Arabic App Reviews: Analysis and Classification

OTHMAN ALJEEZANI, Information and Computer Sciences, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
DORIEH ALOMARI, Information and Computer Sciences, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
IRFAN AHMAD, Information and Computer Sciences Department, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia and SDAIA-KFUPM JRC for Artificial Intelligence, Dhahran, Saudi Arabia

User opinions and feedback on mobile applications are crucial for application developers, offering insights into issues like bugs, popular features, and enhancement requests. Given the vast number of feedback for each app, it is impractical for developers to manually extract valuable information. To better understand and analyze user opinions, developers can benefit from automatic sentiment analysis and classification of app reviews. Existing research has primarily focused on reviews written in English, with some studies addressing sentiment analysis of Arabic reviews but overlooking the classification task. Given the widespread use and complexity of Arabic compared to English, our work investigates both sentiment analysis and classification of Arabic app reviews. We introduce the AURA (**A**pp **U**ser **R**eview in **A**rabic) dataset. AURA dataset has two versions: AURA-Sentiment with 29,700 labeled reviews for sentiment analysis, and AURA-Classification with 2,900 labeled reviews for classification. Using these datasets, we applied deep learning (DL) and natural language processing (NLP) techniques for analyzing and classifying Arabic app reviews. Leveraging the MarBert model, we achieved an F1-score of 0.89 for sentiment analysis and an F1-score of 0.62 for a four-class classification problem. Our findings provide valuable insights and suggest directions for future research.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Language resources;**

Additional Key Words and Phrases: App reviews, Arabic NLP, Datasets, Sentiment analysis, Text classification

ACM Reference Format:
Othman Aljeezani, Dorieh Alomari, and Irfan Ahmad. 2025. Arabic App Reviews: Analysis and Classification. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 24, 2, Article 14 (February 2025), 28 pages. https://doi.org/10.1145/3708987

The authors would like to thank King Fahd University of Petroleum and Minerals (KFUPM) for supporting this work. Irfan Ahmad would like to additionally thank Saudi Data and AI Authority (SDAIA) and KFUPM for supporting him through SDAIA-KFUPM Joint Research Center for Artificial Intelligence grant number JRC–AI–RFP–10.
Authors' Contact Information: Othman Aljeezani, Information and Computer Sciences, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia; e-mail: iu.othmanj@gmail.com; Dorieh Alomari, Information and Computer Sciences, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia; e-mail: g202213300@kfupm.edu.sa; Irfan Ahmad, Information and Computer Sciences Department, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia and SDAIA-KFUPM JRC for Artificial Intelligence, Dhahran, Saudi Arabia; e-mail: irfan.ahmad@kfupm.edu.sa.

ACM 2375-4699/2025/02-ART14
https://doi.org/10.1145/3708987

1 Introduction

Smartphones are becoming increasingly common worldwide, accompanied by the rapid growth of software **applications** (**apps**) designed for these devices [24]. The global number of smartphone users has recently risen from 2.5 billion to 3.2 billion and is expected to grow further [59]. In their daily lives, these users engage with apps downloaded from distribution platforms, such as Apple's App Store and Google Play Store. Recent statistics [23] show that, as of the first quarter of 2020, Google Play Store offered 2.56 million apps, while Apple's App Store hosted 1.85 million. With such a vast number of available apps, developers must continuously improve their apps—fixing bugs and incorporating user-requested features—to maintain their user base and prevent them from switching to competitors.

App stores serve as an interactive platform facilitating communication between users and developers. Users can provide feedback by rating apps on a scale of one to five (Star Rating) and sharing their experiences through written reviews. On the other hand, developers can encourage users to leave ratings and reviews while also accessing the feedback provided. Additionally, they can respond directly to user reviews. This dynamic interaction enables developers to enhance and maintain their apps, ensuring they align with users' expectations and requirements.

App reviews provide a wealth of information for developers, often including bug reports, feature requests, and issues related to the **user interface** (**UI**) or **user experience** (**UX**). These reviews, paired with their corresponding star ratings, generally reflect user satisfaction and serve as valuable input for developers to better address customer needs. However, not all reviews are relevant or useful for app development purposes [60]. A significant challenge lies in the sheer volume of reviews per app. For instance, the Instagram[1] app on Apple's App Store has approximately 23 million reviews, while the Tawakkalna app[2] , which is a COVID-19 app launched in Saudi Arabia in April 2020, has over 90,000 reviews. Manually reading and analyzing such a large number of reviews to extract valuable insights is practically impossible. To tackle this issue, researchers have proposed automated approaches leveraging **natural language processing** (**NLP**) and **machine learning** (**ML**) techniques to analyze and classify app reviews [18, 30, 31, 34, 47, 49].

One research direction in app reviews is sentiment analysis, which involves assigning a quantitative value to the text of an app review [13, 30]. This value reflects the mood or emotion of the author (app user). Sentiments in user feedback are beneficial for developers to measure user satisfaction levels with their apps. Researchers have used sentiment analysis in exploring and analyzing app reviews. Hoon et al. [36] manually extracted sentiments from app reviews to test the alignment between review text and its associated star rating. Liang et al. [39] proposed a sentiment analysis approach to examine the influence of sentiments on app sales. Guzman and Maalej [30] extracted app features from app reviews and then assigned a sentiment value to each feature. This work was extended by Guzman et al. [29] to provide developers with a feature-based information retrieval system, enabling them to search for specific features and explore users' sentiments and opinions about them.

Another area of research focuses on app review classification, which involves categorizing app reviews based on the topics they address. As a foundational step, many studies establish a taxonomy of app review categories. For example, Maalej and Nabil [42] identified four key categories: bug reports, feature requests, user experiences, and ratings. Classification is typically performed after defining these categories or by adopting a predefined taxonomy from previous studies. For instance, Oh et al. [46] proposed three categories for app reviews: functional bug, functional demand, and nonfunctional request. Similarly, Panichella et al. [49] introduced a broader taxonomy,

[1] https://apps.apple.com/us/app/instagram
[2] https://apps.apple.com/sa/app/tawakkalna-covid-19-ksa

including categories such as information seeking, information giving, feature request, problem discovery, and others.

The Arabic language, spoken by approximately 275 million people worldwide and the sixth most spoken language in 2021 [25], is the fourth most-used language on the internet as of January 2020, with more than 5% of internet users worldwide [22]. Previous studies have investigated sentiment analysis and related opinion mining techniques in Arabic across diverse domains, including book reviews [10], hotel reviews [45], movie reviews [53], news articles [26], and several other areas [7, 9, 11, 14, 50]. Although the research interest in analyzing Arabic app reviews is expanding, it is still in its early stages with few publicly available datasets for Arabic app reviews to the best of our knowledge. Indeed, the research community has well explored the field of app reviews in the English language, but the Arabic language has its characteristics and challenges; hence, it needs to be investigated.

The primary objective of this research is to introduce datasets for Arabic App review analysis and classification tasks and to perform experiments using modern deep learning-based systems to serve as baselines. This article introduces the **AURA** (**A**pp **U**ser **R**eview in **A**rabic) dataset, which comprises two versions: AURA-Sentiment for sentiment analysis and AURA-Classification for classification task. We evaluate the performance of **Recurrent Neural Networks** (**RNNs**) and pretrained transformer models on these datasets and explore the impact of data augmentation and balancing techniques. The main contributions of this article can be summarized as follows:

(1) Presenting the AURA dataset: a comprehensive public dataset for Arabic app review analysis and classification.
(2) Evaluating the performance of RNNs and pretrained transformer models for sentiment analysis of Arabic app reviews.
(3) Evaluating the performance of RNNs and pretrained transformer models for classification of Arabic app reviews.
(4) Investigating the role and efficacy of preprocessing, focal loss, and data augmentation techniques in enhancing the performance of DL models, particularly in the context of Arabic app review classification.

The remainder of the article is structured as follows: Section 2 reviews previous work and identifies existing gaps. Section 3 details the proposed datasets and their collection. Section 4 discusses our methodology. Section 5 presents the experiments and results. Section 6 concludes the article.

2 Related Work

In the domain of Arabic app review sentiment analysis, many studies have employed ML and DL techniques. Al-Hagree and Al-Gaphari [5] proposed a model for Arabic review sentiment analysis of bank apps. The authors collected 3,197 reviews from the Google Store, covering eight Yemeni banking apps, and classified them into positive, negative, and neutral sentiments. Four ML models were evaluated: **Support Vector Machines** (**SVM**), **Decision Trees** (**DT**), **Naïve Bayes** (**NB**), and **K-Nearest Neighbors** (**KNN**). The models' performances were compared in terms of accuracy, precision, recall, and F1-score. The NB model excelled, achieving 89.65% accuracy, 88.25% precision, 88.08% recall, and an 88.25% F1-score. Similarly, Saudy et al. [54] proposed a hybrid ML and DL model for Egyptian app review sentiment analysis. The authors collected 2,469 Egyptian app reviews from the Google Store across nine different app categories. These reviews were categorized as positive, negative, and neutral. The experiments involved five ML models and one DL model, including SVM, KNN, DL, **Random Forest** (**RF**), **Logistic Regression** (**LR**), and **Multi-Layer Perceptron** (**MLP**). The best-performing models were combined into an ensemble for enhanced results. The hybrid RF-LR-MLP model showed superior performance with 74.3% accuracy,

74.8% precision, 74.3% recall, 74.5% F1-score, and an 89.1% **Area Under the Curve** (**AUC**). Likewise, Chader et al. [17] proposed a model for Algerian app review sentiment analysis. The authors collected 50,000 reviews from the Google Store, the reviews include Modern Standard Arabic, Algerian Dialect, and French reviews, classified into positive and negative sentiments. Pre-processing techniques were applied to the dataset, including normalization, stopword removal, and stemming, and used TF-IDF technique for feature extraction. Five ML algorithms were included in the experiments, namely, SVM, KNN, ANN, NB, and DT. The SVM technique outperformed other techniques with an accuracy of 72%. Furthermore, Ramzy and Ibrahim [52] investigated sentiment analysis on COVID-19 Arabic app reviews. The authors collected 114,499 reviews of 18 COVID-19 apps from different Arab countries from the Google Store. A sample of 8,220 reviews was pre-processed and normalized, and the features were extracted using the **Bag of Words** (**BOW**) technique. Six ML techniques were included in this study, namely, LR, KNN, SVM, RF, NB, and ANN. The ANN model achieved the highest performance with an accuracy and f1-score of 89%.

Moreover, Hadwan et al. [32] proposed an ML model for Arabic review sentiment analysis of Saudi government apps used during the COVID-19 pandemic. The dataset comprised 7,759 Arabic reviews from both the Google Store[3] and Apple Store,[4] covering six Saudi government apps. These reviews were classified into positive, negative, and neutral sentiments. Four ML models, namely SVM, DT, KNN, and NB, were included in this study. The KNN model outperformed the others, achieving 78.46% accuracy, 79.94% precision, 78.01% recall, and 78.96% F1-score. Hadwan et al. [33] improved their previous work [32] by expanding the dataset to include 51,767 reviews of the same apps. To address the imbalance issue, the authors employed the **Synthetic Minority Over-sampling Technique** (**SMOTE**). Five ML models were tested: SVM, LR, NB, RF, and Bagging. The SVM model was the most effective, achieving 94.30% for precision, recall, and F1-score, and 94.38% accuracy. Additionally, Al-Hagree and Al-Gaphari [6] used the same dataset to propose a new approach for Arabic app review sentiment analysis. This approach, an NB model based on the **Levenshtein distance** (**LD**), utilized 1,000 reviews from the original dataset for testing. The proposed approach with K=9 outperformed the solo NB model, yielding a 96.40% accuracy. Using the same dataset [33], Al-Shalabi et al. [8] proposed a KNN with LD model for Arabic app reviews sentiment analysis. The authors used 8,566 reviews from the original dataset. They applied some pre-processing techniques to normalize the dataset, including stopword removing, case folding, and stemming, finally they used the TF-IDF technique for feature representation. The proposed KNN-LD model with K=3 outperformed the single KNN model with an accuracy of 88.11%, precision of 85.10%, recall of 66.30%, and f1-score of 73.53%.

In another study on multi-label Arabic app review classification, Voskergian and Saheb [63] developed a model for **Aspect-Based Sentiment Analysis** (**ABSA**) targeting Arabic app reviews. This study analyzed 2,000 user reviews out of collected 100,000 reviews from three different apps on the Google Store, focusing on five key aspects: Support and Update, UX, UI, Functionality and Performance, and Security. These aspects were categorized into four sentiment polarities: Neutral, Negative, Positive, and Conflict. Various ML models, such as LR, DT, RF, SVM, and **Multinomial Naïve Bayes** (**MNB**), were employed. The MNB model with Term Presence vectorization yielded the best results, with an accuracy of 64.42% and a Hamming loss of 0.1.

Table 1 summarizes the datasets used and their details and Table 2 summarizes the previous work on Arabic app review classification.

In reviewing the existing literature, we have identified several critical gaps that our study aims to address:

[3] https://play.google.com/store/apps
[4] https://www.apple.com/sa/app-store

Table 1. Summary of the Datasets Used for Arabic App Reviews Analysis and Classification

Reference	Year	Dataset	Task	Dialect	Apps Category	Number of Apps	Source	Number of Reviews
Chader et al. [17]	2021	Collected by Authors	Sentiment Analysis	Algerian	–	–	Google Store	50,000
Saudy et al. [54]	2022	MASR	Sentiment Analysis	Egyptian	9 Categories	12	Google Store	2,469
Voskergian and Saheb [63]	2022	AMAR_ABSA	Multi-label Aspect-Based Sentiment Analysis	Multi-Dialects	Musical Apps	3	Google Store	100,000
Al-Hagree and Al-Gaphari [5]	2022	Collected by Authors	Sentiment Analysis	Yemeni	Bank Apps	8	Google Store	3,192
Hadwan et al. [32]	2022	Arb-AppsReview-V1	Sentiment Analysis	Multi-Dialects	COVID-19 Government Apps	6	Google Store	7,759
Hadwan et al. [33]	2022	Arb-AppsReview-V2	Sentiment Analysis	Multi-Dialects	COVID-19 Government Apps	6	Google Store	51,767
Ramzy and Ibrahim [52]	2024	Collected by Authors	Sentiment Analysis	Multi-Dialects	COVID-19 Government Apps	18	Google Store	114,599
This study	2024	AURA-Sentiment	Sentiment Analysis	Multi-Dialects	99 Categories	306	Google and Apple Stores	29,700
	2024	AURA-Classification	Reviews Classification	Multi-Dialects	99 Categories	306	Google and Apple Stores	2,900

— **Limited Availability of Arabic App Review Classification Datasets:** There is a notable scarcity of datasets in the area of Arabic app reviews, particularly those designed for classification tasks. The majority of available datasets are focused exclusively on sentiment analysis. To bridge this gap, we introduce two comprehensive datasets: one focused on sentiment analysis of Arabic app reviews, and another dedicated to the classification of such reviews into four main categories.

— **Lack of Diversity in Existing Datasets:** Prevailing datasets typically concentrate on a narrow selection of categories, encompassing a limited number of apps. Furthermore, all these datasets source their reviews from the Google Play Store, neglecting the perspectives of Apple App Store users. In contrast, our datasets aggregate reviews of 306 apps across 99 categories from both the Google Play and Apple App Stores, rendering it the most diverse and encompassing dataset in the field.

Table 2. Summary of the Works on Arabic App Review Analysis and Classification

Reference	Dataset	Task	Best Model	Results
Chader et al. [17]	Collected by Authors	Sentiment Analysis	SVM	Acc= 72%
Saudy et al. [54]	MASR	Sentiment Analysis	Hybrid RF-LR-MLP	Acc= 74.8% F1-score= 73.5%
Voskergian and Saheb [63]	AMAR_ABSA	Multi-label Aspect-Based Sentiment Analysis	MNB	Acc= 64.42% Hamming Loss= 0.1
Al-Hagree and Al-Gaphari [5]	Collected by Authors	Sentiment Analysis	NB	Acc=89.65% F1-score= 88.25%
Hadwan et al. [32]	Arb-AppsReview-V1	Sentiment Analysis	KNN	Acc= 78.46% F1-score= 78.96%
Hadwan et al. [33]	Arb-AppsReview-V2	Sentiment Analysis	SVM	Acc= 94.38% F1-score= 94.30%
Al-Hagree and Al-Gaphari [6]	Arb-AppsReview-V2	Sentiment Analysis	NB-LD	Acc=96.4%
Al-Shalabi et al. [8]	Arb-AppsReview-V2	Sentiment Analysis	KNN-LD	Acc=88.11% F1-score= 73.53%
Ramzy and Ibrahim [52]	Collected by Authors	Sentiment Analysis	ANN	Acc= 89% F1-score= 89%
This study	AURA-Sentiment	Sentiment Analysis	MarBert	Acc= 89% F1-score= 89%
	AURA-Classification	Reviews Classification	MarBert	Acc= 67% F1-score= 62%

— **Underexplored Arabic App Review Classification:** Existing studies focus on sentiment analysis of Arabic app reviews, with minimal attention paid to classification tasks. Given that app reviews play a crucial role in identifying bugs and enhancing app quality, incorporating classification analysis can significantly advance the field and aid developers. Our research, therefore, explores both sentiment analysis and classification of Arabic app reviews.

— **Underutilization of RNNs and Transformers:** RNNs and transformers have established themselves as paramount models in NLP [2], adept at managing textual data. Despite this, their app in the analysis and classification of Arabic app reviews remains underexplored. Our study seeks to fill this void by investigating the efficacy of RNNs and transformers in this domain.

By addressing these gaps, we aim to contribute valuable insights and resources that will enhance the field of Arabic app review analysis and classification.

3 AURA Dataset

This section presents the AURA dataset we prepared for sentiment analysis and classification of app user reviews in Arabic. In the following subsections, we will present the criteria employed for data collection and details regarding the data collection.

3.1 Criteria of Data Collection

Defining criteria for data collection requires surveying previous literature to determine the guidelines and rules applied for selecting mobile apps. Based on that, we decide and define our criteria. Criteria for selecting mobile apps can vary based on the objectives of the research work. The most common practice used in the literature was to choose apps based on app popularity and rating [27, 30, 42, 43, 51, 57]. This study aims to collect and construct a dataset of Arabic app reviews by focusing on apps that are widely used among Arab users. Additionally, we defined our criteria for selecting mobile apps as follows:

(1) Select popular apps from both platforms (iOS and Android) to ensure the generalization of app reviews.
(2) Apps were selected from different categories to ensure diversity in the collected app reviews.
(3) Select government apps that are developed by the government of Saudi Arabia to target more Arab users.

After applying these criteria, the total number of selected apps was 306 apps, where 191 were Android apps and 115 were iOS apps. Additionally, the included government apps are 42 apps from both platforms.

3.2 Data Collection

App reviews can be found on major app distribution platforms, such as Google Play Store (for Android) and Apple's App Store (for iOS). These platforms serve as the primary repositories for mobile apps on the two leading mobile operating systems globally. In previous research, some studies have employed manual web crawling techniques to extract app reviews directly from these repositories.

In this work, we used manual web scrapping techniques[5] to obtain the list of top apps from the official websites of Google's Play Store and Apple's App Store. We also applied manual web scrapping on the official Saudi national portal for government services[6] to collect the list of Saudi government apps. After that, we collected Arabic app reviews for these apps from both the Android and iOS platforms. For Google's Play Store, we used a Python tool called Google Play Scraper,[7] which supports different languages, including Arabic. For Apple's App Store, we used another Python tool called the App Store Scraper.[8]

3.3 AURA-Sentiment Dataset

Sentiment analysis is the action of assigning a quantitative (positive or negative) value to a short text reflecting the emotions or mood of the author [30]. We plan to utilize supervised ML techniques to analyze the sentiments of Arabic app reviews, which requires labeling the dataset beforehand. In this section, first, we will explain the detailed process for annotating the dataset. After that, we will discuss the data preparation details. Finally, details and statistics about the dataset will be presented.

3.3.1 Data Annotation for the AURA-Sentiment Dataset. We used the star rating field from each review to automatically assign sentiments for app reviews. There are two approaches. The first one is to assign "positive" sentiment for reviews with ratings of five and four stars and assign "negative" sentiment for reviews with ratings of one and two stars. In the second approach, the "positive"

[5] https://www.crummy.com/software/BeautifulSoup
[6] https://www.my.gov.sa/wps/portal/snp/content/mobileGovernment
[7] https://github.com/JoMingyu/google-play-scraper
[8] https://github.com/cowboy-bebug/app-store-scraper

Table 3. Accuracy of Automatically Labeling the Sentiments of App Reviews Based on Star Rating

Star Rating	# of Reviews	Correctly Classified	Wrongly Classified	Accuracy
1	100	95	5	0.95
2	100	73	27	0.73
4	100	69	31	0.69
5	100	84	16	0.84

Table 4. Comparison of Two Approaches for Automatically Labeling the Sentiments of App Reviews

Approach	# of Reviews	Correctly Classified	Wrongly Classified	Accuracy
First Approach	100 / star	321	79	0.80
Second Approach	100 / star	179	21	0.90

sentiment is assigned to reviews with only five stars, and the "negative" sentiment is assigned to reviews with only one star. Reviews with ratings of three stars ("neutral" sentiment) are ignored in both approaches because our sentiment classification problem is a binary classification problem considering only positive and negative sentiments.

To select one of the two approaches, we conducted an experiment to determine which star rating class is more accurate in reflecting the sentiment of the review text. In our experiment, we randomly selected 400 reviews from our dataset and 100 reviews from each star rating class except the three-star class because it is not included in either approach. After that, two participants were trained and asked to manually label the 400 reviews as "positive" or "negative" sentiment. To avoid bias in the labeling process, the samples were randomly reordered and the star rating column was removed from the data. Then, the prepared data were given to each participant to carry out the labeling process independently. After the labeling process was finished by both the trained participants, they resolved all conflicts in the labeling and agreed on the final labels for all reviews. The final version of the manually labeled data will be used as a ground truth to compare the two approaches of automatic labeling. Finally, the same 400 reviews were automatically labeled based on the star rating of each review, where reviews with five and four stars were labeled as "positive" and reviews with one and two stars were labeled as "negative". The accuracy of the automatic labeling process is computed and presented in Table 3. It is clear from the table that using reviews of five and one stars only to assign sentiments is significantly more accurate than using reviews of two and four stars. Hence, we selected the second approach to automatically label our data because it uses reviews of only one and five stars, and it is 10% more accurate than the first approach as shown in Table 4.

3.3.2 AURA-Sentiment Dataset Preparation. To prepare our data for the sentiment analysis task, we first filtered out reviews with short text lengths of two words or less. The resulting dataset size compared to the original size was 192,171 out of 418,804 Android reviews and 23,746 out of 34,435 iOS reviews. The second step is to select reviews with a star rating of only five stars and one star, where reviews with five stars will be labeled as "positive" and reviews with one star will be labeled as "negative". In the final step, data balancing was performed using undersampling between the "positive" and "negative" classes. Table 5 presents the size of our app review dataset throughout these steps.

Table 5. AURA-Sentiment Dataset Sizes During Preparation Steps

Platform	Original Size	Short Reviews Excluded	After Balancing
Android	418,804	192,171	14,850
iOS	34,435	23,746	14,850
Total	453,239	215,917	29,700

Table 6. Samples of Reviews from the AURA-Sentiment Dataset

Review Text	Translation	Sentiment	App Name	Platform
تطبيق رائع جدا	Very wonderful app	positive	Tawakkalna	ios
تطبيق رائع وهام جدا .. شكرا للقائمين عليه	A very wonderful and important app.. thanks to those in charge of it	positive	Absher	android
اعلانات مره كثير ومزعجه	Too many annoying ads	negative	Youtube	ios
سيئ جدا جدا..سوق كان افضل منه	Very very bad... Souq was better than it	negative	Amazon	android

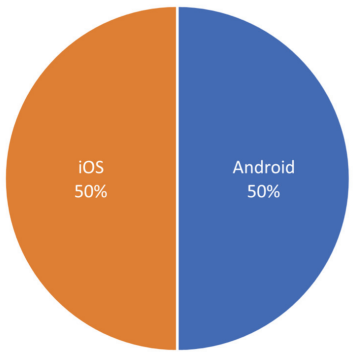

Fig. 1. Distribution of platforms in the AURA-Sentiment dataset.

3.3.3 Statistics on the AURA-Sentiment Dataset. The AURA-Sentiment dataset is a balanced dataset that contains 29,700 reviews where there are 14,850 reviews for each sentiment class. Arabic app reviews in this dataset were collected from 306 different apps on both platforms, and the distribution of platforms is presented in Figure 1. Samples of the dataset are available in Table 6. Some important statistics about the dataset are described in Table 7, which indicates that the dataset contains 337,816 words representing a rich corpus for effective model training. It includes 51,094 unique words, reflecting a diverse vocabulary and the potential to capture language variations relevant to Arabic app reviews. Review lengths vary significantly, with the longest review containing 576 words and the shortest just 3 words, demonstrating that the dataset includes both concise and detailed reviews. On average, reviews are 11 words long, suggesting that most user feedback is brief, a common characteristic in app reviews where users express their opinions concisely.

Table 7. Important Statistics from the AURA-Sentiment Dataset

Field	Value
Number of words	337,816
Number of unique words	51,094
Maximum length	576
Minimum length	3
Average length	11

Table 8. Popular Categories of App Reviews from the Literature

Category Name	Usage Count	References
Improvement request	8	[28, 31, 62]
Bug report	6	[28, 42, 62]
Rating	5	[28, 31, 42]
Others	3	[28, 47, 62]

3.4 The AURA-Classification Dataset

App reviews serve as a valuable source of information for app developers, offering user feedback that includes bug reports, app praise, and feature requests. Organizing app reviews into distinct categories (e.g., bug reports and feature requests) is highly beneficial for developers. In this section, we outline the categories that will be utilized in the labeling process. After that, we will explain in detail the process of labeling app reviews into a predefined set of categories. Finally, we will present statistics and details of the dataset.

3.4.1 Data Annotation for the AURA-Classification Dataset. To define our set of categories for app reviews, we explored previous literature for commonly used categories of app reviews. A summary of popular categories of app reviews is presented in Table 8. The most common category is improvement requests which include recommending improvements to the app or requesting new features by the app users. The bug report category is related to problems in the app to be fixed, such as crashes and errors. In the rating category, users express their opinions about the app either by praising or dispraising the app. Finally, the other category contains reviews that do not fit into any of the previous categories, such as spam and irrelevant reviews. Table 9 shows our identified set of app review categories with descriptions of each category. These categories were selected based on their usage popularity in the literature and their relevance to software development and evolution. Improvement request and bug report categories are important for developers and relevant to software maintenance and evolution, as reported by Panichella et al. [49] in their taxonomy of user review categories.

To construct our ML model for classifying app reviews, we need to label each review in our dataset to one of the predefined sets of categories. To accomplish this task, we utilized crowdsourcing services, with the Appen[9] crowdsourcing platform because it has quality control mechanisms

[9]https://appen.com

Table 9. Selected App Review Categories with the Description of Each Category

Category Name	Category Description
Improvement Request	Requesting new features, recommending enhancements in future versions of the app, asking for content (e.g., books and movies), and suggesting modifications for existing features.
Bug Report	Reporting problems in the app (e.g., crashes and errors).
Rating	Expressing opinion about the app by praising or dispraising.
Others	Reviews that does not fit any of the categories above (e.g., spam and noise reviews).

that will help us attain higher quality results. To ensure labeling quality, test questions were implemented as an entry quiz and to monitor contributors' performance during the job. Contributors with an accuracy below 90% were removed, and their work was discarded.

As a result of using the Appen platform, we labeled 2900 app reviews, and each review was labeled by five contributors. Thus, we have a total of 14,500 judgments (5 judgments per review). For each app review, we select the label (category) with the highest number of judgments (majority vote). Table 10 shows samples of the labeled app reviews with their associated five judgments. In the first four app reviews in the table, all five contributors agreed on the category of each app review. Additionally, in the fifth and sixth examples, there is a majority agreement on the category with four votes. In the seventh and eighth examples, there is a majority agreement on the "rating" category with three votes versus two votes for the "others" category, so both reviews will be labeled under the rating category. However, in the last example (No. 9), there is a clear tie between contributors on the "rating" and "others" categories with two votes for each, and such cases require manual resolution. To resolve tie cases, first, we extracted all tie cases and reordered them randomly. After that, the extracted reviews were given to an expert to manually label them without looking at the five judgments, to avoid bias while labeling. The expert labels for the tie cases were considered instead of the previous labels. Accordingly, following the above process, sample No. 9 in the table was labeled as "rating". To find all tie cases, we analyzed all the labeled data and found that there were approximately 400 tie cases out of 2900 reviews.

As a further step, we performed the auditing process for the dataset to estimate the labeling accuracy. First, we randomly reordered all the labeled app reviews. Second, we sampled 10% of the labeled reviews. Third, the reviews were given to an expert to label them manually. After manually labeling the reviews, we checked the manually labeled reviews against the labels obtained from the Appen platform. We found that the labeling accuracy is promising with an approximately 0.80 agreement.

Finally, to assess the quality of the labels, we employed the Consensus Agreement metric [21]. Consensus Agreement is a metric that evaluates how often the majority of annotators agree on the same label for a given sample. Given our objective to use majority voting to derive single labels for each sample, this metric was selected as it evaluates the proportion of items where a majority of annotators agreed on the same label. This approach provides a practical measure of alignment among annotators, focusing on the consistency and reliability of the labels without requiring full unanimity. To compute the Consensus Agreement, we applied Equation (1) [21] to each sample and then calculated the average across the entire dataset.

$$C_t = \frac{\sqrt{\sum_{i=1}^{K} \left(R_{i,t} - \frac{100}{N}\right)^2}}{\sqrt{\frac{K-1}{K}}}, \tag{1}$$

Table 10. Samples of the Labeling Output Using the Appen Platform

No.	App Review	Translation	Judgments				
			1	2	3	4	5
1	أتمنى إضافة خرائط التغطية في البرنامج لم تعد موجودة	I wish to add coverage maps in the app; they are no longer available.	IR	IR	IR	IR	IR
2	التطبيق لايعمل ويطلب التحديث	The app does not work and asks for an update.	BR	BR	BR	BR	BR
3	شي خرافي يستحق اكثر من خمسة نجوم	Amazing, deserves more than five stars.	R	R	R	R	R
4	سبحان الله و الحمد لله و لا اله الا الله والله أكبر	Glory be to Allah, praise be to Allah, there is no god but Allah, and Allah is the Greatest.	O	O	O	O	O
5	ممتاز لمنع التشتيت	Excellent for preventing distraction.	R	R	R	R	O
6	مع الاسف التطبيق لا يعمل ما الحل	Unfortunately, the app does not work; what is the solution?	BR	BR	BR	BR	O
7	شكرا تويو الى الامام	Thank you, Toyota; keep going forward.	O	R	R	R	O
8	التطبيق مميز واتمنى يدعم تسجيل الدخول عن طريق البصمه	The app is great, and I wish it supported login through fingerprint.	R	R	R	IR	IR
9	اتمنى من الجميع استخدامة و تقييم جميع الدوائر الحكوميه و الخدميه	I hope everyone uses it and rates all governmental and service departments.	IR	R	R	O	O

Categories key: **IR**: Improvement Request, **BR**: Bug Report, **R**: Rating, **O**: Others.

where:

— C_t: The consensus metric for sample t, representing the level of agreement among the labels provided by the N annotators of that sample ($N = 5$ in our case).
— K: The total number of classes ($K = 4$ in our case).
— $R_{i,t}$: The proportion of labels assigned to class i for sample t.

The term $\frac{100}{N}$ denotes the expected proportion of labels for each class if the labels were evenly distributed across all the classes. The term $\frac{K-1}{K}$ is a scaling factor for normalization, ensuring that the metric is comparable across different numbers of classes (K). The analysis revealed a Consensus Agreement of 67.07%, indicating that for the majority of samples, more than half of the annotators reached a consensus. This level of agreement confirms the dataset's suitability for downstream tasks.

3.4.2 Statistics on the AURA-Classification Dataset. The AURA-Classification dataset for the Arabic app review classification is not balanced, and it contains 2900 reviews collected from 289 apps in Apple's app store and Google's Play store. Figure 2 is a pie chart presenting the distribution of platforms in the dataset, and Figure 3 is a bar chart describing the distribution of the categories in the dataset. From the bar chart, we can observe that 42% of the labeled reviews are under the rating category, and 28% are labeled as bug reports. The remaining reviews are almost equally distributed between improvement requests and other categories. Samples of reviews from

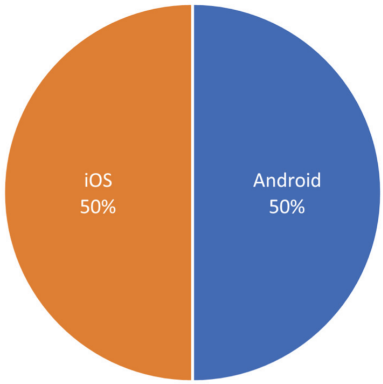

Fig. 2. Distribution of platforms in the AURA-Classification dataset.

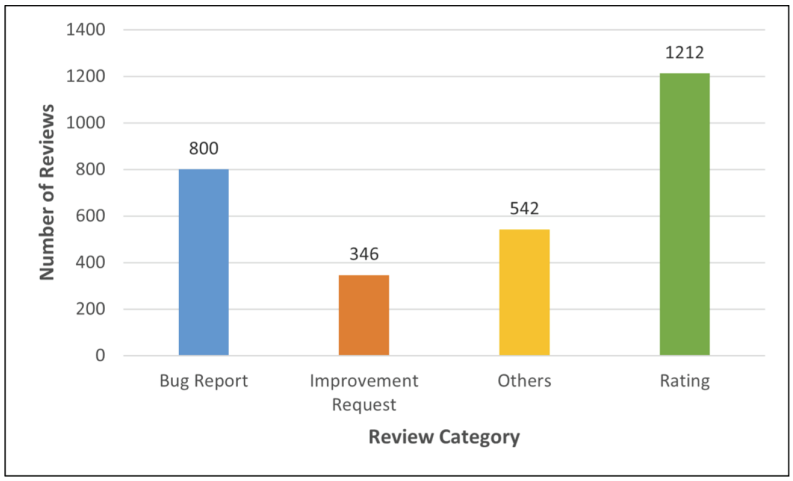

Fig. 3. Distribution of categories in the AURA-Classification dataset.

the dataset are shown in Table 11. Some important statistics about the dataset are described in Table 12, showing that it contains 39,300 words in total, with 12,337 unique words. The reviews range in length from 3 to 576 words, with an average review length of 13 words, indicating that feedback tends to be brief and to the point, typical of app review content.

4 Experiment Methodology

This section discusses the methodology of our sentiment analysis and reviews classification experiments. First, we applied and discussed the preprocessing steps for our datasets. Next, we provide details about our classification models. Finally, we describe the performance measures used to evaluate model effectiveness.

4.1 Data Preprocessing

Preparing the data was essential to ensure that the model learning process was performed correctly. Typically, data contain additional information that is unrelated to the context, which can cause overfitting [61]. To improve the performance of our DL models, we employed some basic

Table 11. Samples of Reviews from the AURA-Classification Dataset

Review Text	Translation	Category	App Name	Platform
نحتاج إلى اللغة العربية في اللعبة	We need the Arabic language in the game.	IR	8 Ball Pool	android
اتمنى اضافة خيار خدمة خرائط التغطية في التطبيق	I hope to add the coverage maps option in the app.	IR	CITC	ios
تعديل الدخل علطول يسجل خروج فيه مشكله	Adjusting the income always logs out; there's an issue.	BR	Citizen Account	android
البرنامج يعلق كثير الفتره الاخيرة	The app freezes a lot lately.	BR	Google Maps	ios
ممتاز جدا يستحق أكثر من ممتاز	Very excellent, deserves more than excellent.	R	ALKAHRABA	android
البرنامج بدائي جداً وفاشل	The app is very primitive and a failure.	R	Mawared	ios
كيفية إعادة تعيين كلمة السر وشكرا	How to reset the password, thank you.	O	Ashanak	android
الله يتقبل من المعتمرين	May Allah accept from the pilgrims.	O	Eatmarna	ios

Categories key: **IR**: Improvement Request, **BR**: Bug Report, **R**: Rating, **O**: Others.

Table 12. Important Statistics from the AURA-Classification Dataset

Field	Value
Number of words	39,300
Number of unique words	12,337
Maximum length	576
Minimum length	3
Average length	13

preprocessing techniques, including text cleaning. For the RNN models, we performed more preprocessing including, stopword removal, Arabic letter normalization, and lemmatization. Details of each preprocessing step are explained in the following sub-sections.

4.1.1 Text Cleaning. App reviews are mostly written using limited smartphone keyboards, thus they include noise such as numbers, extra symbols, and/or non-Arabic words [27]. Our goal in this step is to clean Arabic reviews from such noise and to produce more meaningful and analyzable reviews for our ML models. We used regular expressions to perform most of the following cleaning tasks:

— Remove all **uniform resource locators** (**URLs**) from each app review.
— Remove numbers from each app review.
— Remove punctuation characters using the list of punctuation provided by Python. We also added Arabic punctuation characters such as reversed question marks.

Table 13. Example of Applying Preprocessing Steps on Our Dataset

Original App Review	أقولُ شكراً جزيلاً عَلىَ هَذا التَطبيق الجَميـــــــل والرائع!!
After Text Cleaning	أقول شكرا جزيلا على هذا التطبيق الجميل والرائع
After Stopwords Removal	أقول شكرا جزيلا التطبيق الجميل والرائع
After Normalization	اقول شكرا جزيلا التطبيق الجميل والراءع
After Lemmatization	اقول شكر جزل تطبيق جميل راءع

— Strip diacritical marks in the Arabic language (also called Tashkeel), and strip Kashida which is a type of justification used in the Arabic language (also called Tatweel).
— Normalize Arabic letters (e.g., ا ، ة ، و ، ي). We mean by normalization here, transforming the letter from its different forms to its standard form. An example of,"ا " such transformation is to transform the letter "alif" from its several forms We used the PyArabic[10] Python library ."ا " to its standard form آ ، إ ، أ to perform normalization of Arabic letters.
— Remove non-Arabic words from each app review using the PyArabic Python library.

4.1.2 Stopword Removal. To filter out stopwords from our dataset, we used three lists of Arabic stopwords. The first list contains approximately 250 stopwords, and it was provided by the **Natural Language Toolkit (NLTK)** [15], a Python library. The second list is the largest publicly available[11] list of Arabic stopwords, which includes 750 words. As a third list, we added our selves a few words that we think will not affect the semantics of the sentence when removed, such as app, application, and game.

4.1.3 Lemmatization. Lemmatization is the activity of converting different forms of a word to their roots or, أنصح ، ينصح (advises and I advise), basic form by splitting off prefixes and affixes [3, 41]. For example, after lemmatization, نصح (advise) can become أنصحهم (I advise them) and نصحتكم (I advised you all). This process can help ML classifiers by uniting words with similar meanings but various language forms, thereby reducing the number of unique words in the dataset [30]. This will assist the classifier in focusing on the most important words in the dataset.

To summarize, Table 13 presents an example of applying data preprocessing steps to a sample review.[12] In the text cleaning step, Arabic diacritical marks, Tatweel, and punctuation marks, such as على (on) and هذا (this), were removed. In the normalization step, the Alef letter أ was normalized to its original form ا, and the letter ئ was normalized to its original form ء. In the lemmatization step, five Arabic words were lemmatized.

Finally, the impact of applying data preprocessing reduced the number of unique words in the sentiment analysis dataset by 63% and in the review classification dataset by 68%, as explained in Table 14. This minimizes the number of parameters the ML classifier needs to learn, resulting in a more robust training process.

4.2 Deep Learning Models

In this work, we utilized three RNN models and three transformers for both sentiment analysis and review classification tasks. RNNs were chosen for their strength in capturing sequential word dependencies within sentences, making them well-suited for tasks that require an understanding

[10] https://pypi.python.org/pypi/pyarabic
[11] https://github.com/mohataher/arabic-stop-words
[12] I say thank you very much for this beautiful and wonderful app!!

Table 14. Impact of Text Cleaning and Lemmatization on the Number of Unique Words

Original dataset	# of Unique Words	Reduction Rate
Original dataset	51,094 words	–
Cleaning only	39,869 words	22%
Lemmatization only	21,238 words	58%
Cleaning + Lemmatization	18,836 words	63%

of word order, such as sentiment analysis and text classification. However, RNNs face challenges in handling long sequences due to inherent memory limitations. To address this, we incorporated pre-trained transformers into our experiments. Leveraging the self-attention mechanism, transformers capture relationships between words across long contexts. Additionally, by generating contextualized embeddings, they more accurately capture the meanings of words based on their surrounding context, making them particularly effective at modeling complex linguistic structures. This capability is especially beneficial for text classification, where contextual nuances are crucial. Moreover, pretrained transformers require less training data, making them highly effective in scenarios with limited data availability. In the following subsection, we provide a detailed discussion of each model.

4.2.1 RNN Models. An RNN is a deep learning system that is utilized in different NLP problems [48]. RNN models stimulate the human brain when reading and understanding sentences. This model adopts the same principle of the human brain when reading sentences, word by word, by holding over memories of previous words, which gives the reader a better understanding of the meaning implied by the sentence [20]. RNNs suffer from gradient-exploding and gradient-vanishing problems [35] which make these models more difficult to train especially when capturing long-term dependencies. To address this issue, the **Gated Recurrent Unit** (**GRU**) has been proposed and has shown high effectiveness in various NLP tasks [2]. Another variant of the unidirectional GRU is the **Bidirectional Gated Recurrent Unit** (**Bi-GRU**), which keeps memories from previous states as well as future states and this will improve the accuracy of the final output [19].

Three different RNN models were utilized for sentiment analysis experiments, namely, RNN, GRU, and Bi-GRU. The input of our models is one app review converted to sequences of **word vectors** (**wv**) using word-level one-hot encoding. The first hidden layer in all models is an embedding layer, which is used to represent the words of each app review as dense vectors. Words with similar meanings will have similar wv. The embedding layer can be trained or pre-trained. Weights are randomly initialized for trained word embeddings and updated during training, while in pre-trained word embeddings, weights are precomputed. For most experiments, pre-trained word embeddings were used. We leveraged the pre-trained AraVec[13] model that was trained by Soliman et al. [56] on the Twitter dataset. The dimension of the pretrained embeddings is 300 and it contains approximately 1.5 million words (1,476,715 words).

The output of the embedding layer is the input for the next layer of each model, RNN, GRU, or BiGRU. Each of these layers has 128 units. Next, the output from the neuron layers is passed to a Dense [37] layer with 128 neurons followed by a dropout [58] layer with a 0.7 dropout value. The dropout layer is used to reduce the overfitting of the model by randomly activating and deactivating some neurons in these layers. For the sentiment analysis task, the last layer is

[13]https://github.com/bakrianoo/aravec

Table 15. Summary of the Optimal Hyperparameters for the RNN Models

Hyperparameter	Value
Embedding Dimension	300
Units in RNN Layers	128
Dense Layer Neurons	128
Dropout Rate	0.7
Optimizer	Adam
Learning Rate	1e−3

a dense layer with one neuron that combines all neurons from the previous layer into a fully connected layer and uses a sigmoid as an activation function to ensure that the output is either 0 (negative sentiment) or 1 (positive sentiment). Binary cross entropy was used as a loss function for all models. The sigmoid function can be calculated using Equation (2):

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2}$$

For the classification task, a dense layer with one neuron combines all neurons from the previous layer to a fully connected layer and uses softmax as an activation function, which calculates the probability for each class. The final output of the model will be the class or category which has the highest probability. The softmax function can be calculated using Equation (3):

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{3}$$

where z is the input vector to the Softmax function, and each zi is an element of that input vector. K is the number of classes in the multiclass model. The term in the denominator of the equation is the normalization term, used to ensure that each output value is in the range between 0 and 1, and all outputs sum up to 1. Categorical cross entropy was used as a loss function for most models. For imbalanced datasets, a special loss function was used, called Focal Loss [40], which penalizes instances that are easy-to-classify more lightly than hard-to-classify instances. The selected optimizer was Adam [38] with a learning rate of 1e-3 for optimizing the parameters of the model [44]. Table 15 summarizes the hyperparameters of the models.

4.2.2 Pretrained Transformer Models. Transformers architecture is based on employing a self-attention mechanism, enabling the model to efficiently process and relate all parts of a sequence. simultaneously. This architecture comprises encoder and decoder components for handling various tasks. Encoders are used for understanding and representing input data, while decoders generate output sequences for apps such as text translation and generation. The Transformers library by Hugging Face[14] provides a range of models, including those with just an encoder, just a decoder, or both encoder and decoder components. Encoder-only models are suitable for situations where only the encoding of input sequences is needed. For tasks such as text classification encoder-only models, such as AraBert [12], MarBert [4], and CamelBert [1], are appropriate. The three models were pre-trained on large amounts of Arabic corpora of MSA and different dialects. In this

[14]https://huggingface.co

Table 16. Summary of the Optimal Hyperparameters for the Pretrained Transformers

Hyperparameter	Value
Dense Layer Neurons	128
Dropout Rate	0.3
Optimizer	Adam
Learning Rate	1e−5

study, we fine-tuned the three transformers using our datasets and evaluated their performance. For each model, we added two dense layers and one dropout layer, and the output layer. Table 16 summarizes the hyperparameters of the models.

4.3 Classification Specific Techniques

Due to the small size and imbalanced classes of the AURA-Classification dataset, some extra techniques were applied to overcome these problems. In this subsection, these techniques are discussed in detail.

(1) **Under-sampling:** To balance our dataset, we performed under-sampling on the training dataset by randomly reducing app reviews of each category to the number of app reviews of the minority category.

(2) **Focal Loss:** Another solution to deal with imbalanced datasets is to utilize special loss functions. In our experiment, we leveraged Focal Loss [40]. It applies down-weighting by penalizing easy-to-classify instances more lightly than hard-to-classify instances. Focal Loss can be represented using Equation (4) [40]:

$$FocalLoss = -\sum_{i=1}^{n}(1-p_i)^{\gamma}\log(p_i), \tag{4}$$

where p_i is the predicted probability for the true class, γ is the focusing parameter, and n is the number of samples [40].

(3) **Data Augmentation** The previous techniques tackled the issue of imbalanced datasets. Although some techniques were applied, the data scarcity problem has not yet been resolved. One solution is to label more data for our classification task, but it becomes unlikely because of the shortage of manual labor and the limited resources of this research work. Alternatively, data augmentation techniques were used to produce more training data from our original training dataset. Data augmentation is the process of generating artificial data from available datasets. Augmentation in the NLP field is still immature in contrast to the Computer Vision field [55]. To perform data augmentation, we utilized a pre-trained word embedding model to produce new app reviews that are semantically similar to the original app reviews and we copied the same assigned label from the old app review to the newly produced review. The pre-trained AraVec model was utilized in our data augmentation process. Examples of augmented Arabic app reviews are presented in Table 17.

Two types of data augmentation were applied: **Flat Augmentation** (**FA**) and **Balanced Augmentation** (**BA**). In FA, all app reviews were multiplied by the same multiplication factor, resulting in an unbalanced training set, while in BA, reviews were multiplied by different multiplication factors based on the size of each category, producing a balanced training set.

Table 17. Examples of Augmented Arabic App Reviews

Review Text	Translation	Augmented Text	Augmented Translation	Label
دائما يطلب التحديث	Always asks for updates.	دائما يريد التحديث	Always wants updates.	BR
التطبيق سهل وجميل والحلو فيه انه يدعم العربية	The app is easy and nice, and the good thing is it supports Arabic.	التطبيق صعب وجميل لذيذ في انه ويدعم العربية	The app is difficult and lovely; the great thing is it supports Arabic.	R

Labels key: **IR**: Improvement Request, **BR**: Bug Report, **R**: Rating, **O**: Others.

Table 18. Example of Data Augmentation Process to Balance the AURA-Classification Dataset

	Data Size	BR	IR	O	R
Original Data	3000	700	100	500	1700
FA Multiplication Factor	×2	×2	×2	×2	×2
Output Data	6000	1400	200	1000	3400
Original Data	3000	700	100	500	1700
BA Multiplication Factor	×2	×3	×15	×3	×1
Output Data	6000	1500	1500	1500	1500

BF: Bug Report, IR: Improvement Request, O: Others, R: Rating, FA: Flat Augmentation, and BF: Balanced Augmentation.

We experimented with three multiplication factors (2, 5, and 10). Table 18 provides an example of the datasets produced using a multiplication factor of ×2, although all three factors were applied in the experiments. The distribution of the training set after under-sampling and applying FA and BA is illustrated in Figure 4, and our best model was trained on the resulting datasets.

4.4 Evaluation Measures

To evaluate the performance of our classifiers, we utilized accuracy and F1-score as key metrics. Accuracy represents the ratio of correctly predicted app review samples to the total number of samples and is computed using Equation (5):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5}$$

Here, **True Positives** (**TP**) are instances where the model correctly identifies the positive class, while **True Negatives** (**TN**) correspond to instances where the model correctly identifies the negative class. **False Positives** (**FP**) occur when the model incorrectly classifies a negative instance as positive, and **False Negatives** (**FN**) occur when the model fails to classify a positive instance correctly.

In scenarios involving imbalanced datasets, where one class dominates in the distribution of app reviews, relying solely on accuracy can be misleading, as a model predicting only the majority

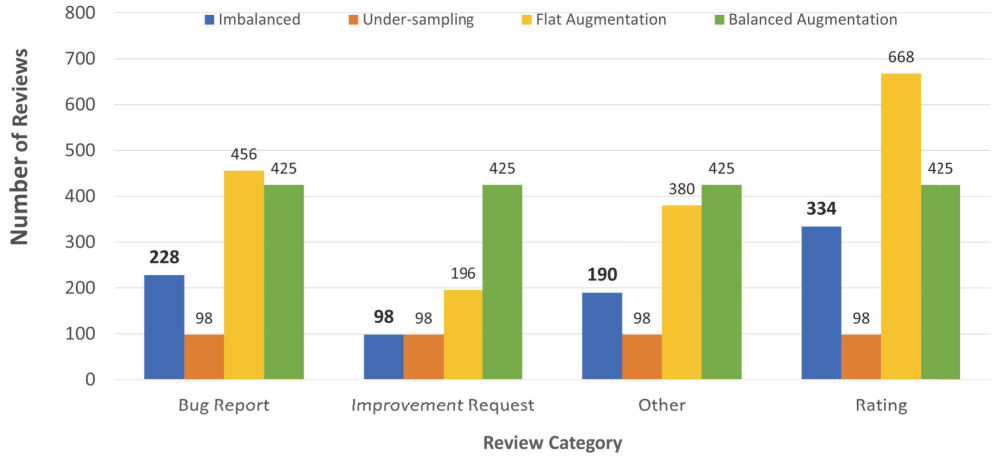

Fig. 4. Distribution of the AURA-Classification train set after applying different data balancing techniques.

class may still achieve high accuracy. To address this, precision and recall provide a more nuanced evaluation. Precision measures the proportion of TP predictions among all positive predictions, while recall quantifies the proportion of TP among all actual positive instances. These metrics are calculated using Equations (6) and (7), respectively:

$$Precision = \frac{TP}{TP + FP}, \tag{6}$$

$$Recall = \frac{TP}{TP + FN}. \tag{7}$$

To provide a balanced measure that considers both precision and recall, we use the F1-score, which is the harmonic mean of these metrics and is computed using Equation (8):

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{8}$$

This approach ensures a robust evaluation of classifier performance, particularly in the presence of class imbalance.

5 Experiments, Results, and Discussions

In this section, we discuss the experimental setup, the results of our experiments, and error analysis and discussion.

5.1 Experimental Setup

To implement our experiments, we used the Keras[15] library, [20], a Python library for deep learning based on TensorFlow.[16] TensorFlow is a Python library for ML and artificial intelligence apps provided by Google. To perform our experiments, we utilized Google Colab,[17] which is a research

[15]https://keras.io
[16]https://www.tensorflow.org
[17]https://colab.research.google.com

Table 19. Pilot Experiments for Sentiment Analysis and Review Classification to Select the Best Performing RNN Model

Task	Model	Accuracy	F1-score
Sentiment Analysis	RNN	0.7	0.7
	GRU	0.76	0.75
	BiGRU	**0.77**	**0.77**
Reviews Classification	RNN	0.43	0.27
	GRU	0.51	0.29
	BiGRU	**0.54**	**0.32**

project from Google suited for running ML projects on powerful hardware resources. The platform employs the Jupyter[18] Notebook service to run Python code interactively through the browser [16].

5.2 Experiments Results

In this section, we discuss the results of our experiments. We conducted pilot experiments using subsets of the training set to compare different RNN versions and to select the best-performing ones for sentiment analysis and review classification tasks.

We compared three RNN models, namely, RNN, GRU, and Bi-GRU to select the best model. For the sentiment analysis task, The Bi-GRU model outperforms other models with an accuracy score of 0.77 and 0.77 F1-score. Similarly, for the classification task, the Bi-GRU model outperforms other models with a 0.54 accuracy score and 0.32 F1-score. Table 19 presents the performance evaluation of the three models on both tasks. The F1-score recorded in the table is the macro average.

5.2.1 Sentiment Analysis. We selected the BiGRU model from the previous experiment and trained it using the complete dataset. The dataset size is 29,700 and it was divided as follows: 11,700 for training, 8000 for validation, and 10,000 for testing. Different embedding configurations were applied, including trained word embeddings, pre-trained word embeddings, and pre-trained word embeddings with fine-tuning. First, trained word embeddings were used, where weights were learned during the training process without setting weights from pre-trained models. Second, pre-trained word embeddings were utilized with static weights from a pre-trained model, and these weights were not updated during training. Third, pre-trained word embeddings with fine-tuning, where weights are initialized from a pre-trained model and updated during training.

Another set of experiments was conducted by fine-tuning AraBert, MarBert, and CamelBert models, each with its embeddings. Table 20 describes the output of the experiments. The best performance was 0.89 accuracy and 0.89 F1-score, achieved when using the MarBert model.

5.2.2 Reviews Classification. We selected the BiGRU model from the previous experiment and trained it using the complete dataset. The dataset size is 2900 and was divided as follows: 1000 for training, 900 for validation, and 1000 for testing. Two embedding configurations were applied, trained word embeddings and pre-trained word embeddings.

We utilized pre-trained word embeddings which improved the performance. After that, we used different techniques to improve our model's results, including using Focal loss, balancing the training dataset using under-sampling, and increasing and balancing the dataset by data augmentation.

[18] https://jupyter.org

Table 20. Models Performance on the AURA-Sentiment Dataset

Model	Accuracy	F1-score
BiGRU- Trained Embedding	0.86	0.86
BiGRU-Pre-trained Embedding	0.86	0.86
BiGRU-Pre-trained Embedding + Fine-tuning	0.87	0.87
AraBERT	0.88	0.88
MarBERT	**0.89**	**0.89**
CamelBERT	0.88	0.88

Table 21. Pilot Experiments with the Different Data Augmentation Techniques on the AURA-Classification Dataset to Select the Best Performing Technique

Model	Multiplication Factor	Accuracy	F1-score
BiGRU + Embed + Loss + FA	×2	0.63	0.55
	×5	0.67	0.58
	×10	**0.66**	**0.60**
BiGRU + Embed + Loss + BA	×2	0.63	0.56
	×5	0.64	0.58
	×10	**0.66**	**0.60**

The performance results of FA and BA experiments, using multiplication factors of 2, 5, and 10, are presented in Table 21. The multiplication factor ×10 resulted in the best performance for both augmentations. The last set of experiments utilized the AraBert, MarBert, and CamelBert models. Table 22 describes the results of our experiments. The best performance was 0.67 accuracy and 0.62 F1-score, achieved using the MarBert model.

In both tasks, the fine-tuned models surpassed the performance of other models, with the MarBert model achieving the highest performance. This outcome was anticipated since these models were pre-trained on a vast corpus of data. Specifically, in the case of review classification datasets, the BiGRU model without the application of any enhancement techniques demonstrated significantly lower performance. Conversely, the fine-tuned models exceeded their performance without the necessity for techniques such as data balancing or data augmentation. This discrepancy could be attributed to the dataset's limited size, which was insufficient for the BiGRU model to learn effectively.

5.3 Error Analysis and Discussion

In this section, we analyze and discuss wrongly classified samples by the MarBert model on both the sentiment analysis and classification datasets. Table 23 presents samples that were misclassified in the sentiment analysis dataset. For the first two examples from the table, the misclassification is due to incorrect labels during the data labeling process which was based on the star rating, in which the reviewer provided a star rating that does not reflect the content of the written app review. For example, in the first review, the reviewer wrote, "A very wonderful application", and he

Table 22. Models Performance on the AURA-Classification Dataset

Model	Accuracy	F1-score
BiGRU + Loss	0.58	0.32
BiGRU + Embed	0.65	0.50
BiGRU + Embed + Loss	0.64	0.51
BiGRU + Embed + Loss + Undersampling	0.56	0.51
BiGRU + Embed + Loss + FA ×10	0.66	0.60
BiGRU + Embed + Loss + BA ×10	0.66	0.60
AraBERT	0.66	0.61
MarBERT	**0.67**	**0.62**
CamelBERT	0.65	0.60

Table 23. Sample Misclassified Reviews in the AURA-Sentiment Dataset

ID	Review	Translation	Actual Label	Predicted Label
1	تطبيق رائع جدا	A very wonderful application	Negative	Positive
2	تحديث سيئ جدا	A very bad update	Positive	Negative
3	انها رائعه لكن مملة	It's wonderful but boring	Negative	Positive
4	حلو بس الاعلانات كثير	Nice, but there are too many ads	Positive	Negative
5	اللعبه لم تعد جميله	The game is no longer beautiful	Negative	Positive
6	لعبه محتاجه زكاء	A game that requires intelligence	Positive	Negative

provided a rating of one star for the app, which resulted in a wrong label. In short, our classification model classified these two examples correctly based on the review content but it was considered a misclassification case because of the incorrect labels provided by the reviewer. In the third and fourth misclassification examples, it is difficult to decide whether the review is positive or negative because each review contains two contradictory opinions. These examples are considered difficult to classify for both humans and machines. Finally, the last two app reviews were quite easy examples and were misclassified by the model. This can be attributed to the existence of some strong polar words in opposite contexts in each review, which confused the model.

Table 24 presents examples of misclassifications from the classification dataset. The first and second examples were incorrectly labeled in the dataset, yet the model correctly predicted their true labels. The third and fourth examples pose a challenge due to their potential applicability to two categories simultaneously, making the model's predictions understandable. However, the final two examples were correctly labeled in the dataset but were wrongly classified by the model.

Table 24. Sample Misclassified Reviews from the AURA-Classification Dataset

ID	Review	Translation	Actual Label	Predicted Label
1	جميل وسهل الاستخدام وفعال	Beautiful, easy to use, and effective	Others	Rating
2	نطلب اعاده الاغاني الخاصه بشركه روتانا	We request the return of the songs owned by Rotana company	Others	Improvement Request
3	ارجو تطويرها وهي ممتعه	I hope it gets developed further, it's enjoyable	Rating	Improvement Request
4	ارجو توقيف اشعار هذي اللعبه زفت	Please stop the notifications of this crappy game	Rating	Improvement Request
5	يعمل في الخلفيه وبدون اعلانات	It works in the background and without ads	Rating	Bug Report
6	ما بيرضي يحط صور ممكن تلاقولي حل	It doesn't allow me to post pictures, can you find me a solution	Bug Report	Others

5.4 Summary and Implications of the Work

The main findings of the work can be summarized as follows:

(1) The AURA dataset is a comprehensive resource of Arabic app user reviews, available in two versions. The first, AURA-Sentiment, is designed for sentiment analysis and consists of 29,700 samples. The second, AURA-Classification, is designed for text classification tasks and includes 2,900 samples. Both versions of the dataset were introduced in this article. The dataset will be made publicly available to researchers, aiming to advance the fields of Arabic app review analysis, classification, and Arabic NLP as a whole.
(2) Various RNN-based models were trained from scratch for app review sentiment analysis, with the BiGRU model using pretrained embeddings delivering the best performance. Furthermore, different transformer-based pretrained models were tested for the task, with MarBert achieving the highest performance, marked by an F1-score of 0.89.
(3) For Arabic app review text classification, the BiGRU model combined with pretrained embeddings, focal loss, and data augmentation provided the best results among RNN models trained from scratch. The overall best result, an F1-score of 0.62, was obtained using the pretrained MarBert model fine-tuned for the task.

The main implications of the work can be listed as follows:

(1) Better app review analysis and classification can lead to enhanced app quality and user experience. By automatically processing the reviews, the developer of an app can identify the most important issues and bugs the user is facing in order to fix them as soon as possible. Similarly, prioritizing the features requested by the users can help improve the app's quality and satisfaction.
(2) It can help companies perform better market and competitor analysis by automated analysis of competitor apps in order to understand the strengths of weaknesses of their app as compared to what is available in the market. It can also help identify common themes

and emerging trends in user feedback thereby influencing strategic decisions and product roadmaps.
(3) Automated analysis and classification enables the detection of patterns and trends within large volumes of reviews, facilitating deeper insights into user opinions and behaviors.
(4) Extracting actionable insights from user reviews contributes to the organization's knowledge base, supporting better decision-making processes.

6 Conclusion and Future Work

Analyzing app reviews is critical for app developers to understand and satisfy users' needs and to facilitate app development and maintenance tasks. It is practically infeasible for app developers to manually extract useful information from app reviews, especially with the massive volume of reviews per app and the amount of noise they include. To solve this problem, automatic approaches for analyzing and classifying app reviews have been proposed and discussed in the literature. Most existing research has discussed English app reviews, and a limited number of studies have explored the sentiment analysis of Arabic app reviews. In this work, we propose the AURA dataset, a large and diverse resource for Arabic app review sentiment analysis and classification tasks. The dataset has two versions: AURA-Sentiment which consists of 29,700 reviews annotated with positive and negative labels for sentiment analysis tasks, and AURA-Classification which contains 2,900 reviews labeled into four categories: bug reports, improvement requests, ratings, and others. Moreover, we trained and evaluated different models, namely, BiGRU, AraBert, MarBert, and CamelBert, on sentiment analysis and classification of Arabic app reviews. We also empirically evaluated the effectiveness of applying data augmentation on the model performance. In both datasets, the fine-tuned MarBert model showed superior performance with an accuracy of 0.89 and an F1-score of 0.89 on the sentiment analysis, and an accuracy of 0.67 and an F1-score of 0.62 on the reviews classification task. The main limitation of this work is the small dataset size, specifically for the reviews classification dataset. Possible future work related to the sentiment analysis of Arabic app reviews is to include a three-class classification problem (positive, negative, and neutral), a five-class problem by predicting the star rating of Arabic app reviews, and multi-label classification where one app review can be labeled as positive and negative. Moreover, the future work for Arabic app review classification is to extend the classification work to multi-label classification where one app review can belong to multiple categories.

Data Availability

The datasets used in this study are publicly available on the Hugging Face Hub. The AURA-Sentiment dataset can be accessed via the following link: https://huggingface.co/datasets/irfan-ahmad/AURA-Sentiment. The AURA-Classification dataset can be accessed via the following link: https://huggingface.co/datasets/irfan-ahmad/AURA-Classification.

References

[1] Amine Abdaoui, Mohamed Berrimi, Mourad Oussalah, and Abdelouahab Moussaoui. 2022. DziriBERT: A pre-trained language model for the algerian dialect. arXiv:2109.12346 [cs.CL]
[2] Mohammed M. Abdelgwad, Taysir Hassan A. Soliman, Ahmed I. Taloba, and Mohamed Fawzy Farghaly. 2022. Arabic aspect based sentiment analysis using bidirectional GRU based models. *Journal of King Saud University-Computer and Information Sciences* 34, 9 (2022), 6652–6662.
[3] M. Abdul-Mageed. 2019. Modeling Arabic subjectivity and sentiment in lexical space. *Information Processing & Management* 56, 2 (2019), 291–307.
[4] Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT MARBERT: Deep bidirectional transformers for Arabic. arXiv:2101.01785 [cs.CL]

[5] Salah Al-Hagree and Ghaleb Al-Gaphari. 2022. Arabic sentiment analysis based machine learning for measuring user satisfaction with banking services' mobile applications: Comparative study. In *Proceedings of the 2022 2nd International Conference on Emerging Smart Technologies and Applications (eSmarTA)*. IEEE, 1–4.

[6] Salah Al-Hagree and Ghaleb Al-Gaphari. 2022. Arabic sentiment analysis on mobile applications using Levenshtein distance algorithm and naive Bayes. In *Proceedings of the 2022 2nd International Conference on Emerging Smart Technologies and Applications (eSmarTA)*. IEEE, 1–6.

[7] Ibrahim Al-Jarrah, Ahmad M. Mustafa, and Hassan Najadat. 2023. Aspect-based sentiment analysis for Arabic food delivery reviews. *ACM Transactions on Asian and Low-Resource Language Information Processing* 22, 7 (2023), 1–18.

[8] Ahmed A. Al-Shalabi, Salah Alhagree, Ghaleb Al-Gaphari, and Fahd Alqasemi. 2023. Investigating the impact of utilizing the K-Nearest neighbor and Levenshtein distance algorithms for Arabic sentiment analysis on mobile applications. *Sana'a University Journal of Applied Sciences and Technology* 1, 2 (2023), 175–188.

[9] Faris Al-Smadi, Bashar Al-Shboul, Duha Al-Darras, and Dana Al-Qudah. 2022. Aspect-based sentiment analysis of Arabic restaurants customers' reviews using a hybrid approach. In *Proceedings of the 14th International Conference on Management of Digital EcoSystems*. 123–128.

[10] M. Al-Smadi, O. Qawasmeh, B. Talafha, and M. Quwaider. 2015. Human annotated Arabic dataset of book reviews for based sentiment analysis. In *Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, 726–730.

[11] Anwar Alnawas and Nursal Arici. 2019. Sentiment analysis of Iraqi Arabic dialect on Facebook based on distributed representations of documents. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 18, 3 (2019), 1–17.

[12] Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. 9–15.

[13] Gilbert Badaro, Ramy Baly, Hazem Hajj, Wassim El-Hajj, Khaled Bashir Shaban, Nizar Habash, Ahmad Al-Sallab, and Ali Hamdi. 2019. A survey of opinion mining in Arabic: A comprehensive system perspective covering challenges and advances in tools, resources, models, applications, and visualizations. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 18, 3 (2019), 1–52.

[14] Mohamed Berrimi, Mourad Oussalah, Abdelouahab Moussaoui, and Mohamed Saidi. 2023. Attention mechanism architecture for Arabic sentiment analysis. *ACM Transactions on Asian and Low-Resource Language Information Processing* 22, 4 (2023), 1–26.

[15] S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc..

[16] E. Bisong. 2019. *Google Colaboratory*. Apress, Berkeley, CA.

[17] Asma Chader, Leila Hamdad, and Abdesselam Belkhiri. 2021. Sentiment analysis in Google play store: Algerian reviews case. In *Proceedings of the 6th International Symposium on Modelling and Implementation of Complex Systems, MISC 2020*. Springer, 107–121.

[18] R. Chandy and H. Gu. 2012. Identifying spam in the iOS app store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*. 56–59.

[19] Y. Cheng, L. Yao, G. Xiang, G. Zhang, T. Tang, and L. Zhong. 2020. Text sentiment orientation analysis based on multi-channel CNN and bidirectional GRU with attention mechanism. *IEEE Access* 8 (2020), 134964–134975.

[20] F. Chollet. 2021. *Deep Learning with Python*. Simon and Schuster.

[21] Oscar Claveria. 2021. A new metric of consensus for Likert-type scale questionnaires: An application to consumer expectations. *Journal of Banking and Financial Technology* 5, 1 (2021), 35–43.

[22] J. Clement. 2020. Most Common Languages used on the Internet as of January 2020, by Share of Internet Users. Retrieved June 2020 from https://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/

[23] J. Clement. 2020. Number of Apps Available in Leading App Stores as of 1st Quarter 2020. Retrieved September 2020 from https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[24] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. 2018. App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In *Proceedings of the 2018 IEEE 26th International Requirements Engineering Conference (RE'18)*. IEEE, 170–181.

[25] E. Duffin. 2022. The Most Spoken Languages Worldwide. Retrieved February 2022 from https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/

[26] Ashraf Elnagar, Ridhwan Al-Debsi, and Omar Einea. 2020. Arabic text classification using deep learning models. *Information Processing & Management* 57, 1 (2020), 102121.

[27] C. Gao, J. Zeng, M. R. Lyu, and I. King. 2018. Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*. IEEE, 48–58.

[28] X. Gu and S. Kim. 2015. "What parts of your apps are loved by users?"(T). In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15)*. IEEE, 760–770.
[29] E. Guzman, O. Aly, and B. Bruegge. 2015. Retrieving diverse opinions from app reviews. In *Proceedings of the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'15)*. IEEE, 1–10.
[30] E. Guzman and W. Maalej. 2014. How do users like this feature? A fine grained sentiment analysis of app reviews. In *Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE'14)*. IEEE, 153–162.
[31] E. Ha and D. Wagner. 2013. Do android users write about electric sheep? Examining consumer reviews in Google Play. In *Proceedings of the 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC'13)*. IEEE, 149–157.
[32] Mohammed Hadwan, Mohammed Al-Hagery, Mohammed Al-Sarem, and Faisal Saeed. 2022. Arabic sentiment analysis of users' opinions of governmental mobile applications. *Computers, Materials and Continua* 72, 3 (2022), 4675–4689.
[33] Mohammed Hadwan, Mohammed Al-Sarem, Faisal Saeed, and Mohammed A. Al-Hagery. 2022. An improved sentiment classification approach for measuring user satisfaction toward governmental services' mobile apps using machine learning methods with feature engineering and SMOTE technique. *Applied Sciences* 12, 11 (2022), 5547.
[34] A. Heydari, M. Tavakoli, and N. Salim. 2016. Detection of fake opinions using time series. *Expert Systems with Applications* 58 (2016), 83–92.
[35] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[36] L. Hoon, R. Vasa, G. Y. Martino, J.-G. Schneider, and K. Mouzakis. 2013. Awesome! Conveying satisfaction on the app store. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*. 229–232.
[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4700–4708.
[38] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG]
[39] T.-P. Liang, X. Li, C.-T. Yang, and M. Wang. 2015. What in consumer reviews affects the sales of mobile apps: A multifacet sentiment analysis approach. *International Journal of Electronic Commerce* 20, 2 (2015), 236–260.
[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. 2020. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 2 (2020), 318–327.
[41] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. 2016. On the automatic classification of app reviews. *Requirements Engineering* 21, 3 (2016), 311–331.
[42] W. Maalej and H. Nabil. 2015. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE'15)*. IEEE, 116–125.
[43] H. Malik, E. M. Shakshuki, and W.-S. Yoo. 2020. Comparing mobile apps by identifying 'hot' features. *Future Generation Computer Systems* 107 (2020), 659–669.
[44] Sanidhya Mangal, Poorva Joshi, and Rahul Modak. 2019. LSTM vs. GRU vs. bidirectional RNN for script generation. arXiv:1908.04332 [cs.CL]
[45] A.-S. Mohammad, O. Qwasmeh, B. Talafha, M. Al-Ayyoub, Y. Jararweh, and E. Benkhelifa. 2016. An enhanced framework for aspect-based sentiment analysis of hotels' reviews: Arabic reviews case study. In *Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST'16)*. IEEE, 98–103.
[46] J. Oh, D. Kim, U. Lee, J.-G. Lee, and J. Song. 2013. Facilitating developer-user interactions with mobile app review digests. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 1809–1814.
[47] D. Pagano and W. Maalej. 2013. User feedback in the appstore: An empirical study. In *Proceedings of the 2013 21st IEEE International Requirements Engineering Conference (RE'13)*. IEEE, 125–134.
[48] Y. Pan and M. Liang. 2020. Chinese text sentiment analysis based on BI-GRU and self-attention. In *Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC'20)*. Vol. 1, IEEE, 1983–1988.
[49] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. IEEE, 281–290.
[50] Ashwin Perti, Amit Sinha, and Ankit Vidyarthi. 2024. Cognitive hybrid deep learning-based multi-modal sentiment analysis for online product reviews. *ACM Transactions on Asian and Low-Resource Language Information Processing* 23, 8, Article No. 113 (2024), 1–14.
[51] K. Phetrungnapha and T. Senivongse. 2019. Classification of mobile application user reviews for generating tickets on issue tracking system. In *Proceedings of the 2019 12th International Conference on Information & Communication Technology and System (ICTS'19)*. IEEE, 229–234.
[52] Mina Ramzy and Bahaa Ibrahim. 2024. User satisfaction with Arabic COVID-19 apps: Sentiment analysis of users' reviews using machine learning techniques. *Information Processing & Management* 61, 3 (2024), 103644.
[53] M. Rushdi-Saleh, M. T. Martín-Valdivia, L. A. Ureña-López, and J. M. Perea-Ortega. 2011. OCA: Opinion corpus for Arabic. *Journal of the American Society for Information Science and Technology* 62, 10 (2011), 2045–2054.

[54] Rabab Emad Saudy, M. Alaa El Din, Eman S. Nasr, and Mervat H. Gheith. 2022. A novel hybrid sentiment analysis classification approach for mobile applications Arabic slang reviews. *International Journal of Advanced Computer Science and Applications* 13, 8 (2022), 423–432.

[55] C. Shorten, T. M. Khoshgoftaar, and B. Furht. 2021. Text data augmentation for deep learning. *Journal of Big Data* 8, 1 (2021), 1–34.

[56] A. B. Soliman, K. Eissa, and S. R. El-Beltagy. 2017. AraVec: A set of Arabic word embedding models for use in Arabic NLP. *Procedia Computer Science* 117 (2017), 256–265.

[57] K. Srisopha, C. Phonsom, K. Lin, and B. Boehm. 2019. Same app, different countries: A preliminary user reviews study on most downloaded iOS apps. In *Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME'19)*. IEEE, 76–80.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[59] Statista. 2024. Number of Smartphone Users Worldwide from 2016 to 2023. Retrieved June 5, 2024 from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

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

[61] Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management* 50, 1 (2014), 104–112.

[62] L. Villarroel Pérez. 2015. *Mining mobile apps reviews to support release planning*. Master's Thesis. Universidad Politécnica de Madrid (UPM), Madrid, Spain.

[63] Daniel Voskergian and Mahmoud H. Saheb. 2022. AMAR_ABSA: Arabic mobile app reviews dataset targeting aspect-based sentiment analysis tasks. In *Proceedings of the 2022 Innovations in Intelligent Systems and Applications Conference (ASYU'22)*. IEEE, 1–7.

Received 28 July 2024; revised 30 November 2024; accepted 8 December 2024