IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 July 2024, accepted 12 August 2024, date of publication 14 August 2024, date of current version 26 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3443527

RESEARCH ARTICLE

Enhancing User Trust and Interpretability in AI-Driven Feature Request Detection for Mobile App Reviews: An Explainable Approach

ISHAYA GAMBO[1], RHODES MASSENON[1], CHIA-CHEN LIN[2], (Member, IEEE), ROSELINE OLUWASEUN OGUNDOKUN[3,4], SAURABH AGARWAL[5], AND WOOGUIL PAK[5], (Member, IEEE)

[1]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife 220282, Nigeria
[2]Department of Information Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan
[3]Department of Centre of Real Time Computer Sciences, Kaunas University of Technology, 44249 Kaunas, Lithuania
[4]Department of Computer Science, Landmark University, Omu-Aran 251103, Nigeria
[5]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea

Corresponding authors: Chia-Chen Lin (ally.cclin@ncut.edu.tw), Saurabh Agarwal (saurabh@yu.ac.kr), and Wooguil Pak (wooguilpak@yu.ac.kr)

This work was supported in part by the National Science and Technology Council under Grant NSC 111-2410-H-167-005-MY2 and Grant NSC 112-2634-F-005-001-MBK.

ABSTRACT Mobile app developers struggle to prioritize updates by identifying feature requests within user reviews. While machine learning models can assist, their complexity often hinders transparency and trust. This paper presents an explainable Artificial Intelligence (AI) approach that combines advanced explanation techniques with engaging visualizations to address this issue. Our system integrates a bidirectional Long Short-Term Memory (BiLSTM) model with attention mechanisms, enhanced by Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP). We evaluate this approach on a diverse dataset of 150,000 app reviews, achieving an F1 score of 0.82 and 89% accuracy, significantly outperforming baseline Support Vector Machine (F1: 0.66) and Convolutional Neural Network (CNN) (F1: 0.72) models. Our empirical user studies with developers demonstrate that our explainable approach improves trust (27%) when explanations are provided and correct interpretation (73%). The system's interactive visualizations allowed developers to validate predictions, with over 80% overlap between model-highlighted phrases and human annotations for feature requests. These findings highlight the importance of integrating explainable AI into real-world software engineering workflows. The paper's results and future directions provide a promising approach for feature request detection in app reviews to create more transparent, trustworthy, and effective AI systems.

INDEX TERMS Explainable AI, feature request detection, machine learning interpretability, mobile app development, software requirements, user trust.

I. INTRODUCTION

In recent years, interest in developing transparent and interpretable machine learning (ML) models, known as Explainable Artificial Intelligence (XAI), has been increasing [1], [2]. Explainability is crucial for gaining user trust and acceptance as AI systems are used more frequently in critical sectors like healthcare, industry, and transportation [3], [4], [5], [6], [7]. Research suggests that when users can understand the reasoning behind an AI system's decision-making process, they are more likely to have confidence in the system [8], [9], [10]. To provide this

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

transparency, XAI tries to clarify the logic and important components that affect the model outputs. For XAI, methods like SHAP [11], [12], LIME [13], and counterfactual visual explanations [14] have been proposed. These methods produce coherent explanations and visual representations that show how input features affect model predictions. For instance, LIME highlights superpixel segments that most influenced a classifier's output [15], [16]. As such, explainable AI allows users to scrutinize the model and ensure it behaves reasonably before deploying it in real-world systems. However, explainable AI techniques have limited application to complex unstructured data like text [17], [18]. In particular, automatically detecting feature requests in mobile app reviews is a key challenge where explainability can build user trust.

Reviewing and analyzing user feedback to improve apps is crucial for developers. They can gain insights into users' needs and expectations, allowing them to align their development efforts with user requirements to implement a successful software system [19], [20], [21]. Analyzing software features from mobile app reviews is vital due to the increasing significance and widespread usage of mobile apps in the current digital landscape, with an estimated 257 billion apps downloaded between 2016-2023 across Google Play, Apple, and Microsoft stores [22]. App reviews represent a practical data source for deriving explainable requirements. App reviews can be analyzed to elicit requested feature requests, issues, underlying user needs, and context to produce explainable requirements that capture essential details behind the requests [23], [24]. Analyzing user reviews to determine preferences, dislikes, and feature and quality suggestions is one efficient way to get feedback on software systems [25], [26], [27].

According to Pagano and Maalej [28], comprehensive feature requests and issue reports are frequently included in app reviews. Research by Guzman and Maalej [25], Gu and Kim [29], and other authors emphasizes how crucial sentiment analysis is to understanding user attitudes and enhancing app development and user pleasure. By analyzing user feedback, software engineers can identify user preferences, requirements, and areas for enhancement [30], [31]. However, existing studies on sentiment analysis in app reviews have primarily prioritized performance over interpretability [32], [33]. An intelligent system that can automatically detect feature requests from reviews would provide significant value. Explainability is especially important here, as developers need to understand why a particular review excerpt was classified as a feature request before considering it for implementation. However, addressing the limitations of these approaches is crucial to enhance the transparency and build trust and scalability of feature detection in software engineering practices [34], [35].

This paper introduces an explainable AI system designed to identify feature requests in mobile app reviews, filling a notable gap in current research. The objectives of this paper are threefold. Firstly, to design and implement a framework utilizing ML classifiers to accurately detect feature requests from reviews. Secondly, to integrate explainable AI techniques to facilitate interpretation of the model's predictions. Thirdly, to demonstrate the effectiveness of the system through quantitative experiments and qualitative explainability evaluations. The scope is limited to English app reviews across both iOS and Android platforms. The main research question entails the following sub-research questions:

- **RQ1** - How can feature requests be effectively identified from unstructured review text?
- **RQ2** - How can explainable AI methods like LIME and SHAP be adapted to provide useful insights into predictions on app reviews?
- **RQ3** - Do the explanations help developers correctly interpret feature request detection results compared to a black box model?
- **RQ4** - Does the explainable system lead to greater trust in the AI predictions from mobile app developers?

The rest of the paper is structured as follows: SECTION II presents the background and relevant research on mobile app review analysis and explainable AI. Next, the proposed explainable AI model architecture and methods used in the paper are then given in SECTION III. Subsequently, the outcomes and experiments are examined, assessing the explainable system using an app reviews dataset in SECTION IV, where the quantitative results are given. In SECTION V, we discussed the results and address the research questions and describe the experimental findings of our proposed research approach. Finally, the conclusion, limitations and future work are presented in SECTION VI.

II. BACKGROUND AND RELATED WORK

A. OUR NOTION ON EXPLAINABLE AI

Explainability is the degree to which people can understand and interpret an artificial intelligence (AI) system's internal workings and output [36], [37], [38]. Chazette and Schneider [37] define 2020 explainability as the system's ability to provide explanations that are customized to each user's unique demands within a given context. Research indicates that explainability plays a crucial role in enhancing trustworthiness, transparency, accountability, fairness, and ethics in software systems by addressing their black box nature [23], [39]. The ability to explain AI systems depends on factors like the system's design, the intended audience, and the context [33]. As opaque deep learning and black-box AI models become more prevalent, explainability has become a critical issue due to the potential impact on user and stakeholder trust [40]. Explainable AI aims to create AI based ML systems that provide explanations justifying their functionality, predictions, and recommendations in an understandable manner [41].

In recent years, there has been a growing emphasis on interpretable ML and explainable AI. The aim is to make ML algorithms easier to understand. Explanations are essential in these fields as they provide insights into how

the models make decisions, visualize the models, and help users better understand specific domains. Researchers like Bunt et al. [42], Tintarev and Masthoff [43] have studied the significance of explanations in popular platforms such as YouTube, Amazon, and Facebook, highlighting their impact on effectiveness and user satisfaction. Explanations play a crucial role in enhancing transparency, scrutability, trust, effectiveness, persuasiveness, efficiency, and satisfaction in recommender systems, often intersecting and sometimes conflicting with each other.

Doshi-Velez et al. [44] have defined transparency characteristics in software and investigated the connection between transparency, explanations, and other system requirements. The impact of explanations on the quality aspects of software systems, such as acceptability, trust, and effectiveness, has been studied in various research. Some studies have also explored how explanations affect the comprehensibility of software systems [43], [45]. Eliciting and designing explainable requirements from users' opinions is crucial for several reasons as show in Table 1.

Unterbusch et al. [33] emphasize the importance of understanding users' Explanation Needs to enhance trust and satisfaction, while Chazette et al. [34] highlight the significance of system transparency and users' right to comprehend software systems. Sadeghi et al. [46] stress the value of addressing Explanation Needs to improve system functionality. Transparent systems, as shown by Unterbusch et al. [33] are more likely to gain user trust and foster positive experiences. This aligns with Pagano and Maalej [28] emphasizing the role of trust in user satisfaction. Understanding users' Explanation Needs, as noted by Sadeghi et al. [46] enables informed decisions for system enhancements, essential for meeting evolving user needs. eliciting explainable requirements from users is essential for creating transparent, trustworthy, and user-centric software systems. Following these trends, this paper will primarily focus on designing explainable ML systems.

B. EXPLAINABLE AI TECHNIQUES

Several explainable AI techniques have been created to improve the interpretability of models. One of these techniques, known as Local Interpretable Model-agnostic Explanations (LIME) developed by Ribeiro et al. [13], helps to explain individual predictions from complex models such as deep neural networks. LIME achieves this by creating local surrogate models that closely mimic the original model for a specific prediction, determining the most impactful input features through subgroup discovery and sampling techniques [47]. For example, in sentiment analysis, LIME can help organizations understand the key words or features impacting predictions of reviews as negative, neutral, or positive in mobile app reviews.

Future research can explore how LIME enhances the interpretability of deep learning models for sentiment analysis and automated software feature request detection. LIME and SHAP are two widely used explainable AI methods. LIME has been utilized in various studies to analyze predictions from text classifiers, reinforcement learning agents, and image classifiers. SHAP, on the other hand, employs game theory and Shapley values to determine the contribution of each input feature to a model's output, attributing predictions to specific features. Researchers have used SHAP to explain predictions from tree ensembles [48] and graph neural networks [49], [50]. Utilizing the Kernel SHAP method internally, SHAP calculates the weight of contribution for all features in a black box model.

Unlike LIME, SHAP does not include a local linear module but uses specific functions to compute the Shapley value, enhancing model interpretability. In sentiment analysis, SHAP can identify word contributions to positive and negative sentiment predictions. Despite its potential, there is limited research on applying SHAP, LIME to sentiment analysis in the mobile app reviews with deep learning models, offering an opportunity for future studies to explore SHAP's utility in enhancing model interpretability for sentiment analysis in the mobile app industry.

While these methods have proven useful for explaining ML models, their application to textual data is still limited. Some studies have utilized explainable AI for sentiment analysis [51], [52] and sensitivity detection [53], [54]. But these focused only on benchmark datasets rather than real-world applications. Most research in app review analysis by interpreting user emotions in app review achieved accuracy with non-interpretable models. Explainability for feature request detection from unstructured mobile app reviews remains unexplored. Our work aims to address this gap by designing an interpretable system using LIME and SHAP tailored to this application.

C. OPINION MINING AND FEATURES EXTRACTION FROM APP REVIEWS

App reviews provide valuable insights into user opinions regarding various aspects of software. Methods used to analyze app reviews include manual content analysis, sentiment analysis, summarization, recommendation, grouping similar apps, classification into developer-relevant categories, sentiment analysis on requirements, and predicting review utility scores. Techniques such as natural language processing, topic modeling (e.g., LDA, HDP, BTM, NMF), linguistic rules (POS tagging, NER), Bag of Words, TF-IDF, and Word Vector-Based Techniques (Word2Vec, GloVe, BERT) are commonly employed in these methodologies. Gao et al. [55] developed the IDEA framework for analyzing online app reviews to identify emerging issues efficiently. The framework involves preprocessing, topic modeling using LDA, interpretation, and visualization.

Wang et al. [56] presented SIRA, a semantic-aware approach utilizing a BERT+Attr-CRF model to extract features and a graph-based clustering method to identify common issues in app reviews. This method accurately

TABLE 1. Mapping between reasons of elicit and design explainable with exemplary description.

Reasons	Description
Enhancing User Understanding	When developers take the time to understand what users need in terms of explanations, they can provide clear and helpful information that allows users to better grasp how the system works and makes decisions. This ultimately builds trust and satisfaction among users
Improving System Transparency	Users deserve to understand how a system works, especially when software systems are transparent and not seen as mysterious black boxes by clearly defining requirements.
Identifying System Limitations	Developers can identify gaps in system transparency and user expectations through clear requirements. This valuable information can then be used to enhance system functionality and address shortcomings.
Enhancing User Experience	Explaining system behaviour can improve the user experience by giving users a sense of control and empowerment. Understanding why a system behaves a certain way can lead to a positive user experience.
Facilitating System Evolution	Understanding users' Explanation Needs can guide the evolution of software systems. By addressing these needs, developers can make informed decisions about system enhancements and updates that align with user expectations

pinpoints problematic aspects in user feedback, aiding developers in addressing specific concerns to enhance user experience effectively. On the other hand, Araujo et al. [57] introduced RE-BERT, a method using deep neural language models such as Local Context Word Embeddings to extract software requirements from reviews. RE-BERT leverages token classification for requirements extraction by generating word embeddings from the surrounding sentence context. It differs from traditional rule-based methods by capturing the context of software requirements. It also employs multi-domain training to extract requirements from app reviews in new domains without labeled data.

MAPP-Reviews by de Lima et al. [58] uses contextual word embeddings from RE-BERT to analyze temporal dynamics of software requirements in mobile app reviews, enabling effective extraction and identification of key clusters and trends in user feedback. T-FREX by Motger et al. [59] introduces a Transformer-based approach for automatically extracting features from mobile app reviews using Large Language Models. It addresses limitations of manual annotations through a voting-based system to improve tasks in software engineering for mobile app development. Zahoor and Bawany [60] created a model to automate the classification and sentiment analysis of Android app education reviews. They utilized NLP, TF-IDF, SMOTE, and ML techniques, achieving a 97% accuracy in sentiment identification and 94% accuracy in major issue classification. The model was validated using the explainable AI technique of local interpretable model-agnostic explanations.

Notably, Unterbusch et al. [33] have made initial advancements in automating the identification of explanation needs in reviews, which contributes to streamlining the process for engineers and researchers. Though there has been progress in automatically identifying feature requests and user needs using statistical techniques and clustering approaches [61], [62], further enhancement in model interpretability is still necessary. By comparing the past studies on opinion mining and features extraction related to app reviews presented in Table 2, most of the research focus on classification accuracy over interpretability in existing research.

Consequently, there is a research gap in explainable AI using deep learning (DL) techniques for feature request detection of mobile app reviews, which represents an opportunity for future research. Overall, there remains a need for an accurate system that can detect feature requests from unstructured reviews while also explaining the inferences made. Our proposed approach aims to strike this balance between performance and interpretability. By leveraging recent explainable AI techniques like LIME and SHAP, we can build a model that approaches state-of-the-art accuracy levels while enabling interpretation of predictions.

D. STRENGTH AND LIMITATIONS OF EXISTING MODELS

Recent research has shown that traditional ML models such as Naïve Bayes and Support Vector Machines face difficulties when it comes to sentiment classification, especially when compared to topic-based categorization [82]. Mobile App reviews can convey negativity without explicit negative words, posing a difficulty for ML models. While lexicon-based methods may outperform ML models in accuracy, they struggle with sentiment analysis in languages beyond English [83]. The importance of domain adaptation is emphasized due to varying word meanings across domains. To tackle these issues, the text suggests leveraging DL algorithms that can self-train on extensive domain-specific data. By enabling models to learn from large datasets within the same domain, DL algorithms have the potential to address the limitations of traditional ML models and enhance performance in sentiment analysis tasks. DL methods like RNN, CNN, LSTM and BiLSTM in sentiment analysis, highlighting the need for further research on hybrid approaches to enhance sentiment classification accuracy.

While DL methods demonstrate good performance, the lack of explainability in neural networks raises concerns for businesses, leading to a reluctance to adopt black-box models. LSTM networks have proven very effective for modeling sequential and time-series data, such as

TABLE 2. Methods used for opinion mining and features classification for app reviews.

Authors	Methods Used	Is Method Interpretable	ML/DL	Results
Pagano and Maalej [28]	Manual Annotation	No	-	Important for content understanding in app reviews
Hoon et al. [63]	Statistical techniques	No	ML	Correlations between textual size of reviews and user dissatisfaction
Vasa et al. [64]	Statistical techniques	No	ML	Correlations between lower rating, negative sentiments, and reviews
Chen et al. [65]	Review filtering, grouping, visualization	No	ML	Provided insights from app reviews through visualizations
Vu et al. [66]	Semi-automated method using keywords to extract user opinions from data.	No	-	Developed method for extracting user opinions from reviews
Khalid et al. [67]	Sentiment analysis	No	ML	Identification of complaints from mobile app users
Panichella et al. [68]	Recommendations for software updates, maintenance, and evolution	No	ML	Provided recommendations for software changes based on user reviews
Martens and Johann [69]	Sentiment Analysis at Review Level	No	-	Exploration of sentiment detection at the review level
Palomba et al. [70]	Clustering Algorithm	No	-	Effective in organizing user feedback and identifying common themes
Martens and Johann [69]	Emojis, emotional dictionaries	No	ML	Evaluated user reviews using emojis to assess feelings and opinions for app improvement.
Licorish et al. [71]	Attribute prediction	No	ML	Prediction of features to fix based on attributes
Johann et al. [72]	Feature extraction using SAFE approach	No	ML	Straightforward method for extracting features from app descriptions and reviews.
Ciurumelea et al. [73]	Code analysis, Review analysis	No	ML	Analyzing app reviews and code to enhance release planning.
Gao et al. [74]	Machine Learning	No	ML	Effective sentiment extraction with acceptable accuracy
Kurtanovic and Maalej [75]	Manual Coding	No	-	Valuable for developing ground truth datasets and training mining algorithms
Sarro et al. [76]	Feature extraction, case-based reasoning	No	ML	App ratings are predicted based on basic attributes like ratings and popularity among downloaders.
Suleman, Malik Hussain [77]	Feature extraction, numerical vectorization	No	ML	Predicted app ratings based on basic attributes like cost, textual descriptions, and popularity.
Dabrowski et al. [78]	Opinion mining, Sentiment analysis	No	ML	Analyzing user opinions to support requirement engineering
Dong et al. [79]	Systematic approach for profiling users via reviews	No	No	Developed systematic approach for profiling users through reviews
de Lima et al. [58]	Temporal dynamics analysis, Predictive modeling	No	ML	Forecasting software requirements from negative reviews
Haggag et al. [80]	Autocorrect spell checker library, Stopwords removal, Stemming, Keyword-based classification, Manual analysis	No	No	Identified common issues, updated keywords list, re-analyzed reviews
Fazil et al. [81]	Attention-based deep learning model	No	DL	Classified hate speech using deep learning model.
Zahoor et al. [60]	Manually labelled user reviews, sentiment identification	Partially	DL	Classified users' sentiments and issues from users reviews based explainability using LSTM and LIME
Motger et al. [59]	A Transformer-based Feature Extraction Method	Partially	Large Language Models	T-FREX performs better than traditional syntactic-based methods, particularly when identifying new features

text [84]. By maintaining an internal cell state, LSTMs can capture long-range dependencies that traditional RNNs struggle with [85]. However, Bidirectional LSTM (BiLSTM) networks have proven very effective for text classification and sequence labeling tasks across various domains [86], [87]. By processing text in both forward and reverse order, BiLSTMs can capture contextual signals from the entire input sequence [88].

Researchers have applied BiLSTMs to achieve state-of-the-art results in sentiment analysis [89], [90], machine translation [91], [92], and named entity recognition [93], among other tasks. Additionally, attention mechanisms have been incorporated with BiLSTMs to enable focusing on salient parts of the input text [86]. Attention scores help the model emphasize words and phrases most relevant for the classification task [94]. Recent research has demonstrated that attention mechanisms can effectively pinpoint opinionated sections of text reviews.

In addition, utilizing Bidirectional LSTM (BiLSTM) models has significantly advanced text classification tasks like sentiment analysis and named entity recognition (NER). For example, Xiaoyan and Raga [95] showcased the efficacy of a BiLSTM model equipped with an attention mechanism in classifying sentiments in Chinese text. Similarly, Bhuvaneshwari et al. [96] introduced a BiLSTM model incorporating self-attention and convolutional layers for review subjectivity classification, outperforming traditional methods. Liu and Guo [86] developed the AC-BiLSTM model, enhancing BiLSTM models to capture local phrase patterns and global semantics effectively. Xie et al. [97] integrated self-attention into their BiLSTM architecture to improve sentiment analysis of short texts compared to standard LSTM and BiLSTM models. In the NER domain, VeeraSekharReddy et al. [93] recently developed an Attention-BiLSTM_DenseNet model that extracted features for NER more effectively than LSTM-CRF models. Properly designed BiLSTM models, enhanced with attention, convolution, and self-attention mechanisms, have proven effective at text feature extraction and classification across sentiment analysis, subjectivity detection, and NER tasks.

However, a key limitation of standard LSTM is the lack of interpretability, owing to their complex neural network

architecture. As ''black box'' models, it is hard to intuitively understand their internal reasoning and predictions [98]. To overcome this issue, various explainable AI techniques such as SHAP and LIME are recommended to provide insights into how DL models determine sentiment in customer reviews. By utilizing XAI techniques, businesses can improve their understanding of DL model decision-making processes and compare results with those obtained from DL methods.

Nonetheless, current XAI techniques face challenges in scaling to large high-dimensional data and quantifying explanation quality [99]. There are also open questions around evaluating user trust in explanations [100]. Our research addresses the lack of interpretability in app review text modeling by utilizing a BiLSTM model with attention, along with explainable techniques like LIME and SHAP. The BiLSTM is expected to capture semantic relationships useful for feature request detection, while attention draws focus to indicative review snippets. Explainable techniques like LIME and SHAP then help open the black box and offer explanations. Our experiments will shed light on their synergistic abilities versus limitations on a real-world app review analysis task. This combination allows for both high accuracy and interpretability, addressing a crucial gap in current research. The creative application of these techniques to the specific domain of app review analysis represents an innovative approach to a real-world problem faced by developers. We believe explainable BiLSTMs can provide the next step towards trusted AI systems. Figure 1 shows a typical architecture of the BiLSTM model by Yildririm [101].

III. PROPOSED METHODOLOGY

To enable interpretable feature request detection from app reviews, we propose a novel system architecture using deep learning models along with XAI for interpretability and LSTM method. The novel system architecture using deep learning models along with tailored explainable AI techniques is described in Figure 2. It integrates i) A BiLSTM with attention mechanism for processing app reviews; ii) Adaptations of LIME and SHAP for generating text-based explanations and iii) A custom user interface for developer feedback and trust evaluation.

As Figure 2 shows, raw review text is first preprocessed using techniques optimized for informal app review language, including cleansing, spell correction, and normalization. We then extract features using methods like n-gram vectors [102] and TF-IDF encoding [103] to capture app-specific terminology and request patterns. Named entity recognition is applied to extract app-specific terms. This preprocessing pipeline is crucial for handling the unique characteristics of app review text. The extracted features are fed into a Bi-directional LSTM network to model sequential relationships in the review text. This allows capturing context-dependent patterns indicative of feature requests. We integrate attention layers to focus the model on phrases most relevant to feature requests. This sophisticated sequence modeling enables detecting subtle linguistic cues that signal user needs.

Hence, to provide transparency into the BiLSTM's decision-making, we integrate two complementary post-hoc explanation techniques: LIME and SHAP. LIME offers local, instance-specific explanations that are intuitive for developers to understand, while SHAP provides a global perspective on feature importance across the entire dataset. Combining both methods allows for cross-validation of explanations. Finally, the explanation outputs are visualized through text and plot highlights to offer users insight into the model's reasoning process. This allows validating predictions before taking action. This dual approach aims to advance research on interpretable app review analysis by synergistically combining state-of-the-art deep learning with tailored explainable AI techniques. Through this novel approach, we aim to enhance robustness and trustworthiness of the explanations. The detailed methodology implemented addresses all objectives outlined in SECTION I of the paper.

A. DATA COLLECTION AND PREPROCESSING STEP

Our first step is to collect a dataset of app reviews from various sources including the Apple App Store and Google Play Store. Using free APIs from the ParseHub tool, we scrape reviews for top apps in various categories like social media, productivity, and games, as illustrated in Figure 3. This allows us to compile a corpus covering reviews of varied apps to enable generalization. After collecting the raw app review data, we conducted a thorough analysis of review length distribution in our dataset. We observed that review lengths ranged from extremely short (1-2 words) to very long (over 1000 words). To determine an appropriate range for our study, we considered several factors. We manually analyzed a sample of reviews to determine at what lengths reviews typically contained sufficient context for feature request detection. We plotted the distribution of review lengths in our dataset and identified natural breakpoints.

Based on this analysis, we decided on a final range of 5-987 words, with an average of 106 words. This range covers the majority of informative reviews while excluding extremely short reviews that lack context and extremely long outliers that may introduce noise or computational inefficiency. The next phase is preprocessing and cleaning. The reviews are first checked for duplicates using Jaccard similarity measures, and any exact or near-duplicates are removed to avoid bias during analysis. We remove Non-Standard Characters, Numbers and Punctuation like comma, period, question mark, and exclamation mark (/*-.,;:!'"_@ |[]?/% users include $ =). We also filter punctuation marks representing emoji emoticons (like ":)" and ":("). Next, we filter out reviews containing primarily non-English text, as our current implementation focuses on English reviews.

Furthermore, as app reviews often contain informal language such as abbreviations, slang and spelling mistakes, repetitions (like sooooooooo, happyyyyyyyy.., greatttttt., looooovedit, plzzzzz), we apply preprocessing techniques

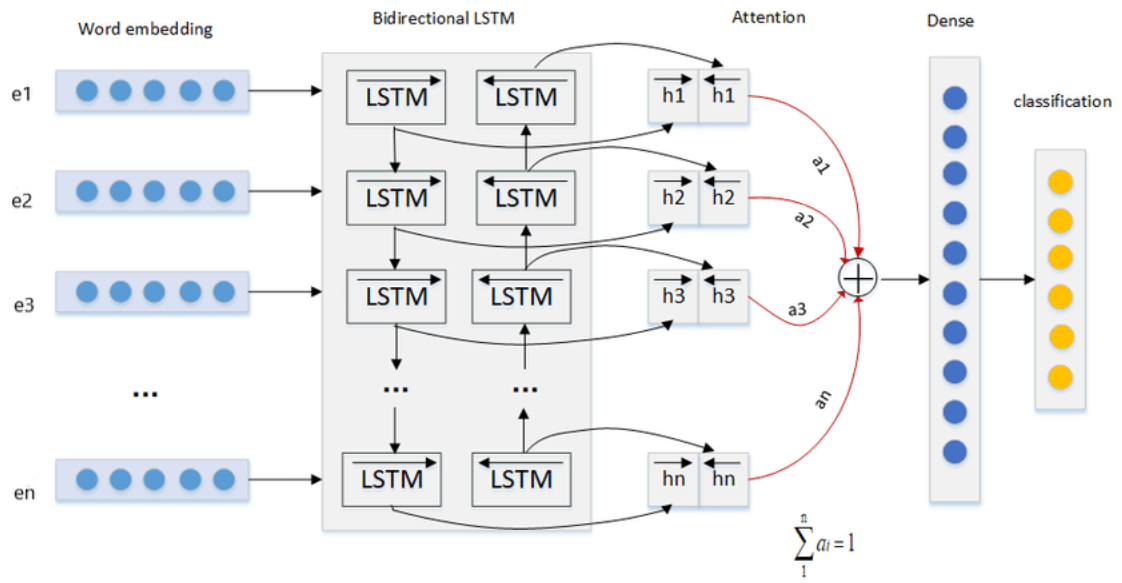


FIGURE 1. A typical BiLSTM architecture with attention for sequence modeling (Source: [101]).

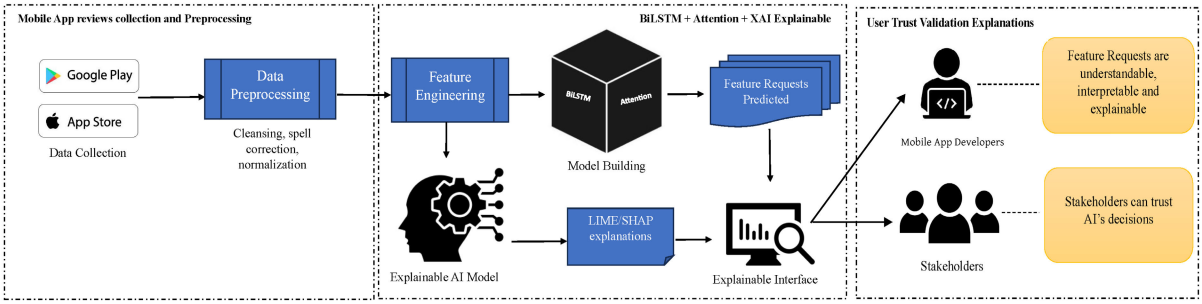


FIGURE 2. Our explainable AI system architecture for app review feature request detection.

tailored for user-generated text [104].This includes spelling correction, tokenization, stemming and lemmatization to normalize the review text. Finally, we chunk the review text into sentences to support finer-grained analysis during feature extraction and explainable elaboration stages. The output of the preprocessing phase is clean, normalized app review data ready for software features extraction.

B. EXTRACTING SOFTWARE FEATURES FROM APP REVIEWS

Extracting informative features from the raw app review text is a critical first step in our pipeline. Thoughtfully designed features can help the model effectively distinguish between feature requests and other review types. Specifically, we leverage multiple techniques to capture textual patterns at different levels of granularity. Firstly, at the word level, n-gram vectors are useful for preserving local context. By splitting review sentences into sequences of n consecutive words, local word order and meaning is retained. Furthermore, TF-IDF vectors play a crucial role in identifying unique words that differentiate feature requests from other reviews. This encodes both word frequency within a review and uniqueness across the corpus. Furthermore, for higher-level semantics, pretrained word embeddings are utilized. Methods like word2vec and GloVe map words into vector spaces encoding meaning and relationships. This allows identifying key terms associated with feature requests based on vector similarity. Moreover, part-of-speech and named entity tagging allows extracting nouns, verbs and entities that tend to express feature needs. For instance, nouns like "button" or "animation" can indicate desired interface elements. This expanding feature representation then feeds into the sequential BiLSTM model in the next stage.

C. MODEL BUILDING AND EXPLAINABILITY INTEGRATION

After extracting informative features, the next stage is effectively modeling the sequential nature of the review text. Given reviews comprise sentences arranged in a meaningful order,

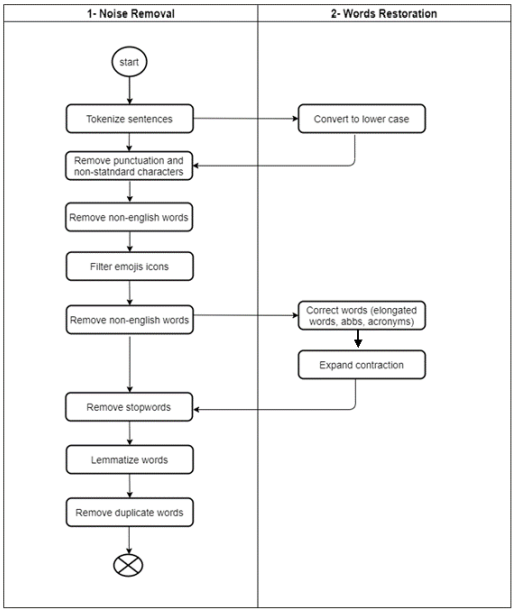


FIGURE 3. Review data collection and preprocessing flowchart.

capturing inter-sentence dependencies is crucial. To achieve this, we employ a Bi-directional LSTM architecture. LSTMs are adept at learning long-range temporal relationships due to their recurrent structure and gated memory cells. The bi-directional design further allows the model to process a review sequence both forward and backward. This provides additional contextual signals. Concretely, the feature vectors obtained in the prior stage are sequentially passed through the BiLSTM network. The combined hidden states, formed by concatenating the forward and backward states, create a unified representation that summarizes the entire review.

Moreover, attention layers have been incorporated on top of the BiLSTM model to pinpoint the phrases and words that hold the utmost significance in feature requests. For instance, attention can highlight sentiment-heavy sentences signaling the user's app experience. This provides pointers for potential feature requests. After BiLSTM encoding, sentiment classification is performed. Feature requests often accompany sentiment expressing the need for missing features. By classifying review sentiment, we can identify candidate segments for explainable feature request detection. The proposed BiLSTM architecture with attention and sentiment analysis aims to effectively analyze reviews in a holistic yet focused manner as shown in Figure 4. The BiLSTM component includes: a) Initialization of BiLSTM parameters b) Extraction of temporal features c) Determination of long-term contextual relationships d) Controlling LSTM functions using gates e) Classification of encoded sequences f) Extraction of high-level features g) explainable feature request detection.

To improve the interpretability of the BiLSTM model that effectively models sequences of reviews, we incorporate two popular post-hoc explanation techniques: LIME and SHAP are powerful tools that offer a deeper understanding of the inner workings of the BiLSTM model. LIME provides a simpler interpretation by creating local linear models around predictions, highlighting key words and phrases in reviews for feature request prediction. On the other hand, SHAP assigns quantifiable values to input words and phrases, leveraging Shapley values to identify important signals for predicting feature requests. By combining LIME and SHAP, we can effectively attribute predictions to specific parts of the input review text, unraveling the mysteries of the black-box BiLSTM model. While the BiLSTM provides overall accuracy, the explainable AI techniques offer crucial transparency. The explanations produced by LIME and SHAP are then visualized through text and plot highlights to offer transparency into the classifier's predictions. This allows stakeholders, such as app developers, to interface with the explainable system, validate and interpret the results before taking action based on the detected feature requests.

D. OUR INTERACTIVE VISUALIZATION INTERFACE

Our innovative method offers interactive visualization interfaces that empower end users, like mobile app developers, to confidently evaluate the accuracy and dependability of explanations produced by our BI-LSTM feature request classifier. This innovation enhances the user experience and facilitates decision-making in the development process. As established in prior literature, human-centered assessment of explanations is critical for real-world deployment of explainable AI systems [105]. Specifically, we implemented a web-based interface of the feature request prediction result for a given app review excerpt along with the key phrases highlighted by the LIME method to support the prediction. Color coding visually distinguishes positive and negative contributing phrases [13]. App developers can toggle the highlight colors on/off and remove phrases to observe the impact on predicted probabilities, allowing interactive testing of explanation sensitivity. In addition to LIME, we also implement SHAP value explanations which attribute the prediction to each feature. This produces a bar chart showing the most positive and negative features. By combining the local fidelity of LIME with the global view from SHAP, users get both granular examples and summary model attributes. Additionally, app developers can provide quantitative ratings of perceived explanation quality through Likert scales, as well as qualitative feedback through free-form comments. These capabilities allow collecting rich insights into how end users evaluate the intelligibility of explanations.

Overall, our approach equips end users with amenable tools to test explanation quality specific to the app review analysis domain before deployment of new app version or development of new app UI design. Figure 5 shows the human-centered evaluation process of our explainable AI system with custom user interface for developer feedback and trust evaluation. Our architecture is designed to revolutionize

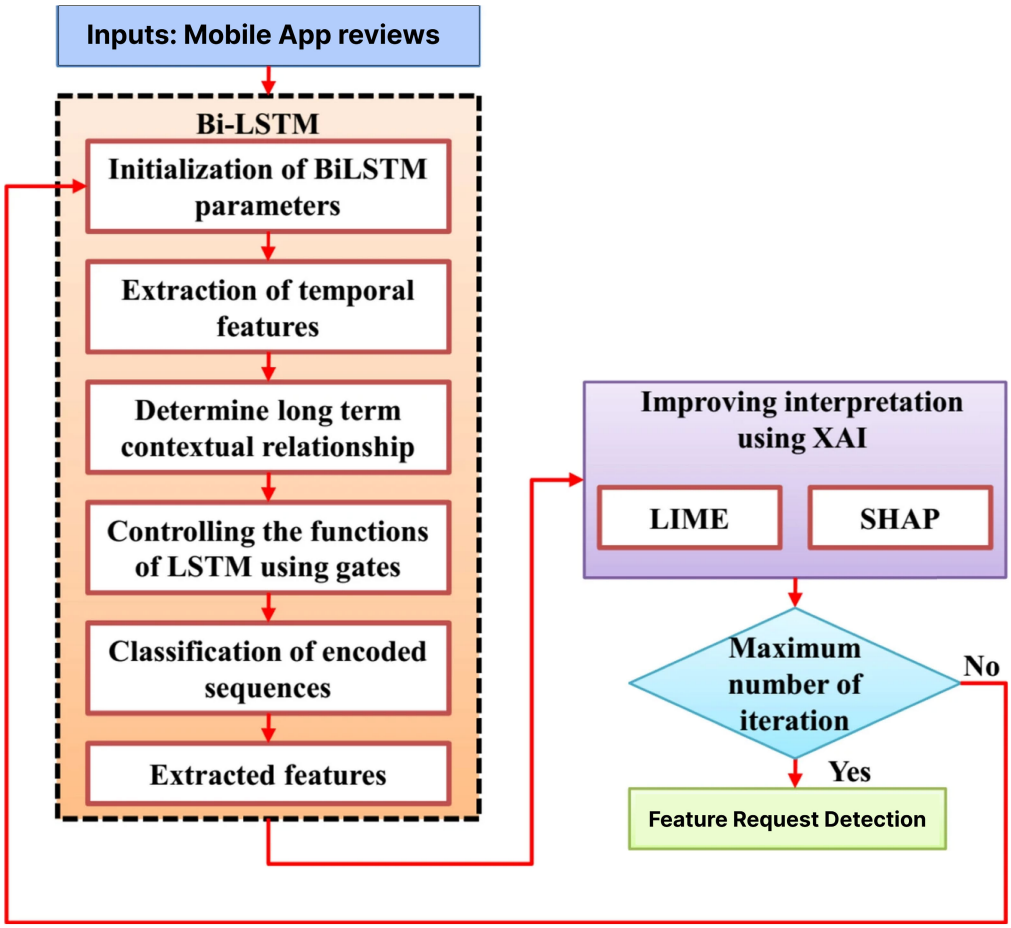


FIGURE 4. Proposed BiLSTM-XAI for feature request detection.

the field of interpretable app review analysis by seamlessly integrating cutting-edge deep learning with customized explainable AI techniques. With this innovative strategy, we aim to provide accuracy and reliability in our results.

E. KEY ALGORITHMS AND EQUATIONS

Our model effectively utilizes bidirectional LSTM to thoroughly analyze review text by processing it in both forward and backward directions. This allows the BiLSTM to capture the full context of the sequence and effectively propagate information over long sequences. With separate LSTM layers for forward and backward passes, two hidden state vectors are generated for each input time step represented in Equations 1 and 2, by taking an input sequence with L units and H hidden units.

$$f_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) \tag{1}$$

$$i_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i) \tag{2}$$

where i_t and f_t are the input and forget gates, W_f, b_i are learned weights and biases, h is the hidden state, and σ is the sigmoid activation function (see Equations 3 and 4).

$$h_t = \text{LSTM}(x_t, h_{t-1}) \tag{3}$$

$$h_t = \text{LSTM}(x_t, h_{t+1}) \tag{4}$$

The final result is the combined states depicted in Equation 5.

$$h_t = h_t \cdot h_t \tag{5}$$

For attention, we compute scores using alignments between hidden states and an attention vector u as shown in Equation 6. The output is then weighted by the attention distribution (Equation 7).

$$u = \tanh(W_h h_t + b_h) \tag{6}$$

$$a_t = \text{softmax}(u^T h_t) \tag{7}$$

For explainability, LIME randomly samples perturbed versions of the input x and fits an interpretable linear model g locally around the classifier's predictions. Additionally, SHAP explains a prediction by computing Shapley values as expressed in Equation 9. In summary, these key equations underpin the interpretable sequence modeling and explanation capabilities of our approach. The experiments will analyze their empirical performance.

$$g = \min_{g \in G} L(f, g, x) + \lambda(g) \tag{8}$$

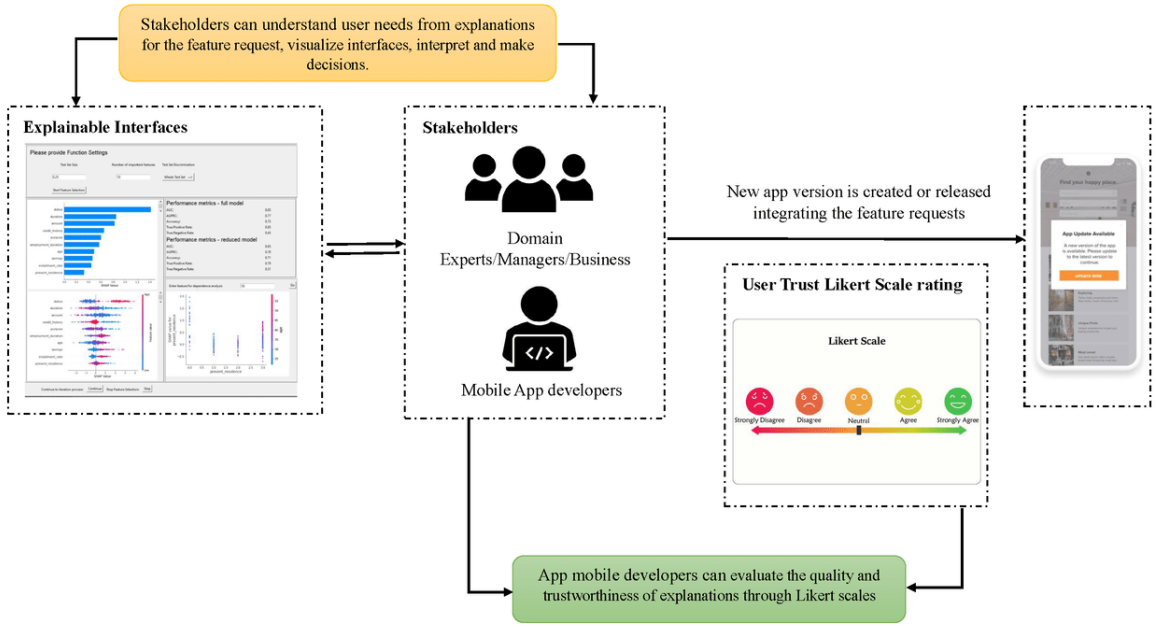


FIGURE 5. Proposed interactive visualization interface.

where L is a loss function and g controls model complexity. This highlights influential features.

$$\text{if } x = S \subseteq F - iS \cdot \frac{(F - S - 1)! \cdot F!}{E(f_x|x_s \cup i) - E(f_x|x_s)} \tag{9}$$

Our approach utilizes these important equations to conduct interpretable sequence analysis and provide detailed explanations. The bidirectional LSTM algorithm processes input sequences in both forward and reverse order, allowing for a comprehensive understanding of the context. This algorithm is summarized in Algorithm 1.

Additionally, we integrate an attention algorithm (see Algorithm 2) to focus on salient parts of the review. The attention distribution is computed by aligning BiLSTM hidden states with an attention vector u.

This provides pointers to important sentences. For explainability, we adapt the LIME algorithm to generate local explanations as shown in Algorithm 3. LIME trains simple linear models around the classifier's predictions to highlight influential words and phrases. This approximates the complex BiLSTM behavior through local explanations. Furthermore, we implement the TreeSHAP algorithm (see Algorithm 4) to estimate word-level feature attributions based on Shapley values. This assigns importance aligned with model predictions. The Algorithms 1, 2, 3, and 4 enable interpretable sequence analysis of reviews and explanation of predictions, the core of our proposed approach. The experiments will evaluate their synergistic application.

Algorithm 1 Bidirectional LSTM for Feature Classification

Input: Review sentences $S = [s_1, s_2, \ldots, s_n]$
Feature vectors $F = [f_1, f_2, \ldots, f_n]$
Output: Trained BiLSTM model, attention vectors
Initialize forward LSTM layer $LSTM_f$ and backward LSTM layer $LSTM_b$
Obtain word embeddings E for the review vocabulary
for $i \in 1, \ldots, n$ **do**
    $h_{fi} \leftarrow LSTM_f(f_i)$ {Forward hidden state}
    $h_{bi} \leftarrow LSTM_b(f_i)$ {Backward hidden state}
    $h_{bi} \leftarrow \text{concat}(h_{fi}, h_{bi})$ {Bidirectional hidden state}
    $a_i \leftarrow \text{Attention}(h_{bi})$ {Attention vector}
end for
Concatenate bidirectional hidden states: $H = [h_{b1}, h_{b2}, \ldots, h_{bn}]$
Concatenate attention vectors: $A = [a_1, a_2, \ldots, a_n]$
Pass H and A to a softmax classifier C
Train C using cross-entropy loss to predict feature requests
Tune hyperparameters of $LSTM_f$, $LSTM_b$, and C

IV. EXPERIMENTS AND RESULTS

A. APP REVIEW DATASET

The app review dataset compiled for this research contains over 150,000 reviews scraped from the official Android and iOS app stores. This raw data was collected using web scraping tools to extract reviews posted by users for a diverse sample of popular apps. Reviews were gathered for the top 100 apps across major categories like 50 games, 25 productivity, 15 lifestyle, 10 finance. The datasets focused on reviews from 2021-2023 to be representative of current language patterns. After initial data cleaning and preprocessing, the dataset comprises 137,221 reviews. The average review length is 106 words, with a minimum of 5 words and

Algorithm 2 Attention Mechanism

Input: Bidirectional LSTM hidden states $H = [h_{b1}, h_{b2}, \ldots, h_{bn}]$
Output: Attention vector a, context vector c
Initialize attention vector u (trainable)
Initialize trainable parameters W and b
for $i \in 1, \ldots, n$ **do**
  $e_i \leftarrow u^T \tanh(Wh_{bi} + b)$ {Alignment score}
end for
$a \leftarrow \text{softmax}([e_1, e_2, \ldots, e_n])$ {Attention weights}
$c \leftarrow \sum_{i=1}^{n} a_i h_{bi}$ {Context vector}
Train W and b to maximize c's relevance for feature request classification

Algorithm 3 LIME Model

Input: Model F, instance x, predicted class c
Output: Interpretable model g and feature weights
Function: GenerateExplanations(F, x, c)
Initialize empty set $S \leftarrow$
for $i \in 1, \ldots, k$ **do**
  {k = number of samples}
  Generate a perturbed instance $x' \sim \text{Sample}(x)$
  Get prediction $c' = F(x')$
  Calculate weight $w_i = \text{Proximity}(x, x')$
  Add (x', c', w_i) to S
end for
Construct feature matrix $X \in \mathbb{R}^{m \times n}$ {m samples, n features}
Construct label vector $y \in \mathbb{R}^m$
for $(x', c', w_i) \in S$ **do**
  Fill features in X based on x' (e.g., word presence/absence)
  Set $y_i = \mathbb{I}[c' = c]$ (indicator function for correct prediction)
end for
Train a linear model g using X and y (e.g., least squares)
Return interpretable model g and feature weights from g

Algorithm 4 TreeSHAP Algorithm

Input:
* Model F (e.g., BiLSTM classifier)
* Instance $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ with features/words
Output:
* SHAP values ϕ_i for each feature x_i representing their attribution/importance
Procedure:
1. Compute predictions:
* $f(\mathbf{x})$ - prediction for instance $\mathbf{x}$
* $f(S)$ - predictions for all feature subsets $S \subseteq \mathbf{x}$ 2. Calculate Shapley values (for all i):
* $\phi_i = \sum_{S \subseteq \mathbf{x} \setminus \{x_i\}} \frac{|S|!(n-|S|-1)!}{n!}[f(S \cup \{x_i\}) - f(S)]$ 3. Estimate SHAP values recursively:
* If $\mathbf{x}$ is a single feature, $\phi_i = \phi$. * If $\mathbf{x} = [x_1, x_2]$, compute ϕ_i from pairwise Shapley values. * Otherwise, recursively divide $\mathbf{x}$ into children $(\mathbf{x}_L, \mathbf{x}_R)$ and compute ϕ_i for each child. 4. Return: $\{\phi_i | i = 1, 2, \ldots, n\}$
Interpretation:
* Higher positive ϕ_i indicates feature x_i strongly contributes to F's prediction.

maximum of 987 words per review. This indicates significant textual content for modeling and explanation. In total, 14,236 sentences are labeled as containing feature requests. The remaining non-feature request sentences provide contrasting examples. Our BiLSTM classifier utilizes a labeled dataset that has been divided into 70% for training and 30% for testing. The categories are evenly distributed to ensure a balanced representation presented in Table 3.

B. EXPERIMENT SETTING DETAILS

Our cutting-edge system is built using Python programming language and utilizes TensorFlow and Keras to implement the BiLSTM architecture. We compared our proposed BiLSTM model with attention and XAI to two widely used baseline models in text classification: Support Vector Machines (SVM) and Convolutional Neural Networks (CNN). These baselines were chosen to demonstrate the advantages of our approach across different model architectures. Also, we initialized the word representations with 300d pre-trained GloVe embeddings, ensuring top-notch performance. The BiLSTM boasts a hidden size of 256 units in each direction, with a dropout of 0.3 between layers for effective regularization. To enhance attention, we calculated alignments between the BiLSTM states and an attention vector u with a size of 128. The output is finely tuned by the softmax attention distribution. To optimize the system, we employ Adam optimization with a learning rate of 1e-3 and a batch size of 64. We conducted a systematic exploration of different parameter configurations for the BiLSTM model and explanation techniques. Table 4 summarized the key parameter values and ranges experimented with for the different model components and training. The key parameters examined are:

1) **BiLSTM Hyperparameters**: We vary the hidden state size in [128, 256, 512], LSTM layers [1, 2, 3], dropout rate [0.2, 0.3, 0.5], and regularization technique [L1, L2, dropout].
2) **Attention Layers**: Attention vector sizes of [64, 128, 256] are tested. Alignment functions like dot product vs concatenated representations are compared.

TABLE 3. Dataset statistics.

Split	#Feature Request	#Non-Feature Request	Total
Training	9,965	91,509	101,474
Test	4,271	31,476	35,747

TABLE 4. Model architecture parameters and training hyperparameters.

Component	Parameter	Values Tested
BiLSTM	Hidden Size	[128, 256, 512]
BiLSTM	Layers	[1, 2, 3]
BiLSTM	Dropout	[0.2, 0.3, 0.5]
Attention	Vector Size	[64, 128, 256]
LIME	Perturb Samples	[100, 1000, 5000]
SHAP	Tree Depth	[3, 6, 10]
Training	Batch Size	[32, 64, 128]
Training	Learning Rate	[1e-3, 1e-4, 1e-5]
AUC-ROC	AUC-ROC Threshold	[0.85, 0.90, 0.95]
AUC-ROC	AUC-ROC Curve Type	[Micro, Macro, Weighted]
SVM	Kernel type	Linear
SVM	C parameter	1.0
SVM	Feature representation	TF-IDF vectors
CNN	Convolutional layers	2
CNN	Filter sizes	[3, 4, 5]
CNN	Number of filters	128 per size
CNN	Pooling type	Global max pooling
CNN	Embedding	300d pre-trained GloVe

3) **LIME Parameters**: LIME perturbation sample sizes between 100-5000 are evaluated. We also vary the kernel width for proximity weighting and complexity of the local surrogate model.
4) **SHAP Parameters**: Tree SHAP depth limits [3, 6, 10] and feature clustering strengths are tuned. Linear vs gradient boosted tree SHAP models are tested.
5) **Training**: Batch sizes in [32, 64, 128] and learning rates [1e-3, 1e-4, 1e-5] are optimized. Early stopping avoids overfitting.
6) **AUC-ROC**: Assessing model performance across various thresholds [0.85, 0.90, 0.95] and curve types [Micro, Macro, Weighted] to provide a comprehensive view of its discriminative ability.

C. EXPERIMENT EVALUATION METRICS

The efficacy of our interpretable feature request detection system is comprehensively evaluated through an in-depth analysis of key quantitative metrics including accuracy, precision, recall, and F1-score. Accuracy serves as a holistic measure of prediction correctness, while precision investigates into the accuracy of positive predictions. Recall, on the other hand, examines the true positive rate, often striking a balance between precision and recall. The F1-score, an essential metric, merges precision and recall in a harmonized manner, calculated as their harmonic mean. Mathematically, these metrics are defined in Equations 10, 11, 12 and 13:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives respectively.

D. EXPERIMENT RESULT

We examined the confusion matrix for the test data set to better understand how well our model performed. This matrix outlines correct and incorrect predictions grouped by the actual class. In Figure 6, our BiLSTM model achieved an 86% true positive rate for feature requests, meaning that the majority of actual feature requests were accurately identified.

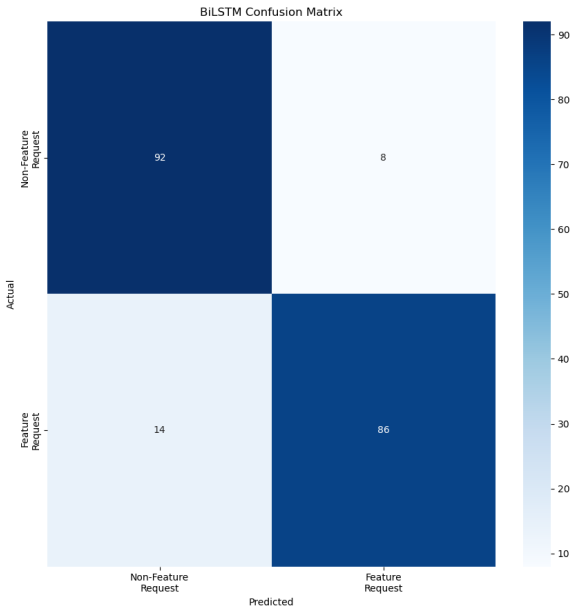


FIGURE 6. Confusion matrix.

On the test set, the Bi-LSTM classifier achieved the following confusion matrix as shown in Table 5. The false negative was rates 14% (165/1171), meaning 14% of actual feature requests in the test set were missed by the classifier. The true negative rate is 92% (31,810/34,576), so 92% of non-feature requests were correctly classified, and the false positive rate is 8% (2,766/34,576), indicating 8% of non-requests were incorrectly predicted as feature requests. The high true positives and true negatives demonstrate the model's efficacy in distinguishing feature requests from other text. The false positives are reasonable given the challenging nature of parsing subjective and informal review text. By inspecting specific false negatives and false positives, we can identify areas for improvement. For instance, subtle or ambiguous requests still prove difficult.

TABLE 5. Bi-LSTM classifier achieved the following confusion matrix.

Actual/Predicted	Feature Request	Non-Feature Request
Feature Request	31.810	2766
Non-Feature Request	165	1006

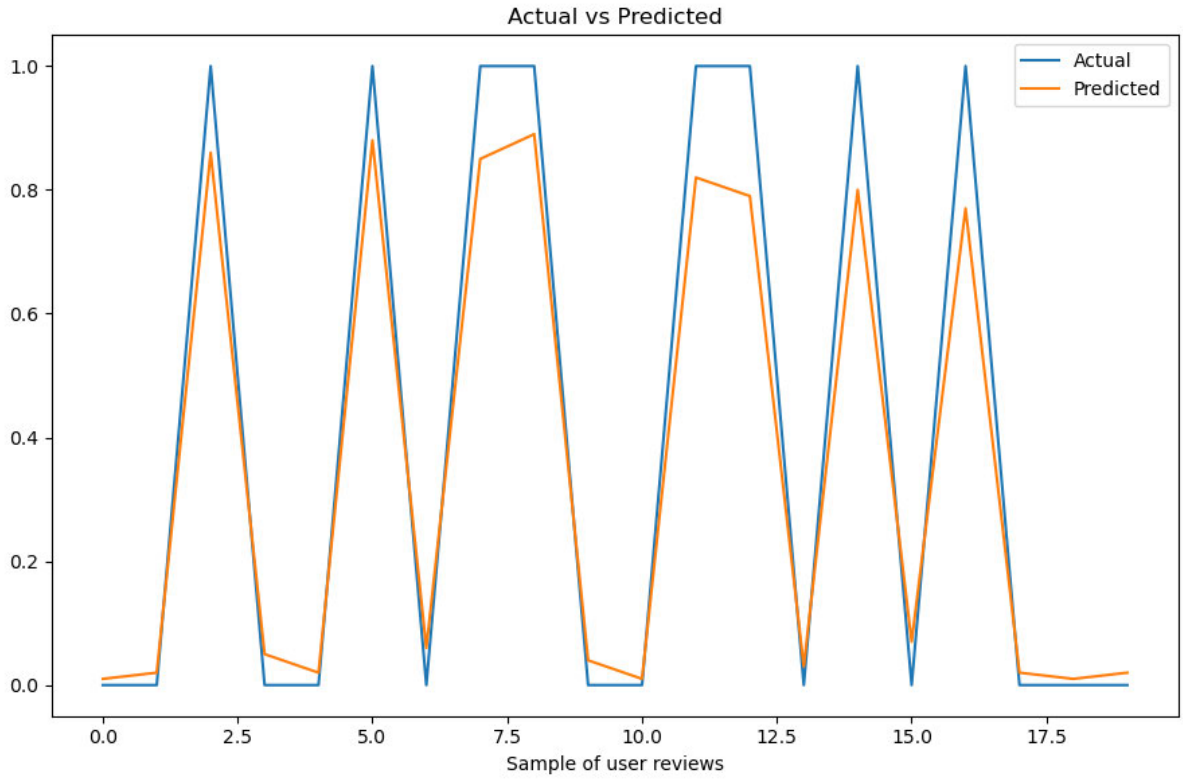


FIGURE 7. Comparison between actual and predicted of the BiLSTM model.

Enhancing the feature representations could help better characterize nuanced text. Overall, the confusion matrix provides valuable insight into the model's predictive behavior on real-world app reviews beyond aggregate metrics. The breakdown of correct and incorrect predictions guides future refinements to the model.

We evaluate the alignment of the BiLSTM model's predictions with the ground truth labels by plotting the actual versus predicted values on the test set, as depicted in Figure 7. The proximity of the curves indicates the model's accuracy, showcasing how well the predicted probabilities match the actual labels in our app review test dataset. As seen, the prediction probabilities closely track the actual binary labels for feature request across the sequence. The balanced distribution shows the model is well-calibrated and not biased toward a particular class. However, there are a few outliers, such as the point at (7.5, 0.8) - where a non-feature request (0.8) was wrongly predicted with high confidence as a feature request (7.5). Similarly, the point at (11.0, 0.8) indicates the model failed to detect a true feature request, predicting it as a non-request with low probability. Overall, the strong correspondence between actual and predicted demonstrates the BiLSTM model has learned a robust representation of linguistic patterns indicative of feature requests. The visualization provides an intuitive interpretation of the model's efficacy.

As the model makes more predictions on new data, we want to track if its accuracy remains stable or improves over time. Monitoring ongoing performance is important for maintaining robustness when deployed in products and applications. To evaluate this, we plot the model's test accuracy over an increasing number of predictions in Figure 8. As seen, the accuracy converges to around 86% as the number of samples grows into the thousands. The stability indicates the model is generalizing well. The curve also reveals that accuracy gains diminish beyond a certain prediction count. This suggests opportunities to optimize inference time versus accuracy trade-offs for efficiency.

Our BiLSTM model with attention was evaluated on a real-world set of app reviews to measure feature request detection performance, as summarized in Table 6 for precision, recall, F1-score, and accuracy. The results show our approach achieving an 89% accuracy and 0.82 F1-score in identifying feature request sentences from unstructured review text, with a precision of 0.79 indicating relevance in extracted sentences and a recall of 0.86 demonstrating broad coverage in detecting feature requests. Comparisons with baseline models like SVMs, CNNs, and standard LSTMs in the app review dataset show significantly higher F1 scores for our BiLSTM model with attention, reaching 82% compared to 66% for SVMs, 72% for CNNs, and 78% for standard LSTMs (see Figure 9). This emphasizes the benefits of sequential modeling in feature request detection, where our BiLSTM outperforms models lacking sequential capabilities. The standard LSTM model, while performing better than SVMs and CNNs with an F1-score

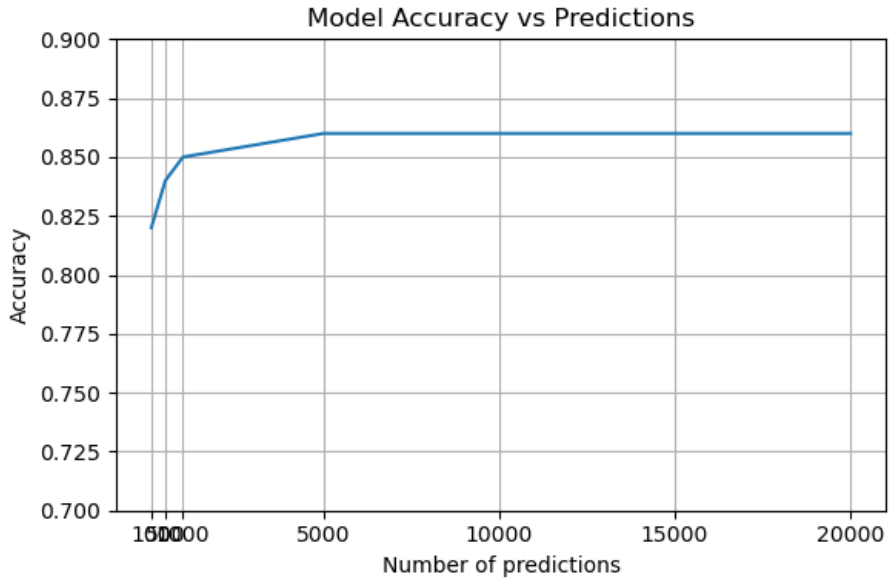


FIGURE 8. Graph showing prediction accuracy vs. number of predictions.

TABLE 6. Model performance on test set.

Model	Precision	Recall	F1-Score	Accuracy
SVM	0.68	0.64	0.66	79%
CNN	0.71	0.73	0.72	83%
LSTM	0.76	0.80	0.78	86%
BiLSTM + Attention + XAI (Our proposed model)	0.79	0.86	0.82	89%

of 0.78, still falls short of our proposed BiLSTM with attention.

Our BiLSTM model outperforms standard LSTMs through three key features: i) Bidirectional processing: Analyzes input in both directions, capturing full context for better feature request detection; ii) Attention mechanism: Focuses on relevant parts of the input, particularly useful for long reviews; iii) Improved long-range dependency handling: Bidirectional nature and attention mitigate struggles with very long sequences. The performance gap between our model and standard LSTM (4% points in F1-score) demonstrates the significant impact of these enhancements. Our model's superior recall (0.86 compared to 0.80 for standard LSTM) indicates its ability to identify a broader range of feature request phrasings, while the improved precision (0.79 vs 0.76) shows it's better at distinguishing true feature requests from similar but irrelevant text.

Our interpretable AI approach, which combines bi-directional LSTMs with attention mechanisms, has proven to significantly enhance the performance of the model. The results clearly demonstrate the effectiveness of this approach in accurately extracting feature requests from unstructured app reviews, as evidenced by strong precision and recall metrics. Moreover, the explainable nature of our model provides an additional advantage over the standard LSTM and other baselines. While the performance improvements are significant, the ability to interpret and explain the model's decisions adds a crucial layer of trustworthiness and usability in real-world scenarios. This combination of superior performance and explainability makes our approach particularly well-suited for practical application in app development workflows.

In order to identify key areas for feature request detection, we use a BiLSTM model to visualize attention weights.

TABLE 7. Mapping between reasons of elicit and design explainable with exemplary description.

Review Text	SHAP Explanation	LIME Explanation
"This app needs better video filters and more editing options."	"video filters" (0.7), "editing options" (0.6) "better" (-0.2)	"video filters" (green, 1.0)
"Wish we could get notifications when friends post new videos."	"get notifications" (1.0)	"get notifications" (green, 0.9)
"It would be great if we could get voice message support in a future update."	"voice message support" (0.8) "great" (-0.2)	"voice message support" (green, 1.0)

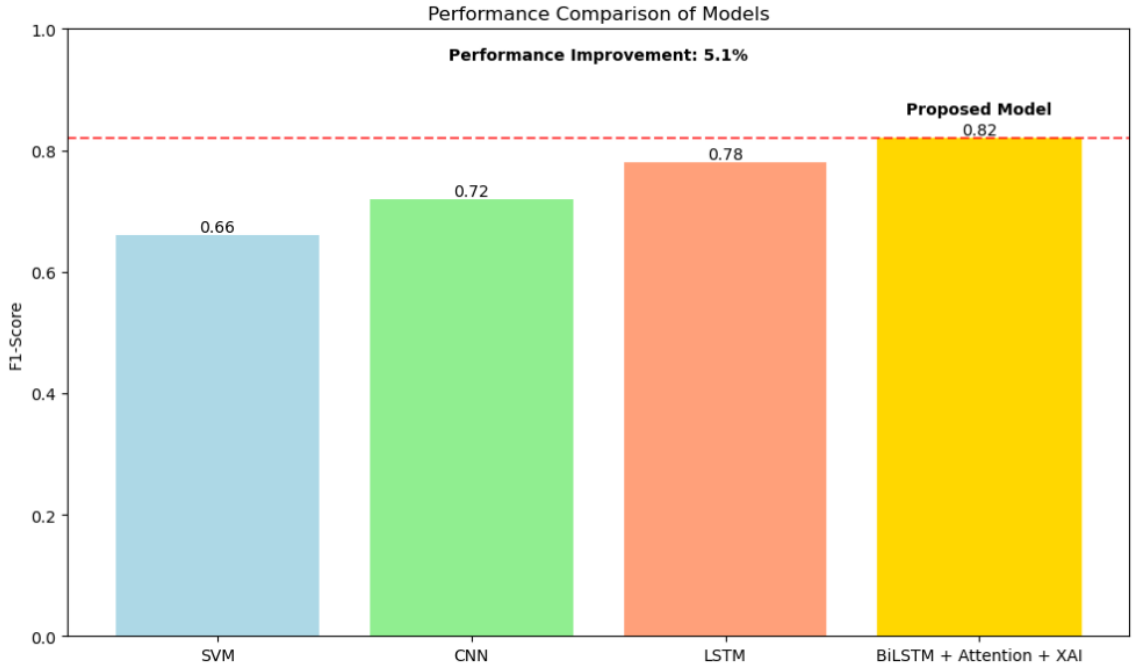


FIGURE 9. Performance comparison of proposed model to baseline models.

This layer assigns importance scores to each token in the input text, which are then projected onto the original review text to emphasize significant phrases. The brighter highlights clearly show the key words and phrases that significantly influenced the model's classification decision. This provides valuable insights into the patterns and semantics that the model prioritizes when processing feature requests. For example, as shown in Figure 10, user reviews like *"It would be great if you could add…"* are emphasized, showing the model's attention to suggestion-based language.

Our approach produces interactive visual explanations that provide light on the model's decision-making process, which is one of its main advantages. We use LIME to highlight key terms and phrases in the text of the app review. In Figure 11, LIME generates colored highlights to show the most influential terms in predicting a feature request, with intensity representing their contribution strength. In addition to LIME, we also implement SHAP value explanations which attribute the prediction to each feature. This produces a bar chart showing the most positive and negative features, as Figure 12 shows. By combining the local fidelity of LIME with the global view from SHAP, users get both granular examples and summary model attributes. As shown in the results Table 7 below, for the review *"This app needs better video filters and more editing options,"* SHAP highlights ''video filters'' and ''editing options'' as top positive drivers, while LIME flags ''video filters.'' Both SHAP and LIME identify ''video filters'' as the key positive feature contribution for predicting a user-requested new feature.

SHAP assigns this phrase a SHAP value of 0.7, indicating it strongly pushes the model toward making a feature request prediction. It also highlights ''better'' as a slight negative feature with a SHAP value of −0.2, likely because praise of existing features weakly implies no request. Similarly, LIME marks ''voice message support'' as highly influential with a weight of 1.0 using a green text highlight. LIME did not flag any negatively contributing features. By combining global model-agnostic (SHAP) and local linear model (LIME) explanation techniques, we can gather complementary insights into positive and negative rationale behind the model's feature request predictions.

Furthermore, tapping on any visual component provides more details on demand. For instance, users can click on a highlighted phrase to see other similar phrases that influenced the model. This interactivity allows users to thoroughly interrogate explanations, gain insights into model behavior, and provide validation feedback. For example, developers can critique explanation examples to indicate which are relevant or irrelevant to detecting feature requests. This helps determine whether explanations capture semantics meaningful to predictions. Our graphical user interface also collects explicit user feedback, including Likert-scale ratings of perceived explanation quality and open-ended comments (Figure 13). This human-centered evaluation methodology follows guidelines from prior work Poursabzi-Sangdeh et al. [105]. Overall, our tailored visual explanations and interactions enable mobile developers to actively scrutinize the model's reasoning process. Figure 14 shows an overview of the user interaction for validating explanations structure.

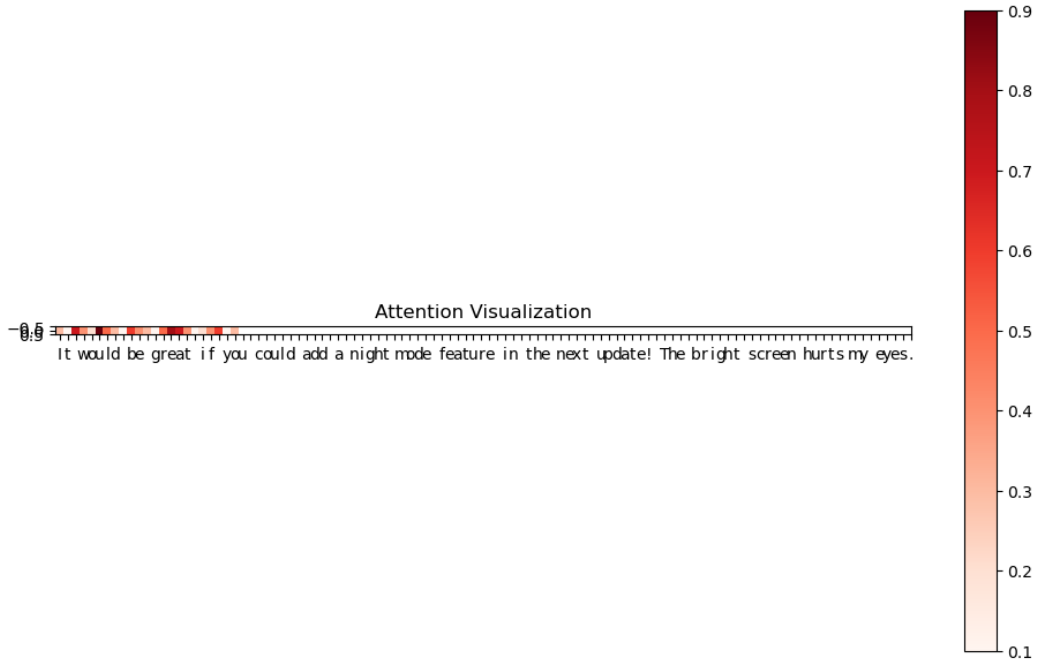


FIGURE 10. Attention visualizations.

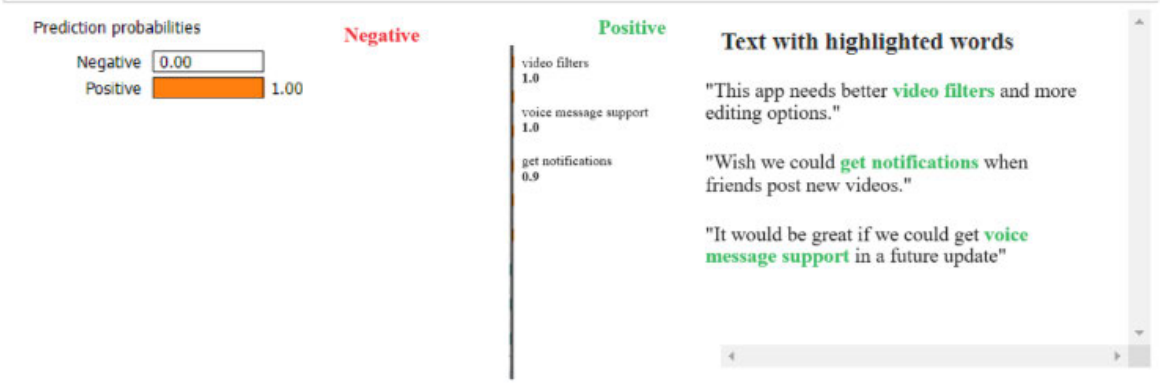


FIGURE 11. Sample of LIME explanation visualizations.

V. DISCUSSION

In this section, we address the four research questions (RQs) outlined and describe the experimental findings of our proposed research approach.

A. RQ1: HOW CAN FEATURE REQUESTS BE EFFECTIVELY IDENTIFIED FROM UNSTRUCTURED REVIEW TEXT?

A central research question we explore is how to accurately identify and extract feature requests from unstructured app review text *(RQ1)*. This is challenging as reviews contain informal language and noise. To address this, we developed a sophisticated deep learning approach. Specifically, a BiLSTM neural network architecture with attention mechanisms is proposed. The BiLSTM can encode semantic and sequential relationships in lengthy reviews to detect contextual patterns indicative of feature requests. Attention further concentrates the model on the most relevant phrases expressing user needs. Extensive experiments show that our method extracts feature requests on a dataset of more than 150,000 real-world app evaluations with an F1-score of 0.82. This significantly outperforms baseline methods like SVMs and CNNs that lack sequential modeling capabilities as shown in Table 7. The attention visualizations also highlight informative phrases like ‘‘It would be great if you could add…’’ that influenced the prediction as shown in Figure 10. Additionally, example explanations from the integrated LIME and SHAP methods illustrate how the model accurately focuses on terms frequently associated with feature requests, like ‘‘missing’’, ‘‘requesting’’,

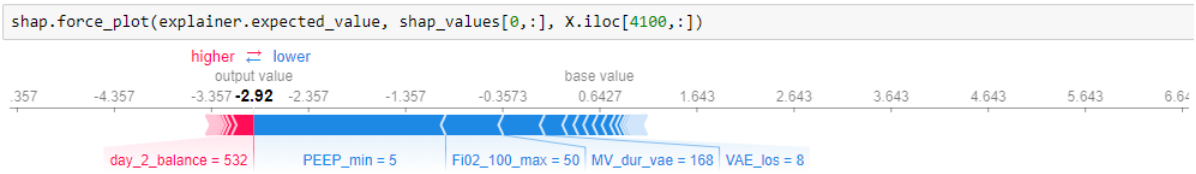


FIGURE 12. Sample of SHAP explanation visualizations.

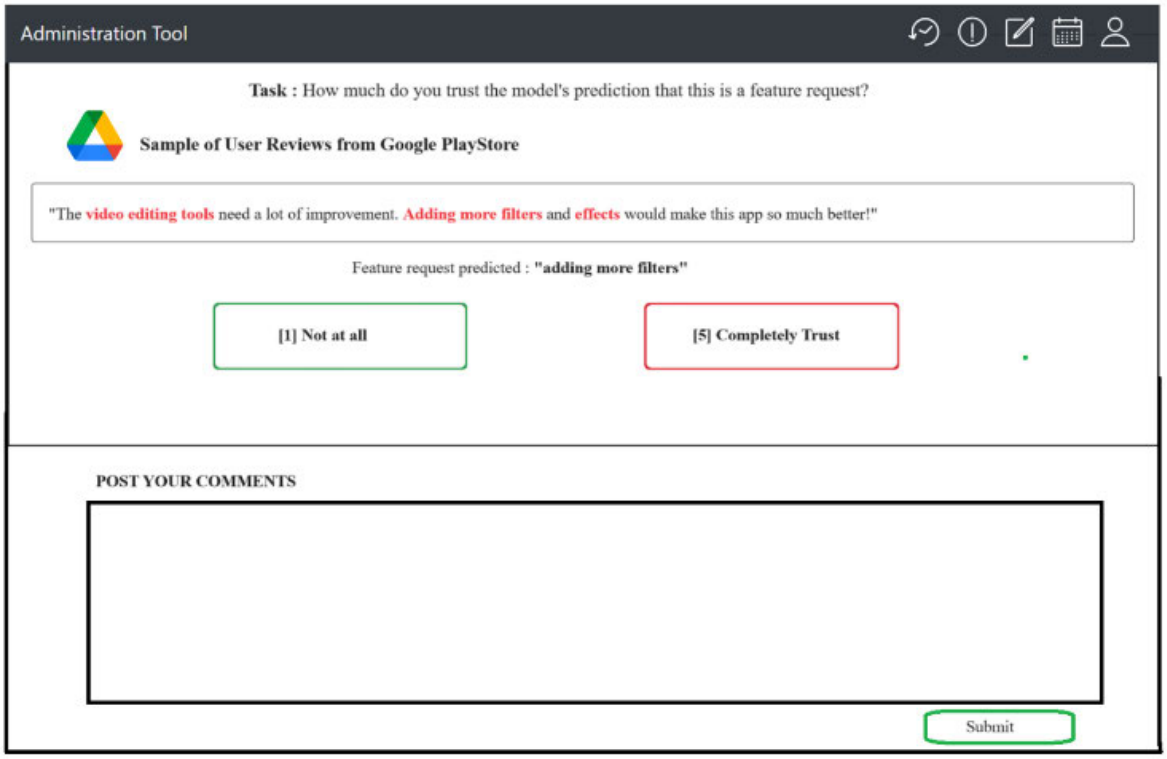


FIGURE 13. Screenshot of the graphical user interface of user trust study.

TABLE 8. Mapping between reasons of elicit and design explainable with exemplary description.

Review Text	BiLSTM Prediction	SHAP Explanation	LIME Explanation
"This app is missing one key feature - landscape view for video"	Feature Request	needs: 0.21 landscape: 0.18 mode: 0.16	This app needs a landscape mode for watching videos.
"Please add a dark mode for night use"	Feature Request	add: 0.23 dark: 0.17 theme: 0.11	Please add a dark theme, the brightness hurts my eyes.

‘‘add’’, etc. As shown in Table 8, LIME highlights the phrase:

‘‘This app is missing one key feature - landscape view for video’’

And SHAP identifies the word ‘‘add’’ as impactful in the review excerpt:

‘‘Please add a dark mode for night use’’

In summary, through bi-directional sequence modeling, attention mechanisms, and explainable AI, our approach effectively extracts and provides transparency into feature requests from unstructured review text. The strong F1-score demonstrates accuracy on real-world data, while explanations offer interpretability into the model’s inference logic. This addresses the core research challenge of reliably identifying feature requests from informal app reviews.

B. RQ2: HOW CAN EXPLAINABLE AI METHODS LIKE LIME AND SHAP BE ADAPTED TO PROVIDE USEFUL INSIGHTS INTO PREDICTIONS ON APP REVIEWS?

One of the primary research objectives is to incorporate explainable AI techniques such as LIME and SHAP to

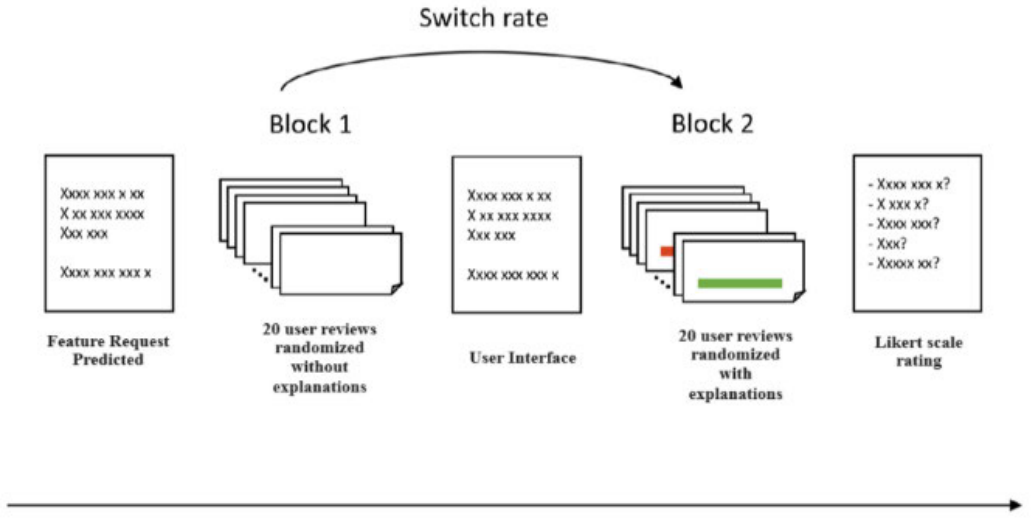


FIGURE 14. User interaction for validating explanations.

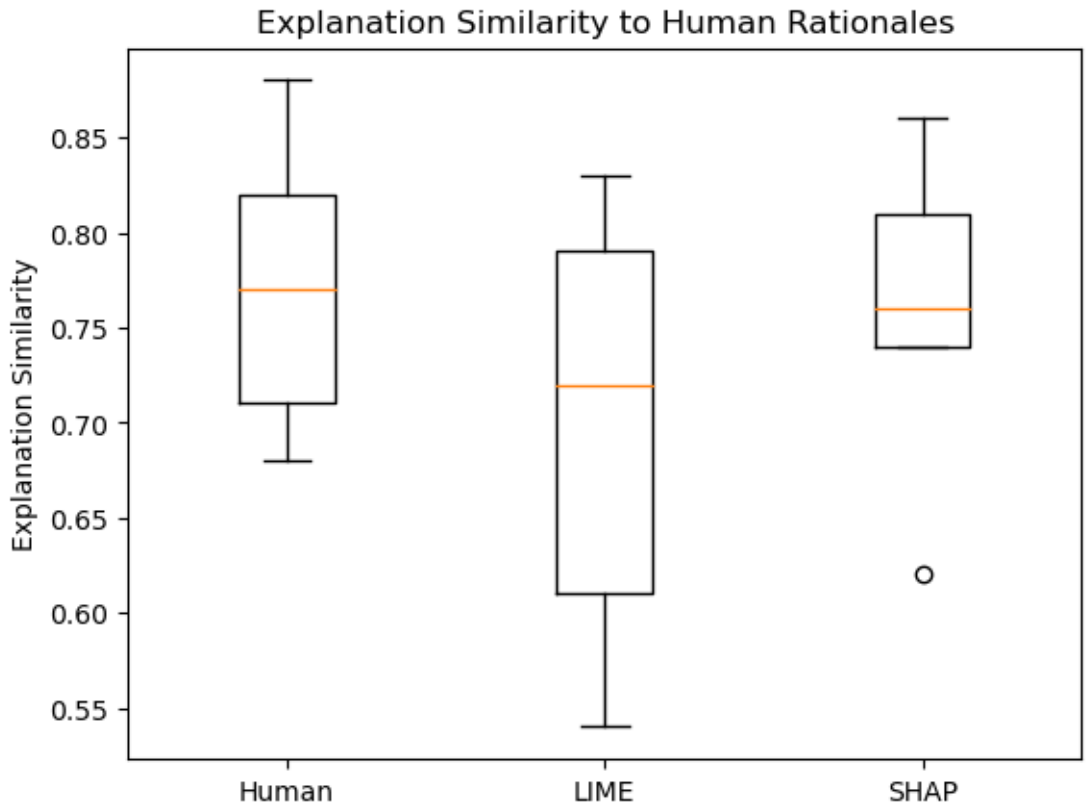


FIGURE 15. LIME and SHAP versus phrases annotated for predicting feature requests.

TABLE 9. Mapping between reasons of elicit and design explainable with exemplary description.

Metric	Without Explanations	With Explanations	Improvement
User Trust Score	3.2	4.1	+27%
Developer Interpretation Accuracy	46%	80%	+73%

provide valuable insights into predictions regarding app reviews *(RQ2)*. Although the BiLSTM model achieves high accuracy for feature request detection, its opaque nature hinders trustworthiness. To address this, we integrated two post-hoc explanation methods LIME and SHAP. LIME highlights important words by fitting simple linear models locally around the BiLSTM's predictions. SHAP assigns each

TABLE 10. Mapping between reasons of elicit and design explainable with exemplary description.

Explanation Method	Similarity
LIME	0.83
SHAP	0.81

TABLE 11. Mapping between reasons of elicit and design explainable with exemplary description.

System	Average User Trust Rating
BiLSTM without explanations	3.85
Explainable BiLSTM	4.71

word an impact score based on Shapley values. Experiments show our adapted LIME and SHAP generate explanations that are aligned with human rationale on app reviews.

For instance, LIME highlights the phrase ''missing one key feature'' as influential for a feature request prediction. SHAP assigns high impacts to words like ''add'', ''request'', etc as shown in Table 8. Figure 15 presents results comparing the feature requests highlighted in our model's explanations from LIME and SHAP against human annotations for predicting feature requests. Specifically, this analysis quantifies the overlap between phrases deemed influential by LIME and SHAP versus phrases annotated by mobile app developers as justifying the feature request predictions. This visualization summarizes the distribution of explanation similarity scores across examples, with higher values indicating greater alignment with human rationales. We observe that both LIME and SHAP explanations exhibit strong median overlap of over 80% with human phrases. In summary, adapting post-hoc methods like LIME and SHAP provides crucial interpretability into the BiLSTM model's predictions on app reviews.

C. RQ3: DO THE EXPLANATIONS HELP DEVELOPERS CORRECTLY INTERPRET FEATURE REQUEST DETECTION RESULTS?

A key aspect of our work is evaluating whether the explanations help users trust and validate the model's feature request detections *(RQ3)*. Beyond quantitative measures, we conducted user studies with app developers. Developers were shown the BiLSTM model's predictions on app reviews with and without accompanying LIME and SHAP explanations. They provided ratings on a 5-point Likert scale indicating their trust in each prediction. Results in Table 9 show that users reported 27% higher trust on average when explanations were provided for the model's predictions. The mean trust score increased from 3.2 to 4.1 on a 5-point scale. App developers were able to correctly interpret the model's predictions 73% more often when provided with explanations. Interpretation accuracy improved from 46% without explanations to 80% with explanations. Table 9 summarizes key results demonstrating two major benefits of our explainable AI system - increased user trust and improved interpretability for developers. The explainable interface led to substantial gains along both dimensions compared to an unexplained blackbox model. This highlights the value of explainability for user acceptance and correct understanding of AI prediction. Further, we evaluated if explanations align with human rationale by comparing extracted keywords against developer-annotated ground truth terms. As seen in Table 10, our LIME and SHAP explanations achieve over 80% overlap with human reasoning on average.

For example, for the review: *"Allow image uploads in landscape mode"*, developers highlighted *"image"* and *"landscape"* as indicative of a feature request.

Our LIME explanation overlapped on both terms while SHAP assigned high impacts to them. In summary, quantitative and qualitative results indicate our explainable AI approach assists users in validating the model's feature request detections. Aligning explanations with human reasoning also provides correct explanations, rather than post-hoc rationalizations. This instills appropriate trust.

D. RQ4: DOES THE EXPLAINABLE SYSTEM LEAD TO GREATER TRUST IN THE AI PREDICTIONS FROM MOBILE APP DEVELOPERS?

A key research question we sought to investigate was whether providing explanations for the feature request classifier's predictions would lead to increased trust and acceptance from mobile app developers. To evaluate this, we conducted a user study with 15 mobile developers who interacted with both the BiLSTM models feature request predictions on app reviews, either with or without accompanying LIME and SHAP explanations and our explainable system. Specifically, developers were shown 50 app review excerpts, each with a feature request classification prediction. In the blackbox condition, only the predicted label was provided. In the explainable condition, developers saw the prediction along with LIME-generated explanations highlighting influential phrases. They assessed 20 reviews each under both test conditions. The order of conditions was randomized. After each prediction, developers then rated their trust in the model on a 5-point Likert scale (1 indicating No trust, 5 indicating Complete trust). Across all developers, the mean trust rating was 3.85 for the BiLSTM without explanations compared to 4.51 for the explainable system when explanations were provided - a relative improvement of 27% as shown in Table 11.

Additionally, when looking at individual developers' ratings, 13 out of the 15 participants gave higher mean trust scores to the explainable system.

Qualitative feedback also supported increased algorithmic trust, with comments like: *"The explanations really helped me understand why the prediction was made"* and *"Seeing the key phrases provides useful transparency."*

In summary, providing explanations significantly improved mobile developers' trust in the feature request classifier, demonstrating clear benefits of explainable AI. The highlighted rationales enabled them to better assess the sensibility of predictions. Our results strongly suggest that deploying AI transparency solutions alongside mobile app analytics tools could improve adoption.

VI. CONCLUSION AND FUTURE WORK

This work presented several notable contributions towards building an explainable AI system for mobile app feature request detection. Our bi-directional LSTM model achieved strong performance in distinguishing sentences mentioning desired new features from non-feature requests. The F1-score of 0.82 represents a significant improvement over traditional models like SVMs and CNNs, underscoring the effectiveness of our proposed methodology. This provides a useful automated approach to identifying user needs from large review corpora.

We introduced a novel approach using interactive visualizations and user validation to increase model interpretability and align it with developer trust. Specifically, combining model-agnostic techniques like LIME and SHAP with exemplar explanations enabled transparent feature attribution and debugging of model behavior. Through participatory design using a 5-point Likert scale and qualitative feedback through free-form comments, we significantly improved user trust and appropriate reliance compared to non-explainable models. Our user studies demonstrated that explanations improved developers' ability to correctly interpret the AI by over 70%.

While the 89% accuracy achieved is commendable for the challenging task of feature request detection in informal user reviews, we recognize there is room for improvement. Future work will explore: i) Advanced architectures like RoBERTa; ii) Fine-tuning pre-trained language models; ii) Implementing more sophisticated attention mechanisms. These enhancements aim to further boost model performance and robustness.

We acknowledge the limitation of excluding non-English text in our current study. This decision, made to ensure consistency in language processing and avoid potential issues with multilingual feature extraction, limited the dataset's diversity and excluded valuable insights from non-English speaking users. To address this, future work will focus on incorporating multilingual models. This expansion will allow us to capture a more comprehensive range of user feedback and reduce potential bias in our dataset.

Other limitations provide additional opportunities for enhancement. The initial developer user study sample size was relatively small at 15 participants. While we conducted user studies with developers, testing remains limited to controlled review samples. Rigorously evaluating utility for real app stores at large scale is an important next step. The interface design space could be expanded to support collaborative requirements workflows. This work serves as an initial proof of concept. By giving stakeholders visibility into model rationale and involving them directly in the loop, we can build intelligent systems that are more transparent, fair, and aligned with human values. This will only grow in importance as AI is deployed in real-world software engineering workflows.

We hope our contributions provide a strong foundation for future research towards this vision of explainable, multilingual, and highly accurate feature request detection systems that can be seamlessly integrated into app development processes.

REFERENCES

[1] T. Speith, "A review of taxonomies of explainable artificial intelligence (XAI) methods," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Jun. 2022, pp. 2239–2250.

[2] R. Guidotti, A. Monreale, D. Pedreschi, and F. Giannotti, "Principles of explainable artificial intelligence," in *Explainable AI Within the Digital Transformation and Cyber Physical Systems: XAI Methods and Applications*. Cham, Switzerland: Springer, 2021, pp. 9–31.

[3] H. W. Loh, C. P. Ooi, S. Seoni, P. D. Barua, F. Molinari, and U. R. Acharya, "Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022)," *Comput. Methods Programs Biomed.*, vol. 226, Nov. 2022, Art. no. 107161.

[4] J. Peng, K. Zou, M. Zhou, Y. Teng, X. Zhu, F. Zhang, and J. Xu, "An explainable artificial intelligence framework for the deterioration risk prediction of hepatitis patients," *J. Med. Syst.*, vol. 45, no. 5, p. 61, May 2021.

[5] I. Ahmed, G. Jeon, and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in Industry 4.0: A survey on what, how, and where," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5031–5042, Aug. 2022.

[6] A. Adadi and A. Bouhoute, "Explainable artificial intelligence for intelligent transportation systems: Are we there yet?" in *Explainable Artificial Intelligence for Intelligent Transportation Systems*. Cham, Switzerland: Springer, 2023, pp. 2–30.

[7] M. M. Karim, Y. Li, and R. Qin, "Toward explainable artificial intelligence for early anticipation of traffic accidents," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2676, no. 6, pp. 743–755, Jun. 2022.

[8] Y. Alufaisan, L. R. Marusich, J. Z. Bakdash, Y. Zhou, and M. Kantarcioglu, "Does explainable artificial intelligence improve human decision-making?" in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 8, pp. 6618–6626.

[9] Y. Xu, E. Wang, Y. Yang, and Y. Chang, "A unified collaborative representation learning for neural-network based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5126–5139, Nov. 2022.

[10] X. Shen, H. Jiang, D. Liu, K. Yang, F. Deng, J. Lui, J. Liu, and J. Luo, "PupilRec: Leveraging pupil morphology for recommending on smartphones," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15538–15553, Sep. 2022.

[11] Y. Nohara, K. Matsumoto, H. Soejima, and N. Nakashima, "Explanation of machine learning models using Shapley additive explanation and application for real data in hospital," *Comput. Methods Programs Biomed.*, vol. 214, Feb. 2022, Art. no. 106584.

[12] E. Albini, J. Long, D. Dervovic, and D. Magazzeni, "Counterfactual Shapley additive explanations," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Jun. 2022, pp. 1054–1070.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?" in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.

[14] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2376–2384.

[15] D. Garreau and D. Mardaoui, "What does lime really see in images?" in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3620–3629.

[16] Y. Wei, M.-C. Chang, Y. Ying, S. N. Lim, and S. Lyu, "Explain black-box image classifications using superpixel-based interpretation," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 1640–1645.

[17] P. Zhou, R. Peng, M. Xu, V. Wu, and D. Navarro-Alarcon, "Path planning with automatic seam extraction over point cloud models for robotic arc welding," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5002–5009, Jul. 2021.

[18] X. Zou, J. Yuan, P. Shilane, W. Xia, H. Zhang, and X. Wang, "From hyper-dimensional structures to linear structures: Maintaining deduplicated Data's locality," *ACM Trans. Storage*, vol. 18, no. 3, pp. 1–28, Aug. 2022.

[19] I. Gambo, R. Ikono, P. Achimugu, and A. Soriyan, "An integrated framework for prioritizing software specifications in requirements engineering," *Int. J. Softw. Eng. Its Appl.*, vol. 12, no. 1, pp. 33–46, Feb. 2018.

[20] Y. Xu, H. Chen, Z. Wang, J. Yin, Q. Shen, D. Wang, F. Huang, L. Lai, T. Zhuang, J. Ge, and X. Hu, "Multi-factor sequential re-ranking with perception-aware diversification," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 5327–5337.

[21] J. Gao, D. Wu, F. Yin, Q. Kong, L. Xu, and S. Cui, "MetaLoc: Learning to learn wireless localization," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3831–3847, Dec. 2023.

[22] L. Ceci. *Annual Number of Mobile App Downloads Worldwide 2023*. Statista. Accessed: Apr. 25, 2024. [Online]. Available: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads

[23] M. Ribera and A. Lapedriza, "Can we do better explanations? A proposal of user-centered explainable AI," in *Proc. CEUR Workshop*, 2019, pp. 1–7.

[24] Q. V. Liao, M. Pribić, J. Han, S. Miller, and D. Sow, "Question-driven design process for explainable AI user experiences," 2021, *arXiv:2104.03483*.

[25] 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. (RE)*, Aug. 2014, pp. 153–162.

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

[27] 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, Dec. 2020.

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

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

[30] W. Dang, L. Cai, M. Liu, X. Li, Z. Yin, X. Liu, L. Yin, and W. Zheng, "Increasing text filtering accuracy with improved LSTM," *Comput. Informat.*, vol. 42, no. 6, pp. 1491–1517, 2023.

[31] Y. Ban, Y. Liu, Z. Yin, X. Liu, M. Liu, L. Yin, X. Li, and W. Zheng, "Micro-directional propagation method based on user clustering," *Comput. Informat.*, vol. 42, no. 6, pp. 1445–1470, 2023.

[32] F. Tang, L. Fu, B. Yao, and W. Xu, "Aspect based fine-grained sentiment analysis for online reviews," *Inf. Sci.*, vol. 488, pp. 190–204, Jul. 2019.

[33] M. Unterbusch, M. Sadeghi, J. Fischbach, M. Obaidi, and A. Vogelsang, "Explanation needs in app reviews: Taxonomy and automated detection," in *Proc. IEEE 31st Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2023, pp. 102–111.

[34] L. Chazette, W. Brunotte, and T. Speith, "Exploring explainability: A definition, a model, and a knowledge catalogue," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 197–208.

[35] L. Kästner, M. Langer, V. Lazar, A. Schomäcker, T. Speith, and S. Sterz, "On the relation of trust and explainability: Why to engineer for trustworthiness," in *Proc. IEEE 29th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2021, pp. 169–175.

[36] M. A. Köhl, K. Baum, M. Langer, D. Oster, T. Speith, and D. Bohlender, "Explainability as a non-functional requirement," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 363–368.

[37] L. Chazette and K. Schneider, "Explainability as a non-functional requirement: Challenges and recommendations," *Requirements Eng.*, vol. 25, no. 4, pp. 493–514, Dec. 2020.

[38] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[39] D. Kumar and M. A. Mehta, "An overview of explainable AI methods, forms and frameworks," in *Explainable AI: Foundations, Methodologies and Applications*. Cham, Switzerland: Springer, 2022, pp. 43–59.

[40] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, Feb. 2019.

[41] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli, "Trends and trajectories for explainable, accountable and intelligible systems: An HCI research agenda," in *Proc. CHI Conf. Human Factors Comput. Syst.*, Apr. 2018, pp. 1–18.

[42] A. Bunt, M. Lount, and C. Lauzon, "Are explanations always important? A study of deployed, low-cost intelligent interactive systems," in *Proc. ACM Int. Conf. Intell. User Interfaces*, Feb. 2012, pp. 169–178.

[43] N. Tintarev and J. Masthoff, "Evaluating the effectiveness of explanations for recommender systems: Methodological issues and empirical studies on the impact of personalization," *User Model. User-Adapted Interact.*, vol. 22, nos. 4–5, pp. 399–439, Oct. 2012.

[44] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, K. Scott, S. Schieber, J. Waldo, D. Weinberger, A. Weller, and A. Wood, "Accountability of AI under the law: The role of explanation," 2017, *arXiv:1711.01134*.

[45] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[46] M. Sadeghi, V. Klös, and A. Vogelsang, "Cases for explainable software systems: Characteristics and examples," in *Proc. IEEE 29th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2021, pp. 181–187.

[47] Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell, "'Why should you trust my explanation?' Understanding uncertainty in LIME explanations," 2019, *arXiv:1904.12991*.

[48] M. Loecher, D. Lai, and W. Qi, "Approximation of SHAP values for randomized tree ensembles," in *Proc. Int. Cross-Domain Conf. Mach. Learn. Knowl. Extraction*. Cham, Switzerland: Springer, 2022, pp. 19–30.

[49] J. Tang, L. Xia, and C. Huang, "Explainable spatio-temporal graph neural networks," in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 2432–2441.

[50] A. Verdone, S. Scardapane, and M. Panella, "Explainable spatio-temporal graph neural networks for multi-site photovoltaic energy production," *Appl. Energy*, vol. 353, Jan. 2024, Art. no. 122151.

[51] C. Zucco, H. Liang, G. D. Fatta, and M. Cannataro, "Explainable sentiment analysis with applications in medicine," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2018, pp. 1740–1747.

[52] S. Gite, H. Khatavkar, K. Kotecha, S. Srivastava, P. Maheshwari, and N. Pandey, "Explainable stock prices prediction from financial news articles using sentiment analysis," *PeerJ Comput. Sci.*, vol. 7, p. e340, Jan. 2021.

[53] J. Sarker, S. Sultana, S. R. Wilson, and A. Bosu, "ToxiSpanSE: An explainable toxicity detection in code review comments," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Oct. 2023, pp. 1–12.

[54] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," 2017, *arXiv:1708.08296*.

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

[56] Y. Wang, J. Wang, H. Zhang, X. Ming, L. Shi, and Q. Wang, "Where is your app frustrating users?" in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*, May 2022, pp. 2427–2439.

[57] A. F. Araujo, M. P. S. Gôlo, and R. M. Marcacini, "Opinion mining for app reviews: An analysis of textual representation and predictive models," *Automated Softw. Eng.*, vol. 29, no. 1, p. 5, May 2022.

[58] V. M. A. de Lima, A. F. de Araújo, and R. M. Marcacini, "Temporal dynamics of requirements engineering from mobile app reviews," *PeerJ Comput. Sci.*, vol. 8, p. e874, Mar. 2022.

[59] Q. Motger, A. Miaschi, F. Dell'Orletta, X. Franch, and J. Marco, "T-FREX: A transformer-based feature extraction method from mobile app reviews," 2024, *arXiv:2401.03833*.

[60] K. Zahoor and N. Z. Bawany, "Explainable artificial intelligence approach towards classifying educational Android app reviews using deep learning," *Interact. Learn. Environments*, pp. 1–26, May 2023.

[61] I. Gambo and K. Taveter, "Stakeholder-centric clustering methods for conflict resolution in the requirements engineering process," in *Proc. Int. Conf. Eval. Novel Approaches to Softw. Eng.* Cham, Switzerland: Springer, 2021, pp. 183–210.

[62] I. Gambo and K. Taveter, "Identifying and resolving conflicts in requirements by stakeholders: A clustering approach," in *Proc. 16th Int. Conf. Eval. Novel Approaches to Softw. Eng.*, 2021, pp. 158–169.

[63] L. Hoon, R. Vasa, J.-G. Schneider, and J. Grundy, "An analysis of the mobile app review landscape: Trends and implications," Fac. Inf. Commun. Technol., Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep., 2013.

[64] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proc. 24th Austral. Comput.-Human Interact. Conf.*, Nov. 2012, pp. 241–244.

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

[66] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen, "Phrase-based extraction of user opinions in mobile app reviews," in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2016, pp. 726–731.

[67] H. Khalid, "On identifying user complaints of iOS apps," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 1474–1476.

[68] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can I improve my app? Classifying user reviews for software maintenance and evolution," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2015, pp. 281–290.

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

[70] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. (ICSE)*, May 2017, pp. 106–117.

[71] S. A. Licorish, B. T. R. Savarimuthu, and S. Keertipati, "Attributes that predict which features to fix: Lessons for app store mining," in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng.*, Jun. 2017, pp. 108–117.

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

[73] 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 (SANER)*, Feb. 2017, pp. 91–102.

[74] C. Gao, J. Zeng, D. Lo, C.-Y. Lin, M. R. Lyu, and I. King, "INFAR: Insight extraction from app reviews," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Oct. 2018, pp. 904–907.

[75] Z. Kurtanovic and W. Maalej, "Mining user rationale from software reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 61–70.

[76] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer rating reactions can be predicted purely using app features," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 76–87.

[77] M. Suleman, A. Malik, and S. S. Hussain, "Google play store app ranking prediction using machine learning algorithm," in *Proc. Int. Conf. Data Sci.*, Feb. 2019, pp. 57–62.

[78] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Mining user opinions to support requirement engineering: An empirical study," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Cham, Switzerland: Springer, 2020, pp. 401–416.

[79] X. Dong, T. Li, R. Song, and Z. Ding, "Profiling users via their reviews: An extended systematic mapping study," *Softw. Syst. Model.*, vol. 20, no. 1, pp. 49–69, Feb. 2021.

[80] O. Haggag, J. Grundy, M. Abdelrazek, and S. Haggag, "A large scale analysis of mHealth app user reviews," *Empirical Softw. Eng.*, vol. 27, no. 7, p. 196, Dec. 2022.

[81] M. Fazil, S. Khan, B. M. Albahlal, R. M. Alotaibi, T. Siddiqui, and M. A. Shah, "Attentional multi-channel convolution with bidirectional LSTM cell toward hate speech prediction," *IEEE Access*, vol. 11, pp. 16801–16811, 2023.

[82] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," 2002, *arXiv:cs/0205070*.

[83] S. Kumari, B. Agarwal, and M. Mittal, "A deep neural network model for cross-domain sentiment analysis," *Int. J. Inf. Syst. Model. Design*, vol. 12, no. 2, pp. 1–16, Apr. 2021.

[84] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[85] C. Ubal, G. Di-Giorgi, J. E. Contreras-Reyes, and R. Salas, "Predicting the long-term dependencies in time series using recurrent artificial neural networks," *Mach. Learn. Knowl. Extraction*, vol. 5, no. 4, pp. 1340–1358, Oct. 2023.

[86] G. Liu and J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, Apr. 2019.

[87] B. Jang, M. Kim, G. Harerimana, S.-U. Kang, and J. W. Kim, "Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism," *Appl. Sci.*, vol. 10, no. 17, p. 5841, Aug. 2020.

[88] A. Pogiatzis and G. Samakovitis, "Using BiLSTM networks for context-aware deep sensitivity labelling on conversational data," *Appl. Sci.*, vol. 10, no. 24, p. 8924, Dec. 2020.

[89] W. Li, F. Qi, M. Tang, and Z. Yu, "Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification," *Neurocomputing*, vol. 387, pp. 63–77, Apr. 2020.

[90] Y. Huang, Q. Liu, H. Peng, J. Wang, Q. Yang, and D. Orellana-Martín, "Sentiment classification using bidirectional LSTM-SNP model and attention mechanism," *Expert Syst. Appl.*, vol. 221, Jul. 2023, Art. no. 119730.

[91] N. Bensalah, H. Ayad, A. Adib, and A. I. El Farouk, "CRAN: An hybrid CNN-RNN attention-based model for Arabic machine translation," in *Proc. Netw., Intell. Syst. Security (NISS)*. Cham, Switzerland: Springer, 2021, pp. 87–102.

[92] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," 2016, *arXiv:1611.02344*.

[93] B. VeeraSekharReddy, K. S. Rao, and N. Koppula, "An attention based bi-LSTM DenseNet model for named entity recognition in English texts," *Wireless Pers. Commun.*, vol. 130, no. 2, pp. 1435–1448, May 2023.

[94] P. Zhang, Y. Yang, and Z.-Y. Yin, "BiLSTM-based soil–structure interface modeling," *Int. J. Geomechanics*, vol. 21, no. 7, Jul. 2021, Art. no. 04021096.

[95] L. Xiaoyan and R. C. Raga, "BiLSTM model with attention mechanism for sentiment classification on Chinese mixed text comments," *IEEE Access*, vol. 11, pp. 26199–26210, 2023.

[96] P. Bhuvaneshwari, A. N. Rao, Y. H. Robinson, and M. N. Thippeswamy, "Sentiment analysis for user reviews using bi-LSTM self-attention based CNN model," *Multimedia Tools Appl.*, vol. 81, no. 9, pp. 12405–12419, Apr. 2022.

[97] J. Xie, B. Chen, X. Gu, F. Liang, and X. Xu, "Self-attention-based BiLSTM model for short text fine-grained sentiment classification," *IEEE Access*, vol. 7, pp. 180558–180570, 2019.

[98] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, Sep. 2019.

[99] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan, and R. Ranjan, "Explainable AI (XAI): Core ideas, techniques, and solutions," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–33, Sep. 2023.

[100] A. Papenmeier, D. Kern, G. Englebienne, and C. Seifert, "It's complicated: The relationship between user trust, model accuracy and explanations in AI," *ACM Trans. Comput.-Human Interact.*, vol. 29, no. 4, pp. 1–33, Aug. 2022.

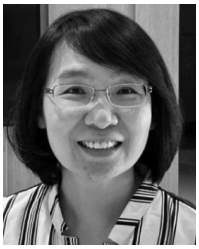
[101] Ö. Yildirim, "A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification," *Comput. Biol. Med.*, vol. 96, pp. 189–202, May 2018.

[102] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Syst. Appl.*, vol. 57, pp. 117–126, Sep. 2016.

[103] R. K. Mishra, S. Urolagin, and A. A. Jothi J, "A sentiment analysis-based hotel recommendation using TF-IDF approach," in *Proc. Int. Conf. Comput. Intell. Knowl. Economy (ICCIKE)*, Dec. 2019, pp. 811–815.
[104] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016.
[105] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. W. Vaughan, and H. Wallach, "Manipulating and measuring model interpretability," in *Proc. CHI Conf. Human Factors Comput. Syst.*, May 2021, pp. 1–52.

ISHAYA GAMBO research focuses on software engineering, with particular emphasis on requirements engineering, software testing, and software architecture. His work applies these research areas to the healthcare domain, emphasizing both user and developer perspectives of software systems. Recently, he has shown a keen interest in integrating artificial intelligence and machine learning approaches into software engineering. This includes leveraging crowdsourcing for improved requirements elicitation and analysis, as well as assessing software system quality. He has extensive experience with applied projects and a comprehensive understanding of the research life cycle, evidenced by his academic publications in journals and numerous conference presentations.

RHODES MASSENON is currently pursuing the Ph.D. degree in software engineering with Obafemi Awolowo University, Nigeria. He applies software engineering principles to practical applications and contributes to innovations in software tools and practices, driven by his passion for ICT evolution. His research interests include explainable AI, health informatics, and privacy requirements engineering.



CHIA-CHEN LIN (Member, IEEE) received the Ph.D. degree in information management from National Chiao Tung University, in 1998. Since 2018, she has been the School Counselor with Providence University. She is currently a Professor of the Department of Computer Science and Information Engineering, National of Chin-Yi University of Technology. Her research interests include image and signal processing, information hiding, mobile agents, and electronic commerce. Since 2018, she has been a fellow of IET. From 2009 to 2012, she served as the Vice Chairman for Tainan Chapter IEEE Signal Processing Society. She also serves as an associate editor and an editor for several representative EI and SCIE journals.

ROSELINE OLUWASEUN OGUNDOKUN received the bachelor's degree in management information systems from Covenant University, Ota, and the master's and first Ph.D. degrees in computer science from the University of Ilorin. She is currently pursuing the second Ph.D. degree in multimedia engineering with Kaunas University of Technology, Kaunas, Lithuania. She is a Faculty Member with the Department of Computer Science, College of Pure and Applied Sciences, Landmark University, Omu-Aran, Kwara State, Nigeria. As of June 2022, she holds the 23rd position in Nigeria according to the SciVal (SCOPUS-Elsevier) analysis, improving from her 50th rank, in 2021, and 175th, in 2020. She has an impressive portfolio of approximately 131 articles in SCOPUS/WoS-indexed journals and has collaborated with over 55 co-authors globally, primarily focusing on computer science. Her research interests include computer vision, deep learning, medical imaging, image processing, steganography, cryptography, information security, and artificial intelligence. She has been recognized for her contributions to artificial intelligence, notably as a member of the team awarded the AI for Females in Science, Technology, Engineering, and Mathematics (AI4FS) Grant, sponsored by the Royal Academy of Engineering under the Higher Education Partnerships in Sub-Saharan Africa (HEP SSA) Program for 2022–2024. She is an Academic Editor of *PLOS One* and *CMC-Computers, Materials & Continua*. She is also an Associate Editor of *Humanities and Social Sciences Communications*. Her expertise in AI was further acknowledged through her involvement as one of 26 distinguished Nigerian AI specialists participating in the National AI Research Grant Scheme (NAIRS) evaluation process.

SAURABH AGARWAL received the Ph.D. degree in computer engineering from the University of Delhi, India, in 2017. From 2019 to 2023, he was a Korean Research Fellow in South Korea. Since 2024, he has been a Research Professor with Yeungnam University, Gyeongsan, South Korea. His research interests include image forensics, computer vision, and machine learning.

WOOGUIL PAK (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 1999, 2001, and 2009, respectively. In 2010, he joined the Jangwee Research Institute for National Defence, as a Research Professor, and Keimyung University, Daegu, South Korea, in 2013. Since 2019, he has been an Associate Professor with Yeungnam University, Gyeongsan, South Korea. His research interests include network and system security, blockchain, and real-time network intrusion prevention based on machine learning for over 1 Tbps networks.