Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.059351

ARTICLE

Hybrid Deep Learning Approach for Automating App Review Classification: Advancing Usability Metrics Classification with an Aspect-Based Sentiment Analysis Framework

Nahed Alsaleh[1,2], Reem Alnanih[1,*] and Nahed Alowidi[1]

[1]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

[2]Department of Computer Science, College of Computer Science and Engineering, University of Hail, Hail, 81451, Saudi Arabia

*Corresponding Author: Reem Alnanih. Email: ralnanih@kau.edu.sa

Received: 05 October 2024 Accepted: 06 December 2024 Published: 03 January 2025

ABSTRACT

App reviews are crucial in influencing user decisions and providing essential feedback for developers to improve their products. Automating the analysis of these reviews is vital for efficient review management. While traditional machine learning (ML) models rely on basic word-based feature extraction, deep learning (DL) methods, enhanced with advanced word embeddings, have shown superior performance. This research introduces a novel aspect-based sentiment analysis (ABSA) framework to classify app reviews based on key non-functional requirements, focusing on usability factors: effectiveness, efficiency, and satisfaction. We propose a hybrid DL model, combining BERT (Bidirectional Encoder Representations from Transformers) with BiLSTM (Bidirectional Long Short-Term Memory) and CNN (Convolutional Neural Networks) layers, to enhance classification accuracy. Comparative analysis against state-of-the-art models demonstrates that our BERT-BiLSTM-CNN model achieves exceptional performance, with precision, recall, F1-score, and accuracy of 96%, 87%, 91%, and 94%, respectively. The significant contributions of this work include a refined ABSA-based relabeling framework, the development of a high-performance classifier, and the comprehensive relabeling of the Instagram App Reviews dataset. These advancements provide valuable insights for software developers to enhance usability and drive user-centric application development.

KEYWORDS

Requirements Engineering (RE); app review analysis; usability metrics; hybrid deep learning; BERT-BiLSTM-CNN

Abbreviation	Definition
ML	Machine Learning
DL	Deep Learning
SRs	Software Requirements
FRs	Functional Requirements
NFRs	Non-Functional Requirements
SRS	Software Requirements Specifications

ABSA	Aspect-Based Sentiment Analysis, used to analyze sentiments associated with specific aspects
BERT	Bidirectional Encoder Representations from Transformers, a language model for NLP tasks
CNN	Convolutional Neural Network, used for extracting local features in text classification
LSTM	Long Short-Term Memory network, a type of RNN that captures sequential dependencies
BiLSTM	Bidirectional Long Short-Term Memory, captures context from both forward and backward directions
BERT-BiLSTM-CNN	Proposed hybrid model combining BERT, BiLSTM, and CNN for comprehensive text classification

1 Introduction

Requirements engineering is a human-centered process in software development that involves eliciting, analyzing, specifying, and validating software requirements (SRs). SRs play a crucial role in guiding the development of a software system and are typically categorized into two main types: Functional requirements (FRs) and non-functional requirements (NFRs). FRs define the software's behavioral aspects, specifying what the system should do. NFRs, on the other hand, cover quality attributes that the system must possess, such as availability, performance, usability, security, scalability, and more [1]. Identifying NFRs early in the development process is essential for accelerating progress and reducing future modification costs. Additionally, prioritizing NFRs ensures the final application meets desired quality standards, resulting in a better user experience [1,2].

Among NFRs, usability holds particular importance. The ISO 9241-11 standard defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [3].

When evaluating software usability, the following key factors are typically assessed:

1. **Effectiveness (Completeness):** This measures a user's ability to accurately and thoroughly complete a task within a given context [3]. Completeness, as a component of effectiveness, ensures the system meets all defined requirements and provides comprehensive functionality to address user needs.
2. **Efficiency (Correctness):** This evaluates the resources required for a user to achieve a goal accurately and completely [4]. Correctness focuses on the internal accuracy and error-free operation of the system, emphasizing resource optimization and waste minimization [3].
3. **Satisfaction (Rating):** This refers to the absence of discomfort and the overall positive experience users have with the product. User ratings serve as a key indicator, reflecting satisfaction with the software's effectiveness and its ability to meet user needs.

Usability plays a pivotal role in determining the quality of software applications. By enhancing user satisfaction, reducing errors and frustrations, and improving both effectiveness and efficiency, usability drives adoption and retention. Organizations that prioritize usability throughout the development process are more likely to deliver high-quality applications that meet user expectations and foster positive user experiences.

App reviews are valuable resources for eliciting user requirements, playing a crucial role in shaping software design and future releases. These reviews provide developers with essential insights into users'

needs and preferences [1]. Incorporating feedback from app reviews is key to engaging new users and improving retention rates. As a result, app reviews have become one of the most important sources of information for developers during the maintenance phase.

Martens et al. categorize reviews into four main maintenance tasks: bug reports, feature requests, user experience, and ratings [5]. Similarly, Guzman et al. identify seven categories of maintenance feedback, including bug reports, feature strengths, feature shortcomings, user requests, praise, complaints, and usage scenarios [6]. Users not only provide star ratings but also offer detailed reviews that include feature requests, bug reports, and their overall experiences with the application [7,8]. By analyzing and classifying these reviews, developers can gain insights that improve software quality and identify gaps in features [9].

Intelligence-based techniques, including machine learning (ML), deep learning (DL), and data mining, have become prevalent in analyzing and classifying app reviews. Much of the research has focused on sentiment analysis, classifying reviews into predefined sentiment categories—typically positive, negative, or neutral [10]. It is also common practice to classify app reviews into SRs [10]. However, recent studies indicate that NFRs have received limited attention and are often inadequately understood or considered during development.

To our knowledge, classifying app reviews based on usability factors—such as effectiveness, efficiency, and satisfaction—using metrics like completeness, correctness, and ratings has not been thoroughly explored. Furthermore, no studies have focused on sentiment analysis-based classification of these usability factors. Using a structured classification approach, such as aspect-based sentiment analysis (ABSA), could help developers, requirements engineers, and product maintenance teams more effectively identify, prioritize, and address usability issues. This approach would ultimately improve product quality and enhance user satisfaction over time.

ABSA is an advanced technique that goes beyond traditional sentiment analysis by focusing on specific aspects or components of an entity. While traditional sentiment analysis provides an overall sentiment polarity for an entire text, ABSA identifies and analyzes the sentiment associated with distinct aspects or features mentioned within the text. ABSA is widely applied across various domains, including social media monitoring, market research, and brand management, where it offers valuable insights into customer opinions and feedback. These insights enable organizations to make data-driven decisions to enhance their products and services.

The ABSA process typically involves two main steps: 1) Aspect Identification: Identifying the specific aspects or features being discussed in the text, and 2) Sentiment Classification: Determining the sentiment polarity (positive or negative) associated with each identified aspect. ML and DL models are commonly employed for ABSA, often trained on labeled datasets that link specific aspects to sentiment polarities.

While app reviews provide rich feedback for developers, existing methods primarily focus on general sentiment analysis or functional requirements, often neglecting non-functional usability metrics critical for user satisfaction. Despite advancements in sentiment analysis, the classification of app reviews based on usability metrics remains underexplored.

This study addresses this gap by leveraging advanced hybrid deep learning models to analyze and classify app reviews effectively. This research introduces a novel ABSA framework for classifying app reviews based on key usability metrics: Effectiveness (measured by completeness), efficiency (measured by correctness), and satisfaction (measured by rating). ABSA allows for a deeper understanding of sentiment by analyzing reviews at the level of specific usability features. This insight is invaluable for businesses aiming to improve their products, manage reputation, tailor marketing campaigns, and make informed decisions based on customer feedback.

The study also compares various state-of-the-art DL models, including bidirectional encoder representations from transformers (BERT) and hybrid architectures combining BERT with recurrent neural networks (RNNs) and convolutional neural networks (CNNs), to determine the most effective approach for ABSA classification of app reviews.

This research aims to answer the following research question (RQ):

RQ: Can DL models, particularly hybrid architectures combining BERT with RNNs and/or CNNs, deliver a high-performing ABSA classifier for app reviews based on usability factors and their corresponding metrics?

The major contributions in this research are as follows:

- **Advanced Usability Insights:** This study introduces the ABSA relabeling framework based on usability metrics, providing valuable insights into user experiences. This framework enables software developers and requirements engineers to make informed decisions to enhance app functionality and user satisfaction.
- **Advancements in Usability Classification:** A high-performance ABSA classifier is proposed by integrating BERT with BiLSTM and CNN layers. This hybrid model outperforms existing methods, offering a powerful tool for developers focusing on user-centric application development.
- **Comprehensive ABSA Dataset Relabeling:** The Instagram App Reviews dataset [11] has been extensively relabeled with aspect labels tied to specific usability metrics and sentiment polarities. This enriched dataset provides new opportunities for research and practical applications in sentiment and usability analysis.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of existing research on NFR classification derived from SRSs and app reviews. Section 3 outlines the methodology, highlighting the key components and the rationale behind the study. Section 4 presents and discusses the results. Finally, Section 5 concludes the paper and offers insights into its limitations and potential directions for future research.

2 Related Work

This section reviews the existing literature on the classification of NFRs derived from SRSs and app reviews.

2.1 Classification of NFRs from SRSs

Kurtanovic et al. [12] developed a supervised ML algorithm using support vector machine (SVM), achieving a precision of 93% and a recall of 90% in identifying NFR classes. A systematic review [13] highlighted both the challenges and benefits of using ML for security-related NFR identification, revealing that supervised learning (SL) is the most widely used ML approach, appearing in 17 studies (71%). SVM was found to be the most popular ML algorithm overall.

In the evaluation of security requirements, one study [14] found high accuracy rates for both supervised and unsupervised ML algorithms. Long short-term memory (LSTM) networks achieved the highest accuracy rate (84%) among unsupervised algorithms, while the Boosted Ensemble algorithm performed best among supervised methods, reaching 80% accuracy. When comparing word embeddings, a study [15] concluded that FastText is a promising model, achieving the highest F1-score of 92.8%.

Another study [16] proposed an enhanced method for classifying NFRs, utilizing a CNN-based multi-label classifier to automatically categorize stakeholder requirements into different NFR classes. This method aimed to minimize misclassification and improve NFR management. In [17], a two-phase DL system was implemented to classify SRs as either FRs or NFRs. The two-phase system demonstrated greater robustness, achieving 95.7% accuracy in the binary classification phase and 93.4% accuracy in the NFR/FR multiclass classification phase.

A different approach was proposed in [18], where a method for automating the extraction of requirement sentences from SRS documents was developed using natural language processing (NLP). The study achieved precision values ranging from 64% to 100% and recall values from 64% to 89%, indicating the effectiveness of the technique. Additionally, a comparative analysis of ML methods in [19] showed that a CNN-based approach achieved a remarkable 99% accuracy on the PROMISE dataset. Haque et al. [20] developed an automated methodology for NFR classification, comparing different feature extraction methods and ML algorithms. Among the classifiers, the SGD SVM achieved the best results, with precision, recall, F1-score, and accuracy of 0.66, 0.61, 0.61, and 0.76, respectively. Notably, the TF-IDF feature extraction technique at the character level outperformed other methods. In [21], Fahmi et al. compared ML techniques for identifying NFRs statements in SRS documents. The SVM approach consistently outperformed others, with an average accuracy rate of 96%. Another study [22] found that using TF-IDF followed by logistic regression (LR) produced the best performance in classifying SRs, with an F-measure of 0.91 for binary classification, tying with SVM. For NFR classification, the F-measure was 0.74, and for general classification, it was 0.78. Shreda et al. [23] analyzed feature extraction methods and classification algorithms for NFRs, concluding that the CNN approach outperformed traditional ML methods, achieving a precision of 92%, compared to 87% for traditional ML techniques. Rahman et al. [24] applied an RNN-LSTM model to classify NFRs, achieving a recall of 71.5%, precision of 71.7%, F1-score of 70%, and accuracy of 71.5%, demonstrating strong performance in NFR classification. Baker et al. [25] utilized a CNN model for NFR classification, achieving high performance with precision ranging from 82% to 94%, recall from 76% to 97%, and F-scores between 82% and 92%.

In [26], Khayashi et al. explored the use of DL methods for classifying SRs using the PURE dataset. While emphasizing the need to enhance the accuracy and efficiency of the sorting process in computer systems, the study did not provide a comprehensive comparison of different techniques, nor did it delve deeply into real-world challenges. The paper introduced a key approach called science-informed deep learning (ScIDL), which integrates scientific knowledge with DL techniques. In [27], the authors compared and evaluated the effectiveness of NLP and ML techniques in software engineering. They concluded that these techniques offer a viable solution for supporting system analysts during requirement elicitation, achieving precision and recall metrics exceeding 95%. A study in [28] proposed a transfer learning approach, where XLNet outperformed other models, reaching an accuracy, precision, recall, and F1-score of 0.91489. The work presented in [29] introduced a supervised categorization approach, obtaining accuracy rates between 85% and 98% in the security, performance, and usability domains. In [30], the authors compared various methods and concluded that multinomial naive Bayes was the most practical option, delivering excellent precision, recall, and execution time. Finally, in [31], the authors explored feature extraction methods and supervised ML algorithms, proposing three combinations that achieved recall and precision values above 0.90, along with fast execution times.

In [32], Rahman et al. proposes an approach to classify NFRs in SRS documents. This approach leverages pre-trained word embedding models to extract relevant features, which are then fed into various neural network architectures, including RPCNN, RPBiLSTM, RPLSTM, and RPANN. The

findings demonstrate that integrating pre-trained GloVe models with RPBiLSTM yields the highest performance, achieving an average AUC score of 96%, along with a precision of 85% and recall of 82%. In [33], Saroth et al. proposed a multi-label classification approach for categorizing NFRs into product, process, and external factors. They develop four ML algorithms—LR, SVM, DT, and K Nearest Neighbor—as well as LSTM model for classification. The results indicate that the LSTM model performs best, achieving an accuracy of 99.69%, making it highly effective for NFR classification.

In [34], Rahman et al. proposed a DL framework for automating the classification of NFRs in SRS documents. The framework aims to address the limitations of traditional ML approaches, which often rely on manual feature extraction. By leveraging DL techniques, the proposed framework automatically extracts features from the text data, leading to improved performance. The results demonstrate that the DReqANN model outperformed alternative models, achieving precision between 81% and 99.8%, recall between 74% and 89%, and F1-scores between 83% and 89%. In [35], García et al. specifically focused on enhancing the classification of NFRs using a CNN. It highlights the significance of preprocessing techniques, sampling strategies, and the use of pre-trained word embedding models such as FastText, GloVe, and Word2Vec. The best performance was achieved with the Word2Vec + CNN combination, which yielded a recall of 0.88, precision of 0.90, and an F1-score of 0.88. Table 1 summarizes the key findings from the preceding studies on classifying NFRs from SRSs.

Table 1: Summary of related work on classifying NFRs from SRSs

Ref.	NFR		Usability attributes			METHOD		Best meth.	RESULT			Accuracy
	Usability	Other	Sat.	Corr.	Comp.	ML	DL		Precision	Recall	F1	
[12]	✓	✓	✓	✓	✓	✓	✗	SVM	93%	90%	—	—
[13]	✗	✓	✗	✗	✗	✓	✗	SVM	—	—	—	71%
[14]	✓	✓	✗	✗	✗	✗	✓	LSTM	—	—	—	84%
[15]	✗	✓	✗	✗	✗	✓	✗	SVM, NB, and LR	—	—	92.8%	—
[16]	✓	✓	✓	✓	✓	✗	✓	CNN	—	—	—	70%
[17]	✓	✓	✗	✗	✗	✗	✓	LSTM, BiLSTM, GRU, and CNN	—	—	—	93.4%
[18]	✓	✓	✗	✗	✗	✓	✗	NLP	64%–100%	64%–89%	—	—
[19]	✗	✓	✗	✗	✗	✗	✓	CNN	—	—	—	94%
[20]	✗	✓	✗	✗	✗	✓	✗	SGD SVM	0.66	0.61	0.61	0.76
[21]	✗	✓	✗	✗	✗	✓	✗	SVM	—	—	—	96%
[22]	✗	✓	✗	✗	✗	✓	✗	SVM	—	—	0.91	—
[23]	✓	✓	✗	✗	✗	✗	✓	CNN	92%	—	—	—
[24]	✓	✓	✗	✗	✓	✗	✓	RNN-LSTM	71.7%	71.5%	70%	71.5%
[25]	✓	✓	✗	✗	✗	✗	✓	CNN	82%–94%	76%–97%	82%–92%	—
[26]	✓	✓	✗	✗	✗	✗	✓	ScIDL	—	—	—	—
[27]	✗	✓	✗	✗	✗	✓	✗	Multinomial NB, SVM, LR, and DT	95%	95%	—	—
[28]	✓	✓	✗	✗	✗	✗	✓	XLNet	0.914	0.914	0.914	0.914
[29]	✓	✓	✗	✗	✗	✓	✗	Supervised categorization approach	—	—	—	85%–98%
[30]	✓	✓	✗	✗	✗	✓	✗	Multinomial Naive Bayes	—	—	—	—
[31]	✓	✓	✗	✗	✗	✓	✗	SVM with TF-IDF, LR with POS and BoW, and MNB with BoW	0.92	0.90	—	—
[32]	✗	✓	✗	✗	✗	✓	✓	GloVe & RPBiLSTM	85%	82%	—	—
[33]	✗	✓	✗	✗	✗	✓	✓	LSTM	—	—	—	99.69%
[34]	✗	✓	✗	✗	✗	✓	✗	DReqANN	81%–99.8%	74%–89%	83%–89%	—
[35]	✗	✓	✗	✗	✗	✓	✓	Word2Vec + CNN	88%	90%	88%	—

Note: Sat: Satisfaction, Corr: Correctness, Comp: Completeness.

2.2 Classification NFRs from App Reviews

In one study, a combination of four classification techniques—bag-of-words (BoW), term frequency-inverse document frequency, Chi-square, and AUR-BoW—was utilized alongside three machine learning algorithms (naive Bayes, J48, and Bagging). The findings indicate that using AUR-BoW with Bagging achieved the highest F-measure of 71.8%, with a precision of 71.4% and a recall of 72.3% [36].

Another research paper [37] proposed a method called AUG-AC, which enhances requirement classification in app reviews by utilizing app changelogs. The experimental results demonstrated that the AUG-AC approach improved the accuracy of requirement classification, achieving a precision of 0.656, a recall of 0.678, and an F-measure of 0.652.

In another study by Aslam et al. [38], a CNN-based approach for classifying app reviews was introduced. This method incorporates both textual and non-textual information, including sentiment analysis and reviewer history. The results showed significant improvements: average recall increased from 69.40% to 93.94%, average precision rose from 75.72% to 95.49%, and the F-measure increased from 72.41% to 94.71%.

Another research effort [39] explored the use of app changelogs to categorize requirements in app reviews, highlighting that SVM is the most widely used technique for NFR classification. The study found that supervised learning outperforms unsupervised learning, achieving over 70% accuracy.

Additionally, a study [40] demonstrated that the SVM algorithm, in combination with TF-IDF feature extraction, exhibited the best performance for classifying Functional Requirements (FRs), achieving precision and recall rates of 0.92 and 0.93, respectively.

A paper employing text mining techniques to extract and classify NFR descriptions into nine categories found that the Naive Bayes classifier provided the most accurate predictions for all types of NFRs except one. AUC (Area Under the Curve) is a performance measurement for classification models at various threshold settings. Specifically, it refers to the area under the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate. The AUC value ranges from 0 to 1: An AUC of 1 indicates perfect model performance, where the model can perfectly distinguish between the positive and negative classes. The highest AUC values were reported for the types “A”, “LF”, “L”, “MN”, “O”, “PE”, “SC”, and “US”, with scores of 0.97, 0.83, 0.97, 0.95, 0.81, 0.86, 0.88, and 0.77, respectively [41].

Dave’s study [42] utilized the TF-IDF technique alongside various machine learning algorithms (SVM, SGD, RF) and NLP methods to accurately identify SRs. The SGD algorithm achieved the highest accuracy of 83%.

A hybrid BERT-RCNN model introduced in another study demonstrated superior classification performance by leveraging BERT for word embeddings and combining BiLSTM and CNN layers. The BERT-RCNN approach achieved precision, recall, and F1-scores of 0.90, 0.87, and 0.88, respectively [43].

In a different study [44], the loop-matching classification technique (LMC) achieved a precision of 74.2%, a recall rate of 82.5%, and an F-measure of 78.1% for categorizing user comments into NFRs.

Moreover, a study [45] presented a shallow ARTIFICIAL NEURAL NETWORK (ANN) for NFR classification, which achieved higher accuracy compared to existing approaches using evolutionary algorithms and decision tree classifiers. The results indicated that the ANN technique achieved a precision of 0.469, a recall rate of 0.471, an F-measure of 0.458, and an accuracy of 0.590.

The authors in [46] addressed the need for extracting NFRs from app reviews in mobile app stores through a two-phase approach. The first phase involved qualitative analysis of 6000 app reviews using binary relevance (BR) classification, incorporating features such as text preprocessing, sentiment scores, and app categories. Results indicated that 40% of the reviews expressed NFRs, with different app categories showing distinct types of NFRs. In the second phase, an optimized dictionary-based multi-label classification approach was developed, achieving an average precision of 70% and an average recall of 86% in identifying NFRs from 1100 reviews. This study demonstrated the effectiveness of capturing NFRs from app reviews and suggested integrating this information into the design and development workflow of mobile applications to enhance overall quality.

A study [47] introduced multi-label active learning as a solution for classifying mobile app reviews. Using the MAREVA algorithm and RF classifier, they achieved a high F-measure score of 0.75. Recent research indicates a growing interest in applying machine learning techniques to categorize NFRs, with SVMs emerging as the most commonly used technique. Supervised learning approaches generally yield better results than unsupervised learning methods, with accuracy rates exceeding 70% [48].

In [49], Yahya et al. focus on detecting and classifying NFRs—specifically usability, reliability, performance, and supportability—in user reviews of mobile apps. The authors propose a hybrid DL model that combines recurrent neural network (RNN) and long short-term memory (LSTM) architectures to effectively analyze textual reviews in Arabic. They conducted experiments using various ML classifiers and DL models, including artificial neural networks (ANN) and bidirectional LSTM. The results indicate that the hybrid model outperformed all other models, achieving a remarkable F1-score of 96%.

In [50], Kaur et al. proposed a transfer learning approach using BERT to classify multi-label app reviews into four NFR categories: Dependability, Performance, Supportability, and Usability. The proposed model outperforms traditional machine learning techniques such as binary relevance and keyword-based approaches. By leveraging BERT's pre-trained language understanding capabilities, the model achieves an F1-score of 0.74, demonstrating its effectiveness in classifying NFRs from user reviews.

In [51], Rahman et al. investigated algorithmic hybridization, a technique that combines multiple machine learning algorithms to enhance performance by leveraging their individual strengths. The study proposes a framework that integrates Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) networks with Artificial Neural Networks (ANN) to classify NFRs into categories, specifically focusing on maintainability, operability, performance, security, and usability. To address the challenge of limited labeled NFR data, the authors compare the performance of their integrated model against standalone LSTM and BiLSTM models. Using two datasets containing 1000 NFRs, the experimental results demonstrate the effectiveness of the proposed hybrid approach. The BiLSTM-ANN model achieved superior performance with precision, recall, and F1-score values of 0.8, 0.78, and 0.78, respectively. Table 2 summarizes the key findings from these studies related to the extraction of NFRs from app reviews.

Table 2: Summary of related work on the classification NFRs from app reviews

Ref.	NFR		Usability attributes			METHOD		Best meth.	RESULT			Accuracy
	Usability	Other	Sat.	Corr.	Comp.	ML	DL		Precision	Recall	F1	
[36]	✓	✓	×	×	×	✓	×	AUR-BoW with Baggin	71.4%	72.3%	71.8%	—
[37]	✓	✓	×	×	×	✓	×	AUG-BoW	0.656	0.678	0.652	—
[38]	×	✓	×	×	×	×	✓	CNN	95.49%	93.94%	94.71%	—
[39]	✓	✓	×	×	×	✓	×	SVM	—	—	—	70%
[40]	✓	✓	×	×	×	✓	×	SVM + TF-IDF	0.92	0.93	0.92	—

(Continued)

Table 2 (continued)

Ref.	NFR		Usability attributes			METHOD		Best meth.	RESULT			Accuracy
	Usability	Other	Sat.	Corr.	Comp.	ML	DL		Precision	Recall	F1	
[41]	✗	✓	✗	✗	✗	✓	✗	AUC	—	—	—	—
[42]	✓	✓	✗	✗	✗	✓	✗	SGD	—	—	—	83%
[43]	✗	✓	✗	✗	✗	✗	✓	BERT-RCNN	0.90	0.87	0.88	—
[44]	✗	✓	✗	✗	✗	✓	✗	LMC	74.2%	82.5%	78.1%	—
[45]	✓	✓	✗	✗	✗	✗	✓	ANN	0.469	0.471	0.458	0.590
[46]	✓	✓	✗	✗	✗	✗	✗	BR + dictionary-based multi-label classification	70%	86%	—	—
[47]	✗	✓	✗	✗	✗	✓	✗	MAREVA + RF	—	—	0.75	76%
[48]	✓	✓	✓	✓	✓	✓	✗	SVM	—	—	—	70%
[49]	✓	✓	✓	✓	✓	✗	✓	SVM	—	—	96%	—
[50]	✓	✓	✓	✓	✓	✓	✓	MNoR-BERT	—	—	0.74%	—
[51]	✗	✓	✗	✗	✗	✗	✓	BiLSTM-ANN	80%	78%	78%	—

Note: Sat: Satisfaction, Corr: Correctness, Comp: Completeness.

2.3 Research Gap

Despite significant progress in classifying app reviews, there remains a notable gap in the literature regarding the classification of app reviews based on usability factors, specifically effectiveness, efficiency, and satisfaction, within the context of ABSA. These usability factors are widely recognized as key contributors to software quality [52]. By effectively classifying app reviews according to these factors, developers can gain valuable insights into user experiences and identify areas for improvement, ultimately enhancing app development. This research addresses this gap by providing the ABSA framework that accurately classifies app reviews based on the identified usability factors: effectiveness (measured by completeness), efficiency (measured by correctness), and satisfaction (measured by rating).

3 Research Methodology

This section outlines the methodology employed for classifying app reviews within the framework of ABSA for usability metrics. Fig. 1 presents a visual representation of this methodology.

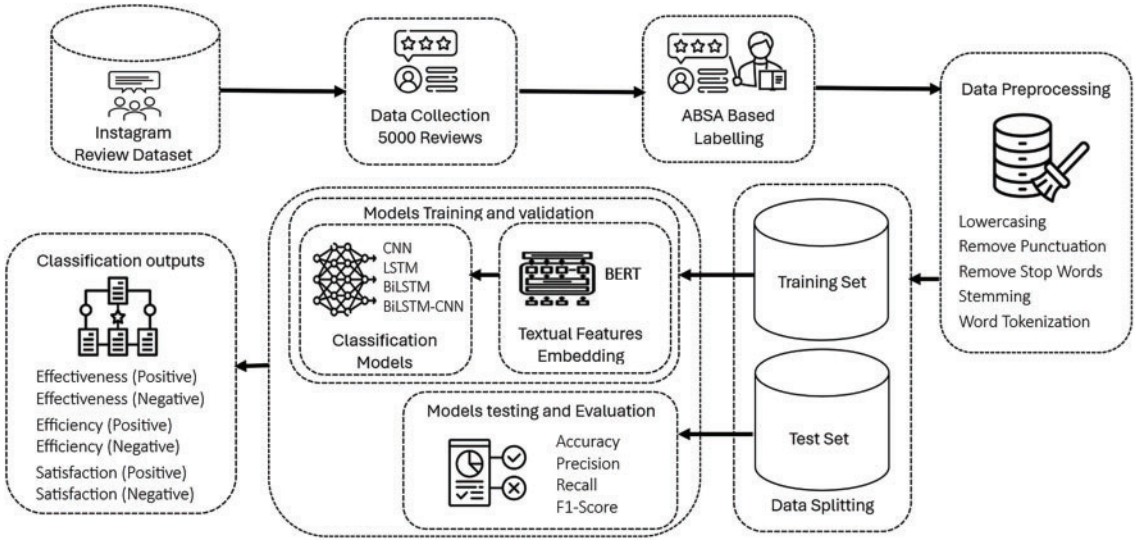

Figure 1: Research methodology

The process begins with the careful collection of an appropriate app review dataset, taking into account both the volume of samples and the variety of review types. Next, the dataset is specifically labeled for ABSA classification. Following this, the data undergoes preprocessing to ensure it is correctly formatted for subsequent analysis. Once the data is thoroughly prepared, the focus shifts to developing and training classification models. These models are rigorously evaluated, with their performance analyzed to identify the most effective model for ABSA classification of usability metrics derived from app reviews. A detailed breakdown of each step is presented in the following subsections.

3.1 Dataset Collection

The Instagram App Reviews dataset was selected for this research due to its comprehensive nature and diversity. It consists of app reviews from the Google Play Store, providing valuable insights into user sentiment and satisfaction.

The dataset spans from 12 September 2018, to 27 July 2023, and includes three columns:

- Reviews: The text content of app reviews.
- Ratings: The star ratings assigned by users.
- Dates: The dates on which the reviews were submitted.

The dataset was collected by scraping reviews from the Google Play Store and encompasses a total of 209,956 reviews, which are publicly available on Kaggle [11]. The latest update was recorded on 29 July 2023. This dataset offers a robust foundation for analyzing user experiences and preferences.

For this research, a subset of 5000 reviews was extracted from the dataset. This sample size was chosen to provide a diverse and representative sample of usability feedback while maintaining computational feasibility. This number ensures the inclusion of sufficient variation in sentiment polarity and usability aspects, facilitating effective model training and evaluation.

Fig. 2 showcases a sample from the Instagram App Reviews dataset before it was labeled according to the ABSA framework.

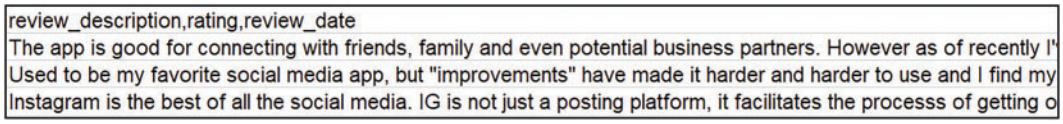

Figure 2: Sample from instagram app reviews dataset before ABSA-based labeling

3.1.1 Dataset Distribution and Visualization

To visualize the dataset's structure, the distribution of key usability metrics—completeness, correctness, and satisfaction—was plotted using a histogram. As shown in Fig. 3, this visual representation illustrates the distribution across these categories, highlighting both positive and negative feedback for each metric. The histogram reveals a balanced distribution within each category, with comparable counts for positive and negative feedback. However, when examining the distribution across categories, there is a noticeable skew towards user satisfaction, with 2594 reviews, followed by correctness with 930 reviews, and completeness with 527 reviews. This distribution reflects real-world user behavior, where users typically prioritize expressing satisfaction, followed by identifying errors, and finally commenting on the app's completeness.

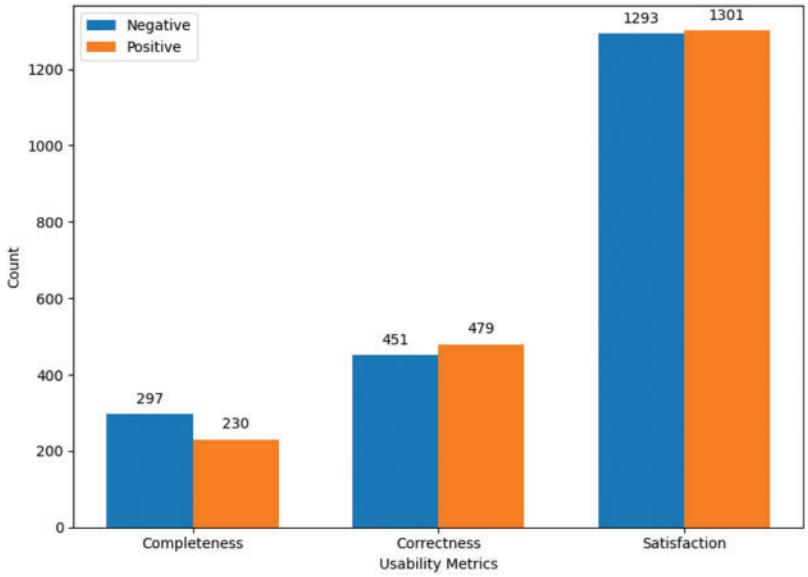

Figure 3: Distribution of usability metrics in the dataset: completeness, correctness, and satisfaction

3.1.2 Dataset Splitting

The dataset was partitioned using a combination of a 70:30 split and 10-fold cross-validation. Initially, the data was randomly divided into a training set (70%) for model training and a test set (30%) for evaluating the model on unseen data. The training set was then further split into 10 equal folds to facilitate cross-validation and identify the optimal model configuration.

This combined approach maximizes data utilization and enhances model robustness by mitigating overfitting, reducing bias, and providing a more accurate estimate of performance. Such techniques are standard practices in machine learning and deep learning to ensure the development of reliable and effective models.

3.2 Dataset Labeling Process

The labeling process was carried out by three experts with backgrounds in software engineering and user experience. Their expertise ensured a comprehensive understanding of usability factors relevant to applications. The experts systematically reviewed each app review to identify key usability aspects and their corresponding sentiment polarities (positive or negative) using the following methodology:

1. Develop Labeling Guidelines: To ensure consistency and reliability, clear and comprehensive guidelines were created. These guidelines specified the criteria for labeling each review and included:

    1.1 Usability Aspects Identification Guidelines: The guidelines detailed how to identify aspects related to key usability factors: Effectiveness (measured by Completeness), Efficiency (measured by Correctness), and Satisfaction (measured by Rating) [52].

    1.2 Sentiment Polarities Assignment Guidelines: Each usability factor was associated with specific sentiment polarities. The guidelines outlined the criteria for determining whether sentiment was positive or negative for each factor.

2. Provide Illustrative Examples: To assist experts in applying the guidelines, illustrative examples for each usability factor with corresponding sentiment polarities were included. These examples served as practical references, ensuring a standardized approach during the labeling process.
3. Solve Inter-Rater Variations: To enhance inter-rater reliability, all reviews were labeled independently by the three reviewers. Any discrepancies were resolved through discussion and voting.

By following this structured and well-defined labeling methodology, the reliability and validity of the dataset labeling process were enhanced.

3.2.1 Guidelines for Identifying Aspects and Sentiments Related to Effectiveness, as Measured by Completeness, in App Review Labeling

Definition: Effectiveness refers to the degree to which an app achieves its intended purpose, with Completeness as a metric indicating whether the app includes all necessary features and functions as expected [53].

Labeling Process

1. *Identify Aspect:*
   - Look for mentions of features, functions, or overall capabilities of the app, specifically addressing whether the review discusses the app's ability to meet user needs in terms of completeness.
   - Consult the guideline rules detailed in Table 3 to understand how to accurately identify the effectiveness factor.
   - Refer to the keywords in Table 4 to aid in the accurate identification of the effectiveness factor.
2. *Sentiment Analysis:*
   - Consult the guideline rules detailed in Table 3 to understand how to classify the sentiments expressed by users regarding the effectiveness factor of the application.
   - Refer to the keywords in Table 4 to assist in the accurate classification of sentiment related to the effectiveness factor.
   - Determine the sentiment expressed towards the app's completeness and assign a sentiment label based on the tone:
     - Positive: The review indicates satisfaction with the completeness of features.
     - Negative: The review indicates dissatisfaction due to missing features or functionality.

Examples of Reviews and Labels

Review: "The app has all the features I need for photo editing; it's very comprehensive!"

Label: Aspect: Effectiveness

Metric: Completeness

Sentiment: Positive

Review: "While the app is helpful, I find it lacking in some areas".

Label: Aspect: Effectiveness

Metric: Completeness

Sentiment: Negative

Table 3: Guideline rules for identifying and categorizing the effectiveness aspect and its associated sentiment from app reviews

Aspect	Sentiments	
	Positive	Negative
Effectiveness (Completeness)	If the review highlights aspects of efficiency, high quality, or excellent service.	If the review points out missing features or functionalities that hinder the app's effectiveness.
	If the review mentions that the app includes all necessary features or functions that meet user needs.	If the review expresses frustration over a lack of options or inadequate functionality.
	If the review mentions that the app effectively assists users in achieving their goals or completing tasks.	If the review reflects issues with accuracy, poor service, or complexity.
		If the review mentions failures in completing tasks, incomplete steps, or low quality.

Table 4: Keywords for identifying and categorizing the effectiveness aspect and its associated sentiment from app reviews

Aspects	Sentiments	
	Positive	Negative
Effectiveness (Completeness)	Enhance, completed tasks, successfully completed, steps complete, productivity, speed, faster, expertise, improvement, quickly, high quality, low cost, more efficient, reduce costs, better serve, easy, small steps, clear, accuracy, effective, task completion, accuracy (performance).	Unenhanced, uncomplete tasks, completion failed, steps incomplete, less productivity, delay, slower, less experience, problems, slow, low quality, high costs, less efficient, expensive costs, bad server, complex, long steps, unclear, inaccurate.

3.2.2 *Guidelines for Identifying Aspects and Sentiments Related to Efficiency, as Measured by Correctness, in App Review Labeling*

Definition: Efficiency refers to how well an app performs its tasks in terms of speed and resource utilization, while Correctness indicates whether the app functions as intended without errors or issues [53].

Labeling Process

1. *Identify Aspect:*
   - Look for mentions of the app's performance in executing tasks accurately and quickly. Focus on whether the review addresses the correctness of the app's functionalities.
   - Consult the guideline rules detailed in Table 5 to understand how to accurately identify the efficiency factor.
   - Refer to the keywords listed in Table 6 to aid in the accurate identification of the efficiency factor.
2. *Sentiment Analysis:*
   - Consult the guideline rules in Table 5 to understand how to accurately classify the sentiments expressed by users regarding the efficiency factor of the application.
   - Refer to the keywords detailed in Table 6 for assistance in classifying sentiment related to the efficiency factor.
   - Determine the sentiment expressed towards the app's efficiency and correctness, and assign a sentiment label based on the tone:
     – Positive: The review indicates satisfaction with the app's speed and accuracy.
     – Negative: The review indicates dissatisfaction due to errors or slow performance.

Table 5: Guideline rules for identifying and categorizing the efficiency aspect and its associated sentiment from app reviews

Aspects	Sentiments	
	Positive	Negative
Efficiency (Correctness)	If the review includes words or phrases that indicate the product's functionality is operating as expected.	If the review includes words or phrases that suggest issues or problems with the product's core features.
	If the review contains terms that suggest the app is easy to use and performs tasks correctly.	If the review mentions bugs, errors, or crashes that affect the app's functionality.
	If the review highlights consistency in the app's performance without errors	If the review highlights that the app is non-engaging or fails to capture user interest due to performance issue.
		If the review highlights that the app is non-engaging or fails to capture user interest due to performance issue.

Table 6: Keywords for Identifying and categorizing the efficiency aspect and its associated sentiment from app reviews

Aspects	Sentiments	
	Positive	Negative
Efficiency (Correctness)	Few bugs, few errors, quickly perform tasks, easy to use, engaging, effective, clean and intuitive design, seamless, more enjoyable, correct tasks, easiness.	More bugs, more errors, slow, hard, non-engaging, non-effective, not clear, not seamless.

Examples of Reviews and Labels

Review: "The app runs smoothly and processes my requests accurately every time".

Label: Aspect: Efficiency

Metric: Correctness

Sentiment: Positive

Review: "I often encounter errors when trying to save my work; it's very frustrating".

Label: Aspect: Efficiency

Metric: Correctness

Sentiment: Negative

3.2.3 *Guidelines for Identifying Aspects and Sentiments Related to Satisfaction, as Measured by Rating, in App Review Labeling*

Definition: Satisfaction refers to how pleased users are with an app, while rating serves as a metric indicating users' overall evaluations, often represented through star ratings or numerical scores [53].

Labeling Process

1. *Identify Aspect:*
   - Look for mentions of overall user satisfaction with the app. This includes direct references to the user's satisfaction based on their rating, as well as implied satisfaction derived from their experiences.
   - Consult the guideline rules detailed in Table 7 to understand how to accurately identify the satisfaction factor.
   - Refer to the keywords outlined in Table 8 to aid in the accurate identification of the satisfaction factor.
2. *Sentiment Analysis:*
   - Consult the guideline rules detailed in Table 7 to understand how to classify the sentiments expressed by users regarding the satisfaction factor of the application.
   - Refer to the keywords provided in Table 8 to assist in accurately classifying the sentiment related to the satisfaction factor.

- Determine the sentiment expressed towards the app's satisfaction and rating, and assign a sentiment label based on the tone:
  - Positive: The review indicates a high level of satisfaction or enjoyment.
  - Negative: The review indicates dissatisfaction or disappointment.

Table 7: Guideline rules for identifying and categorizing the satisfaction aspect and its associated sentiment from app reviews

Aspects	Sentiments	
	Positive	Negative
Satisfaction (Rating)	If the review contains affirmative statements about the user experience.	If the review contains clear statements of dissatisfaction
	If the review contains descriptors that indicate a pleasant user experience.	If users indicate they have stopped using the app or are considering uninstalling it.
	If the review includes a recommendation to others.	If the review expresses that the app fails to engage the user.
	If the rating is 5, 4, or 3, the user is completely satisfied with the app.	If the rating is poor, 1 or 2

Table 8: Keywords for identifying and categorizing the satisfaction aspect and its associated sentiment from app reviews

Aspects	Sentiments	
	Positive	Negative
Satisfaction (Rating)	User satisfaction, enjoyable, attention (engagement), pleasurable, exciting, entertaining, motivating, challenging, enhancing sociability, enhancement of understanding, supporting creativity, cognitively stimulating, fun, cutesy, gratifying, rewarding, emotionally fulfilling, experience, good, favorite, the best, helpful, perceived enjoyment, interest.	Unsatisfying, boring, unpleasant, frustrating, annoying, gimmicky, childish, confusing, bad, patronizing.

Examples of Reviews and Labels

Review: "I absolutely love this app! It makes my life so much easier".

Label: Aspect: Satisfaction

Metric: Rating

Sentiment: Positive

Review: “I’m not very happy, I expected more from this app”.

Label: Aspect: Satisfaction

Metric: Rating

Sentiment: Negative

Fig. 4 showcases samples from the Instagram App Reviews dataset after being labeled using the ABSA framework. The labels highlight the sentiment associated with the identified key usability factors, which include effectiveness (measured by completeness), efficiency (measured by correctness), and satisfaction (measured by the assigned rating).

reviews	completeness	correctness	satisfaction	rating
Eventhough sometimes it works most of the time it doesn't. The quality of videos uploaded to	Negative	Negative	Negative	2
The app is good for connecting with friends family and even potential business partners. How	None exists	Positive	Positive	3

Figure 4: Sample from instagram app reviews dataset after ABSA-based labeling

3.3 Dataset Preprocessing

To ensure that the review text is clean, consistent, and well-suited for subsequent analysis and modeling, the following preprocessing pipeline was implemented:

First: Text Cleaning

- Lowercasing: All review text was converted to lowercase for consistency.
- Removal of Punctuation: All punctuation marks were removed.
- Removal of Stop Words: Common words (such as “the,” “and,” and “is”) that carried little semantic meaning were removed.

Second: Text Normalization

Stemming technique was used to transform the words in the review text to their root forms.

Third: Word Tokenization

The review text was split into tokens using word-based tokenization.

3.4 Developing Classification Models

This research aims to identify the most effective model architecture for categorizing app reviews based on their alignment with key usability factors, as evaluated using the corresponding usability metrics within the ABSA framework. To achieve this, the study explores advanced hybrid DL model architectures, with the BERT model serving as the foundational base for embedding textual features. BERT was selected for its groundbreaking capabilities in the field of NLP. Its bidirectional contextual understanding allows for nuanced interpretation of language, while its robust generalization from extensive pre-training on large datasets enhances its effectiveness. Furthermore, BERT’s flexibility for fine-tuning and its state-of-the-art performance across a variety of NLP tasks solidify its position as a leading choice in the domain.

The hybrid DL models explored in this research include BERT as a baseline, BERT paired with CNNs (BERT-CNN), BERT paired with LSTM networks (BERT-LSTM), BERT paired with bidirectional LSTM networks (BERT-BiLSTM), and BERT paired with both BiLSTM and CNN (BERT-BiLSTM-CNN). A detailed explanation of each model is provided below:

1. BERT-Base: The BERT model was implemented using the TensorFlow framework, featuring a robust architecture designed for effective ABSA. It begins with an input layer that processes tokenized review text, accompanied by attention masks that differentiate valid tokens from padding. The core of the model is the BERT layer, utilizing the TFBertForSequenceClassification architecture, initialized with pretrained weights from the "bert-base-uncased" variant. This layer effectively processes the input to generate rich contextual embeddings. The output embeddings are then passed through a fully connected layer, followed by a log softmax output layer that generates log probabilities for each usability factor, facilitating multi-class classification.
2. BERT-CNN: This model integrates a BERT layer with Convolutional Neural Network (CNN) components for effective text classification. It starts with an input layer and a BERT layer, similar to those in the BERT-Base model. The output is then passed through a 2D convolutional layer with 13 feature maps and a kernel size of (3, 768), followed by batch normalization and ReLU activation. Max pooling reduces the dimensionality, and dropout is applied to mitigate overfitting. The pooled output is flattened and fed into a fully connected layer that maps to the specified number of classes. Finally, a log softmax output layer generates log probabilities for each class, facilitating multi-class classification tasks.
3. BERT-LSTM: This model combines a BERT layer with an LSTM layer for effective text classification. Similar to the BERT-Base architecture, it begins with an input layer and attention masks that represent tokenized sentences, followed by the BERT layer. The contextual embeddings produced by the BERT layer are then passed to an LSTM layer, which processes the input with a hidden size of 128, utilizing two layers without bidirectionality. A dropout layer with a rate of 0.1 helps prevent overfitting. The output from the last time step of the LSTM is fed into a fully connected layer that maps to the specified number of classes, followed by a log softmax output layer that produces log probabilities for each class. This architecture effectively captures sequential dependencies in text.
4. BERT-BiLSTM: The architecture of this model closely resembles that of BERT-LSTM, with a notable distinction: the LSTM layer is configured for bidirectionality, allowing the model to process input sequences in both forward and backward directions, capturing contextual information from both the past and the future. The output from the BiLSTM layer is passed through a fully connected layer, followed by a log softmax output layer that produces log probabilities for each class, enabling effective multi-class classification. This enhancement improves the model's ability to understand nuances in text.
5. BERT-BiLSTM-CNN: This model architecture combines the strengths of BERT, BiLSTM, and CNN layers for enhanced text classification. It begins similarly to the BERT-Base model, with an input layer and attention masks that feed into the BERT layer, generating rich contextual embeddings. These embeddings are then processed by a BiLSTM layer to capture contextual information from both past and future sequences. The output from the BiLSTM is subsequently fed into a CNN layer, where a ReLU activation function introduces non-linearity to extract local features from the sequence. Following the CNN layer, batch normalization is applied to stabilize and accelerate training, while max pooling reduces the dimensionality of the feature maps. To mitigate overfitting, a dropout layer is applied after the pooling layer. Finally, the model includes a fully connected layer that maps the pooled features to the specified number of classes, followed by a log softmax output layer that generates log probabilities for each class. This comprehensive architecture leverages the strengths of BERT, BiLSTM, and CNN, making it particularly effective for complex tasks such as sentiment analysis and sequence classification.

The architecture of the BERT-BiLSTM-CNN model, identified as the best-performing model in Section 4, is illustrated in Fig. 5, highlighting its key components.

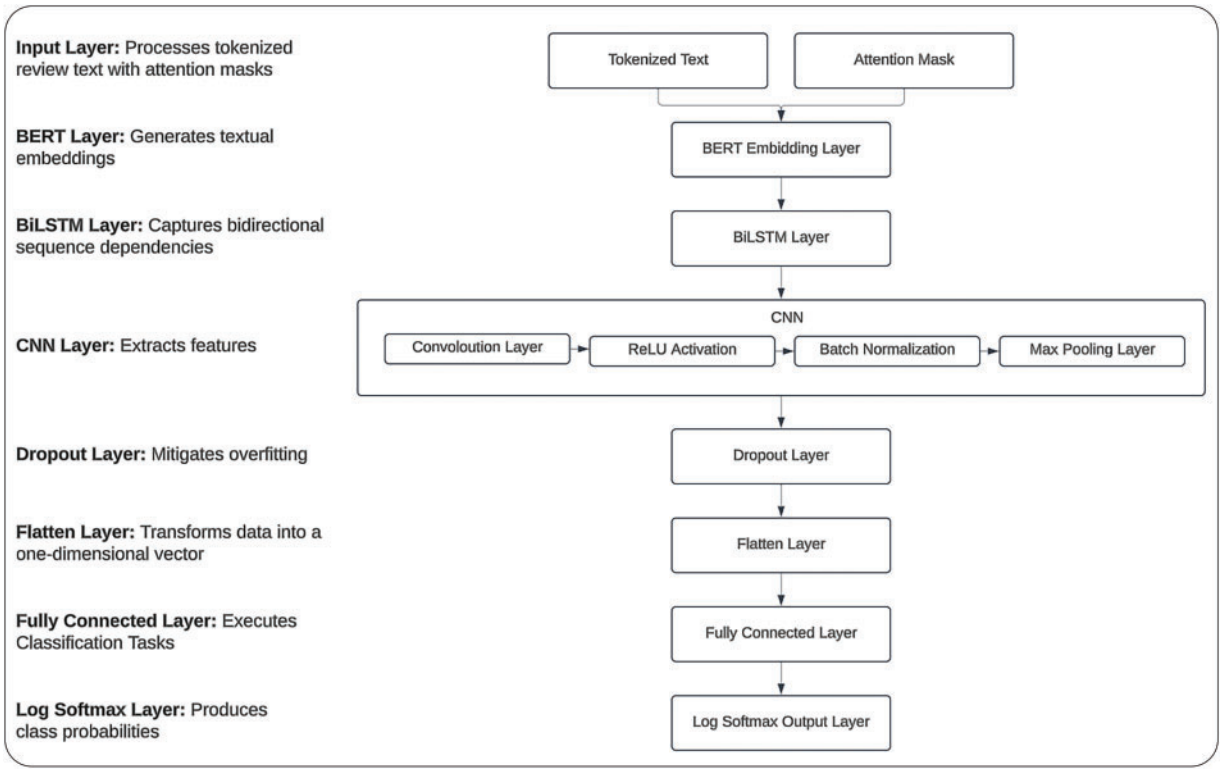

Figure 5: Architecture of the BERT-BiLSTM-CNN model

Fig. 5 illustrates the architecture of the BERT-BiLSTM-CNN model, showcasing the sequential flow from input text tokenization to the final classification output. Key components include the BERT layer for contextual embedding, the BiLSTM layer for capturing sequential dependencies, the CNN layer for feature extraction, and the fully connected layer for classification.

3.5 Performance Metrics

Evaluating classification models for usability metrics in app reviews requires a combination of key performance metrics: accuracy, precision, recall, and F1-score. Together, these metrics provide a comprehensive understanding of the model's ability to classify sentiments associated with usability factors, including effectiveness, efficiency, and satisfaction.

- **Accuracy** measures the overall rate of correct classifications and provides a general sense of reliability. However, it can be misleading for imbalanced datasets, which are common in sentiment classification tasks.
- **Precision** evaluates the proportion of true positive predictions out of all predicted positives, reflecting the model's ability to minimize false positives.
- **Recall** measures the proportion of true positives out of all actual positives, ensuring that critical sentiments are not missed.
- The **F1-score**, as the harmonic mean of precision and recall, balances these metrics and is particularly effective in handling imbalanced datasets.

These metrics collectively offer a robust and interpretable measure of classifier performance, guiding the selection of models capable of reliably capturing nuanced sentiments in app reviews. Naser et al. [54] and Botchkarev [55] emphasize their importance in applications like app review classification, where detecting sentiment polarity across diverse usability factors is critical.

Presented below are the formulas corresponding to these metrics:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{2}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{3}$$

$$F1 - score = 2\left(\frac{Precision \times Recall}{Precision + Recall}\right) \tag{4}$$

where:

- **TP (True Positive)**: Correctly predicted positive cases.
- **TN (True Negative)**: Correctly predicted negative cases.
- **FP (False Positive)**: Incorrectly predicted positive cases.
- **FN (False Negative)**: Incorrectly predicted negative cases.

4 Results and Discussion

The results of evaluating various hybrid DL architectures, summarized in Table 9 and visually represented in Fig. 6, reveal important insights into their performance. Although the BERT-Base model achieved a high precision score of 0.86, its low recall of 0.55 indicates significant challenges in identifying relevant sentiments within the reviews. This suggests that, despite its capabilities, BERT-Base may not be sufficiently fine-tuned for the nuances of the ABSA framework in app review classification, highlighting the need for hybrid approaches to enhance its effectiveness.

Table 9: Results of the hybrid DL models evaluated in this study

Model	Precision	Recall	F1-score	Accuracy
BERT-Base	0.86	0.55	0.67	0.77
BERT-CNN	0.78	0.78	0.78	0.78
BERT-LSTM	0.76	0.74	0.75	0.74
BERT-BiLSTM	0.76	0.78	0.77	0.78
Proposed model: BERT-BiLSTM-CNN	0.96	0.87	0.91	0. 94

In contrast, all dual hybrid models that paired BERT with a single DL architecture—namely, BERT-CNN, BERT-LSTM, and BERT-BiLSTM—demonstrated promising results, particularly in recall. This improvement indicates their enhanced ability to capture relevant instances compared to standalone models. Each hybrid model achieved F1-scores around 0.75, showcasing their effectiveness in balancing precision and recall. However, their performance still fell short of that achieved by the trio hybrid model, BERT-BiLSTM-CNN.

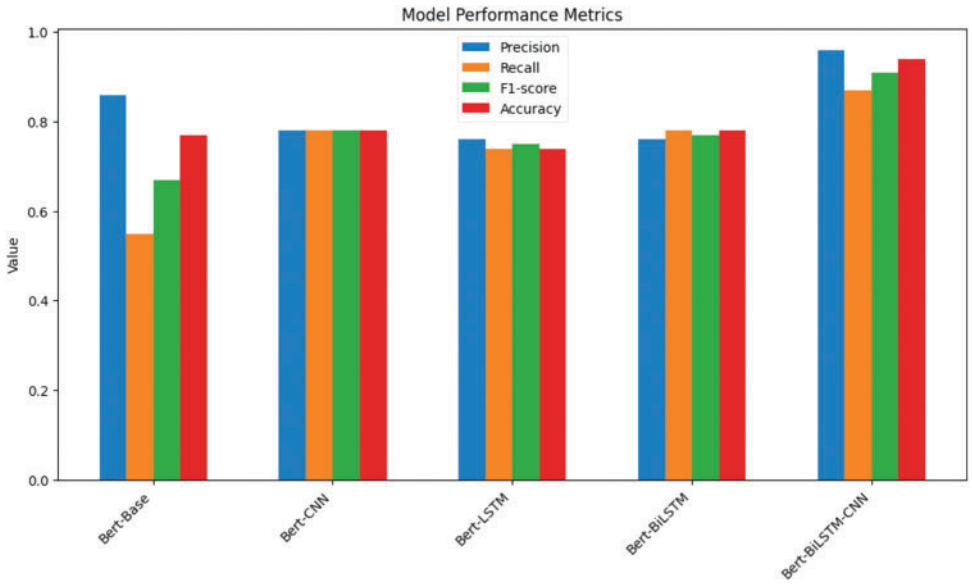

Figure 6: Visualization of the performance of hybrid DL models

The BERT-BiLSTM-CNN hybrid model emerged as the standout performer, achieving impressive metrics: a precision of 0.96, a recall of 0.87, an F1-score of 0.91, and an accuracy of 0.94. This model's ability to integrate BERT's contextual embeddings with the sequential processing capabilities of BiLSTM and the feature extraction power of CNNs allows it to capture both global context and local patterns in app reviews, resulting in a more comprehensive understanding of user sentiment.

To evaluate the efficacy of the ABSA-based methodology and the hybrid architecture, the proposed BERT-BiLSTM-CNN model was analytically compared against state-of-the-art app review classification models that categorize reviews based on NFRs. These models typically classify reviews into primary NFR types or granular metrics, such as bug reports, feature requests, and user experiences, offering valuable insights for defining NFRs.

This comparative analysis addresses the lack of existing research on classifying app reviews using usability factors within the ABSA framework. Table 10 presents the results of this analysis, visually represented in Fig. 7. The findings demonstrate that the proposed model, despite the increased complexity of the ABSA classification framework, consistently outperforms existing methods across all evaluation metrics, including precision, recall, F1-score, and accuracy.

Table 10: Comparative performance of the proposed models across metrics

Ref.	Method	Precision	Recall	F1	Accuracy
[31]	AUR-BoW with Baggin	0.714	0.723	0.718	—
[32]	AUG-BoW	0.656	0.678	0.652	—
[33]	CNN	0.9549	0.9394	0.9471	—
[34]	SVM	—	—	—	0.70
[35]	SVM + TF-IDF	0.92	0.93	0.92	—
[36]	AUC	—	—	—	—

(Continued)

Table 10 (continued)

Ref.	Method	Precision	Recall	F1	Accuracy
[37]	SGD	—	—	—	0.83
[38]	BERT-RCNN	0.90	0.87	0.88	—
[39]	LMC	0.742	0.825	0.781	—
[40]	ANN	0.469	0.471	0.458	0.590
[41]	BR + dictionary-based multi-label classification	0.70	0.86	—	—
[42]	MAREVA + RF	—	—	0.75	0.76
[43]	SVM	—	—	—	0.70
Proposed model	**BERT-BiLSTM-CNN**	**0.96**	**0.87**	**0.91**	**0. 94**

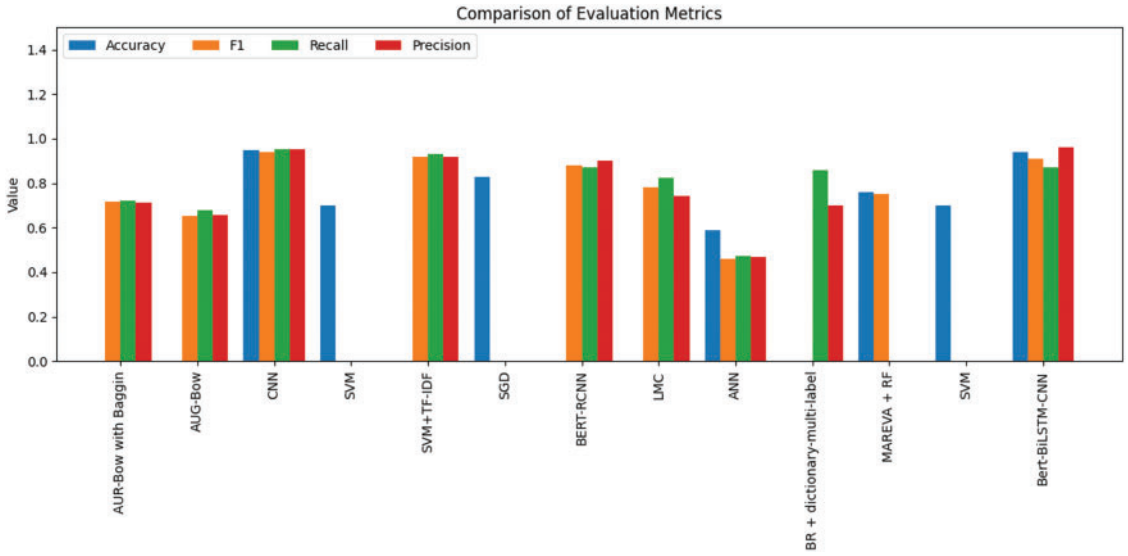

Figure 7: Visualization of the comparative performance of the proposed models across metrics

Table 10 further highlights the superior performance of the BERT-BiLSTM-CNN model, which achieves an F1-score of **91%** and an accuracy of **94%**. This performance is attributed to its hybrid architecture, combining BERT's contextual embeddings with BiLSTM's sequential learning capabilities and CNN's efficient feature extraction. In contrast, traditional machine learning models, such as Random Forest and Logistic Regression, underperform due to their reliance on manual feature engineering, limiting their ability to process complex textual data. Similarly, standalone BiLSTM or CNN models, while effective in some areas, lack the contextual richness provided by pre-trained embeddings like BERT, reducing their ability to capture nuanced sentiments and usability aspects.

These findings emphasize the importance of hybrid architectures in tackling the complexities of ABSA, particularly in domains where usability metrics are critical. However, limitations remain in terms of computational cost and reliance on labeled datasets, which may hinder scalability.

Notably, the BERT-BiLSTM-CNN model significantly surpasses traditional machine learning models, such as SVM, ANN, AUR-BoW with bagging, and LMC. Compared to other deep learning models, the hybrid architecture also demonstrates superior performance. However, in comparison to the CNN model proposed in [33], the results are comparable in terms of precision, though the model in [33] achieves better recall. This discrepancy may stem from the inherent complexity of ABSA classification and the multimodal approach employed in [33], which enhances the CNN model's ability to capture a broader range of features.

The primary research question in this study aimed to evaluate the effectiveness of the proposed hybrid deep learning architecture (BERT-BiLSTM-CNN) in classifying app reviews based on usability metrics within the ABSA framework. The results clearly demonstrate that the BERT-BiLSTM-CNN model significantly outperforms other configurations, achieving a precision of 96%, recall of 87%, F1-score of 91%, and accuracy of 94%. These metrics highlight the model's ability to accurately capture user sentiments and usability feedback, addressing the research question by confirming the efficacy of hybrid architectures in sentiment analysis. Furthermore, the superior performance of the model validates the feasibility of using such advanced techniques for improving user-centric application development.

5 Conclusions

This research successfully identifies and evaluates advanced hybrid deep learning models for classifying app reviews based on usability metrics within the ABSA framework. The proposed BERT-BiLSTM-CNN model outperformed other models and state-of-the-art techniques, achieving precision of 96%, recall of 87%, F1-score of 91%, and accuracy of 94%. These results emphasize the advantages of hybrid architectures in sentiment analysis, providing a robust foundation for future advancements.

The insights derived from this research are invaluable for organizations seeking to refine their products and services. By adopting the ABSA framework and leveraging the BERT-BiLSTM-CNN model, businesses can gain a deeper understanding of user sentiments associated with specific usability factors. This knowledge enables targeted improvements in app features, driving higher user satisfaction and long-term customer loyalty. However, scalability challenges and computational demands may impact real-world implementation, requiring further optimization.

The BERT-BiLSTM-CNN model demonstrated superior performance, achieving a precision of 96%, recall of 87%, F1-score of 91%, and accuracy of 94%. This model surpassed other configurations, including BERT-CNN, BERT-LSTM, and BERT-BiLSTM. These results underscore the model's capability to effectively balance precision and recall, ensuring comprehensive coverage of both positive and negative sentiments related to usability factors. While these results are promising, some challenges and limitations need to be addressed for broader applicability.

Despite these promising results, potential limitations include scalability challenges when applied to larger datasets or across diverse languages. Additionally, the model's reliance on labeled data can restrict adaptability to new domains where labeled data may be limited or unavailable. Furthermore, real-time processing demands may require significant computational resources, potentially increasing deployment costs.

Future research could focus on enhancing the BERT-BiLSTM-CNN model's performance by integrating additional layers, such as attention mechanisms or transformer-based enhancements, to improve its ability to handle complex and nuanced app reviews. Expanding the dataset to include app reviews from diverse platforms—such as social media, e-commerce websites, and industry-specific

feedback systems—could further improve the model's generalizability and robustness. These advancements would address scalability challenges and optimize the model for real-world applications, making it adaptable to a broader range of user feedback scenarios.

To ensure clarity and accessibility, Table 11 outlines the metrics applied to evaluate model performance and usability factors, ensuring a comprehensive understanding of their relevance.

Table 11: List of metrics and their definitions

Metric		Definition
Model performance metrics	Accuracy	The proportion of correct predictions (TP and TN) out of all predictions
	Perception	The proportion of true positives out of all predicted positive cases; TP/(TP+FP)
	Recall	The proportion of true positives out of all actual positives; TP/(TP + FN)
	F1-score	Harmonic mean of precision and recall, providing a balanced metric: 2 ∗ (Precision ∗ Recall)/(Precision+Recall)
Usability factors metrics	Completeness	Metric for **Effectiveness**, indicating how fully an app meets user needs
	Correctness	Metric for **Efficiency**, reflecting the accuracy and reliability of an app's performance
	Rating	Metric for **Satisfaction**, representing user ratings or scores

In conclusion, this study introduces a robust hybrid deep learning model for classifying app reviews within the ABSA framework. This approach not only enhances the ability to analyze user feedback but also provides a practical tool for improving app design, boosting customer satisfaction, and fostering long-term user engagement.

Acknowledgement: This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah. The authors sincerely acknowledge the DSR for their financial and logistical support, which made this research possible.

Funding Statement: This work was supported by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (GPIP: 13-612-2024). The funding provided support for technical resources, materials, and overall project execution.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Nahed Alsaleh, Reem Alnanih, Nahed Alowidi; Data collection: Nahed Alsaleh; Analysis and interpretation of results: Nahed Alsaleh, Reem Alnanih, Nahed Alowidi; Draft manuscript preparation: Nahed Alsaleh; Paper reviewing and editing: Reem Alnanih, Nahed Alowidi; Paper proofreading: Nahed Alsaleh, Reem Alnanih, Nahed Alowidi. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials used in this study are available to interested researchers upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] L. Chung, P. Do, and J. C. S. Leite, "On non-functional requirements in software engineering," in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, E. S. Yu, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, vol. 5600, pp. 363–379. doi: 10.1007/978-3-642-02463-4_19.

[2] O. Oyebode, F. Alqahtani, and R. Orji, "Using machine learning and thematic analysis methods to evaluate mental health apps based on user reviews," *IEEE Access*, vol. 8, pp. 111141–111158, 2020. doi: 10.1109/ACCESS.2020.3002176.

[3] ISO, "Ergonomics of human-system interaction—Part 11," Usability: Definitions and Concepts (ISO 9241-11:2018), 2018.

[4] S. L. Pfleeger, *Software Engineering: The Production of Quality Software*. New York, NY, USA: Macmillan Publishing Co., Inc., 1987.

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

[6] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution," in *Proc. 30th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, 2015, pp. 771–776. doi: 10.1109/ASE.2015.88.

[7] K. E. Wiegers and J. Beatty, *Software Requirements*. Upper Saddle River, NJ, USA: Pearson Education, 2013.

[8] P. Borque and R. Fairley, *Guide to the Software Engineering Body of Knowledge Version 3.0*. Los Alamitos, CA, USA: IEEE Computer Society Staff, 2014.

[9] M. Hosseini, K. T. Phalp, J. Taylor, and R. Ali, "Towards crowdsourcing for requirements engineering," in *Proc. 20th Int. Working Conf. Requirements Eng.: Found. Softw. Quality (REFSQ)*, Essen, Germany, 2014.

[10] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdiscip. Rev. Data Mining Knowl. Discovery*, vol. 8, no. 4, 2018, Art. no. e1253.

[11] Saloni1712, "Instagram play store reviews," 2023. Accessed: Jun. 14, 2024. [Online]. Available: https://www.kaggle.com/datasets/saloni1712/instagram-play-store-reviews?resource=download

[12] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *Proc. 25th IEEE Int. Requirements Eng. Conf. (RE)*, 2017, pp. 490–495.

[13] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Syst. Appl. X*, vol. 1, 2019, Art. no. 100001.

[14] A. Kobilica, M. Ayub, and J. Hassine, "Automated identification of security requirements: A machine learning approach," in *Proc. Eval. Assessment Softw. Eng.*, Trondheim, Norway, ACM, 2020, pp. 475–480. doi: 10.1145/3383219.3383288.

[15] S. Tiun, U. A. Mokhtar, S. H. Bakar, and S. Saad, "Classification of functional and non-functional requirements in software requirements using Word2vec and fastText," *J. Phys.: Conf. Ser.*, vol. 1529, 2020, Art. no. 042077.

[16] M. Sabir, C. Chrysoulas, and E. Banissi, "Multi-label classifier to deal with misclassification in non-functional requirements," in *Proc. Eval. Assessment Softw. Eng.*, 2020.

[17] N. Rahimi, F. Eassa, and L. Elrefaei, "One- and two-phase software requirement classification using ensemble deep learning," *Entropy*, vol. 23, no. 10, Oct. 2021, Art. no. 1264.

[18] M. Syauqi Haris and T. A. Kurniawan, "Automated requirement sentences extraction from software requirement specification documents," in *Proc. 5th Int. Conf. Sustain. Inf. Eng. Technol.*, 2020, pp. 142–147.

[19] F. Baskoro, R. A. Andrahsmara, B. R. P. Darnoto, and Y. A. Tofan, "A systematic comparison of software requirements classification," *IPTEK J. Technol. Sci.*, vol. 32, no. 3, pp. 184–193, 2021. doi: 10.12962/j20882033.v32i3.13005.

[20] M. A. Haque, M. A. Rahman, and M. S. Siddik, "Non-functional requirements classification with feature extraction and machine learning: An empirical study," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, vol. 33, no. 1, pp. 1–5, 2019. doi: 10.47489/p000s331z702-1-5mc.

[21] A. A. Fahmi, A. Achmad, and D. Siahaan, "Algorithms comparison for non-requirements classification using the semantic feature of software requirement statements," *IPTEK J. Technol. Sci.*, vol. 31, no. 3, pp. 343–352, 2021. doi: 10.12962/j20882033.v31i3.7606.

[22] E. D. Canedo and B. C. Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, no. 9, 2020, Art. no. 1057. doi: 10.3390/e22091057.

[23] Q. A. Shreda and A. A. Hanani, "Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches," *IEEE Access*, vol. 9, pp. 1–22, 2021.

[24] M. A. Rahman, M. A. Haque, M. N. A. Tawhid, and M. S. Siddik, "Classifying non-functional requirements using RNN variants for quality software development," in *Proc. 3rd ACM SIGSOFT Int. Workshop Mach. Learn. Tech. Softw. Quality Eval.*, 2019, pp. 25–30.

[25] C. Baker, L. Deng, S. Chakraborty, and J. Dehlinger, "Automatic multi-class non-functional software requirements classification using neural networks," in *Annu Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, pp. 610–615, 2019. doi: 10.1109/COMPSAC.2019.10275.

[26] F. Khayashi, B. Jamasb, R. Akbari, and P. Shamsinejadbabaki, "Deep learning methods for software requirement classification: A performance study on the PURE dataset," 2022, *arXiv:2211.05286*.

[27] L. Tóth and L. Vidács, "Comparative study of the performance of various classifiers in labeling non-functional requirements," *Inf. Technol. Control.*, vol. 48, no. 3, pp. 432–445, 2019. doi: 10.5755/j01.itc.48.3.21973.

[28] M. A. Khan, M. S. Khan, I. Khan, S. Ahmad, and S. Huda, "Non-functional requirements identification and classification using transfer learning model," *IEEE Access*, vol. 11, no. 3, pp. 74997–75005, 2023. doi: 10.1109/ACCESS.2023.3295238.

[29] M. Younas, K. Wakil, and D. N. A. Jawawi, "An automated approach for identification of non-functional requirements using Word2Vec model," *J. Adv.*, vol. 10, no. 8, 2019. doi: 10.14569/issn.2156-5570.

[30] L. Tóth and L. Vidács, "Study of various classifiers for identification and classification of non-functional requirements," in *Computational Science and Its Applications—ICCSA 2018*, O. Gervasi *et al.*, Eds. Cham, Switzerland: Springer International Publishing, 2018, vol. 10964, pp. 492–503, 2018. doi: 10.1007/978-3-319-95174-4_39.

[31] M. EzzatiKarami and N. H. Madhavji, "Automatically classifying non-functional requirements with feature extraction and supervised machine learning techniques: A research preview," in *Requirements Engineering: Foundation for Software Quality*. Cham: Springer, Apr. 12–15, 2021, pp. 71–78.

[32] K. Rahman, A. Ghani, A. Alzahrani, M. U. Tariq, and A. U. Rahman, "Pre-trained model-based NFR classification: Overcoming limited data challenges," *IEEE Access*, vol. 11, pp. 81787–81802, 2023. doi: 10.1109/ACCESS.2023.3301725.

[33] M. A. F. Saroth, P. M. A. K. Wijerathne, and B. T. G. S. Kumara, "Automatic multi-class non-functional software requirements classification using machine learning algorithms," in *2024 Int. Res. Conf. Smart Comput. Syst. Eng. (SCSE)*, Colombo, Sri Lanka, 2021, pp. 1–6.

[34] K. Rahman, A. Ghani, S. Misra, and A. U. Rahman, "A deep learning framework for non-functional requirement classification," *Sci. Rep.*, vol. 14, no. 1, 2024, Art. no. 3216.

[35] S. M. García, C. A. Fernández-y-Fernández, and E. R. Pérez, "Classification of non-functional requirements using convolutional neural networks," *Program. Comput. Softw.*, vol. 49, no. 8, pp. 705–711, 2023. doi: 10.1134/S0361768823080133.

[36] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng.*, 2017, pp. 344–353.
[37] C. Wang, T. Wang, P. Liang, M. Daneva, and M. Van Sinderen, "Augmenting app reviews with app changelogs: An approach for app review classification," in *Proc. 31st Int. Conf. Softw. Eng. Knowl. Eng. (SEKE)*, vol. 2023, pp. 398–512, 2019. doi: 10.18293/SEKE.
[38] N. Aslam, W. Y. Ramay, K. Xia, and N. Sarwar, "Convolutional neural network-based classification of app reviews," *IEEE Access*, vol. 8, pp. 185619–185628, 2020. doi: 10.1109/ACCESS.2020.3029634.
[39] C. Wang, F. Zhang, P. Liang, M. Daneva, and M. Van Sinderen, "Can app changelogs improve requirements classification from app reviews? An exploratory study," in *Proc. 12th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2018, pp. 1–4.
[40] D. Dave, V. Anu, and A. S. Varde, "Automating the classification of requirements data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2021, pp. 5878–5880.
[41] R. Jindal, R. Malhotra, A. Jain, and A. Bansal, "Mining non-functional requirements using machine learning techniques," *e-Informatica Softw. Eng. J.*, vol. 15, no. 1, pp. 85–114, 2021. doi: 10.37190/e-Inf210105.
[42] D. J. Dave, "Identifying functional and non-functional software requirements from user app reviews and requirements artifacts," 2022. Accessed: Jun. 14, 2024. [Online]. Available: https://digitalcommons.montclair.edu/etd/1012/
[43] K. Kaur and P. Kaur, "BERT-RCNN: An automatic classification of app reviews using transfer learning based RCNN deep model," 2023. Accessed: Jun. 14, 2024. [Online]. Available: https://www.researchsquare.com/article/rs-2503700/v1
[44] Y. Yao, W. Jiang, Y. Wang, P. Song, and B. Wang, "Non-functional requirements analysis based on application reviews in the Android app market," *Inf. Resour. Manag. J.*, vol. 35, no. 2, pp. 1–17, 2022. doi: 10.4018/IRMJ.291694.
[45] D. A. López-Hernández, E. Mezura-Montes, J. O. Ocharán-Hernández, and A. J. Sánchez-García, "Non-functional requirements classification using artificial neural networks," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, vol. 5, pp. 1–6, 2021.
[46] N. Jha and A. Mahmoud, "Mining non-functional requirements from app store," *Empir. Softw. Eng.*, vol. 24, no. 6, pp. 3659–3695, 2019. doi: 10.1007/s10664-019-09716-7.
[47] M. B. Messaoud, I. Jenhani, N. B. Jemaa, and M. W. Mkaouer, "A multi-label active learning approach for mobile app user review classification," in *Knowledge Science, Engineering and Management*. Cham: Springer, Aug. 28–30, 2019, pp. 805–816.
[48] M. K. M. Binkhonain, "Using machine learning algorithms for classifying non-functional requirements—Research and evaluation," Ph.D. dissertation, Univ. of Manchester, UK, 2021.
[49] A. E. Yahya, A. Gharbi, W. M. Yafooz, and A. Al-Dhaqm, "A novel hybrid deep learning model for detecting and classifying non-functional requirements of mobile apps issues," *Electronics*, vol. 12, no. 5, 2023, Art. no. 1258. doi: 10.3390/electronics12051258.
[50] K. Kaur and P. Kaur, "MNoR-BERT: Multi-label classification of non-functional requirements using BERT," *Neural Comput. Appl.*, vol. 35, no. 30, pp. 22487–22509, 2023. doi: 10.1007/s00521-023-08833-1.
[51] K. Rahman, A. Ghani, R. Ahmad, and S. H. Sajjad, "Hybrid deep learning approach for nonfunctional software requirements classifications," in *Proc. Int. Conf. Commun., Comput. Digit. Syst. (C-CODE)*, IEEE, 2023, pp. 1–5.
[52] T. Rana, "Devising a usability development life cycle (UDLC) model for enhancing usability and user experience in interactive applications," *Sir Syed Univ. Res. J. Eng. Technol.*, vol. 12, no. 2, pp. 81–94, 2022.
[53] W. Yafooz and A. Alsaeedi, "Leveraging user-generated comments and fused BiLSTM models to detect and predict issues with mobile apps," *Comput. Mater. Contin.*, vol. 79, no. 1, pp. 735–759, 2024. doi: 10.32604/cmc.2024.048270.

[54] M. Z. Naser and A. H. Alavi, "Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences," *Archit. Struct. Constr.*, vol. 3, no. 4, pp. 499–517, 2023. doi: 10.1007/s44150-021-00015-8.

[55] A. Botchkarev, "A new typology design of performance metrics to measure errors in machine learning regression algorithms," *Interdiscip. J. Inform. Knowl. Manage.*, vol. 14, pp. 45–76, 2019. doi: 10.28945/4184.