaries from user reviews. The experimental settings of summarization mechanism with GPT-3 model and its parameters such as *prompt, max_tokens, temperature, frequency_penalty* and *presence_penalty* have been discussed in Section VI-F.

*2) Key Phrase Extraction:* The next process of Key phrase extraction is carried out in the following four steps:

i. *Text Embedding Model:* Since T5 shows the highest performance with respect to semantic similarity task [61], we choose the T5 model for text embedding. However, any variants of the transformer-based model can also be used for the text embedding task.
ii. *Candidate Key-Phrase Extraction:* In the next step, we aim to extract $n$-grams from the text. We use count-vectorizer technique for extracting the $n$-grams from the text. Additionally, to enhance the selection list, we use Yake [45] to extract key phrases.
iii. *Similar Key-Phrases:* The extracted candidate phrases are embedded by T5 model. Additionally, the particular review is also embedded by the same model. Since the candidate phrases and reviews are in the same vector space, we apply cosine similarity measure to find the most similar phrases that exceed a certain threshold.
iv. *Diversifying Key-Phrases:* It is also important to extract the most relevant key-phrases that are the least similar to each other. In order to achieve it, we start by specifying a value for $K$. Then we extract $2 \times K$ keywords from the set of $n$-grams or candidate key phrases. Pairwise semantic similarities are measured between these key-phrases to obtain a set of the most relevant key phrases that are least similar to each other. Table V shows the extracted key phrases from a sample review content. The detailed architecture of the *Key Phrase Extraction* mechanism has been provided in Annexure-II of the provided supplementary document, available online.

TABLE V
KEY PHRASES EXTRACTED FROM REVIEW WHERE K = 2

| | |
|---|---|
| **Review** | In Zoom App Screen freezes when I try to share the screen |
| **Diverse Key Phrases** | App screen freezes, screen freezes when, share the screen, try to share. |
| **Key Phrases** | App screen freezes, share the screen. |

It is important to note that the summarized highlights or the key phrases extracted from the reviews serve as a crucial guide for the development team to understand which tasks are essential for the immediate evolution and maintenance of the software, thereby answering Research Question 1 (**RQ1**). The experimental evaluation of this module is discussed in Section VI-F.

## V. FRAMEWORK VALIDATION METHODOLOGY

In this section, we provide a comprehensive methodology of the experimental evaluation process for the framework. We emphasize the steps taken to bring the framework to full operation, as follows:

- *Module Implementations:* Each module is meticulously scrutinized and based on this examination, we make deliberate and informed selections of the tools and technologies necessary to implement the required functionalities. The integration of these packages results into a tool prototype[5] that instantiates our generic framework and serves as the foundation for the experimental assessment of its efficacy. It is worth noting that alternative implementation choices for individual modules have the potential to enhance the experimental results even further.
- *Selection of Use cases:* In order to evaluate the effectiveness of our framework, we conducted an analysis of reviews for sixteen popular applications from various genres over a long period of time (2 years span)- three video conferencing apps, one note-taking app, and two apps each from OTT platforms, video players, instant messaging, online job search, photo editing, and online gaming. A detailed list of the apps included in our experiments can be found in Table VIII.
- *Experimental Evaluation and Validation:* In this phase, we perform experiments on the reviews of each selected application and conduct a thorough evaluation of each module within the framework. In Section VI, we provide a comprehensive overview of the evaluation criteria and

[5]https://github.com/svk-cu-nlp/reviews-to-app-maintenance

discuss their reasoning. The detailed experiment on the effectiveness evaluation of the framework is also presented in Annexure IV of the supplementary document, available online[6] available online. Additionally, we answer the research questions raised in Section III and highlight how the different modules of the framework efficiently address these questions.

- *Comparison with Baseline Approaches:* Finally, we evaluate the efficacy of the framework by comparing its results with those of other baseline approaches. In Section II, we have included a comprehensive table (Table I) that outlines a detailed comparison between our framework and other approaches. Specifically, we closely examine the work proposed by Sorbo et al. [25], which is the most relevant to our study.

In the next section, this methodology will be applied to validate the framework discussed in Section IV.

## VI. Implementation and Experimental Results

This section provides a thorough explanation of the experimental process and evaluation of the proposed framework. It is worth mentioning that we are introducing a new technology framework that has the potential for significant improvement in an emerging software development process in the industry. Hence, at this point, it seems too early to evaluate the framework's impact and effectiveness in the industry. However, we have conducted extensive experimental evaluations to validate the proposition and also to gain some initial insights. In this section, we provide a thorough documentation of the experimental setup, which includes the inputs and expected outputs for each module. Additionally, we present an analysis of the experimental results.

### A. Review Fetcher Module

Our tool prototype begins with this pre-processing component (refer Section IV-A) where developers are able to provide the name of the application and specify other parameters like time window, location, number of reviews to be considered, etc. Based on the developer preferences, reviews are fetched from Google Playstore. For our experiments, the module fetches 3400 reviews using web scraping. The extracted review dataset contains information that is irrelevant to our framework. These include *id, createdAt, reply, replyAt*, and *rating* and may be removed from our dataset.

After collecting the reviews, the grammar correction mechanism is applied to these reviews to rectify the grammatical mistakes. We fine-tune the T5 model on cleaned NAIST Lang-8 [47] dataset. The research paper also provides detailed instructions for the fine-tuning process. The training parameters are specified as follows.

- *Learning Rate:* 0.001
- *Epoch:* 7
- *tokens_per_batch:* 1048576
- *finetune_steps:* 2000

[6] 10.5281/zenodo.7836451

TABLE VI
Hyperparameter Settings for Training for Sentiment Analysis

| Hyperparameters | Value |
|---|---|
| learning_rate | 3e-5 |
| train_batch_size | 32 |
| eval_batch_size | 32 |
| max_seq_length | 256 |
| adam_epsilon | 1e-8 |

- *Optimizer:* adam
- *adam_epsilon:* 1e-6

To summarize, we recapitulate the inputs required for the module and the corresponding expected output.

- *Inputs:* Name of the application and additional parameters such as time window, language, region, and the values for the number of reviews to be extracted.
- *Output:* Grammar corrected review text without other information such as *id, createdAt, reply, replyAt*, and *rating*.

### B. Review Sentiment Analyzer Module

As presented in the framework, the first major module of our workflow is the sentiment analyzer of reviews. We carry out our experiments with four transformer models - BERT [13], RoBERTa [14], DistilBERT [40], and BART-large-mnli [41]. BERT, RoBERTa, and DistilBERT models have been fine-tuned on SemEval 2016 [49] and SemEval 2017 [50] datasets, which contained approximately 75,000 labeled tweets with five classes of sentiments. As part of our experimentation, we utilized the BART model, which was trained on the MNLI dataset and functioned as a zero-shot classifier. While testing various models, we found that the RoBERTa model outperformed the others, achieving an impressive accuracy of 77.45% for tweet sentiment classification. Based on this superior performance, we made the decision to incorporate the RoBERTa model into our framework for accurately analyzing the sentiments expressed in user reviews.

In the fine-tuning part, we use AdamW [62] as our optimizer and Categorical Crossentropy [63] as our loss function. We keep the learning rate as 3e-5. We add a softmax layer as the activation function. We trained the selected pre-trained models for different numbers of epochs, ranging between 1 and 5. The reason for choosing such a small number of epochs is that passing through a short fine-tuning process avoids the overfitting problem. We only take the optimized training outcome for each model within the range of 1 to 5 epochs. We observe that we achieve optimal training accuracy for 4 epochs. The hyper-parameter settings are mentioned in Table VI. In case of BART-large-mnli, since it is a zero-shot learner model, we provide 5 desired classes to the pipeline of classification task. The performance of these four models on SemEval dataset is given in Table VII. RoBERTa-large uncased model achieves the highest accuracy for the SemEval dataset. On the other hand, BART-large-mnli model has the advantage that it does not require any fine-tuning.

In order to classify user reviews into five (5) possible sentiment classes, we choose the RoBERTa-large uncased model. In Table VIII, we present the performance of RoBERTa for

TABLE VII
SENTIMENT ANALYSIS RESULT OF TRANSFORMER MODELS ON SEMEVAL 5 CLASS DATASET

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| BERT | 76.40% | 76.40% | 75.90% | 76.15% |
| RoBERTa | 77.45% | 77% | 77.56% | 77.28% |
| DistillBERT | 74.68% | 74.28% | 74.75% | 74.51% |
| BART | 73.80% | 73.26% | 73.60% | 73.43% |

TABLE VIII
PERFORMANCE OF ROBERTA MODEL FOR FINE-GRAINED SENTIMENT ANALYSIS

| App | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Zoom | 82.50% | 83.60% | 82.49% | 83.04% |
| Skype | 82.60% | 81.25% | 81.25% | 81.25% |
| Webex | 81.45% | 81.76% | 80.45% | 81.10% |
| WhatsApp | 82.67% | 80.00% | 82.35% | 81.16% |
| Telegram | 84.21% | 81.69% | 84.06% | 82.86% |
| PicsArt | 84.77% | 84.51% | 83.33% | 83.92% |
| Photoshop | 81.58% | 81.08% | 81.08% | 81.08% |
| VLC | 84.00% | 82.35% | 82.35% | 82.35% |
| MX player | 82.99% | 81.69% | 82.86% | 82.27% |
| LinkedIn | 84.21% | 80.56% | 85.29% | 82.86% |
| Indeed | 83.54% | 81.58% | 83.78% | 82.67% |
| Netflix | 81.88% | 80.56% | 81.69% | 81.12% |
| Prime Video | 83.33% | 80.56% | 81.58% | 82.89% |
| NFS | 83.75% | 82.05% | 84.21% | 83.12% |
| eFootball | 85.00% | 84.42% | 84.42% | 84.42% |
| Evernote | 82.12% | 81.08% | 82.19% | 81.63% |

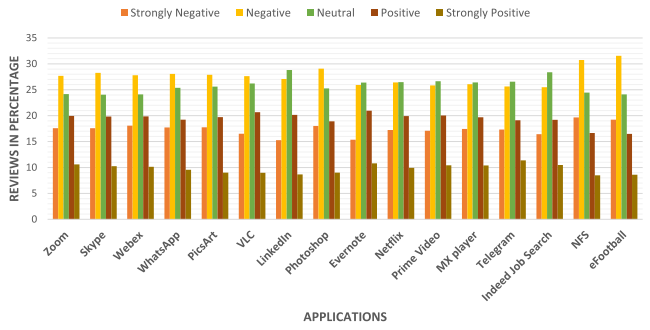



Fig. 2. Distribution of reviews (in percentage) for different apps across 5 class sentiments.

sentiment analysis of reviews concerned with the chosen 16 apps of our case study. We manually checked the labels and counted the number of *False Positives, False Negatives, True Positives* and *True Negatives*. Based on the confusion matrix, we estimated the accuracy, precision, recall, and F1-score. The distribution of reviews across the five (5) classes of sentiments is presented in Fig. 2.

At this stage, we would like to emphasize the inputs that were provided to the module, as well as the outputs generated by the module that will be utilized for further analysis.

- *Inputs:* Review text from the previous *Review Fetcher* module. In order to classify the sentiment of the reviews, we also provide the RoBERTa model that is already trained on the Twitter sentiment analysis dataset [49].
- *Output:* Review text tagged with one of the 5 classes of sentiment.

TABLE IX
REVIEW CATEGORIES FOR REMOVAL

| App | Strong Sentiment Reviews | Non-informative Reviews | Short Reviews | Removed Reviews |
|---|---|---|---|---|
| Zoom | 840 | 635 | 302 | 221 |
| Skype | 836 | 594 | 284 | 197 |
| Webex | 837 | 721 | 324 | 234 |
| WhatsApp | 819 | 746 | 298 | 201 |
| Telegram | 858 | 657 | 300 | 208 |
| PicsArt | 799 | 610 | 316 | 214 |
| Photoshop | 804 | 615 | 319 | 226 |
| VLC | 764 | 588 | 308 | 204 |
| MX player | 833 | 700 | 304 | 207 |
| LinkedIn | 715 | 580 | 305 | 214 |
| Indeed | 803 | 625 | 312 | 216 |
| Netflix | 812 | 674 | 311 | 210 |
| Prime Video | 824 | 652 | 296 | 204 |
| NFS | 836 | 741 | 338 | 228 |
| eFootball | 831 | 758 | 321 | 215 |
| Evernote | 780 | 549 | 309 | 218 |

### C. Review Filter Module

This module filters out non-informative user reviews identified by three criteria (as mentioned in Section IV-C). The sentiment of reviews is determined by the previous Review Sentiment Analyzer module. On the other hand, Non-informative reviews are classified by the BART-large-mnli zero-shot classifier. In order to identify short reviews, we use the spaCy[7] tokenizer library to parse and count the number of tokens in each review.

We choose to remove only those reviews which are identified as potential removal candidates according to *all the three criteria* mentioned above. This is done to ensure that we selectively remove only those reviews which are assured to be irrelevant for our framework.

Table IX shows the number of reviews with strong positive and negative sentiments (column 2), the number of non-informative reviews (column 3), and the number of short reviews (column 4) for each of the apps of our case study, separately. Column 5 shows the number of reviews from these apps which satisfy all the three classifications listed in columns 2–4 and are thus eliminated. The rest of the reviews, for each app, is ready for the topic classification task. To summarize, we reiterate the inputs and outputs of the *Review Filter* module for improving the clarity of the process.

- *Inputs:* The input to this module is the reviews tagged with the sentiment. In order to classify the informative and non-informative reviews, we also provide separately the BART-large-mnli zero-shot classification model that is already trained on MNLI [54] dataset.
- *Output:* As output, the module provides the filtered sentiment-tagged reviews that are relevant for further analysis.

### D. Review Topic Classifier Module

In topic classification task, due to the lack of sufficient supervised labelled dataset, we choose to use zero-shot classification. In our tool prototype, we utilize the BART-large-mnli model

[7]https://spacy.io/api/tokenizer

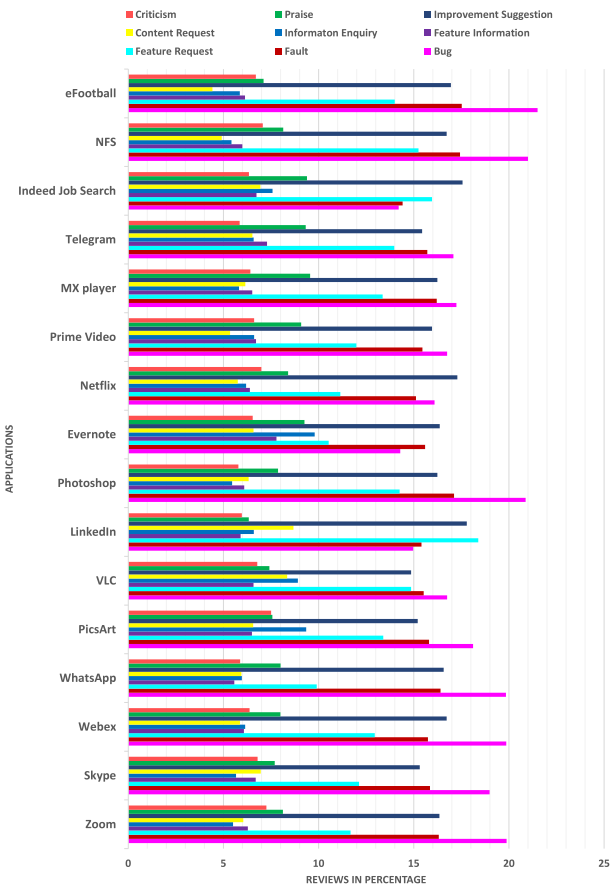



Fig. 3. Distribution of reviews across different topics.

to build our zero-shot classifier. A detailed documentation of building our zero-shot classifier is provided in Annexure II of the supplementary document, available online. In this zero-shot classifier, we feed 9 (7 + 2) labels (as discussed in detail in Section IV-D) to the classification pipeline of Huggingface library [64]. The zero-shot classifier considers the review as *Premise* and each label is treated as *Hypothesis*. As the pre-trained BART model is trained on MNLI dataset, it tries to find whether the *Premise* entails the *Hypothesis*. Based on the highest entailment score, it classifies the review into one of these nine classes. It is noteworthy that the classification result of each of the reviews is accompanied by scores for each topic. A low threshold of 0.3 has been set to increase the likelihood of reviews being classified under any of the topics. Reviews that have scores below this threshold for a particular topic are automatically discarded. Reviews that fail to be classified into any of the nine topics due to their scores for all topics being lower than the pre-defined threshold of 0.3, are not considered for further analysis.

Distribution of reviews, for the 16 apps of our case study, across the nine topics is presented in Fig. 3. We can observe that reporting of bugs is the most visible among the App reviews. Users have reported the highest number of bugs for e-football Android gaming app. More feature requests can be seen for LinkedIn app. On the other hand, users suggest more improvements on Indeed Job Search and LinkedIn apps. In Table X, we present the performance of our zero-shot classification model. We manually checked the *False Positives, False Negatives, True Positives* and *True Negatives*. The zero-shot classifier achieves upto ∼81% accuracy for topic classification of reviews associated with the 16 apps of our case study (refer Table X).

TABLE X
PERFORMANCE OF THE ZERO-SHOT CLASSIFIER

| App | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Zoom | 80.45% | 81.25% | 78.79% | 80.00% |
| Skype | 79.85% | 80.00% | 78.79% | 79.39% |
| Webex | 79.85% | 80.60% | 79.41% | 80.00% |
| WhatsApp | 78.20% | 77.94% | 79.10% | 78.52% |
| Telegram | 79.71% | 80.00% | 80.00% | 80.00% |
| PicsArt | 81.25% | 81.54% | 81.54% | 81.54% |
| Photoshop | 81.10% | 82.54% | 80.00% | 81.25% |
| VLC | 80.45% | 80.00% | 82.35% | 81.16% |
| MX player | 78.79% | 78.14% | 80.82% | 79.41% |
| LinkedIn | 79.85% | 80.00% | 81.16% | 80.58% |
| Indeed | 79.41% | 78.57% | 80.16% | 79.77% |
| Netflix | 79.41% | 81.69% | 79.45% | 80.56% |
| Prime Video | 79.39% | 81.82% | 78.26% | 80.00% |
| NFS | 81.06% | 81.82% | 80.60% | 81.20% |
| eFootball | 81.68% | 81.25% | 81.25% | 81.25% |
| Evernote | 78.91% | 80.00% | 78.79% | 79.39% |

Next, we take each of the seven topic classes of reviews from Fig. 3, and present their distribution across the five classes of sentiments identified earlier (refer Fig. 4). In this figure, we have presented the distribution of sentiment for the four most significant topic classes. The same distribution of sentiment across the other three topic classes can be found in Annexure-III of the online supplementary document, available online.[8] Each of the subfigures from Fig. 4(a), (b), (c), and (d) represents the distribution for one particular topic class.

We observe that most of the reviews regarding *bugs* (Fig. 4(a)) and *faults* have negative sentiments. The sentiments for *feature request* (Fig. 4(b), *improvement suggestion* (Fig. 4(c)), are around neutral and negative. The sentiment for *feature information*, *information enquiry* (Fig. 4(d)) are centered around neutral sentiments. This implies that the users' sentiments are balanced. We can infer that the users are satisfied to some extent; however, the app lacks some content or user guides for specific groups of users.

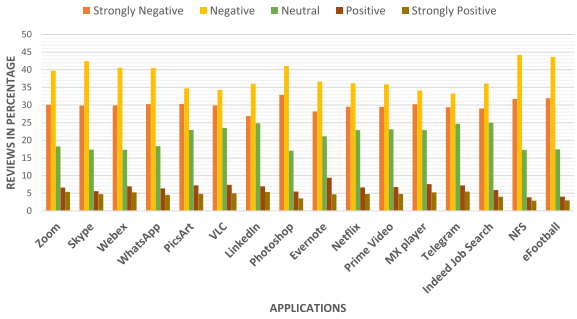
At this point, it is important to emphasize the inputs that were fed into the module, as well as the resulting outputs that will be utilized for subsequent analysis.

- *Inputs:* The relevant informative reviews tagged with sentiment. In order to perform the topic classification, we also provide separately the BART-large-mnli zero-shot classification model.
- *Output:* As output, reviews will be tagged with corresponding topics identified by the module.

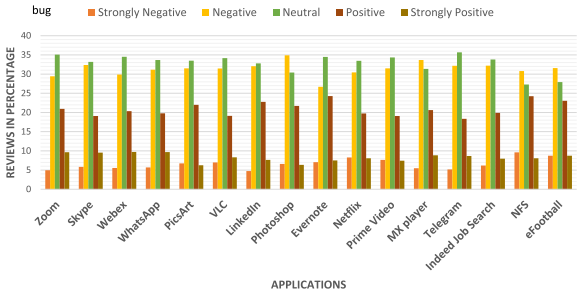
### E. Topic Feature Mapper Module

In order to carry out our experiments with 16 apps of our case study, we manually collect the documentation of app features consisting of official user guides, FAQs, and support documents

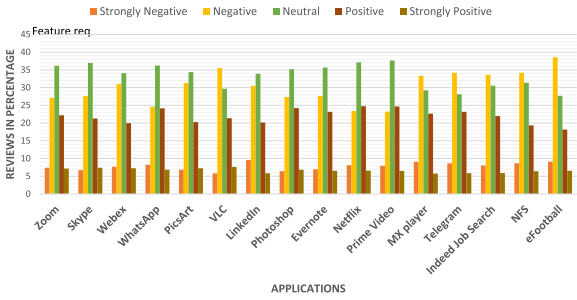
[8] 10.5281/zenodo.7836451



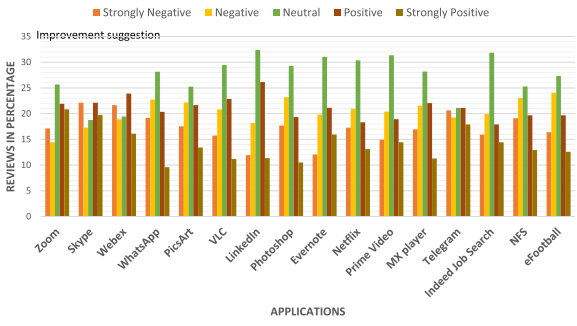

(a) Bug




(b) Feature Request




(c) Improvement Suggestion




(d) Information Enquiry

Fig. 4. Distribution of sentiments across different topics.

from web pages created for the respective Apps. We take user reviews and perform a semantic search on the respective app documentation to map the particular review to specific app features. Semantic search works by embedding all elements in the corpus, whether they are phrases, paragraphs, or texts, into a vector space. The query is embedded into the same vector space as the corpus at search time, and the closest embeddings from the corpus are discovered.

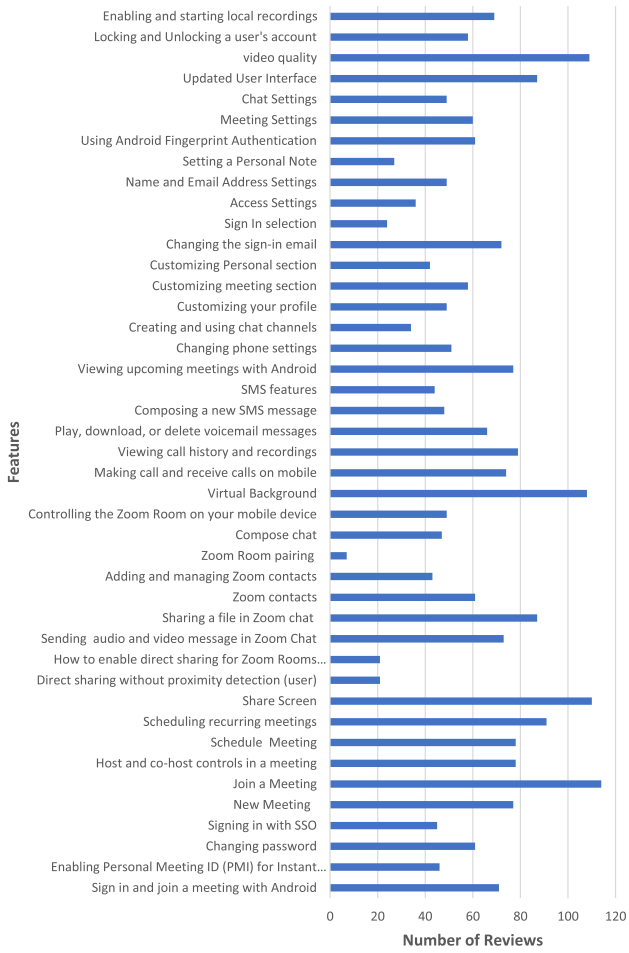



Fig. 5. Mapping of reviews with app features of zoom.

We use a RoBERTa-based transformer model [65] trained on MS MARCO [42] passage retrieval dataset to embed both the user review and documentation set. Cosine scores are evaluated in order to measure the semantic similarity between a particular user review and the documentation of app features. The cosine scores are rated on a scale of 0 to 1, and we have established a threshold of 0.5 to identify multiple app features that are associated with a particular review. Through this process, we can accurately identify multiple app features that are referenced in a particular review by ranking the semantic search results.

Fig. 5 shows the distribution of reviews across 44 different features of the Zoom Android application. From the figure, we can observe that some of the more reported issues include *Join a Meeting, Sharing Screen, Sharing a file, Virtual Background, Video Quality, User Interface, Scheduling recurring meetings* and many more. The development team can use this feedback to decide on upcoming iterations and potentially shippable software releases. Fig. 6 shows the distribution of sentiments across the most frequently discussed app features of Zoom. Furthermore, we have presented the distribution of topics across the most frequently reported features in Fig. 7. This information allows the development team to take a call on the urgency of addressing an issue based on the negativity of the sentiments associated with a particular feature.

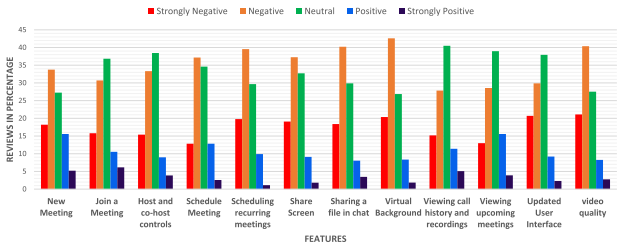



Fig. 6. Distribution of sentiments across specific features.

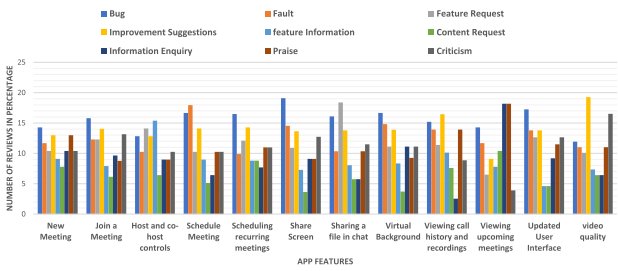



Fig. 7. Distribution of topics across specific features.

For the purposes of brevity, the distribution of reviews across the features of other Apps is detailed in Annexure III (refer to supplementary material, available online[9]). We provide a summary of the module by specifying both the inputs accepted by the module and the resulting output, which is of utmost importance for the software development team.

- *Inputs:* The essential input for our module is the application documentation, which is in the form of text pairs denoting feature names and their corresponding descriptions. Additionally, for performing semantic searches, we provide a fine-tuned RoBERTa model based on the MS MARCO [42] dataset. Lastly, we also include review text as input to the module.
- *Output:* This module provides a list of feature names along with their corresponding relatedness scores concerning a specific review.

### F. Review Key Phrase Summarizer Module

This module begins with the generation of summaries from the reviews. We have set a threshold of twenty (20) words to determine the appropriate method for analyzing each review. If a review contains fewer than the threshold number of words, we will extract key phrases from it. On the other hand, if a review exceeds the threshold, we will subject it to a summarization process.

*1) Review Summary Generation:* GPT-3, as a generative model, creates a summary of a text according to a specific prompt. To optimize the quality of the summarization, we can adjust the following parameters.

- *prompt:* This is the input text that the model will use as a starting point for generating the summary. We specify the prompt as "*Summarize the given review and extract key points*"
- *model:* This parameter specifies which version of GPT-3 to use. we specify *text-davinci-003* model for the summarization task.
- *temperature:* The setting controls the level of randomness in the generated text. A value of 0 will result in deterministic output, meaning the same output will be generated every time for a given input text. In contrast, a value of 1 will cause the engine to incorporate more creative elements in the generated text. We continue our task with the default setting of 0.7 for summarization.
- *top_p:* The proportion of the most likely tokens to keep in the generated summary. We keep the default value of one (1) for this parameter.
- *frequency_penalty:* it is used to control the repetition of certain tokens in the generated text. We do not change the default setting and keep it zero (0).
- *presence_penalty:* It is used to control the likelihood of certain tokens appearing in the generated text. We continue our experiment with its default setting i.e. zero (0).

With the above-mentioned parameters, we accomplish our experiment for review summarization. Table XI presents a summarization of a sample review.

*2) Key Phrase Extraction:* We use the Yake [45] algorithm for identifying these key phrases as it achieves higher accuracy than other existing approaches. However, Yake does not perform as expected when applied on short-length reviews. Thus, we combine Yake with the countvectorizer[10] technique, which can extract $n$-gram sequences from short reviews as well. Table XII shows such short-length examples where the combination of Yake and countvectorizer extract more meaningful key phrases than applying Yake alone. The $n$-grams extracted (by countvectorizer) and key phrases extracted (by Yake) are combined to enhance the list of key phrases. We keep the size of $n$ as 3 for

TABLE XI
SUMMARIZATION OF REVIEWS USING GPT-3

| Review | Summary |
|---|---|
| Crashes at least once a week. Devices used have to be rebooted due to app randomly turning the screen white while making a static noise before turning device off completely while in use. | - The app crashes frequently.<br>- Using the app can cause devices to malfunction.<br>- Static noise causing the device to turn off completely while in use. |
| One of the worst apps I have ever used, it's so bad that it is suprising. They have so much money from all the school contracts but still cannot make a good UI or any features that are good. Just use discord, sure it's not as "professional", but at least it actually works and is not a torture to use. | - The app is considered to be very poor in quality.<br>- The app's user interface and features are not well-designed or functional.<br>- Discord is a better alternative to the app because it works well and is not difficult to use. |

TABLE XII
COMPARISON OF KEY PHRASE EXTRACTION BY YAKE ALONE AND COMBINED WITH COUNTVECTORIZER

| Sentence | Extracted Key phrases by Yake | Extracted Key phrases by n-grams + Yake |
|---|---|---|
| The App should remember permissions, settings and user preferences. | settings and user, permission, user, preferences | remember permission settings, settings and user, user preferences |
| The button in the webpage is no longer needed | longer needed, needed, button | button in webpage, no longer needed, |

[9]10.5281/zenodo.7836451

[10]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

TABLE XIII
COMPARING MODELS FOR SEMANTIC SIMILARITY

| Models | STSB | MNLI |
|---|---|---|
| BERT-Large | 87.6 | 85.9 |
| RoBERTa-Large | 92.4 | 90.2 |
| $\text{SMART}_{RoBERTa}$ | 92.9 | 90.8 |
| T5 | 92.5 | 92 |
| BART-Large | 91.2 | 90.1 |

TABLE XIV
KEY PHRASE EXTRACTION EXPLAINED

| | |
|---|---|
| **Review Content** | Zoom is a good video chat app for secure meet--ings, but unfortunately the user interface is awkward. It should remember permissions, settings and user preferences. Despite the flaws, I do recommend the Zoom app. |
| **Extracted Key Phrases** | video chat app, 0.7625, chat app secure, 0.665, zoom good video, 0.6192, remember permissions settings, 0.5955, secure meetings unfortunately, 0.5638, user interface awkward, 0.5318, unfortu--nately user interface, 0.5287, recommend zoom app, 0.5151, permissions settings user, 0.5028 |
| **Diverse Set** | video chat app, zoom good video, remember permissions settings, secure meetings unfortu--nately, user interface awkward, recommend zoom app |

extracting $n$-grams as Yake also achieves optimal performance for $n = 3$.

In the next phase, we proceed to identify the similar key phrases using semantic similarity scores. This requires embedding both the reviews as well as the key phrases extracted previously. We compare different models for text embedding by applying them on two well-known semantic similarity datasets - STSB [66] and MNLI [54]. Table XIII shows that T5 [61] and RoBERTa [14] models achieve similar performance in terms of accuracy in the STSB dataset. However, T5 outperforms other models in the MNLI dataset. Thus, we use the T5 model for text embedding purposes. We apply the cosine similarity approach on these text embeddings to find the similar key phrases corresponding to each review. In Table XIV, row 2 shows the similarity scores of the extracted key phrases to the sample review content shown in row 1.

Next, we aim to identify the most diversified set of key phrases for each review. We observe that most reviews contain a key phrase within a fixed interval of words, say $K$. We diversify the extracted similar key phrases by specifying the value of $K$. We keep the default value of $K$ as $\lceil \frac{1}{6} \rceil$-th of the number of words in the review. The value of $K$ can be altered by specifying it in the module explicitly. In our sample review content shown in Table XIV, row 3 enlists the most diverse set of key phrases.

We evaluate the end-to-end key phrase extraction module (which combines the three activities mentioned above) with the T5, RoBERTa, and BART pre-trained language models. In Table XV, we refer to these end-to-end models as KPT5, KPRoBERTa, and KPBART (marked with *), respectively. We provide a comparative analysis of these three model with state-of-the-art key phrase extraction models by applying them on three benchmark datasets (Inspec [67], SemEval2010 [68] and SemEval2017 [50]). We observe that our proposed KPT5 model achieves the highest F1-score for SemEval 2010 dataset. At this stage, it's crucial to emphasize the inputs and outputs generated

TABLE XV
F1-SCORE OF DIFFERENT APPROACHES FOR KEY PHRASE EXTRACTION

| Model | Inspec | SemEval 2010 | SemEval 2017 |
|---|---|---|---|
| RoBERTa+BiLSTM-CRF | 59.5 | 27.8 | 50.8 |
| RoBERTa+TG-CRF | 60.4 | 29.7 | 52.1 |
| SciBERT+Hypernet-CRF | **62.1** | 36.7 | **54.4** |
| RoBERTa-extended-CRF | 62.09 | 40.61 | 52.32 |
| RoBERTa+Hypernet-CRF | 62.3 | 34.8 | 53.3 |
| KPT5* | 61.24 | **40.72** | 52.56 |
| KPRoBERTa* | 58.74 | 35.26 | 51.46 |
| KPBART* | 60.58 | 39.41 | 52.64 |

by the implemented *Review Key Phrase Summarizer* module. These details enable the development team to comprehend the significant concerns raised in the reviews.

- *Inputs:* The reviews are the most important input for this module. In order to generate the summary of the reviews, GPT-3 model is fed into the module. The proposed key phrase extraction process (KPT5) involves the Yake [45] algorithm and T5 model beforehand. The Yake algorithm is implemented using the instruction provided in their GitHub repository.[11]
- *Outputs:* The module provides the summary for the long text reviews using GPT-3. For short reviews, the module provides the key phrases using our proposed key phrase extraction mechanism KPT5.

This concludes our set of experiments for each individual module of our framework. We have proposed to use certain specific models and algorithms based on our experimental evaluations. However, these are only suggestive in nature and by no means restrictive on the users of this framework.

## VII. DISCUSSION

In this section, we discuss some results and observations associated with our framework. The framework uses advanced language models like BERT, T5, BART, or GPT-3 to handle NLP tasks without requiring a separate parsing technique. These models are capable of directly working with raw text inputs as they have been trained on large amounts of text data, enabling them to extract information and make predictions by identifying patterns and relationships within the text. Therefore, the framework does not need an additional NLP parsing technique, thanks to the efficient use of these language models. Some of the key findings and implications from our experiments are summarized in Fig. 8.

In light of the experimental results, we can return to the initial research questions.

**RQ1:** *How can the end-user app reviews help the development team to precisely identify the app features which are of immediate concern?*

The proposed framework tries to identify the specific features of a software application that should be the major focus of the software development team in the upcoming deployments of the technology value stream. The only human involvement

[11]https://github.com/LIAAD/yake

- Except for online job search applications, reviews of all other app categories is dominated by bug and fault reports.
- All apps have a high percentage of reviews that suggests improvements in the respective apps.
- Bug reports generally have *negative* and *strongly negative* sentiments associated with them.
- Like bug reports, faults are dominated by *negative* and *strongly negative* sentiments.
- Feature requests have a relatively even distribution of *neutral* and *negative* sentiments. The percentage of such requests with *positive* sentiments is observed to be higher than that of bug reports.
- Improvement suggestions have more *negative* sentiments than *neutral* sentiments for video playing and online gaming applications.
- Information enquiries have a somewhat uniform distribution across all user sentiments.
- Feature information and content requests are dominated by *neutral* sentiments, accompanied by an even mix of *negative* and *positive* sentiments.

Fig. 8. Key findings and implications.

required in this framework is to provide app documentation. Besides, the framework categorizes reviews into distinct classes, as demonstrated in the Topic Classification module. It also identifies the specific app features referenced in the reviews, a task accomplished by the Topic Feature Mapper module. The framework takes it a step further by highlighting the particular aspects of the identified features that are causing concern among end-users. The summarization and key phrase extraction modules enable the identification of these aspects and key concerns, resulting in a well-defined set of recommendations to consider in their planning of future iterations in the CI/CD pipeline.

**RQ2:** *How can the user sentiments be integrated into the recommendation system to identify the more relevant reviews that are significant for the technology value stream?*

Users' intentions are often reflected by their sentiments, which get embedded in their reviews. Our proposed framework begins with sentiment analysis of the reviews. We observe that most of the informative reviews belong to the category of neutral and negative sentiments. We filter out those reviews which have very strong sentiments (negative or positive) embedded in them. The framework focuses and prioritizes the *negative, neutral* and *positive* reviews in order to perform further analysis. Another aspect of sentiment analysis is that it helps to identify the theme of the review. In order to prevent collaborative attacks on the framework by groups of malicious end-users, we identify the sentiments associated with the reviews and use them for coming up with the right set of recommendations. This prevents the framework from developing a bias towards irrelevant reviews which are typically submitted with the intention of misguiding the business customers and end-user market. Thus, there is a two-fold purpose for performing sentiment analysis of app reviews and using the outcome of that analysis to drive the technology value stream.

**RQ3:** *How can we integrate the entire framework within the software development pipeline to automate the continuous maintenance process?*

Modern software delivery and deployment rely heavily on the Principles of Flow within DevOps environments [69]. In order to shift the focus of the technology value stream towards delivering business value to the customer, there is a need to concentrate on meeting users' needs by amplifying and reducing feedback loops at the same time [70]. DevOps Research and Assessment (DORA) [71] shows that development teams achieve higher performance when they work in organizations that utilize the following things.

1) Collection of user's satisfaction metrics.
2) Analyze and respond to customer reviews on the quality of the product and its features.
3) Utilize the analysis of the feedback to help design products and features.

With this prior discussion, we observe that our proposed framework fits into this feedback loop of DevOps platforms.

## VIII. Threats to Validity

*Threat related to framework validity:* One of the threats to validity of the framework is that the efficiency of the proposed framework is bounded by the efficiency of the different machine learning models that have been deployed across the different components of the framework. On the contrary, the framework is designed to be flexible and adaptable to the constantly evolving NLP field and its cutting-edge advancements. Its underlying language models are based on the transformer architecture, enabling easy replacement with more efficient, compatible models.

Moreover, in the actual implementation, the framework is restricted to handling only the English language and is not equipped to process other languages. In order to counteract this issue, the language models utilized by the framework must be designed for multilingual processing. This represents a promising area for future research efforts.

*Threat related to external dependencies:* One major external dependency of the framework is the availability of app documentation. The development team has to extract texts or paragraphs about specific features from the app documentation. App documents, which can be readily used in NLP tasks, are sometimes not available. Documents exist in unstructured format such as web pages and PDFs. It is a laborious job to extract texts from such unstructured documents and feed them into the proposed framework for further processing.

## IX. Conclusion

In this article, we presented a framework which combines several NLP techniques such as Sentiment Analysis, Text Analysis, and Text Classification in order to decipher meaningful information that often remain embedded in app user reviews. The purpose of the framework is to use this information and automate

the process of guiding software developers to deliver maximum customer value through successive software releases. The framework helps to pinpoint which app segment requires attention from the development and maintenance team. Our experiments demonstrate that the combination of sentiment analysis and topic classification on user reviews greatly improves the quality of recommendations for continuous app development. In addition, the framework aids developers in identifying specific app feature segments that demand attention for maintenance tasks. It also offers a succinct summary of the review and underscores crucial issues raised by the reviewer. It is essential to highlight that the paper presents a novel framework designed to address the research problem, which is not confined to the performance of any singular module within the framework. As time progresses, the deep learning models employed in the framework can be substituted with even more high-performing and compatible models, thereby ensuring scalability.

As a first direction for future work we plan to extend our work by developing a complete DevOps pipeline that integrates our framework with existing CI/CD pipelines. We also intend to evaluate our framework on a variety of app reviews for apps belonging to different categories such as mobile gaming, e-commerce applications, financial and payment applications, and so on. We also aim to eliminate human intervention by automating the collection of app documentation. Finally, we aim to enhance the summarization and key-phrase extraction process and introduce text generation models to provide developers an exact direction towards App maintenance.

## REFERENCES


[1] W. Cunningham, "The wycash portfolio management system," *ACM SIGPLAN OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1992.

[2] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *J. Syst. Softw.*, vol. 125, pp. 207–219, 2017.

[3] X. Gu and S. Kim, ""what parts of your apps are loved by users?"(T)," in *Proc. IEEE/ACM 30th Int. Conf. Automated Softw. Eng.*, 2015, pp. 760–770.

[4] L. V. G. Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Proc. IEEE 35th Int. Conf. Softw. Eng.*, 2013, pp. 582–591.

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

[6] S. Scalabrino, G. Bavota, B. Russo, M. Di Penta, and R. Oliveto, "Listening to the crowd for the release planning of mobile apps," vol. 45, no. 1, pp. 68–86, Jan. 2017.

[7] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proc. IEEE 10th Work. Conf. Mining Softw. Repositories*, 2013, pp. 41–44.

[8] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party Android marketplaces," in *Proc. 2nd ACM Conf. Data Appl. Secur. Privacy*, 2012, pp. 317–326.

[9] 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.*, 2016, pp. 1023–1027.

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

[11] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 48–58.

[12] S. Panichella, J. Aponte, M. Di Penta, A. Marcus, and G. Canfora, "Mining source code descriptions from developer communications," in *Proc. IEEE 20th Int. Conf. Prog. Comprehension*, 2012, pp. 63–72.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018.

[14] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019.

[15] 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.*, 2014, pp. 153–162.

[16] Z. Peng, J. Wang, K. He, and M. Tang, "An approach of extracting feature requests from app reviews," in *Proc. 12th Int. Conf. Collaborate Comput. Netw. Appl. Worksharing*, Beijing, China, Springer, Nov. 10–11, 2017, pp. 312–323.

[17] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Softw. Eng.*, vol. 21, no. 3, pp. 1067–1106, 2016.

[18] H. Khalid and M. Shihab, "What do mobile app users complain about? A study on free IoS apps," *IEEE Softw.*, vol. 32, no. 3, pp. 70–77, May/Jun. 2015.

[19] T. Johann et al., "SAFE: A simple approach for feature extraction from app descriptions and app reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf.*, 2017, pp. 21–30.

[20] 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.*, 2015, pp. 281–290.

[21] 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.*, 2016, pp. 1023–1027.

[22] A. Di Sorbo, C. A. Visaggio, M. Di Penta, G. Canfora, and S. Panichella, "An NLP-based tool for software artifacts analysis," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, 2021, pp. 569–573.

[23] Y. Zhou, Y. Su, T. Chen, Z. Huang, H. Gall, and S. Panichella, "User review-based change file localization for mobile applications," *IEEE Trans. Softw. Eng.*, vol. 47, no. 12, pp. 2755–2770, Dec. 2021.

[24] 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, 2020.

[25] A. Di Sorbo et al., "What would users change in my app? Summarizing app reviews for recommending software changes," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 499–510.

[26] C. Tao, H. Guo, and Z. Huang, "Identifying security issues for mobile applications based on user review summarization," *Inf. Softw. Technol.*, vol. 122, 2020, Art. no. 106290.

[27] E. Guzman, O. Aly, and B. Bruegge, "Retrieving diverse opinions from app reviews," in *Proc. IEEE/ACM Int. Symp. Empirical Softw. Eng. Meas.*, 2015, pp. 1–10.

[28] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H. C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *Proc. IEEE 24th Int. Conf. Softw. Anal. Evol. Reengineering*, 2017, pp. 91–102.

[29] H. Jiang et al., "Recommending new features from mobile app descriptions," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 4, pp. 1–29, 2019.

[30] C. Gao, B. Wang, P. He, J. Zhu, Y. Zhou, and M. R. Lyu, "PAID: Prioritizing app issues for developers by tracking user reviews over versions," in *Proc. IEEE 26th Int. Symp. Softw. Rel. Eng.*, 2015, pp. 35–45.

[31] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. Jan, pp. 993–1022, 2003.

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

[33] 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, 2021, Art. no. e2316.

[34] D. Martens and W. Maalej, "Towards understanding and detecting fake reviews in app stores," *Empirical Softw. Eng.*, vol. 24, no. 6, pp. 3316–3355, 2019.

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

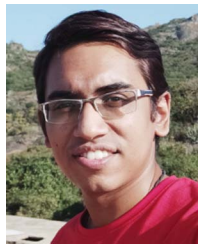
[36] E. Guzman, P. Bhuvanagiri, and B. Bruegge, "FAVe: Visualizing user feedback for software evolution," in *Proc. IEEE 2nd Work. Conf. Softw. Vis.*, 2014, pp. 167–171.

[37] 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./Oct. 2019.

[38] J. Dabrowski, E. Letier, A. Perini, and A. Susi, "Mining user opinions to support requirement engineering: An empirical study," in *Proc. 32nd Int. Conf. Adv. Inf. Syst. Eng.*, Grenoble, France, Springer, Jun. 8–12, 2020, pp. 401–416.

[39] D. Franzmann, A. Eichner, and R. Holten, "How mobile app design overhauls can be disastrous in terms of user perception: The case of snapchat," *ACM Trans. Social Comput.*, vol. 3, no. 4, pp. 1–21, 2020.
[40] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019.
[41] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.
[42] T. Nguyen et al., "MS MARCO: A human generated machine reading comprehension dataset," in *Proc. Workshop Cogn. Comput. Conf. Neural Inf. Process. Syst.*, 2016.
[43] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining Appl. Theory*, vol. 1, no. 1/20, pp. 10–1002, 2010.
[44] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2004, pp. 404–411.
[45] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "YAKE! keyword extraction from single documents using multiple local features," *Inf. Sci.*, vol. 509, pp. 257–289, 2020.
[46] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "KEA: Practical automated keyphrase extraction," in *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*. Hershey, PA, USA: IGI Global, 2005, pp. 129–152.
[47] S. Rothe, J. Mallinson, E. Malmi, S. Krause, and A. Severyn, "A simple recipe for multilingual grammatical error correction," 2021, *arXiv:2106.03830*.
[48] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," in *Proc. 18th Conf. Comput. Natural Lang. Learn. Shared Task*, Baltimore, Maryland, Association for Computational Linguistics, 2014, pp. 1–14. [Online]. Available: https://aclanthology.org/W14--1701
[49] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "Semeval-2016 task 4: Sentiment analysis in twitter," 2019.
[50] S. Rosenthal, N. Farra, and P. Nakov, "Semeval-2017 task 4: Sentiment analysis in twitter," 2019.
[51] W. Yin, J. Hay, and D. Roth, "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach," 2019.
[52] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.
[53] H. Bai et al., "BinaryBERT: Pushing the limit of BERT quantization," 2020.
[54] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, Association for Computational Linguistics, 2018, pp. 1112–1122. [Online]. Available: http://aclweb.org/anthology/N18--1101
[55] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: A large-scale dataset for fact extraction and verification," 2018.
[56] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2018.
[57] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," vol. 28, 2015.
[58] A. Bhaskar, A. R. Fabbri, and G. Durrett, "Zero-shot opinion summarization with GPT-3," 2022, *arXiv:2211.15914*.
[59] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop Text Summarization Branches Out*, 2004, pp. 74–81.
[60] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 3075–3081.
[61] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
[62] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017.
[63] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.
[64] T. Wolf et al., "Huggingface's transformers: State-of-the-art natural language processing," 2019.
[65] K. Wang, N. Reimers, and I. Gurevych, "TSDAE: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning," 2021.
[66] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation," 2017, *arXiv: 1708.00055*.
[67] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2003, pp. 216–223.
[68] I. Hendrickx et al., "SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," 2019, *arXiv: 1911.10422*.
[69] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," in *Proc. IEEE 5th Int. Conf. Innov. Comput. Technol.*, 2015, pp. 78–82.
[70] F. Beetz and S. Harrer, "GitOps: The evolution of DevOps?," *IEEE Softw.*, vol. 39, no. 4, pp. 70–75, Jul./Aug. 2022.
[71] N. Forsgren, M. C. Tremblay, D. VanderMeer, and J. Humble, "DORA platform: DevOps assessment and benchmarking," in *Proc. Int. Conf. Des. Sci. Res. Inf. Syst. Technol.*, Springer, 2017, pp. 436–440.

**Souvick Das** received the BSc and MSc degrees in computer science from West Bengal State University, India in the year 2012 and 2014 respectively. He is a research associate with the Department of Environmental Science, Informatics, and Statistics, of Ca' Foscari University, Venice, Italy and is currently working toward the PhD degree in Computer Science and Engineering from University of Calcutta, India. He has qualified UGC NET-JRF in the year of 2016.



**Novarun Deb** (Member, IEEE) received the master's and PhD degrees in eequirements engineering from the Department of Computer Science and Engineering, University of Calcutta. He is an assistant professor with the Indian Institute of Information Technology, Vadodara, India. He was a research associate with the Department of Environmental Science, Informatics, and Statistics, of Ca' Foscari University, Venice, Italy, for more than two years.

**Nabendu Chaki** (Senior Member, IEEE) is a professor with the University of Calcutta, Kolkata, India. He is sharing 7 international patents including 4 US patents. He has authored 7 books and nearly 200 Scopus Indexed papers in Journals and International conferences. He is the founder chair of ACM Professional Chapter in Kolkata. He was active in 2009–2015 for developing international standards in Software Engineering and Service Science as a Global (GD) member for ISO-IEC.

**Agostino Cortesi** is a professor of computer science with Ca' Foscari University, Venice, Italy. He has extensive experience in the area of static analysis and software verification techniques, with particular emphasis on security applications. He published more than 150 papers in high level international journals and proceedings of international conferences. Currently, he serves as co-editor in chief of the book series "Services and Business Process Reengineering" published by Springer-Nature.