Array 29 (2026) 100572

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

Exploring and identifying fine-grained accessibility issues in app store using fine-tuned deep learning

Mumrez Khan [a], Zhixiao Wang [a,*], Javed Ali Khan [b], Nek Dil Khan [c]

[a] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China
[b] School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield, UK
[c] Faculty of Information Technology, Beijing University of Technology, Beijing, China

ARTICLE INFO

Dataset link: https://github.com/nekdil566/accessibility_issues

Keywords:
App store reviews analysis
Accessibility issues detection
Deep learning classification
Software evaluation

ABSTRACT

The Apple App Store (AAS) allows users to provide feedback on applications, offering developers insights into improving software performance. Researchers have utilized this feedback for software evolution activities, including features, issues, and nonfunctional requirements. However, end-user feedback has not been explored to identify accessibility-related challenges. This study proposes an automated approach to detect and classify accessibility issues by analyzing end-user reviews in the AAS. We crawled 178667 user reviews from 85 apps across 18 categories to represent a diverse sample. We developed a coding guideline to identify common accessibility issues, including Navigation and Interaction Problems (NAV), Input and Control Issues (INPUT), Compatibility with Assistive Technologies (CAT), Audio and visual accessibility issues (AUDIOVISUAL), and UI Accessibility Issues (UI). We manually annotated reviews using coding guidelines and content analysis to create a labeled dataset for training and evaluating deep learning(DL) algorithms to detect accessibility in user comments and classify them into categories. The experiments showed that fine-tuned DL classifiers achieved high accuracy in detecting accessibility and classifying them into specific types. For binary classification, the CNN classifier achieved 93% precision, while LSTM, BiLSTM, GRU, and BiGRU achieved accuracies from 76% to 87%. In fine-grained classification, CNN performed better with 97% accuracy, followed by BiGRU and BiLSTM at 96%. The BiLSTM and LSTM models demonstrated strong performance, with accuracies of 96% and 95%. These results show the potential of automated methods to improve identification of accessibility challenges, helping developers address these issues effectively and enhance user experience.

1. Introduction

Mobile applications have become essential tools in modern life, serving critical roles in domains such as education, online banking, healthcare, and communication [1]. For improved user satisfaction and exclusivity, it is vital that these applications demonstrate easy access to the different services they offer, particularly for end users who suffer from specific disabilities. An estimated 1.3 billion (16%) of the world's population experience a particular disability [2], which is one percent higher than that reported in 2011 [3]. Software vendors must incorporate accessibility-related requirements when developing software applications to give a voice to people with disabilities. Recently, developers and researchers have proposed accessibility standards, guidelines, approaches, and tools to help software providers and developers incorporate accessibility-related requirements when developing software applications [1,4–6]. However, software developers face challenges in intertwining accessibility into existing software development and evolution processes due to a lack of resources and awareness [7]. In contrast, App Stores such as AAS allow users to share instant feedback on apps, providing developers with valuable information related to software evolution, including new features, issues, user experience, and accessibility [1,8,9]. However, identifying actionable feedback from a large volume of user reviews remains a significant challenge [10]. Although automated tools have been developed to analyze user feedback [9,11], existing approaches often overlook nuanced aspects, such as fine-grained accessibility issues, disproportionately affecting users with disabilities. This gap highlights the need for advanced methods to detect and classify accessibility-related feedback into frequently occurring accessibility issues, enabling developers to better understand user challenges and improve software exclusivity [12].

Moreover, existing studies on the analysis of end-user feedback for software evolution have focused mainly on sentiment analysis [11,

* Corresponding author.
E-mail addresses: mumrez.khan@stu.xaut.edu.cn (M. Khan), wangzhx@xaut.edu.cn (Z. Wang), j.a.khan@herts.ac.uk (J.A. Khan), nekdilkhan@emails.bjut.edu.cn (N.D. Khan).

https://doi.org/10.1016/j.array.2025.100572
Received 18 July 2025; Accepted 4 November 2025
Available online 11 December 2025

13], issue and bug detection and classification [14,15], and feature requests [16], leaving accessibility issues largely unexplored. Accessibility challenges in mobile applications remain persistent, particularly for users with visual, motor, and cognitive impairments. Common barriers include poor compatibility with screen readers, inadequate color contrast, unresponsive touch interfaces, and complex navigation structures [17]. Users in app store reviews frequently report these issues; however, they remain unexplored in automated feedback analysis systems. For example, Oliveira et al. highlighted the failure of many educational applications to effectively support screen readers, limiting their usability for visually impaired students [17]. Similarly, Alghamdi et al. emphasized the gaps in adherence to accessibility standards, such as WCAG, as identified through Stack Overflow posts and app reviews [18]. Despite the availability of guidelines such as the WCAG, ensuring compliance remains a persistent challenge for developers. In addition, Alomar et al. [7] proposed a machine learning-based approach that analyzes end-user feedback to detect accessibility-related review. In contrast, Aljedanni et al. [19] analyzed end-user reviews to classify them into various reviews of accessibility guidelines. However, fine-grained exploration and analysis of end-user reviews must be conducted to identify frequently occurring accessibility issues and improve app exclusivity. Addressing these gaps requires a systematic approach to automatically identify and classify accessibility-related feedback, allowing developers to prioritize inclusive design. This limitation might hinder developers' ability to prioritize and effectively address specific accessibility challenges [20]. In addition, more research is required to address accessibility issues, specifically in application stores such as Amazon software app (ASA) store, Google play (GP) store, and ASS [1]. However, relying on the manual analysis of end-user reviews is impractical because of the enormous volume of data, underscoring the need for automated solutions [21]. These gaps motivate the need for a more comprehensive and fine-grained approach to detect and classify accessibility issues in AAS reviews.

To address this gap, we propose a novel automated approach using fine-tuned DL classifiers to identify and classify accessibility issues that often occur in AAS feedback. A dataset containing 178667 user reviews was curated from 85 applications in 18 different categories, and a novel coding guidelines was developed to identify common accessibility issues, including NAV, INPUT, CAT, AUDIOVISUAL, and UI. Furthermore using the coding guidelines and manual content analysis, a subset of reviews was manually annotated to create a labeled dataset for training and evaluating DL classifiers for the fine-tuned process. The empirical results show that fine-tuned DL classifiers effectively detect accessibility issues and classify them into distinct classes with comparatively high accuracy. These results highlight the potential of the proposed approach to enhance the identification and understanding of accessibility issues in AAS reviews improvements and achieve inclusion following the guidelines provided by Apple [6] when developing mobile applications.

The contributions of this study are as follows:

- A comprehensive data set of end-user reviews annotated for accessibility issues provides a foundation for future studies.
- A novel coding guideline for categorizing accessibility issues, facilitating reproducibility and scalability.
- An evaluation of state-of-the-art fine-tuned DL classifiers to detect and classify accessibility issues shows their effectiveness in real-world scenarios.
- Insights into how accessibility issue detection can improve the performance of software applications for improved usability.

2. Related work

User comments from AAS, GP, and ASA stores are an important resource for identifying accessibility challenges faced by diverse end-user groups, including those with disabilities [22,23]. Systematic studies of app reviews show recurring accessibility issues, such as insufficient screen reader compatibility, unresponsive touch interfaces, impaired color contrast, and complex navigation, which disproportionately affect end users with visual, motor, or cognitive impairments. For instance, Oliveira et al. (2024) [17] highlighted the failure of many educational apps to support screen readers effectively, limiting their usability for visually impaired students, while Alghamdi et al. (2024) [18] showed gaps in observations of accessibility standards, such as WCAG, as identified through Stack Overflow posts and app reviews. Despite the wealth of information available in end-user reviews, analyzing it poses challenges, including the sheer volume and noise of unstructured reviews, the lack of specificity in describing accessibility issues, and the diverse needs of users with different disabilities. To address these challenges, some researchers have proposed automated tools for classifying accessibility-related reviews and involving end-users with disabilities in usability testing to ensure a more inclusive app design [19]. These understandings are of significant importance to developers, who can prioritize features such as customizable interfaces, alternative input techniques, commitment to accessibility standards, and policymakers to support stronger accessibility regulations [17].

Furthermore, forthcoming research should focus on developing state-of-the-art tools for automatic end-user review analysis, conducting longitudinal analyses to track accessibility progress over time, and investigating cross-platform accessibility challenges to provide inclusivity in Android app stores [24]. Using end-user feedback, developers and researchers can create more accessible applications, promote digital inclusion, and ensure that mobile applications address the needs of all end users, including those with a disability. Furthermore, research such as those of Reyes et al. (2022) [25] and Eler et al. (2019) [26] have demonstrated the efficacy of analyzing end-user comments on platforms such as GP Store to identify accessibility requirements related to vision, hearing, and cognition, further emphasizing the importance of addressing these concerns to improve user experience. By synthesizing findings of previous research, we gain a comprehensive understanding of the challenges developers face, underscoring the ongoing need for greater focus on accessibility in mobile applications [1,22,27].

Our proposed approach differs from existing state-of-the-art methods by focusing on detecting and classifying accessibility issues in AAS reviews, which have been largely ignored in prior studies. Our study presents a novel coding guideline that systematically classifies accessibility-related issues into five distinct types: UI, NAV, CAT, AUDIOVISUAL and INPUT. Similarly, fine-tuned DL classifiers, such as LSTM, BiLSTM, CNN, RNN, BIRNN, GRU, and BIGRU, are used to automatically detect and classify these issues with high accuracy. This fine-grained analysis, combined with advanced DL models, provides actionable insights into the most prevalent accessibility-related issues. Software developers can directly apply this understanding to prioritize and address issues more effectively, thereby improving the accessibility of applications for all end users. This approach represents a substantial advancement over existing methods, offering a more comprehensive and scalable solution for enhancing application accessibility.

3. Research methodology

This section outlines the research questions and proposed approach for detecting and classifying accessibility issues in AAS reviews. Through a systematic analysis of end-user reviews, this study aims to provide developers with actionable insights into accessibility issues. The foremost goal is to encourage more inclusive software innovation, thereby enabling a future in which software is accessible to all users

3.1. Research questions

RQ1: How do end-users express accessibility issues in AAS reviews?

To address RQ1, we conducted an essential study of end-user reviews in the AAS. We determined how end users express their concerns about accessibility issues in AAS software applications. To achieve this goal, we performed a manual analysis of user reviews in the AAS, which led to the development of novel coding guidelines. This novel coding guideline facilitates the detection and classification of accessibility issues into five primary types, promoting scholarly interest in their potential implications for future research.

RQ2: What commonly reported accessibility issues can be identified in AAS reviews?

RQ2 aims to analyze AAS reviews to elucidate the accessibility challenges reported by users and identify the various types of accessibility issues and their frequency in end-user feedback. To accomplish this objective, we systematically gathered end-user reviews from the AAS and manually annotated them in accordance with the developed coding guidelines and a content analysis approach [28]. Coding guidelines facilitate the identification of accessibility issues, including navigation problems, input difficulties, compatibility with assistive tools, and user interface-related challenges. These insights have the potential to improve software quality and facilitate a more comprehensive understanding and prioritization of accessibility-related requirements for software enhancement.

RQ3: How effective are fine-tuned DL classifiers in detecting accessibility issues?

Currently, end users submit numerous comments in the AAS, containing valuable information for software developers to enhance the quality of existing and future versions of software applications. However, the manual analysis of such a large volume of end-user comments is time-intensive and presents significant challenges. To address RQ3, we evaluated the performance of various fine-tuned DL algorithms in automatically detecting and classifying accessibility issues in end-user feedback. For this research question, we aimed to examine the performance of several state-of-the-art DL classifiers, including CNN, LSTM, BiLSTM, RNN, BIRNN, GRU, and BIGRU, to identify and categorize end-user feedback into accessibility-related issues automatically. The proposed approach assists developers in promptly identifying accessibility-related issues, thereby facilitating improved decision-making and enhancing software inclusivity and its usability.

3.2. Research methodology overview

The proposed research methodology for identifying and categorizing accessibility issues in AAS reviews to various frequently occurring accessibility-related issues is illustrated in Fig. 1. This schematic representation provides a visual overview of the three primary phases of this study. Initially, we developed a dataset by extracting user reviews from the AAS utilizing a Python script,[1] targeting 85 applications across 18 diverse categories, with details of each application presented in Table 1. Subsequently, we established a coding guideline[2] employing grounded theory and content analysis to categorize reviews into five types of accessibility issues: NAV, INPUT, CAT, AUDIOVISUAL, and UI to develop an annotated dataset for automated classification of end-user reviews in various kinds. Finally, we experimented with various fine-tuned DL classifiers, including LSTM, BiLSTM, CNN, RNN, BIRNN, GRU, and BIGRU, to automatically detect and classify accessibility issues after preprocessing the data and applying feature engineering techniques, and reported their results. The proposed methodology can be integrated with existing software evolution processes to address and enhance the accessibility and usability challenges of applications with low ratings. Each phase is comprehensively detailed in the subsequent sections.

[1] Pythonscript

[2] GroundedTheoryandCodingGuidelines

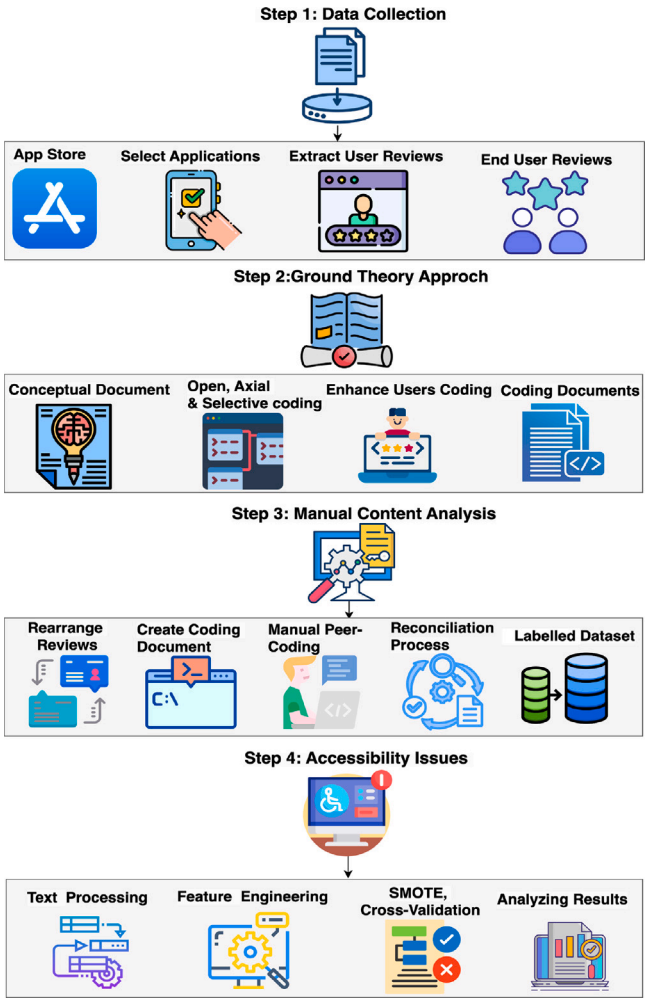

Fig. 1. Research Methodology Diagram.

4. Dataset creation and preparation

This section elucidates the development and preparation of the dataset employed to identify the continually emerging accessibility issues in AAS reviews. This also explains the creation of a detailed coding guideline that employs grounded theory and content analysis, serving as the foundation for annotating the dataset. The annotation process ensures that the dataset is structured and labeled for training and evaluating DL algorithms.

4.1. Data collection

A Python script was employed to extract user reviews from the AAS to develop a research dataset. The dataset contains user reviews of 85 applications across 18 diverse categories, ensuring a broad range of reviews on accessibility challenges. Each user review was extracted along with its associated metadata, including the author's name, review heading, rating, and complete review text. The details of the selected applications are listed in Table 1. The apps and categories were chosen to represent the overall needs of end-users as accurately as possible.

4.2. Grounded theory approach

To develop a systematic approach for identifying and classifying accessibility issues in app store reviews, we employed Strauss and

Table 1
Details of end-user reviews dataset.

No.	Applications Category	Applications Name	Total Reviews
1	Business Apps	1: WH Careers, 2: Uber Eats Orders, 3: Document Pro, 4: Print Works	2247
2	Health and Fitness	1: LA Fitness Mobile, 2: Crunch Fitness, 3: VeryFitPro, 4: DayBand	4778
3	Lifestyle Apps	1: Toyota, 2: Lexus, 3: LotusLanternX, 4: HappyLighting, 5:The Pattern, 6:Pinterest: Lifestyle Ideas, 7:Evite: Party Invitation Maker	20 641
4	References Apps	1: Ohio Lottery, 2: Dictionary, 3: Hallow: Prayer & Meditation, 4: Chat: Daily Devotional, 5: Google Translate	4750
5	Games Apps	1: Mobile Strike, 2: Bejeweled Blitz, 3: Super Mario Run, 4: Subway Surfers, 5: Paper.io	8500
6	Education Apps	1: Google Classroom, 2: Clever, 3: i-Ready Connect for Students, 4: RoadReady, 5: Edpuzzle, 6: Campus Student, 7:Five Star Student, 8:Aeries Mobile Portal, 9:Minga, 10:Schoology	33 358
7	Entertainment Apps	1: We TV, 2: VH1, 3: Lottery, 4: Meta Horizon, 5: Steam Mobile, 6: Lottery Player Color	14 881
8	Social Networking Apps	1: LYSN, 2: QQ, 3: Steam Chat, 4: LINE	7603
9	Sports Apps	1: Madden NFL 24, 2: HryFine, 3: Legends Arena PK, 4: A Sports HD, 5: FotMob - Football Live Scores	4453
10	Utility Apps	1: Love Spouse, 2: SmartHub, 3: SuperVPN Fast VPN Client, 4: Pi Browser, 5: ETA	3883
11	Photo and Video Apps	1: LumaFusion, 2: CapCut - Video Editor, 3: InShot - Video Editor, 4: iMovie, 5: Videoleap: AI Video Editor	17 303
12	Social Platforms	1: MyShake, 2: Emergency: Severe Weather App	1262
13	News Apps	1: Business Insider, 2: Stock Market and Business, 3: Bloomberg: Business News Daily, 4: Reddit, 5: X	19 000
14	Shopping Apps	1: Marketplace Buy and sell, 2: Daraz Online Shopping App, 3: Meesho:Online Shopping	2352
15	Productivity Apps	1: ESS 41 - Reflexis One, 2: 1Password: Password Manager	2904
16	Finance Apps	1: Connect Notwork, 2: The Check Cashing Store, 3: Tomo Card, 4: Wood Forest Mobile Banking, 5: Zenith Bank, 6:Access Bank, 7:First Mobile App, 8:Direct Auto Insurance	3871
17	Travel Apps	1: Allegiant, 2: Ventra, 3: Virgin Voyages	4587
18	Music Apps	1: RØDE Central Mobile, 2: Spotify for Artists, 3: Skullcandy	1316
Total Applications		85	178 667

Corbin's Grounded Theory methodology [29]. This qualitative and systematic approach facilitated the creation of a coding guideline tailored to the unique context of accessibility challenges reported by end users. This process involved an iterative analysis of user reviews to identify common patterns and themes related to accessibility issues. Initially, a random sample of 650 end-user reviews from the AAS was manually analyzed to identify frequently occurring accessibility issues. These issues were categorized into five primary types: UI, NAV, INPUT, AUDIOVISUAL, and CAT. The categories identified in this process closely align with the Web Content Accessibility Guidelines (WCAG),[3] ensuring that the study's findings and methodology contribute to universally recognized accessibility standards. We further refined each category through multiple rounds of review and validation to ensure clarity and consistency. The resulting coding guidelines, a result of the grounded theory-driven approach, include a comprehensive framework for annotating end-user reviews and creating a labeled dataset for training and evaluating DL models. It contains a wide range of accessibility issues and provides a comprehensive analysis. Similarly, the empirical validation of the guidelines showed their efficacy in improving inter-coder agreement and annotation accuracy. These guidelines not only enhance classification accuracy but also contribute to the broader goal of improving the inclusivity of mobile apps for users with disabilities. The coders agreed on the final coding findings, which provided a robust and reliable methodology.

4.3. Processing end-user reviews for accessibility issue detection

In app stores, user comments can be categorized based on their intentions and semantics [30]. While previous research has concentrated on classifying reviews into new features [31], issues [9,11,15], and non-functional requirements [32], this study specifically focuses on accessibility-related concerns. To achieve this goal, end-user reviews were systematically analyzed to identify common accessibility issues and classify them into distinct categories: INPUT, NAV, CAT, AUDIO-VISUAL, and UI. This classification aims to enhance the understanding of the challenges users face regarding accessibility and usability, and to improve the overall quality of software applications by incorporating feedback into the software development process.

UI The code "UI" is assigned to reviews describing problems related to the app's visual and structural design that create barriers for users, particularly those with visual impairments or cognitive disabilities. For example, one user commented, *"The fonts are too small to read comfortably, and I can't adjust them in settings. It's especially difficult for me as someone with low vision."* Another user expressed frustration with the layout, stating, *"The layout is confusing, and many buttons are so small that I have to zoom in to click them, which is inconvenient."* Such feedback highlights the need for customizable font sizes, more apparent visual elements, and more intuitive layouts to enhance usability for users with disabilities.

NAV The code "NAV" is assigned to end-user reviews describing navigation barriers and interactive elements that hinder users from completing tasks or accessing content. For instance, a user reported, *"The app has a confusing layout, making it difficult to find key features. I kept getting lost in different sections of the app."* Another user emphasized unclear labels, stating, *"I couldn't understand the function of some buttons because they had no labels, just icons. when trying to send a message, I accidentally tapped the wrong button multiple times."* This type of review highlights the importance of instinctive navigation systems and clear labelling of interactive elements, particularly for end-users with motor impairments or mental disabilities.

CAT The code "CAT" is assigned to end-user reviews, highlighting the apps in capacity to integrate with assistive technologies such as

[3] WCAGGuidelines

screen readers, voice control systems, or switch devices. For instance, one end-user stated, *"I tried using a screen reader with this app, but it simply doesn't work. The text is not properly announced, and I can't navigate to the different sections."* Another user expressed frustration with voice control, saying, *"This app does not respond to voice control commands, which is frustrating since I rely on voice commands due to my limited dexterity."* Such a review highlights the urgent need for apps to support assistive technologies, providing exclusivity for end-users who rely on them.

AUDIOVISUAL The code "AUDIOVISUAL" is assigned to user reviews, emphasizing the lack of basic features, such as captions, transcriptions, or audio descriptions in media content. For example, a user complained: *"I tried to watch a video on the app, but there were no captions or subtitles. As a deaf user, this makes it impossible for me to understand the content."* Another review pointed out harmful visual content, stating that *"While using this app, I noticed several flashing images. As someone with photosensitive epilepsy, this triggered a seizure, which was very dangerous."* This feedback demonstrates the necessity of providing accessible multimedia content to users with sensory impairment.

INPUT The code "INPUT" is assigned to end-user reviews discussing difficulties related to input methods and controls, disproportionately affecting users with motor impairments or limited dexterity. For example, one user noted that *"The character limit in text fields is too restrictive. I cannot even type half a sentence before I hit the limit, which is frustrating when using assistive technologies."* Another user highlighted unresponsive input fields, saying, *"The app's text input field is too small, especially when typing on a smartphone. It's difficult to see what I'm typing, and I often miss key presses."* Such reviews highlight the need for flexible input options and improved responsiveness to accommodate the diverse user needs.

Identifying these accessibility issues is crucial for improving the inclusivity of software applications. Unlike general usability concerns, accessibility issues directly impact users with disabilities, creating significant barriers that hinder their full engagement with the app. By systematically analyzing and categorizing these issues, developers can prioritize inclusive design practices and enhance the overall user experience.

Key Research Finding 1

We developed a novel coding guideline to identify key accessibility issues, including NAV, INPUT, CAT, AUDIOVISUAL, and UI. These categories align with the WCAG guidelines, which focus on accessible navigation, input controls, assistive technology compatibility, multimedia accessibility, and user-interface design. The novelty of this study lies in automating the detection and classification of these issues directly from user reviews, offering a practical solution for developers to enhance the inclusivity of their applications.

4.4. End-user reviews annotation

We developed a comprehensive annotation process to systematically analyze end-user reviews for accessibility-related problems. This process involves categorizing user feedback into distinct accessibility issues. For this purpose, we utilized the coding guidelines outlined in the developed grounded theory document, which provided clear definitions and examples for each category. The annotation process commenced with a manual evaluation of each review to determine whether it contained accessibility-related issues. If a review is identified as having accessibility concerns, it is further classified into one of five categories: UI, NAV, CAT, AUDIOVISUAL, or INPUT. Reviews that do not relate to accessibility are labeled as Non-Accessibility-Related Issues (NO).

For this purpose, two annotators (the third and fourth authors of the study) independently reviewed and labeled a subset of 13,000 user comments from the dataset. Discrepancies were resolved through discussion, and inter-coder agreement was calculated using Cohen's kappa (κ). Besides achieving an inter-coder agreement of 87%, the annotators recorded a Cohen's kappa of 69%, which signifies substantial agreement based on Cohen's kappa scale. This finalized annotated dataset was employed to train and validate the automated classifiers. Table 2 provides examples of reviews, their accessibility classifications, and the corresponding types of accessibility issues. These examples illustrate the diversity of user feedback and the challenges faced by users with disabilities when interacting with software.

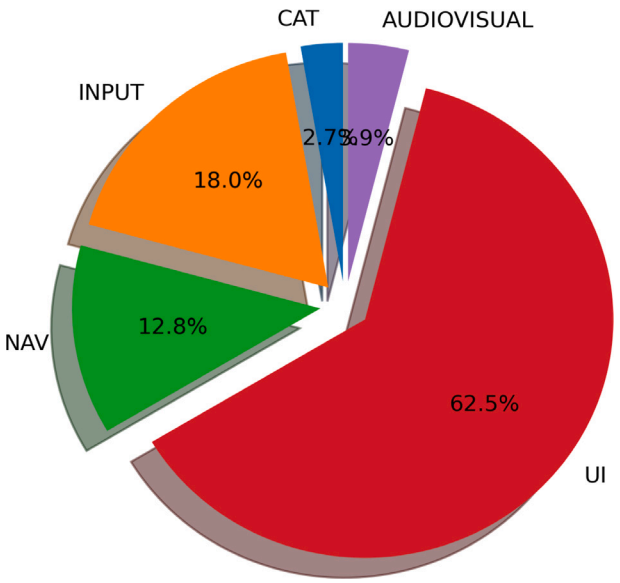

Fig. 2. Frequency of annotated dataset.

4.5. Frequency of accessibility issues: Annotation results

The annotation process revealed a diverse distribution of accessibility issues in the dataset, as shown in Fig. 2. Among the types identified, UI was the most prevalent, accounting for 3136 instances (62.5%), indicating the significant challenges users face with interface design. INPUT followed closely with 904 (18%) instances, reflecting the difficulties related to user input mechanisms. NAV was also notable, with 645 (12.8%) occurrences, suggesting persistent challenges in the navigation of the functions of the apps. AUDIOVISUAL and CAT appeared less frequently, with 198 (9%) and 137 (2.7%) instances. These findings underscore the varying degrees of impact that different accessibility issues have on user experience, providing critical information for developers looking to prioritize improvements in app exclusivity.

5. Automatic classification of user reviews

The increasing volume of end-user feedback on platforms such as the ASA store, the GP store, and the AAS presents both an opportunity and a challenge for software developers in extracting useful information for software evaluation. While user reviews offer valuable insights into accessibility issues, manually analyzing this feedback is often impractical due to the sheer volume of data, and it can be both time-consuming and error-prone [9,15]. To tackle this challenges, we propose and automated approach using fine-tuned DL algorithms to detect and classify accessibility issues in end-user reviews in the AAS. This section details the experimental framework, including the pre-processing steps, selection of DL algorithms, and performance evaluation metrics used to validate the proposed approach.

Table 2
Examples of annotated end-user reviews.

Review	Is Accessibility?	Accessibility Type
"The fonts are too small to read comfortably, and I can't adjust them in settings."	Yes	UI
"The app has a confusing layout, making it difficult to find key features."	Yes	NAV
"I tried using a screen reader with this app, but it simply doesn't work."	Yes	CAT
"I tried watching a video on the app, but there are no captions or subtitles."	Yes	AUDIOVISUAL
"The character limit in text fields is too restrictive."	Yes	INPUT
"This app is frustrating to use. It crashes frequently and is very slow."	No	

5.1. Accessibility issues identification using binary classification

The initial experiment sought to access the effectiveness of various DL algorithms in identifying whether end-user reves from AAS reported accessibility issues. To ensure data reliability, a dataset of 13000 annotated end-user reviews was carefully curated and classified into two categories: those contain accessibility issues and those without. This classification task forms the basis for identifying relevant feedback that can inform developers about potential accessibility issues.

5.2. Experimental setup

The first step in designing a DL experiment involves a comprehensive process to identify classification algorithms that have demonstrated strong performance on short-text data. To accomplish this, we meticulously selected seven cutting edge DL algorithms based on their effectiveness in processing short texts for various software engineering task such as fake reviews identification [33], sarcasm detection and classification [30], issue classification [9] and emotions classifications [15]. The DL classifiers shortlisted included the CNN, GRU, Bi-GRU, RNN, Bi-RNN, LSTM and Bi-LSTM.

5.3. Data pre-processing

We executed a series of essential pre-processing steps to ready the input data for the DL models, thereby improving the identifications and classifications of accessibility issues, these steps significantly influence the accuracy of the DL models. Initially we removed URLS and HTML tags from user reviews to ensure the textual data remained relevant and effectiveness. Next we normalized the text by converting it to low case, eliminating inconsistencies due to case sensitivity. We also addresses extraneous elements that could distort the analysis, such as brackets, punctuation marks, alphanumeric characters and special symbols. By removing these superfluous components, we aimed to focus the DL models attention on the primary detection and classifications task with improved accuracy. We incorporated lemmatization into pre-processing pipeline to ensure consistency among various word forms bys reducing words to their root of dictionary form. This steps is crucial for enhancing the performance of the DL models by maintain dataset uniformity . Additionally we removed stop words commonly used words like "in", "the", "is" etc which offer minimal contextual value during the training of the DL models. Removing stop-words is important for reducing noise in-textual data, allowing DL models to prioritize more impactful words and improves their accuracy.

5.3.1. Fine tuning the DL classifiers

This section outlines the approach for refining DL classifiers in both binary and detailed experiments to detect and categorize accessibility issues within the AAS framework. We focused on optimizing the hyperparameters of various DL classifiers to improve their ability to identify accessibility issues. In particular, we adjusted the embedding layer by experimenting with different "output_dim" values. We tested the DL classifiers' performance with output dimensions set at 32, 64, and 100. An output dimension of 100 consistently delivered better accuracy across different DL classifiers compared to dimensions of 64 and 32. Additionally, researchers have suggested several optimizers for training DL classifiers, with studies in software engineering indicating that some optimizers perform exceptionally well [15,30]. Therefore, we evaluated the Adam and root mean square propagation (RMSProp) optimizers to determine their effectiveness in identifying accessibility issues and their types. The experimental results demonstrate that the Adam Optimizer consistently surpasses RMSProp in identifying accessibility issues and their types, as reported in the literature [30].

Table 3
Settings and hyperparameters for detecting accessibility issues.

Setting/Hyperparameter	Binary Classification
Output Dimension of Embedding Layer	100
Optimizer for DL	Adam Optimizer
Learning Rate for DL	Default value from Adam optimizer
Dropout Layer	0.2
Dense Layer Units	1
Number of DL Units	100
Dropout	0.2
Loss Function	Binary Cross-Entropy
Output Layer Activation Function	Sigmoid
K-fold	5
Oversampling and Under-sampling	Applied as needed

Additionally, a significant obstacle in training DL classifiers is the issue of overfitting, where classifiers tend to memorize the training data instead of learning to generalize from it. To tackle this problem, we employed several strategies, such as simplifying the classifiers and integrating dropout layers into the DL algorithms. By adding a dropout layer with a dropout rate of 0.2, the DL classifiers' ability to differentiate between reviews with and without accessibility issues in the AAS dataset was enhanced, which is consistent to the previous study [30]. Furthermore, the DL classifiers showed better generalization in categorizing AAS reviews into specific types of accessibility issues, including AUDIOVISUAL, CAT, INPUT, NAV, and UI, by reducing the impact of overfitting. Table 3 provides a summary of the hyperparameters used for the training of DL algorithms to detect whether an end-user review contains accessibility-related issues.

5.4. Data imbalanced

One frequent issue encountered by supervised algorithms during the training phase is dealing with imbalanced datasets, which occur when there is an unequal distribution of user reviews across classification labels in the dataset [34]. In the binary classification experiment, the distribution of end-user comments across the two labels (accessibility and non-accessibility) was imbalanced. Specifically, 34.94% of the 13,000 comments were labeled accessibility-related, while 65.40% were non-accessibility-related. Similarly, the distribution of accessibility issues is disproportionate. As shown in Fig. 2, UI issues were the most prevalent, accounting for 3136 instances (62.5%), highlighting the significant challenges users face in interface design. INPUT issues closely followed 904 instances (18%), reflecting difficulties related to user input mechanisms. NAV issues were also notable, with 645 occurrences (12.8%) indicating persistent challenges in navigating the app's functionalities. AUDIOVISUAL and CAT issues appear less frequently, with 198 (9%) and 137 (2.7%) instances, respectively. The natural imbalance in data distribution heightens the risk of bias in DL classifiers, as they often prioritize the majority class and overlook

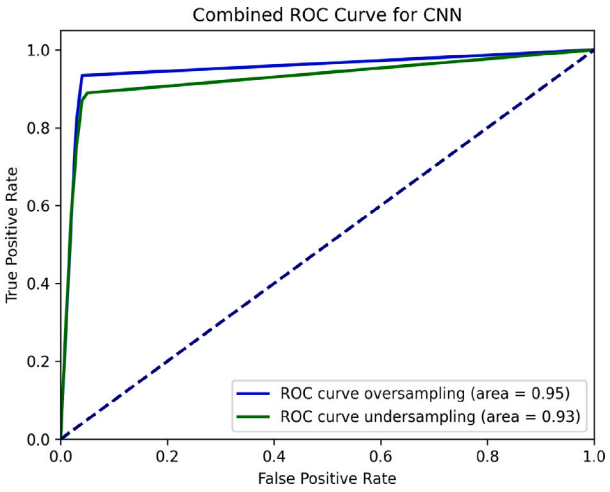

Fig. 3. ROC curve for sampling methods (CNN).

minority classes with less data. To tackle this problem, we utilized two commonly adopted methods: oversampling and undersampling [15, 35]. Oversampling involves duplicating instances from minority classes, whereas undersampling involves excluding cases from the majority class. Additionally, we assessed the performance of the DL classifiers using receiver operating characteristic (ROC) and precision–recall curves to determine the optimal configurations, which led to enhanced and more generalized detection and classification of accessibility issues. Fig. 3 illustrates the ROC curve of the CNN model. According to the experimental findings, DL classifiers that use oversampling consistently achieve better performance than those that rely on undersampling. This is likely due to the loss of important textual information that occurs with the undersampling method [36].

5.5. Training and validation of algorithms

To perform classification tests at the binary and multiclassification levels, supervised DL algorithms require training on manually annotated comments. For this purpose, we used the widely utilized 5-fold cross-validation method, applying it to 13,000 manually annotated user comments. These comments were meticulously annotated using the content analysis approach, ensuring the highest quality of the data. In this process, four folds were used to train the classifiers, while the remaining fold was reserved for the validation and verification of the classifiers. A systematic rotation of the training and validation folds was performed to repeat the procedure five times, ensuring optimal training and validation of the DL classifiers. The performance of these algorithms was assessed by calculating the average outcomes across the five folds. Additionally, standard evaluation metrics such as precision (P), recall (R), and F-measure (F) were utilized to evaluate the effectiveness of the DL algorithms in classifying user comments concerning accessibility issues. The formulations for P and R are as follows:

$$P_k = \frac{TP_j}{TP_j + FP_j}$$

$$R_k = \frac{TP_j}{TP_j + FN_j}$$

Where: - TP_j: The balance of end-user feedback related to accessibility issues that are correctly classified as being of type j. - FP_j: This metric highlights the impact of end-user comments misclassified in the AAS that are incorrectly classified as of type j. It underscores the need for precision and accuracy in our work. - FN_j: This metric, along with the F1 score, is a key component in our evaluation process. It provides a balanced measure performance of the classifier's, which is crucial in our work.

Table 4
Classification results of binary classification.

Classifier	Accuracy	Precision	Recall	F1 Score
CNN	93%	92%	93%	93%
LSTM	87%	85%	89%	87%
BiLSTM	86%	84%	89%	86%
RNN	83%	80%	88%	84%
BiGRU	80%	77%	86%	81%
GRU	76%	74%	81%	78%
BiRNN	77%	73%	86%	79%

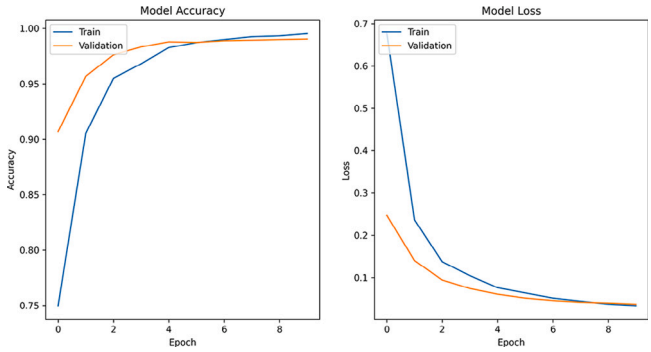

Fig. 4. Training and validation accuracies and loss graphs for CNN.

5.6. Binary classification results

Table 4 summarizes the results of the binary classification experiment, which used fine-tuned DL algorithms to categorize app reviews into two classes: those related to accessibility issues and those that were unrelated. Among the DL classifiers evaluated, the fine-tuned CNN classifier exhibited superior performance, achieving an overall precision of 93%, a precision of 92%, a recall of 93%, and an F1 score of 93%. This makes CNN the most appropriate choice for implementation in a tool designed to detect accessibility-related issues in the end-user reviews. The LSTM and BiLSTM classifiers also demonstrated efficacy, with accuracies of 87% and 86%, respectively. However, their F1 scores (87% and 86%, respectively) were marginally lower than those of CNN, indicating a slightly reduced effectiveness in balancing precision and recall. The RNN and BiGRU classifiers achieved accuracies of 83% and 80%, respectively, with corresponding F1 scores of 84% and 81%. Although these classifiers are effective, they do not achieve the level of performance exhibited by CNN. Finally, the GRU and BiRNN classifiers showed the lowest performance metrics with accuracies of 76% and 76%, respectively, and F1 scores of 78% and 79%, respectively. These results suggest that GRU and BiRNN may face challenges in processing accessibility-related feedback, particularly in distinguishing relevant reviews from irrelevant ones.

To further evaluate the performance of the CNN classifier, we analyzed its training and validation accuracy and loss graphs, as shown in Fig. 4. The training accuracy steadily increased, whereas the validation accuracy remained consistent, indicating that the model generalized well to unseen data. Similarly, the training and validation losses decreased over epochs, confirming the model's convergence.

The ROC curve for the CNN classifier is also presented in Fig. 5. The curve illustrates the classifier's ability to effectively distinguish between accessibility-related and non-accessibility-related reviews, with an area under the curve (AUC) of approximately 95%. This indicates excellent discriminatory power.

Finally, the confusion matrix for the CNN classifier is shown in Fig. 6. The matrix offers an overview of the classifier's effectiveness by detailing both accurate and inaccurate predictions. The high numbers along the diagonal suggest that the classification of reviews related to accessibility and those not related to accessibility is precise.

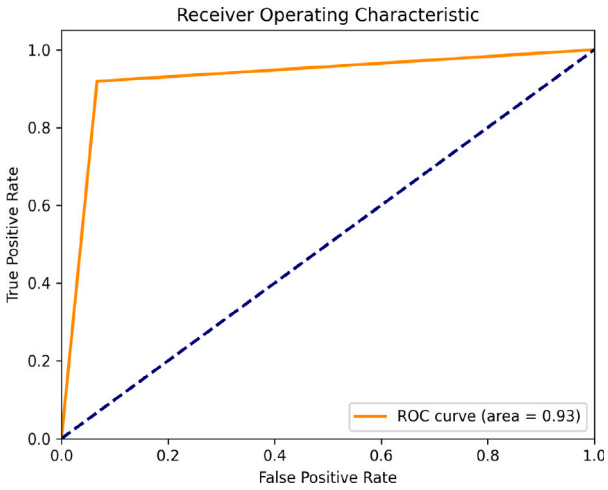

Fig. 5. ROC curve for CNN.

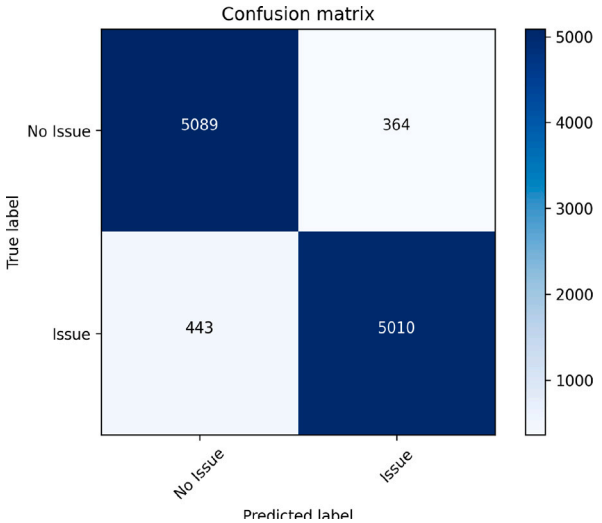

Fig. 6. Confusion matrix for CNN.

5.7. Multi-classification using DL algorithms

In the second DL experiment, we assessed the effectiveness of different DL algorithms in sorting user comments with accessibility-related issues into the following specific categories: NAV, INPUT, CAT, UI, and AUDIOVISUAL. Out of the 13,000 annotated user comments, 5020 (around 38.6%) were found to have accessibility issues concerning feedback. This experimental study aimed to test the capability of this DL classifier to distinguish and categorize the identified accessibility-related user comments into detailed types of accessibility issues. For the detailed DL experimentation, we employed the same data preparation and preprocessing techniques as those used in the initial testing setup. Additionally, we fine-tuned the DL classifiers using particular hyperparameters and configurations, as detailed in Table 5.

5.7.1. Fine-grained accessibility classification results

Table 6 presents the results of a fine-grained classification experiment using DL algorithms to classify app reviews into specific accessibility-related categories: NAV, INPUT, CAT, UI and AUDIOVISUAL. Among the DL classifiers evaluated, the CNN model demonstrated superior performance in all categories, with an average precision of 97%. The BiGRU and BiLSTM models also performed well, with an average accuracy of 96% each, followed by LSTM with 95%,

Table 5
Settings and configurations for classifying accessibility issues.

Setting/Configuration	Value/Details
Output Dimension of Embedding Layer	128
Number of Model Units	64
Cross-Validation Folds	5-Fold
Optimization Algorithm	Adam
Dense Layer Units	64 (with ReLU Activation)
Activation Function in Output Layer	Softmax
Dropout Rate	0.2 (Input)/0.2 (Recurrent)
Loss Function Used	Categorical Cross-Entropy
Learning Rate	Default Value (0.001)
Batch Size	128
Data Balancing Technique	Oversampling/Undersampling

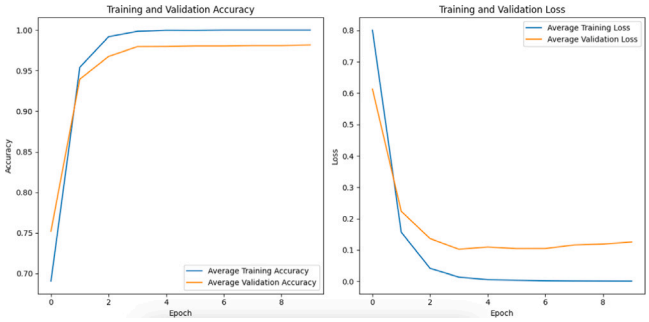

Fig. 7. Training and validation accuracy and loss graphs for CNN.

GRU with 94%, RNN with 89%, and BiRNN with 81%. The CNN model achieved precision, recall, and F-measure scores of 0.99% for most categories, including NAV, INPUT, CAT, and AUDIOVISUAL. For the UI category, the CNN model achieved a high precision of 99% and recall of 99%, resulting in an F-measure of 99%. These results indicate the efficacy of the CNN model in capturing textual features related to accessibility issues in user reviews. To further evaluate the performance of the CNN classifier, we analyzed its training and validation accuracy and loss graphs, as shown in Fig. 7. The training accuracy steadily increased while the validation accuracy remained consistent, indicating that the model generalized well to unseen data. Additionally, the training and validation losses decreased over epochs, confirming the model's convergence.

Furthermore, the ROC curve for the CNN classifier is presented in Fig. 8. The curve demonstrates the classifier's ability to distinguish between different accessibility-related categories effectively, with an area under the curve (AUC) of 1. This indicates excellent discriminatory power.

Finally, the confusion matrix for the CNN classifier is shown in Fig. 9. The matrix provides insights into the classifier's performance by summarizing correct and incorrect predictions. The high values along the diagonal indicate an accurate classification in all accessibility-related categories.

Key Research Finding 2

The fine-tuned CNN classifier demonstrated superior performance over other fine-tuned DL classifiers in detecting and classifying end-user reviews into fine-grained accessibility types. The findings can help developers intertwine the proposed fine-tuned CNN classifier into the existing software evolution process to identify frequently occurring accessibility and usability issues and provide remedies to possibly improve the quality of existing apps.

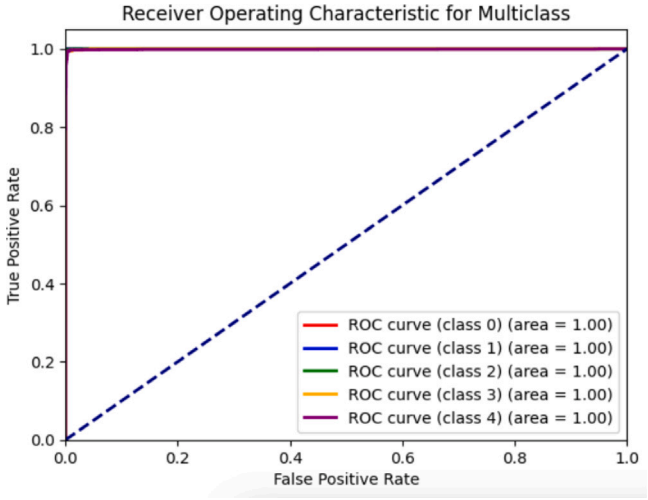

Fig. 8. ROC curve for CNN.

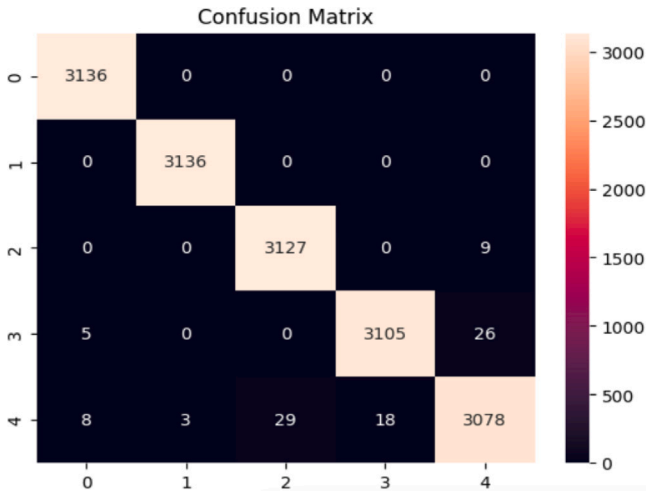

Fig. 9. Confusion matrix for CNN Model for Multi classification.

Table 6
Results comparison of DL algorithms.

Labeled Tags	DL Algorithms	Precision	Recall	F-Measure
NAV	CNN Model	99%	99%	99%
	BiGRU Model	99%	99%	99%
	BiLSTM Model	96%	99%	98%
	LSTM Model	95%	99%	97%
	GRU Model	99%	99%	99%
	RNN Model	94%	98%	96%
	BiRNN Model	95%	98%	96%
INPUT	CNN Model	99%	99%	99%
	BiGRU Model	99%	99%	99%
	BiLSTM Model	94%	97%	96%
	LSTM Model	90%	99%	94%
	GRU Model	99%	99%	99%
	RNN Model	93%	95%	94%
	BiRNN Model	91%	96%	93%
CAT	CNN Model	99%	99%	99%
	BiGRU Model	98%	99%	99%
	BiLSTM Model	99%	99%	99%
	GRU Model	98%	99%	99%
	LSTM Model	96%	99%	98%
	RNN Model	97%	99%	98%
	BiRNN Model	96%	99%	98%
UI	CNN Model	99%	99%	99%
	BiGRU Model	99%	98%	99%
	BiLSTM Model	97%	88%	92%
	GRU Model	98%	96%	97%
	LSTM Model	99%	76%	86%
	RNN Model	93%	81%	87%
	BiRNN Model	94%	79%	86%
AUDIOVISUAL	CNN Model	99%	98%	98%
	BiGRU Model	99%	97%	98%
	BiLSTM Model	99%	99%	99%
	GRU Model	96%	96%	96%
	LSTM Model	97%	99%	99%
	RNN Model	97%	99%	99%
	BiRNN Model	98%	99%	99%

DL Classifiers	Average Accuracy
CNN Model	97%
BiGRU Model	96%
BiLSTM Model	96%
LSTM Model	95%
GRU Model	94%
RNN Model	89%
BiRNN Model	81%

6. Discussion

This section discusses the key findings of our research, their importance, and potential limitations.

6.1. Insights into accessibility issue detection

This study indicates that fine-tuned DL classifiers can effectively detect and classify accessibility issues in AAS user reviews. The analysis of 178667 user reviews across 85 mobile applications from 18 different categories revealed common types of accessibility issues, including UI, CAT, NAV, AUDIOVISUA and INPUT. The results show that the DL classifiers achieve an accuracy of 93% in binary classification and up to 97% in multi-classification for the CNN model. These findings underscore the potential of automated methods to improve the identification and understanding of accessibility issues in the built environment. It also highlights the critical role of contextual understanding, as provided by the CNN model, in accurately classifying delicate accessibility issues.

6.2. Improving software evaluation through timely accessibility issue detection

The early detection of accessibility issues in software development offers significant advantages for evaluating software and improving its quality. By identifying these usability and accessibility issues during the initial stages of development, developers can adopt a user-centric design approach, ensuring that applications are inclusive and meet the accessibility and usability needs of users. This proactive approach will reduce costs by avoiding extensive rework after deployment and enhance key software quality metrics, such as usability, reliability, and maintainability. For example, detecting NAV or CAT enables developers to refine app functionalities, ensuring compliance with accessibility standards like WCAG and fostering user trust. In addition, a crucial benefit of early detection is that it enables developers to track end-user feedback through an iterative process, which helps resolve future update issues. The iterative progress through user feedback generates better versions of the application, demonstrating a commitment to inclusivity and resulting in increased user satisfaction and improved market competitiveness.

Key Research Finding 3

This study highlights the potential of automated methods to improve identification and understanding. of accessibility challenges , enabling developers to prioritize and address these issues more effectively. The use of DL models offers a promising solution for processing large volumes of feedback in a timely manner , providing actionable insights to improve the inclusivity of the application.

6.3. Implications for software development

The automatic detection and classification of accessibility issues in AAS is crucial to software development. By incorporating accessibility issue detection into AAS analysis, developers can prioritize and address these issues more effectively, ensuring that their applications are inclusive and user-friendly. This method not only enables the identification of different accessibility issues, but also provides insights into the types of issues that end-users encounter most frequently. For instance, the user interface is often cited as one of the most significant and usability challenges, emphasizing the need for better integration with assistive technologies like screen readers. Similarly, the proposed approach can be extended to other platform, such as the GB store, to facilitate cross-platform analysis of accessibility issues. This capability aids in making well-informed choices during software assessment, which in turn enhances user focus and satisfaction.

Key Research Finding 4

This study enhances the creation of more inclusive and user-friendly software applications by integrating accessibility issues detection into app store analysis. Developers can leverage these insights to make informed decisions and ensure that their applications meet the accessibility and usability needs of users, fostering a more inclusive digital environment.

6.4. Threats to validity

Despite these promising results, several threats to the validity of our research must be acknowledged. First, the data set used in this study was curated exclusively from the AAS, which may limit the generalizability of our findings to other platforms. Second, the coding guidelines developed for identifying accessibility issues focused on five main categories, potentially overlooking less frequent but equally important issues. Expanding the scope of the coding guidelines to include additional accessibility concerns could enhance the comprehensiveness of our approach. Third, the performance of DL classifiers depends heavily on the quality and size of the training dataset. Although oversampling techniques were used to balance the dataset, the limited availability of labeled data remains challenging. Finally, the subjective nature of accessibility issue feedback introduces inherent variability in manual annotation, which could affect the reliability of the ground truth used for training and validation.

7. Conclusion and future work

In conclusion, this study presents an automated approach for detecting and classifying accessibility issues in end user comments from the AAS using fine-tuned DL classifiers. We curated a dataset of 178667 user reviews from 85 applications across 18 diverse categories. We developed a novel coding guideline to identify common accessibility issues, including NAV, INPUT, CAT and UI. We manually annotated a subset of reviews using this guideline to create a labeled dataset for training and evaluating the DL algorithms. The experiments demonstrated that fine-tuned DL classifiers achieved comparatively higher accuracy in detecting accessibility issues and classifying them into specific types than conventional classifiers. For binary classification, the fine-tuned CNN classifier stood out with an impressive accuracy of 93%, surpassing GRU, BiGRU, LSTM, and BiLSTM, which attained accuracies ranging from 76% to 87%. In fine-grained classification, the CNN model continued to impress, obtaining an average accuracy of 97%, followed by BiGRU and BiLSTM with 96%. Similarly, the BiLSTM and LSTM models demonstrated strong performance, with average accuracies of 96% and 95%, respectively. These results highlight the potential of automated methods to enhance the identification and understanding of accessibility challenges, thereby enabling developers to prioritize and address these issues more effectively. By integrating accessibility issue detection into app store analysis, this study contributes to creating more inclusive and user-friendly software applications.

In our future research, we plan to broaden the scope of our proposed method by gathering a wider variety of user reviews. This will include different app categories and platforms, such as the Google Play Store and the App Store, to improve the flexibility and real-world usefulness of the techniques we have developed. Furthermore, this study will investigate advanced transfer learning-based models, including BERT and RoBERTa, to further improve the efficacy of accessibility issue detection. Broadening the scope of sarcasm types and accessibility issues will necessitate data collection from diverse social media platforms, such as Reddit and Twitter, to capture a broader spectrum of expression. Moreover, this study aims to incorporate explainable artificial intelligence (XAI) methodologies to provide interpretable insights into the decision-making processes of DL classifiers, thereby increasing transparency for software vendors and developers. Finally, longitudinal studies will examine the long-term effects of improvements derived from end-user comments and investigate the connection between end-user ratings and software evolution. This expanded research endeavour will contribute to a more comprehensive understanding of accessibility challenges and promote the development of inclusive software applications.

CRediT authorship contribution statement

Mumrez Khan: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Zhixiao Wang:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Javed Ali Khan:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Nek Dil Khan:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Funding

"The author(s) received no financial support for the research, authorship, and/or publication of this article"

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The complete dataset and details of the experimental work can be accessed through the following: https://github.com/nekdil566/accessibility_issues.

References

[1] Alshayban A, Ahmed I, Malek S. Accessibility issues in android apps: state of affairs, sentiments, and ways forward. In: Proceedings of the ACM/IEEE 42nd international conference on software engineering. 2020, p. 1323–34.
[2] Health Organization W. Disability. 2023.
[3] Health Organization W. World Report on Disability. 2011.
[4] Ross AS, Zhang X, Fogarty J, Wobbrock JO. Examining image-based button labeling for accessibility in android apps through large-scale analysis. In: Proceedings of the 20th international ACM SIGACCESS conference on computers and accessibility. 2018, p. 119–30.
[5] Yan S, Ramachandran P. The current status of accessibility in mobile apps. ACM Trans Access Comput (TACCESS) 2019;12(1):1–31.
[6] Store AP. Building accessible apps (Apple). 2025, accessed on 12/01/2025.
[7] AlOmar EA, Aljedaani W, Tamjeed M, Mkaouer MW, El-Glaly YN. Finding the needle in a haystack: On the automatic identification of accessibility user reviews. In: Proceedings of the 2021 CHI conference on human factors in computing systems. 2021, p. 1–15.
[8] Khan JA, Liu L, Wen L, Ali R. Conceptualising, extracting and analysing requirements arguments in users' forums: The CrowdRE-Arg framework. J Softw: Evol Process 2020;32(12):e2309.
[9] Khan ND, Khan JA, Li J, Ullah T, Zhao Q. Mining software insights: uncovering the frequently occurring issues in low-rating software applications. PeerJ Comput Sci 2024;10:e2115.
[10] Alrumayh AS, Lehman SM, Tan CC. Emerging mobile apps: Challenges and open problems. CCF Trans Pervasive Comput Interact 2021;3(1):57–75.
[11] Khan ND, Khan JA, Li J, Ullah T, Alwadain A, Yasin A, Zhao Q. How do crowd-users express their opinions against software applications in social media? A fine-grained classification approach. IEEE Access 2024.
[12] Khalajzadeh H, Shahin M, Obie HO, Agrawal P, Grundy J. Supporting developers in addressing human-centric issues in mobile apps. IEEE Trans Softw Eng 2022;49(4):2149–68.
[13] Guzman E, Maalej W. How do users like this feature? a fine grained sentiment analysis of app reviews. In: 2014 IEEE 22nd international requirements engineering conference. IEEE; 2014, p. 153–62.
[14] Khan JA, Liu L, Wen L. Requirements knowledge acquisition from online user forums. Iet Softw 2020;14(3):242–53.
[15] Khan ND, Khan JA, Li J, Ullah T, Zhao Q. Leveraging large language model ChatGPT for enhanced understanding of end-user emotions in social media feedbacks. Expert Syst Appl 2025;261:125524.
[16] Iacob C, Harrison R. Retrieving and analyzing mobile apps feature requests from online reviews. In: 2013 10th working conference on mining software repositories. IEEE; 2013, p. 41–4.
[17] Oliveira ADA, Eler MM. Exploring accessibility of mobile applications through user feedback: Insights from app reviews in a systematic literature review. In: Proceedings of the XXIII Brazilian symposium on human factors in computing systems. 2024, p. 1–15.
[18] Alghamdi AM, Aljedaani W, Eler MM, Ludi S. Accessibility guidelines and standards: Analyzing stack overflow posts. In: Proceedings of the 21st international web for all conference. 2024, p. 118–22.
[19] Aljedaani W, Mkaouer MW, Ludi S, Javed Y. Automatic classification of accessibility user reviews in android apps. In: 2022 7th international conference on data science and machine learning applications. IEEE; 2022, p. 133–8.
[20] Seixas Pereira L, Matos M, Duarte C. Exploring mobile device accessibility: Challenges, insights, and recommendations for evaluation methodologies. In: Proceedings of the CHI conference on human factors in computing systems. 2024, p. 1–17.
[21] Abdelaziz O. Integrating user feedback to enhance software quality and user satisfaction in mobile application development. [Ph.D. thesis], University of Saskatchewan Saskatoon; 2024.
[22] Chandarana N, Gada T. Accessibility Challenges in Current Mobile Applications: A Comprehensive Overview.
[23] Khamaj A, Ali AM. Examining the usability and accessibility challenges in mobile health applications for older adults. Alex Eng J 2024;102:179–91.
[24] Haggag O, Grundy J, Abdelrazek M, Haggag S. Better addressing diverse accessibility issues in emerging apps: A case study using covid-19 apps. In: Proceedings of the 9th IEEE/ACM international conference on mobile software engineering and systems. 2022, p. 50–61.
[25] Reyes Arias JE, Kurtzhall K, Pham D, Mkaouer MW, Elglaly YN. Accessibility feedback in mobile application reviews: A dataset of reviews and accessibility guidelines. In: CHI conference on human factors in computing systems extended abstracts. 2022, p. 1–7.
[26] Eler MM, Orlandin L, Oliveira ADA. Do android app users care about accessibility? an analysis of user reviews on the google play store. In: Proceedings of the 18th Brazilian symposium on human factors in computing systems. 2019, p. 1–11.
[27] Santiago MT, Marques AB. Exploring user reviews to identify accessibility problems in applications for autistic users. J Interact Syst 2023;14(1):317–30.
[28] Neuendorf K. The content analysis guidebook sage publications, inc. In: Library of congress. CA: United states. 2002.
[29] Strauss A, Corbin J, et al. In: Basics of qualitative research, vol. 15, sage Newbury Park, CA; 1990.
[30] Fatima E, Kanwal H, Khan JA, Khan ND. An exploratory and automated study of sarcasm detection and classification in app stores using fine-tuned deep learning classifiers. Autom Softw Eng 2024;31(2):69.
[31] Khan JA, Xie Y, Liu L, Wen L. Analysis of requirements-related arguments in user forums. In: 2019 IEEE 27th international requirements engineering conference. IEEE; 2019, p. 63–74.
[32] Parsaei Z, Jangi M, Tahmasebian S, Ehteshami A. Functional and nonfunctional requirements of virtual clinic mobile applications: A systematic review. Int J Telemed Appl 2024;2024(1):7800321.
[33] Khan JA, Ullah T, Khan AA, Yasin A, Akbar MA, Aurangzeb K. Can end-user feedback in social media be trusted for software evolution: Exploring and analyzing fake reviews. Concurr Comput: Pr Exp 2024;36(10):e7990.
[34] Chawla NV, Japkowicz N, Kotcz A. Special issue on learning from imbalanced data sets. ACM SIGKDD Explor Newsl 2004;6(1):1–6.
[35] Ullah T, Khan JA, Khan ND, Yasin A, Arshad H. Exploring and mining rationale information for low-rating software applications. Soft Comput 2023;1–26.
[36] Khan JA, Yasin A, Fatima R, Vasan D, Khan AA, Khan AW. Valuating requirements arguments in the online user's forum for requirements decision-making: the CrowdRE-VArg framework. Softw: Pr Exp 2022;52(12):2537–73.