An Empirical Study of Accessibility Features and User Satisfaction in Android Applications

Rahul Ramesh Jois
EECS Department
University of California, Irvine
Irvine, USA
rjois@uci.edu

Sagar Krishna
EECS Department
University of California, Irvine
Irvine, USA
sakrish1@uci.edu

Rajath Ganapathi Hegde
EECS Department
University of California, Irvine
Irvine, USA
rajathg@uci.edu

***Abstract*—Accessibility features in mobile applications are vital to ensure an inclusive and barrier-free experience for users with disabilities. Android, being the most widely used mobile operating system, offers several accessibility features to aid such users in accessing mobile applications. These features can significantly improve the usability of mobile applications for users with vision, hearing, mobility, and cognitive impairments. Despite the growing importance of accessibility in mobile applications, there is a lack of research on how users perceive accessibility features and how they are used in Android applications. It is essential to investigate the user reviews provided for accessibility features in Android applications to gain insights into user experiences and to identify any shortcomings of the features implemented in the app. This study aims to fill the gap in the literature and provide a better understanding of how accessibility features are implemented in Android applications and how it is perceived by the users. A total of 30 apps with over million reviews from different categories including finance, shopping, fitness are used in the study. Furthermore, the study evaluates the possibility of using user reviews to reliably detect bugs in the usage of accessibility features provided in Android applications.**

***Index Terms*—API, model, text extraction, Mobile apps, Android, Functionalities, User feedback, App store**

I. Introduction

Accessibility is a significant aspect to consider when developing an app, as it facilitates interaction with the digital world for people with disabilities. Android enables developers to include accessibility features in their apps through its accessibility framework and accessibility service APIs. Accessibility features in mobile apps are a set of design elements and functionalities that make it easier for people with disabilities to use and navigate mobile applications. Text-to-speech capabilities, text magnification, visual alerts, haptic feedback, voice feedback, and closed captions are some of the features that fall into this category. There are multiple developer guidelines, such as the Web Content Accessibility Guidelines, Android Accessibility Guidelines, and iOS Accessibility Guidelines, published by organizations worldwide, which aim to augment and include accessibility features in Android apps. While most apps available in the market advertise a rich set of accessibility features, many of these features and services are found to be rendered ineffective in usage due to reasons including incorrect implementation of APIs and incompatibility with other libraries. Hence, an empirical approach to finding the usage of accessibility APIs used in Android applications will help interpret the reasons for the effectiveness of advertised accessibility features. The application to be studied is decompiled using *apktool* to ascertain the number and types of accessibility APIs utilized. A more detailed explanation of this process will be provided in the methodology section later in the paper.

User reviews are another crucial resource for evaluating the effectiveness of accessibility features provided by an app. User reviews, written by people who have used the app and shared their experiences, are a critical aspect of modern mobile applications. User reviews can also help to raise awareness about the importance of accessibility in mobile apps and encourage developers to prioritize accessibility in their design and development processes. Therefore, extracting user reviews related to accessibility from app stores can help assess the effectiveness of the app's accessibility features.. Despite the importance of user reviews in evaluating accessibility features, many existing studies of Android applications do not incorporate them. The studies that do consider user reviews often do so at a very rudimentary level, using a limited number of keywords related to accessibility. Furthermore, the methods used to extract user feedback often lack accuracy.

Therefore, the general purpose of this work is to investigate the usage of accessibility APIs in Android applications and determine the level of user satisfaction with them. Furthermore, this research examines a method for detecting the sentiment in user feedback and assessing whether such reviews can more efficiently highlight accessibility concerns. To achieve this, a novel text crawling approach is used in combination with sentiment analysis to extract user feedback and identify the connotation of the reviews. Text crawling is a technique for extracting data that utilize automated tools to scrape information from large text collections on a specific topic. One of the main benefits of text crawling is that it can save time and resources compared to manual data collection. Sentiment analysis involves using natural language processing techniques to identify and extract the sentiment expressed in user reviews related to the accessibility features of the Android application. This analysis provides valuable insights into how users perceive the accessibility features of the application and how these features can be improved to better meet the needs

of users with disabilities.

The paper is structured as follows: Section 2 provides a brief literature survey. In Section 3, we describe the methodology used to conduct the study. Section 4 illustrates the experimental setup used for the study. The paper's research questions and their corresponding answers are discussed in Section 5, while Sections 6 and 7 present the study's findings and potential threats to its validity. The future scope of the work is briefly mentioned in Section 7, and concluding remarks are provided in Section 8.

II. Literature Survey

Many existing studies have attempted to evaluate how Android's accessibility framework and services are utilized in apps. However, only a few of these studies incorporate user feedback in the form of app reviews published on platforms such as Google Play Store, Apple Store, and Microsoft Store. Furthermore, the methods used in prior research to determine accessibility API usage are limited by their runtime and app decomposition. This survey can be broadly classified into two categories: studies that detail the accessibility services used in an app, and research that investigates user reviews.

User feedback is a valuable resource for identifying and improving accessibility features in Android applications. However, manually analyzing large volumes of user reviews can be both time-consuming and expensive [1]. Reviews related to accessibility tend to be concentrated around just a few popular apps, and users often give these apps a reasonably good rating despite reporting accessibility issues, except in cases where the issues render the app unusable [2]. Recent approaches [1, 3] involve classifying user reviews by extracting them from app stores and using text crawling and manual examination of user sentiment to determine if they are accessibility-related feedback. Researchers collect datasets of user reviews from the Google Play Store and manually annotate them as either positive or negative sentiments. They then train deep learning models on the obtained dataset, using only four types of guidelines related to accessibility. However, they do not provide a detailed explanation of the selection of the hyperparameters and their effects on training the classifier.

Accessibility services are an important feature in Android applications that allow users with disabilities to interact with applications in ways that might otherwise be difficult or impossible [4]. However, the improper implementation of these services can also pose risks to users. The paper [4, 5] investigates the use of accessibility APIs in Android applications and the potential risks associated with their use. These approaches manually extract the accessibility APIs used in Android applications. The papers also survey related work on security and privacy issues in Android applications, such as the use of permissions and the risks associated with third-party libraries.

III. Methodology

The methodology section of the paper is structured into two distinct parts: one focused on Accessibility API usage and the other on user reviews. The section on Accessibility API usage describes the process of identifying and classifying API usage for the chosen apps. The section on user reviews focuses on the model used to develop the sentiment analysis of the reviews. It describes the data collection process, which involves text crawling from the Google Play Store to extract reviews for the selected apps.

A. Methodology to find Accessibility API Usage

We used a rigorous and thorough process to analyze accessibility-related code references in popular Android applications. The process involved the following steps and are depicted in Figure 1:

1) Obtaining and Verifying APKs: We obtained the latest APK files of popular Android applications from AndroZoo, a trusted public repository. To ensure that we were working with unmodified versions of the applications, we verified the authenticity and integrity of the APK files by checking their digital signatures using the Android Package Manager(APK) tool.

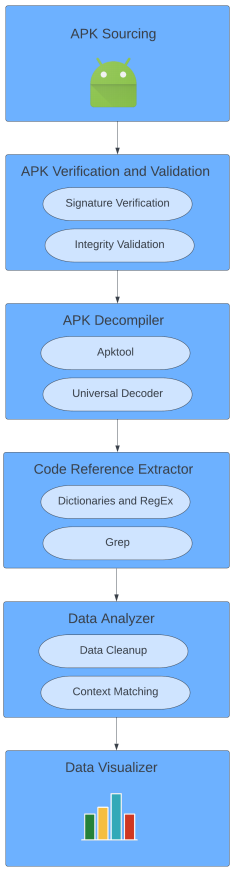

Fig. 1. Accessibility API Usage Flowchart

2) Decompiling APKs: To analyze the applications' source code, we used apktool, a reliable tool that allowed us to decompile the APK files and extract their Java source code. This process involved reverse-engineering the compiled code of the application to recreate its original Java source code.

3) Analyzing the Source Code: We performed a comprehensive analysis of the Java source code of the applications to identify the code segments responsible for invoking accessibility services. We used a combination of automated and manual techniques to locate the accessibility-related code references. First, we converted all the source code files, including binaries, into text files using the Universal Decoder tool. Then, we searched for specific classes and methods related to accessibility in the app's source code. These included *AccessibilityService*, *AccessibilityNodeInfo*, *onAccessibilityEvent*, *AccessibilityManager*, *AccessibilityEvent*, *isAccessibilityEnabled*, *AccessibilityAction*, *AccessibilityDelegate*, *AccessibilityRole*, and *AccessibilityTraversal*.

We also looked for accessibility-related annotations in the source code, such as *@AccessibilityNodeProviderCompat* and *@Keep*, and reviewed the layout files of the apps for the use of accessibility attributes like *android:focusable*, *android:contentDescription*, *android:labelFor*, *android:hint*, and *android:accessibilityLiveRegion*. To ensure the correctness of our automated analysis, we manually inspected the code segments responsible for invoking accessibility services.

4) Analyzing the Data: We converted the accessibility-related code references into a pandas dataframe using Python. We used pandas to extract and process the relevant data from the code references, including the accessibility services called, permissions used, and UI elements accessed.

We used various methods to search for accessibility-related app references, such as searching for specific classes and methods related to accessibility in the app's source code, looking for accessibility-related annotations, and reviewing the layout files.

We used data visualization tools like Matplotlib and Seaborn to generate plots that represented the data in graphical form. These plots helped us gain insights into the accessibility-related features of the applications, such as how often they used accessibility services, what types of services they used, and which UI elements they made accessible.

Overall, our methodology provides a robust and comprehensive approach to investigating the use of accessibility services in Android applications. We automated the entire process using Python scripts, ensuring the accuracy and reproducibility of our analysis.

B. Methodology for Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a process of identifying and extracting opinions and sentiments expressed in text data. In recent years, sentiment analysis has become an important tool for businesses and organizations to analyze customer feedback, monitor brand reputation, and make data-driven decisions. In this paper, we present a methodology for sentiment analysis of app reviews in six different categories using a machine learning model trained on a dataset of 20,000 app reviews. The methodology used in this study is shown in Figure 2.

1) Data Collection and Preprocessing: We collected a dataset of 20,000 app reviews for our study. To obtain these reviews, we used two different sources. Firstly, we collected reviews from the Android-App-Reviews-Dataset [6], which includes reviews from six different categories: Browser, Music, Entertainment, Social Media, Shopping, and accessibility applications. Additionally, we used the google-play-scraper Python package to crawl and extract reviews from the Google Play Store for the selected apps. The collected dataset was then split into two categories: positive and negative reviews. Prior to feature extraction, we performed the following preprocessing steps:

- *Lowercasing*
- *Contraction expansion*
- *Removal of emails*
- *Removal of URLs*
- *Removal of accented characters*
- *Removal of special characters*
- *Removal of repeated characters*

After preprocessing, the data was ready for feature extraction and model training.

2) Feature Extraction: The next step in the sentiment analysis process is feature extraction, which involves transforming the pre-processed text data into a numerical format that can be used as input to a machine learning model. In this study, we used the TfidfVectorizer class from scikit-learn to convert the pre-processed reviews into feature vectors.

The TfidfVectorizer class computes the Term Frequency-Inverse Document Frequency (TF-IDF) value of each word in the reviews. Term frequency (TF) is a measure of how frequently a word occurs in a document, while inverse document frequency (IDF) is a measure of how important the word is in the document relative to the entire dataset. The TF-IDF value for each word in the reviews is computed as the product of the term frequency and inverse document frequency values. This results in a numerical representation of each review that captures the importance of each word in the review relative to the entire dataset.

The TfidfVectorizer class also performs additional preprocessing steps, such as removing stop words (common words that do not carry much meaning, such as "the" and "a") and limiting the number of features based on their frequency in the dataset. This helps to reduce the dimensionality of the feature space and improve the performance of the machine learning model.

After feature extraction, the dataset is represented as a matrix of numerical values, with each row representing a review and each column representing a feature (i.e., a word in the reviews). This matrix can then be used as input to a machine learning model for training and prediction.

3) Model Training and Evaluation: After performing the feature extraction step, the next step in the sentiment analysis process is to train a machine learning model on the preprocessed data. In this study, we trained a Linear Support Vector Classification (LinearSVC) model using the Scikit-Learn library. To evaluate the performance of the trained model, we split the preprocessed data into training and testing sets with a split ratio of 80/20. This means that 80% of the

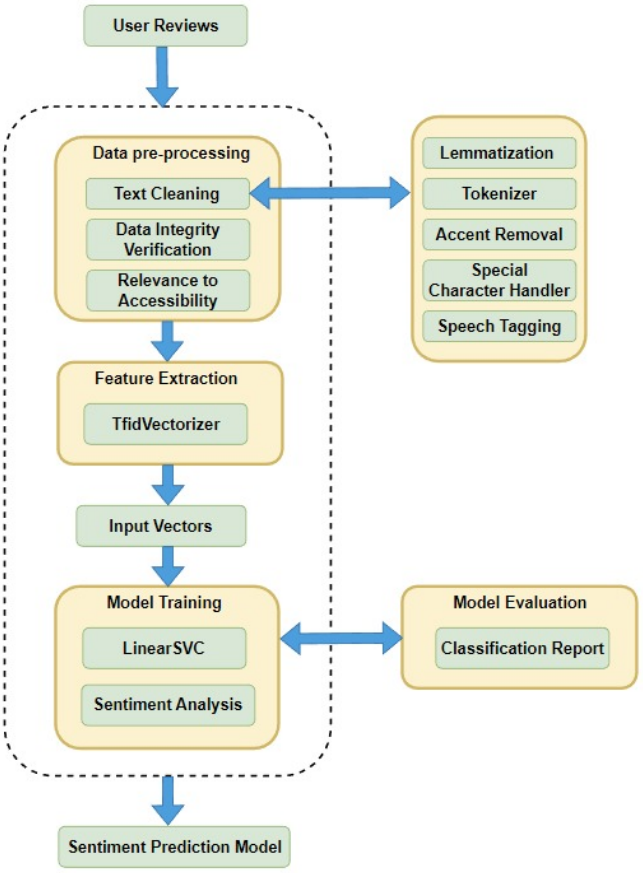

Fig. 2. Generating the Sentiment prediction Model

data was used for training the model, while the remaining 20% was used for testing the model's performance.

We used the classification report from Scikit-Learn to evaluate the performance of the trained model on the testing set. The classification report provides a detailed summary of the model's performance, including precision, recall, and F1-score metrics for each class (positive and negative). By analyzing the classification report, we can determine the accuracy of the model in predicting positive and negative sentiment in the reviews. This evaluation step helps us to assess the effectiveness of the machine learning model and identify areas for improvement. Table I shows the precision, recall, and F1 score for a classification model. The classification report shows that the model achieved an overall accuracy of 89%, with a precision of 0.89 and recall of 0.89 for the negative class, and a precision of 0.90 and recall of 0.90 for the positive class. The F1-score for both classes was also high, with 0.89 for negative and 0.90 for positive. The macro-average F1-score was 0.89, indicating that the model performed well in predicting both classes. These results suggest that the LinearSVC model is effective in predicting sentiment in customer reviews, and can be used for practical applications.

We attempted to use NLTK's Vader [7] and TextBlob [8] libraries for sentiment analysis on app reviews, but found that

TABLE I
CLASSIFICATION REPORT FOR SENTIMENT ANALYSIS MODEL

	precision	recall	f1-score
negative	0.89	0.89	0.89
positive	0.90	0.90	0.90
accuracy			0.89
macro avg	0.89	0.89	0.89
weighted avg	0.89	0.89	0.89

they did not provide satisfactory accuracy for our needs. Vader and TextBlob use rule-based and lexicon-based approaches to analyze text sentiment, respectively. However, these methods can be limited in their accuracy, especially when applied to complex and nuanced language. We found that machine learning models, such as the Linear Support Vector Classification model used in this study, were better suited to our needs as they were able to learn patterns and associations in the text data to make more accurate predictions.

IV. Experimental Setup

The experimental setup for this study involved running the software on a virtual machine with the following specifications: Ubuntu 20.04 operating system, 2 virtual CPU cores, and 8 GB of RAM. The virtual machine was used to ensure a consistent and isolated environment for the experiments, while also minimizing any potential interference from other applications or processes running on the host system. The software was installed and configured on the virtual machine, and the experiments were conducted using this setup.

We evaluated a total of 30 mobile applications across 6 distinct categories, which included Browser, Entertainment, Social Media, Music, Shopping, and Accessibility. For each category, we selected the top 5 most downloaded applications.

V. Evaluation

We evaluate the findings of the study by investigating the following five research questions:

RQ1: How do popular Android apps implement accessibility features, and how do these implementations vary across different app categories?

RQ2: What is the historical trend of accessibility implementation and its frequency in popular mobile applications?

RQ3: What is the level of user satisfaction for the accessibility features in popular categories of Android apps, and to what extent do they express their opinions in user reviews?

RQ4: How do the accessibility feature reviews of popular apps compare to those of less well-known apps within the same category?

RQ5: Can user reviews and accessibility API usage be used to reliably detect accessibility defects in Android applications?

A. RQ1: How do popular Android apps implement accessibility features, and how do these implementations vary across different app categories?

The objective of the study was to examine the prevalence and types of accessibility code implementations found in

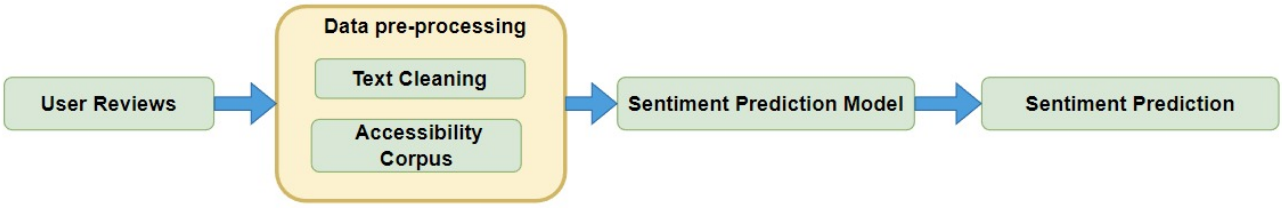

Fig. 3. Sentiment Computation using the generated model

the most popular mobile apps and the differences in these implementations across various app categories. The study's discoveries are significant in comprehending the present state of accessibility in mobile applications and recognizing areas that require enhancement.

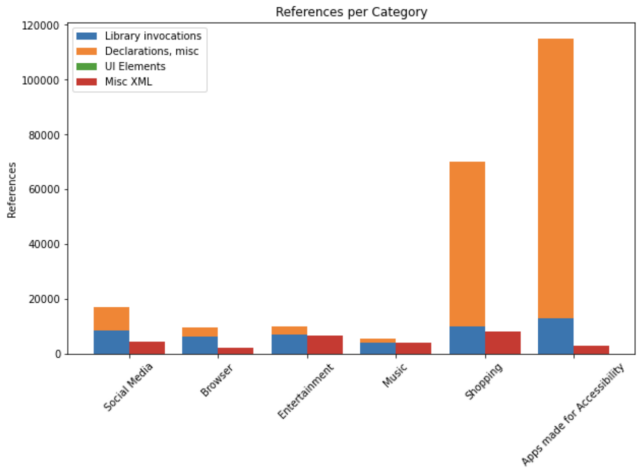

Fig. 4. Number of Accessibility API References by category

The data collected for this study includes information on the number of library method invocations, variable declarations, imports, labels, styling, and UI elements with accessibility attributes present in the 5 most popular mobile applications across different app categories. By analyzing this data, we were able to identify some interesting patterns.

The graph in Figure 4 and Table II shows the number of accessibility-related code references across the different app categories Firstly, it was observed that the number of library method invocations and variable declarations varied significantly across different app categories. For example, the Shopping app category had the highest number of variable declarations and imports, while the Apps made for Accessibility category had the highest number of library method invocations.

Secondly, we found that the number of UI elements with accessibility attributes was relatively low across all app categories, with only 20-25 UI elements with accessibility attributes per app. This suggests that there is a need for

TABLE II
APP METRICS BY CATEGORY

Category	Library invocations	Declarations, misc	UI Elements	Misc XML
Social Media	8540	8747	20	4490
Browser	6348	3276	25	2388
Entertainment	7063	3123	25	6828
Music	3984	1708	20	3991
Shopping	10110	59985	25	8031
Apps for Accessibility	12899	102225	25	3077

developers to pay more attention to the implementation of accessibility attributes in their mobile applications.

Thirdly, it was observed that the Apps made for Accessibility category had the highest number of labels and styling, which is expected given that this category is specifically designed to be accessible for individuals with disabilities.

The study offers valuable perspectives on the present condition of mobile app accessibility and identifies specific areas that require enhancement. The study's results can aid developers and designers in enhancing the accessibility of their mobile applications for people with disabilities.

B. RQ2: What is the historical trend of accessibility implementation and its frequency in popular mobile applications?

We employed a rigorous methodology to analyze the accessibility-related code references in popular Android applications to investigate the use of accessibility services in mobile applications. To answer the research question, we obtained the latest APK files of popular Android applications, and APK files of the same apps with versions published 5 years ago. All the APKs were sourced from AndroZoo.

Our analysis, along with the bar graph presented in Figure 4, demonstrates a significant improvement in the frequency and implementation of accessibility features in the most popular mobile applications over the past five years. We observed a substantial increase in the number of accessibility-related code references in the latest app versions across all app categories, with the greatest number of references in apps designed explicitly for disabled users.

Furthermore, Figure 4 reveals that shopping apps had the highest number of accessibility-related code references, with an average of 15,630 references per app, followed by social

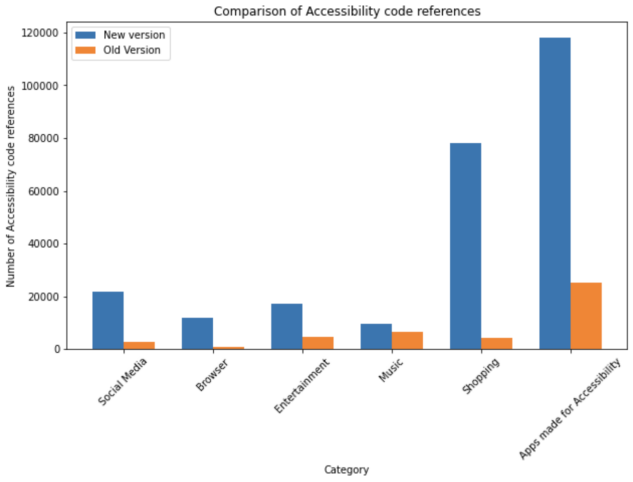

Fig. 5. Analysis of Accessibility code references over time

media and entertainment apps. Browser and music apps had relatively lower numbers of accessibility-related code references.

The total number of accessibility-related code references across all app categories in the latest app versions was 154,753, compared to 19,880 in app versions from 5 years ago, representing an increase of approximately 680%. The average number of accessibility-related code references per app in the latest app versions was 25,792, compared to 3,312 in the app versions from 5 years ago.

Comparing the results to the data from app versions 5 years ago, we found that there has been a substantial increase in accessibility implementation in all categories. The number of accessibility-related code references in the latest app versions was much higher than in the older versions, with apps made for disabled users showing the most significant increase.

Apps made for disabled users had the highest average number of accessibility-related code references per app in both the latest and older app versions, with an average of 59,113 references per app in the latest version and 12,672 references per app in the older versions. The type of accessibility implementation has also evolved, with more apps implementing newer accessibility features such as haptic feedback and voice control.

Overall, our findings suggest that there has been a positive trend in the implementation of accessibility features in top downloaded mobile applications over time, with a particular focus on apps made for disabled users. The results of our study can be used to guide future development efforts aimed at improving the accessibility of mobile applications for all users.

C. RQ3: What is the level of user satisfaction for the accessibility features in popular categories of Android apps, and to what extent do they express their opinions in user reviews?

In order to investigate the level of user satisfaction for the accessibility features in popular categories of Android apps, the study analyzed user feedback for 30 apps across 6 different categories. A total of over 11 million reviews were considered in the study. The results of the study show that the sentiment expressed in user reviews varies by category. The categories with the highest percentage of positive sentiment are Browser, Music, and Shopping, with positive sentiment percentages of 58%, 54.3%, and 61.3%, respectively. In contrast, the categories with the highest percentage of negative sentiment are Entertainment and Social Media, with negative sentiment percentages of 61.3% and 54.7%, respectively. Accessibility Apps have a balanced sentiment distribution, with a slightly higher percentage of positive sentiment (52.6%). In terms of the number of negative and positive reviews, the category with the highest number of positive reviews is Accessibility Apps, with 2,700 positive reviews. The category with the highest number of negative reviews is Entertainment, with 2,724 negative reviews.

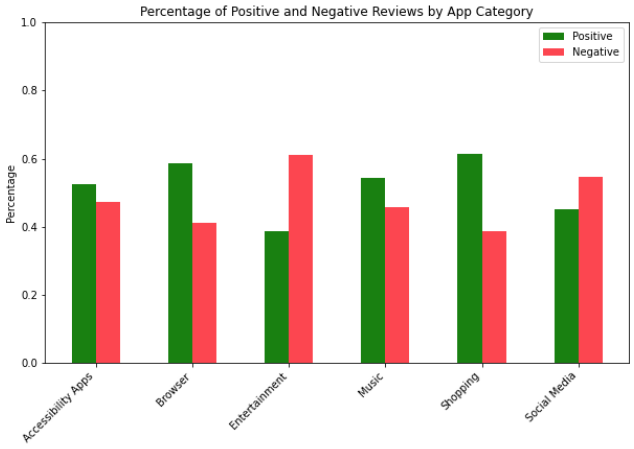

Fig. 6. Comparing the review sentiments across different categories

The graph in Figure 6 shows the percentage of user reviews across different categories. Table III provides a breakdown of the results for all the categories used in the study. The categories included in the study were music, social media, browser, shopping, entertainment, and accessibility apps. In addition to analyzing the overall satisfaction level for each category, the study also provided specific findings for each app considered in the experiment.

TABLE III
USER SATISFACTION FOR ANDROID APP CATEGORIES

Category	Sentiment	Positive	Positive%	Negative	Negative%
Accessibility Apps	5130	2700	52.63%	2430	47.36%
Browser	1823	1072	58.80%	751	41.18%
Entertainment	4446	1722	38.73%	2724	61.26%
Music	1336	725	54.27%	611	45.72%
Shopping	2297	1409	61.34%	888	38.65%
Social Media	2890	1308	45.26%	1582	54.73%
Total	17822	8926	50.02%	8896	49.97%

D. RQ4: How do the accessibility feature reviews of popular apps compare to those of less well-known apps within the same category?

The purpose of this study is to examine the sentiment distribution of popular apps versus lesser-known apps across multiple categories. For the lesser-known apps, we selected applications with an average rating of more than 4 and less than 1 million downloads. We collected sentiment data for both groups of apps from online reviews and used it to calculate the percentage of positive and negative sentiments for each category of apps. We then compared the sentiment distribution between the two groups using descriptive statistics and visualizations. Figure 7 displays a graph of the positive and negative percentages across different distributions of apps and multiple categories.

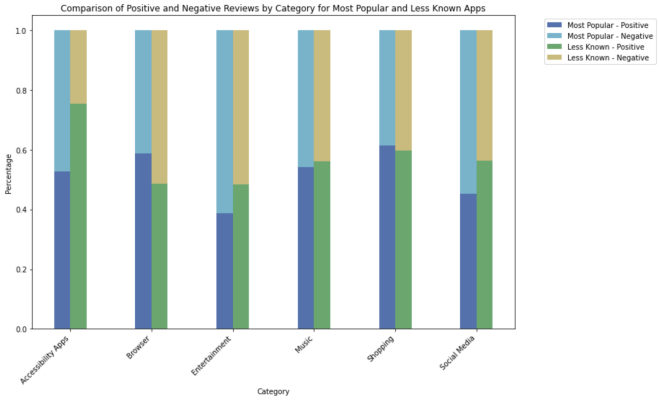

Fig. 7. User satisfaction rates of popular and less known apps

The results show that the sentiment distribution of popular apps differs significantly from less known apps across various categories. For example, in the Accessibility Apps category, the negative sentiment percentage for less-popular apps is 24.62%, while for popular apps it is only 47.36%. Similarly, in the Entertainment category, the negative sentiment percentage for less-popular apps is 51.72%, while for popular apps it is only 38.73%.

Furthermore, we observed variations in the sentiment distribution across categories within each group. For example, in the grouped category, the Shopping category had the highest positive sentiment percentage (61.34%), while the Social Media category had the highest negative sentiment percentage (54.73%). In the less-popular category, the Accessibility Apps category had the highest positive sentiment percentage (75.35%), while the Browser category had the highest negative sentiment percentage (51.33%).

E. RQ5: Can user reviews and accessibility API usage be used to reliably detect accessibility defects in Android applications?

Whether user reviews and accessibility API usage can be used to reliably detect accessibility bugs in Android applications is an important one, as it has significant implications for the development and testing of accessibility features in mobile apps. However, the findings of the study indicate that there is not a strong enough association between the two to definitively detect bugs. Additionally, the lack of user reviews related to accessibility features in many apps further hinders the development of a reliable bug detection mechanism. While some correlations were found between user reviews and API usage, more research and data like app usage metrics, user surveys and resolved bug documentation are needed to develop a reliable method for detecting accessibility bugs in mobile apps.

VI. Threats to Validity

While user reviews are an important source of information for understanding the accessibility of mobile applications, it is important to note that not all apps have reviews related to accessibility. However, the apps we have chosen for our analysis are popular and have a sufficient number of reviews related to accessibility. This ensures that we have enough data to perform a meaningful analysis of the sentiment and content of these reviews. By analyzing the reviews for these apps, we can gain insights into the effectiveness of the accessibility features and identify areas for improvement, ultimately making the apps more accessible and user-friendly for people with disabilities.

One potential threat to the validity of our study is the limited accuracy of the sentiment analysis technique used to identify sarcasm in user reviews. Given that sarcasm can significantly alter the meaning of a review, failing to detect it may lead to inaccurate conclusions about the overall sentiment of the review data. To mitigate this limitation, future research could investigate more advanced sentiment analysis techniques that can effectively detect and account for sarcasm in user reviews.

Another potential threat to the validity of our study is the possibility that users who encounter accessibility issues with an app may uninstall it without leaving a review. This can result in an under-representation of negative reviews, making it challenging to obtain a complete understanding of the user sentiment towards accessibility features. To address this limitation, researchers could consider complementing the analysis of user reviews with other sources of data, such as app uninstall rates or user engagement metrics. Additionally, researchers could reach out to users who have uninstalled the app and conduct surveys or interviews to understand the reasons behind their decision and determine whether accessibility issues played a role.

Lastly, the decompiling process used to identify Accessibility API usage may not be able to discern code obfuscation techniques used by developers to hide their implementation details. To mitigate this issue, the accuracy of the decompiling process was verified by comparing the results with those obtained from other decompiling tools or performing manual code analysis to verify the findings in open-source apps.

VII. Future Work

Version-based user satisfaction analysis can be an area of future work in the study of accessibility features in Android

applications. This approach involves comparing the sentiment of user reviews across different versions of the same app to identify improvements or regressions in the accessibility features over time. It can help developers to understand the impact of changes in their accessibility implementations and prioritize areas for improvement.

Another potential area for future work is the development of more advanced sentiment analysis techniques that are better equipped to handle the nuances of user reviews. While the sentiment analysis used in this study was able to provide a general sense of user satisfaction, it was not able to accurately detect sarcasm or other complex expressions of emotion that may have been present in some reviews. Additionally, the study was limited to the English language, which may not accurately represent the opinions of users who write reviews in other languages.

After decompiling the apps, it is possible to determine the relationship between code elements using advanced techniques such as behavioral analysis. Such techniques can provide more in-depth insights into the usage of accessibility APIs and help identify potential issues that may not be apparent through manual inspection. By analyzing the behavior of the app, researchers can identify patterns in the usage of accessibility APIs and probably correlate them with other sources of information including user surveys and usage metrics, to help gain a better understanding.

VIII. Conclusion

In conclusion, the study analyzed over 30 mobile apps from 6 different categories and examined over a million user reviews to gain insights into the implementation and user satisfaction of accessibility features in Android apps. The analysis revealed the percentage of positive and negative reviews for each category and each app, indicating that entertainment apps had the most negative reviews while shopping apps had the most positive reviews related to accessibility. Furthermore, a sentiment analysis model with a precision of 89% was used to assess the sentiment of the reviews. In addition, accessibility API usage metrics were found by decompiling the apps. However, the study also found that user review and usage metrics were not strongly associated enough to build a reliable bug detection mechanism. The study suggests that further research is needed to develop a more robust method for detecting accessibility defects in mobile apps.

Acknowledgment

We would like to acknowledge the support of the Donald Bren School of Information and Computer Sciences for providing the virtual machines used in this study. Additionally, we would like to express our gratitude to Professor Sam Malek for his guidance throughout the project.

References

[1] W. Aljedaani, M. W. Mkaouer, S. Ludi and Y. Javed, "Automatic Classification of Accessibility User Reviews in Android Apps," *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, Riyadh, Saudi Arabia, 2022, pp. 133-138, doi: 10.1109/CDMA54072.2022.00027.

[2] Marcelo Medeiros Eler, Leandro Orlandin, and Alberto Dumont Alves Oliveira. 2019. "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 (*IHC '19*). Association for Computing Machinery, New York, NY, USA, Article 23, 1–11. https://doi.org/10.1145/3357155.3358477

[3] W. Aljedaani, F. Rustam, S. Ludi, A. Ouni and M. W. Mkaouer, "Learning Sentiment Analysis for Accessibility User Reviews," *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, Melbourne, Australia, 2021, pp. 239-246, doi: 10.1109/ASEW52652.2021.00053.

[4] Diao, Wenrui, Yue Zhang, Li Zhang, Zhou Li, Fenghao Xu, Xiaorui Pan, Xiangyu Liu, Jian Weng, Kehuan Zhang and Xiaofeng Wang. "Kindness is a Risky Business: On the Usage of the Accessibility APIs in Android."*International Symposium on Recent Advances in Intrusion Detection (2019)*.

[5] Huang, J., Backes, M., & Bugiel, S. (2021). A11y and Privacy don't have to be mutually exclusive: Constraining Accessibility Service Misuse on Android. *USENIX Security Symposium*.

[6] A. Mittal, "Android-App-Reviews-Dataset," GitHub. [Online]. Available: https://github.com/amitt001/Android-App-Reviews-Dataset. [Accessed: 11 Mar. 2023]

[7] Hutto, C. J., & Gilbert, E. E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*. Ann Arbor, MI, USA.

[8] Loria, S. (2018). TextBlob: Simplified Text Processing. Accessed on September 5, 2021. [Online]. Available: https://textblob.readthedocs.io/en/dev/