Automatic Classification of Accessibility User Reviews in Android Apps

Wajdi Aljedaani*, Mohamed Wiem Mkaouer†, Stephanie Ludi *, and Yasir Javed ‡
*University of North Texas. Email{wajdi.aljedaani, Stephanie.Ludi@unt.edu}
†Rochester Institute of Technology. Email{mwmvse@rit.edu}
‡Prince Sultan University. Email{yjaved@psu.edu.sa}

***Abstract*—In recent years, mobile applications have gained popularity for providing information, digital services, and content to users including users with disabilities. However, recent studies have shown that even popular mobile apps are facing issues related to accessibility, which hinders their usability experience for people with disabilities. For discovering these issues in the new app releases, developers consider user reviews published on the official app stores. However, it is a challenging and time-consuming task to identify the type of accessibility-related reviews manually. Therefore, in this study, we have used supervised learning techniques, namely, Extra Tree Classifier (ETC), Random Forest, Support Vector Classification, Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression for automated classification of 2,663 Android app reviews based on four types of accessibility guidelines, i.e., Principles, Audio/Images, Design and Focus. Results have shown that the ETC classifier produces the best results in the automated classification of accessibility app reviews with 93% accuracy.**

***Index Terms*—Mobile Applications, User Reviews, Accessibility, Android, Machine Learning.**

I. INTRODUCTION

Many users find it challenging to get complete benefit from mobile applications (apps) having poor accessibility [27], [4], [24], [23]. To address this challenge, researchers have offered variety of methodologies, techniques, frameworks, tools, and guidelines to guide the development of building mobile applications with better accessibility[21]. It is unfortunate that, due to a lack of understanding or resources (e.g., funding and time), many mobile application developers and designers continue to neglect to include accessibility in their mobile app development process [22]. In this paper, we seek to develop a multi-class method that can help app developers to distinguish between the type of accessibility user reviews easily. The proposed method will automatically classify user reviews derived from app stores, such as Google Play[1], and Apple Appstore[2], into the accessibility type based on the accessibility guideline [6].

There are many challenges with detecting accessibility related to user reviews. One of the most common ways of improving applications is by analyzing the feedback given by the users [9], [1], [18], [2]. In many cases, accessibility user reviews for mobile applications are overlooked [12]. It is vital to mention that accessibility user reviews can either be detected automatically or manually [12]. Given the large number of app users' reviews, manual identification becomes more tedious and time-consuming, meaning that automatic identification is often preferred. Automatic identification of reviews means that the system looks for certain keywords in the user reviews that relate to accessibility [12]. The British Broadcasting Network (BBC) accessibility guidelines provide the keywords used for automatic identification [6]. Although automatic identification is convenient, its major disadvantage is that it does not capture words in the user reviews that are not in the accessibility guidelines. In addition, even when the keywords are in a certain user review, it is not a guarantee that the review concerns accessibility. In past studies [12], researchers found phrases in user reviews with keywords from the guidelines, which were not necessarily about accessibility. Hence, researchers should be careful to consider the context of the review so that identification will be effective. Such a challenge can be overcome by introducing learning capabilities, which are trained to know the difference between accessibility user reviews and those that are not, even if they seem similar. Furthermore, not all accessibility problems uniformly occur, and therefore, some accessibility violations tend to be more frequent than others. This can potentially be another challenge for any automated solution that tries to identify them, since one category will be more popular (better represented by data) than another.

To address the above-mentioned challenges, the goal of this paper is to help developers automatically classify user reviews, into what type of accessibility guidelines they are referring to (Principles, Audio/Video, Design, Focus, Forms, Images, Links, Notifications, Dyn.content, Structure, and Text Equivalent). This will help developers quickly distinguish accessibility related problems, and address them on a timely manner.

To design our solution, we rely on supervised learning techniques to effectively enable app developers to correctly identify the underlying accessibility problem in the user reviews. We analyze a corpus of user reviews, extracted from open source apps [12] to extract accessibility problems. Since our goal is to design a model that distinguished between types of accessibility issues, we manually categorized the user reviews based on accessibility guidelines [6]. Then, we employ emerging machine learning techniques, known to perform best in text classification [15], to know the features of the

[1]https://play.google.com/store
[2]https://www.apple.com/ios/app-store/

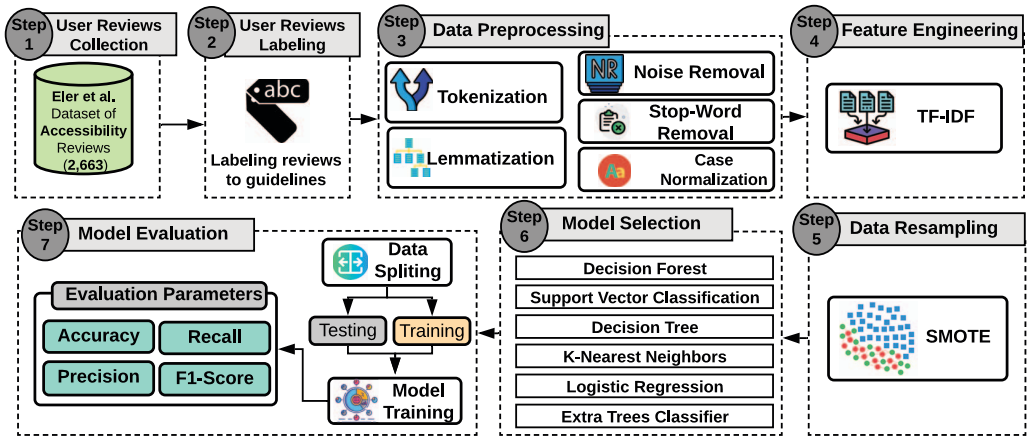

Fig. 1: Overview approach of our study.

reviews that can help in their identification. Our proposed approach gets acquainted with the keywords and patterns that are unique to given type of accessibility guideline, and transform them into features to identify a given class (type of accessibility guideline). Determining unique features is crucial for classification algorithms. The outcome of the algorithm (labeled review) is important for app developers to understand accessibility issues and improve on them.

Following are the key contributions of our research:

- We tackle the identification of accessibility user reviews as a multi-class classification problem, where we analyze the extent to which, machine learning models can accurately distinguish between types of accessibility reviews.
- To handle the unbalance between the number of reviews belonging to each category, we adopt the state-of-the-art re-sampling technique SMOTE. This paper also showcases the potential of class re-balancing on supporting the representation if minority classes (i.e., categories with fewer reviews).

II. RELATED WORK

Accessibility in mobile applications is crucial to ensuring that all users can benefit from them. Accessibility in mobile applications is important in ensuring that all users are able to make use of such apps. In this section, we divide into two subsections: accessibility in user reviews; and text document classification.

Accessibility in user reviews. Providing accessibility feedback on an app can help designers improve it [26], but many people do not leave their reviews if they encounter app problems [12]. A study by Eler et al. has proven that only 1% of mobile app users leave feedback if the app has an accessibility issue [12]. The study, which utilized 214,053 mobile app reviews, had only 2,663 related to accessibility [12]. Given that app development can be significantly improved by analyzing user reviews. We will seek to develop a method that can help distinguish between accessibility types in user reviews. Such an undertaking is significant because developers can easily use the technique to select user reviews related to non-accessibility, analyze them, and utilize them to make better apps that can compete favorably in the market. We will be developing our method using the dataset produced by Eler et al. in this study.

Text documents classification. Different approaches have been developed to classify reviews, depending on the researchers' objective. While some methods seek to classify reviews according to bug reports, others focus on complaints or address future feature requests. It is prudent to note that most of them fail to address accessibility. The use of redefined keywords to classify reviews is an improvement from the automatic classification method. Previous studies by Eler et al. used 213 keywords [12], while Ratzinger et al. utilized 13 keywords to analyze user reviews. Our previous study [3] utilized machine learning to classify reviews into either accessibility or non-accessibility, but it only performs binary classification. In this study, we develop machine learning algorithms to classify the type of accessibility in user review.

III. STUDY DESIGN

This section describes the proposed method to classify the type of accessibility in review on the selected dataset. Figure 1 presents an overview approach of our study.

A. Step (1): User Reviews Collection

In this study, we used a corpus of user reviews, collected from various popular open source projects [12]. These reviews are collected from thousands of users from all over the globe. The dataset contains 2,663 reviews that have been manually inspected for containing a problem related to accessibility. The reviews were gathered from 701 Android applications, belonging to 15 different categories, as shown in Table II.

B. Step (2): User Reviews Labeling

We need to categorize the dataset based on the mobile accessibility guideline [6]. To do so, two authors performed the manual categorization of all user reviews in the dataset. The process of categorizing was spread across seven days to prevent human fatigue. The authors were also provided with the chance to search online for unfamiliar keywords during labeling. After the authors finished the manual categorizing procedure, the dataset was validated using the process of Levin

TABLE I: Summary of accessibility guidelines with corresponding description, relevant keywords, and number of labelled reviews. We followed the BBC standards and guidelines for mobile accessibility [6].

Guideline	Description	Relevant Keywords	# of Labelled Reviews
Principles	These guidelines require a focus on three principles of developing usable and inclusive applications. First, developers should utilize all web standards as required. Secondly, there should be utilization of interact controls. Thirdly, content and functionality in the app should support native features of the app.	Accessibility, disability, operable, screen reader, blind talkback, , impaired, impairment	664
Audio/video	Applications should provide alternative formats such as transcripts, sign language, or subtitles. Autoplay should be disabled, and the user should be provided with play/pause/stop or mute buttons to control audio. There should be no conflict between audio in application media of native assistive technology.	Subtitle, sign language, transcript, audio description, autoplay, mute, volume, can't hear	311
Design	The color in the app background should have appropriate contrast, and touch targets must be large enough to be touched effectively. Visible state change should be experienced in every item in the app that has been focused on. Unnecessary or frequent flickering of content must be avoided.	Contrast, background color, flicker, font size, visual cue, dark/light mode, eyestrain, seizure, can't see, overlap	1,328
Focus	There should be a logical organization of items, and users should be offered alternative input methods. Interactive and inactive elements should be focusable and non-focusable, respectively. Keyboard traps should be eliminated, and focus should not change suddenly when the app is utilized.	Focusable, control focus, focus, keyboard trap, navigable, order, input/type	122
Forms	Every form of control must have a label. All labels must have a logical grouping, and a default input format must be given. Labels should be close to their form controls.	Unique label, missing label, layout, voice-over, visible label	53
Images	Text images should not be included. Any background images that have content should have another accessible alternative.	Image of text, hidden text, background image, text alternative	86
Links	Any navigation links must indicate the function of the link. If a link to an alternative format is clicked, the user should be notified of the redirection to the alternative. Several links that redirect to the same sourceshould be put together in one link.	Link description, unique desc., duplicate link, alternative format	35
Notifications	Error messages should be clear. Any notifications given must be easily seen or heard. There should be standard system notifications where necessary.	Operating inclusive, vibration, feedback, alert dialog, understandable, unfamiliar	49
Dyn. content	Applications should be made in a progressive manner that enables every user to benefit from them. Appropriate notifications should be given for automatic page refreshes. Flexible interaction input control must be given.	Animated content, page refresh, automatic, refresh, timeout, adaptable, input sign	15
Structure	Every page on the application should be uniquely identified. Content should be arranged in a hierarchical and logical manner with appropriate headings. One accessible component should be used to group interface objects, controls or elements.	Page title, screen title, heading, header, unique descriptive	0
Text equivalent	Applications should give the objective of a specific image or its editorial aim. Also, visual formatting must be complemented by other ways to give meaning. There should be no conflict between decorative images with assistive technology. Every element must have well-placed and effective a11y properties.	Alternative text, non-visual, content description, decorative content, no-text-content	0

et al. [17] by randomly choosing 9% sample of accessibility reviews. The sample size was 243 out of the 2,663 accessibility reviews. This number is about equivalent to the size of the sample based on a 95% confidence level with a 6 confidence interval. Following that, the third author categorized them. The chosen reviews were not exposed to the author before. Then, the categorical reviews were evaluated using Cohen's Kappa coefficient [10] with respect to inter-rater agreement level, and the "0.87" agreement level was attained. As per Fleiss et al. [13], (i.e., 0.87–1.00) are perfect values for agreement, and our agreement values are considered nearly *perfect agreement*. Table I presents the accessibility guidelines and the distribution of the categorized user reviews per guideline.

TABLE II: Statistics of the dataset.

Number of Apps	701
App Categories	15
All Reviews	214,053
Accessibility Reviews	2,663

C. Step (3): Data Preprocessing

We used a textual preprocessing strategy after finishing the process of collecting data. It is vital to preprocess and clean the document adequately so a model can execute text categorization successfully. In our method, we combined NLP methods employing the NLTK python (Natural Language Toolkit) to preprocess the reviews of the app. Among the techniques based on NLP are:

- **Tokenization:** This technique involves splitting natural texts into tokens without any white space throughout this procedure. Tokenizing app reviews involves breaking them down into constituent words set.
- **Lemmatization:** Throughout this procedure, a word's suffix is replaced or removed so it's basic form can be obtained. It also lowers the unique occurrence's' count of words that are similar. This approach is used in the suggested strategy for word pre-processing in their canonical format in order to limit the unique occurrences count of identical text tokens.
- **Stop-Word Exclusion:** Stop-words are words that do not assist to the process of classification, such as the, am, and so on.
- **Case Normalization:** Because precise words having various font cases must be treated in a similar way, such as "accessibility" & "Accessibility," the entire text must be converted to lowercase letters. It is commonly referred to as data cleansing because it aids in minimizing the repetition of similar features that vary only in regards to case sensitivity. A person might identify himself as "Deaf" using a capital letter 'D' with in setting of reviews related to accessibility to convey his cultural background with in reviews. Because we have multi-class classifier, the classification outcome for "deaf" and "Deaf would be identical, and case normalization would be secure, there would be no overruling of expressions by the user.
- **Noise Removal:** This stage removes any noise that could degrade performance of the classification or cause the model to become confused during learning. Special characters, numeric data, email id, are examples of noise types deleted in this phase.

D. Step (4): Feature Engineering

Feature engineering helps the model learn patterns for each class it is trying to distinguish, through allocating appropriate unique key words that appear specifically for one category. We intend to train our model and find these unique keywords and use them as features to properly distinguish between classes. The following is feature engineering method that was employed in this study:

1) TF-IDF: is among the most commonly employed scoring metrics for summarization and information retrieval. It is

utilized to convey the significance of the term within each text. The TF-IDF extraction function takes two inputs: IDF and TF. TF-IDF provides tokens that seem to be uncommon within the dataset. When uncommon words appear in multiple documents, their relevance grows.

$$tfidf_{t,d,D} = tf_{t,d} \cdot idf_{t,D} \quad (1)$$

where ***t*** denotes terms, ***d*** denotes each document, and ***D*** is the documents set. The parameter, n-gram range, is used in conjunction with TF-IDF. TF-IDF is used to compute word weights, which offer corpus weights for any given the word. The weighted word matrix is the output. Using TF-IDF vectorizer, an increase within meaning is proportionate of the count, although word frequency in the corpus assists in controlling it. The TF approach is frequently explored for extracting features and therefore is widely utilized for text categorization. During classifier training, the incidence frequency of terms' is used as a parameter. TF function doesn't quite take into account the popularity of a word, contrasting the TF-IDF, which gives less weight to more frequent terms.

E. Step (5): Data Re-Sampling

Imbalanced dataset issues and problems can be resolved through methods of data re-sampling. The problem that can arise in an imbalanced dataset is that it has an uneven ratio of the target classes, which results in the models over-fitting on the majority class during classification [19]. For this, the technique and strategy for re-sampling the dataset have been proposed. Throughout this study, the technique on re-sampling that has been utilized is over-sampling.

1) Synthetic Minority Over-Sampling Technique (SMOTE): In over-sampling, the sample of the class that is a minority increased in the ratio of the majority class. This enlarges the size of the dataset and provides more features that can train the model and improve its accuracy. Over-sampling is implemented in this research using SMOTE, known as the synthetic minority over-sampling technique. SMOTE is a modern approach and was presented in [8] to figure out how over-fitting in the imbalanced dataset could be overcome. It selects the smaller class at random and locates the K-nearest neighbors for each of these classes. The K-nearest neighbor is used to analyze the samples that are chosen to create a new minority class. Figure 2 shows the distribution of the accessibility reviews before and after using SMOTE.

TABLE III: Summary of performance measures, formulas, and definitions.

Measures	Formula	Definition
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Calculates the closeness of a measured value to the standard value.
Recall	$\frac{TP}{TP+FN}$	Calculates the exact number of positive predictions that are actually observed in the actual class.
Precision	$\frac{TP}{TP+FP}$	Calculates the exact no. of correct predictions out of all the input sample.
F1-score	$\frac{2 \cdot P \cdot R}{P+R}$	Calculates the accuracy from the precision and recall.

F. Step (6): Model Selection

In order to build our model, we used the following six learning algorithms:

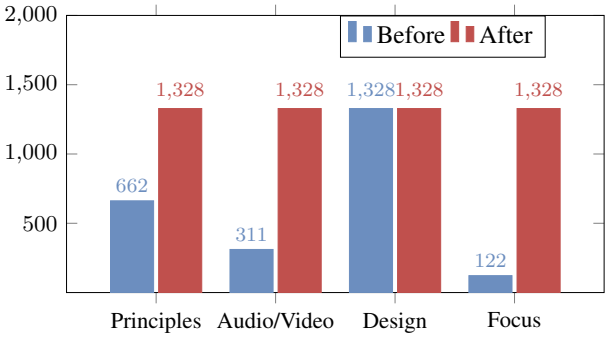

Fig. 2: Distribution of reviews before and after SMOTE.

- **Random Forest (RF):** is a tree-based classifier that constructs a large number of classification trees. Each tree gives a distinct classification [5]. RF selects the classification with the most votes out of all possible trees to classify a new item.
- **Support Vector Classification (SVC):** is a well-known machine learning classifier for tackling linear & non-linear issues. It's suitable for a variety of practical applications [25]. SVC generates a hyperplane or line that splits the data in the section. Higher-dimensional space is obtained by transforming low dimensional input space using the Kernel function; this process changes non-separable problems into separable problems. It primarily aids in the resolving of non-linear differential issues. SVC divides the data into categories according to labels.
- **Decision Tree (DT):** learns basic decision rules to forecast the class. DT utilized node and leaf by descending from the root and utilizing the sum of product representation [7].
- **K-Nearest Neighbors (KNN):** Identifies the similarities between new and existing samples and assigns the new data to a group with a high degree of similarity [11]. The similarity between both the new data and the existing classifications is determined by computing the distance between the two sets of data.
- **Logistic Regression (LR):** is a statistical model which is similar to linear regression and based on the probability concept. By fitting data to something like a logistic function, LR predicts the likelihood of the events.
- **Extra Tree Classifier (ETC):** is a collection of classification algorithms teaching method in which the results of numerous de-correlated random forests gathered within a "forest" gets merged to provide identifications' outcomes.

We chose these algorithms because they are commonly used for a variety of classification issues [20], [28], and they are able to operate effectively with imbalance datasets and NLP in the literature [14], [16].

G. Step (7): Model Evaluation

The four assessment aspects indicated in Table III below are used to evaluate the performance of our chosen models:

- **True Positive (TP):** Parameter determines positive predictions identified accurately using the classifier.

TABLE IV: Detailed classification metrics (Accuracy, Precision, Recall, and F1-Score) of each classifier with TF-IDF feature.

Random Forest (RF)				**Support Vector Classification (SVC)**				**Decision Tree (DT)**			
Category	**Precision**	**Recall**	**F1**	**Category**	**Precision**	**Recall**	**F1**	**Category**	**Precision**	**Recall**	**F1**
Principle	0.89	0.88	0.89	Principle	0.88	0.86	0.87	Principle	0.90	0.69	0.78
Audio/Video	0.94	0.97	0.96	Audio/Video	**0.95**	**0.98**	**0.96**	Audio/Video	**0.94**	**0.91**	**0.92**
Design	0.87	0.84	0.85	Design	0.85	0.82	0.84	Design	0.63	0.85	0.72
Focus	**0.97**	**0.98**	**0.97**	Focus	**0.95**	**0.98**	**0.96**	Focus	0.89	0.83	0.86
Average F1	0.92	0.92	0.92	Average F1	0.91	0.91	0.91	Average F1	0.84	0.82	0.82

K-Nearest Neighbors (KNN)				**Logistic Regression (LR)**				**Extra Tree Classifier (ETC)**			
Category	**Precision**	**Recall**	**F1**	**Category**	**Precision**	**Recall**	**F1**	**Category**	**Precision**	**Recall**	**F1**
Principle	0.50	0.99	0.66	Principle	0.88	0.86	0.87	Principle	0.92	0.89	0.90
Audio/Video	0.98	0.92	0.95	Audio/Video	0.94	**0.98**	**0.96**	Audio/Video	0.94	1.00	0.97
Design	**1.00**	0.03	0.05	Design	0.84	0.82	0.83	Design	0.89	0.85	0.87
Focus	0.97	**0.97**	**0.97**	Focus	**0.95**	0.97	**0.96**	Focus	**0.97**	**0.99**	**0.98**
Average F1	0.86	0.73	0.66	Average F1	0.90	0.91	0.90	Average F1	0.93	0.93	0.93

- **True Negative (TN):** Parameter determines whether or not the classifier accurately labels negative predictions.
- **False Positive (FP):** Parameter determines the quantity of negative cases incorrectly assumed to be positive via the classifier.
- **False Negative (FN):** Parameter determines the quantity of positive instances that the classifier incorrectly interprets as the negative instances.

IV. STUDY RESULTS

RQ_1: To what extent can machine learning models accurately distinguish different types of accessibility reviews?

In this study, we have used six models, namely, Extra Tree Classifier (ETC), Random Forest (RF), Support Vector Classification (SVC), Decision Tree (DT), K-Nearest Neighbors (KNN), and Logistic Regression (LR) for automated classification of accessibility app reviews in four different categories. These categories include Principle, Audi/Video, Design, and Focus. Four metrics, i.e., Accuracy, Precision, Recall, and F1-Score, along with TF-IDF features, are employed for each classifier. The results of detailed classification metrics of each classifier with TF-IDF features are presented in Table IV.

In the case of RF and ETC, the Focus category achieves the highest recall, accuracy, and F1-Score. In the case of RF and ETC, the highest recall, precision, and F1-Score are obtained in the Focus category. SVC classifier exhibits the same trend in Audio/Video and Focus category while DT classifier performs the best result in Audi/Video category. KNN classifier outputs the highest precision in the Design category, while the Focus category produces the highest recall and F1-score. Lastly, LR gives the highest precision in the Focus category, recall in Audio/Video, and the same F1-score in the Audio/Video and Focus category. Overall, the highest precision (1.00) is achieved by KNN in the Design category, and the highest recall (1.00) in the Audio/Video category and F1-Score (0.98) in the Focus category is achieved by ETC. The ETC classifier obtained the highest average F1-score (0.93), while KNN showed the lowest average F1-score (0.66). It can be seen that classifiers have performed well in the Audio/Video, Design, and Focus categories, whereas the lowest results are obtained in the Principle category.

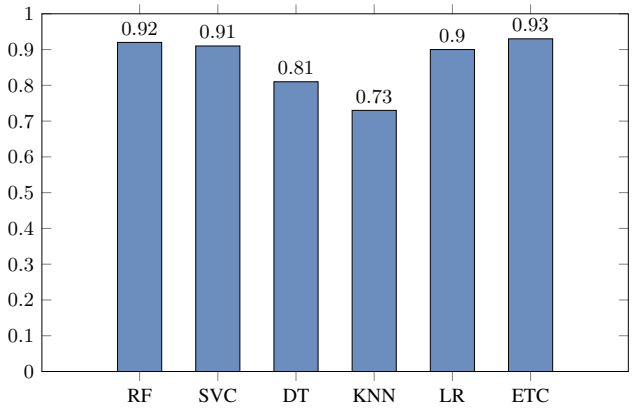

Fig. 3: Comparison of accuracy of all models.

V. DISCUSSION

We chose the four categories/guidelines (principles, audio/video, design, and focus) because they are well-represented, and we know that choosing the others would result in under-representation and an inability to categorize them accurately. At the same time, this is acceptable since the four chosen categories are popular categories that most accessibility user reviews will fit. As a result, the fact that we did not categorize all categories is a limitation. However, we believe that we have captured the primary categories of interest to developers. From the findings of this research, this section presents a discussion of the study takeaways.

Takeaway 1- App reviews represent a valuable source of information which once gathered can give detailed problems related to the accessibility of the mobile app: Accessibility guidelines are numerous, and mobile app designers and developers are in shortage of tools to prevent their appearance. In addition, observing all these guidelines does not always warrant accessibility to the app. Moreover, it is often impractical to undertake usability testing with users with disabilities, e.g., deaf or blind users. A key gap that is not addressed by existing research and testing approaches is listening to user reviews and evaluating them. This new approach is valuable and practical as it allows developers to identify accessibility problems with the app in question.

Takeaway 2- Improving accessibility testing: The current

approaches and strategies for accessibility testing are mainly manual. As a result, developers spend a considerable amount of time and cost in identifying the most appropriate people to test their apps on how well they adhere to accessibility guidelines. Accessibility scanners are already available in the market. However, they are only fit for the web and not the mobile environment. In light of this challenge, online user reviews offer new possibilities in capturing anomalies with the app from a more practical perspective. Using the reviews enables developers to identify test cases they wish to undertake in case of app upgrades. In addition, given the dynamic nature of the mobile environment, recent user reviews can indicate any new anomalies in the recently released apps.

VI. CONCLUSION

This study presented an automated approach for classifying accessibility app reviews in four categories, i.e., Principles, Audio/Video, Design, and Focus, for helping the developers detect app issues and performance improvement by considering user reviews. An existing dataset that comprises manually validated accessibility app reviews has been employed in our work. We employed six classification models, namely Extra Tree Classifier, Random Forest, Support Vector Classification, Decision Tree, K-Nearest Neighbors, and Logistic Regression. To evaluate their performance, we used four classification metrics, i.e., Accuracy, Precision, Recall, and F1-Score for measuring their performance. Evaluation results have shown that KNN exhibits the least accuracy while the ETC model outperformed other models in overall accuracy with TF-IDF features. In the future, we intend to increase the keywords and sample size to improve the selection and analysis process of accessibility reviews and provide a mechanism to check whether the developers have addressed the users' concerns in the subsequent releases by implementing the required features.

REFERENCES

[1] W. Aljedaani and Y. Javed. Bug reports evolution in open source systems. In *5th International Symposium on Data Mining Applications*, pages 63–73. Springer, 2018.

[2] W. Aljedaani, M. Nagappan, B. Adams, and M. Godfrey. A comparison of bugs across the ios and android platforms of two open source cross platform browser apps. In *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 76–86. IEEE, 2019.

[3] E. A. AlOmar, W. Aljedaani, M. Tamjeed, M. W. Mkaouer, and Y. N. El-Glaly. 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*, pages 1–15, 2021.

[4] A. Alshayban, I. Ahmed, and S. Malek. Accessibility issues in android apps: state of affairs, sentiments, and ways forward. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1323–1334. IEEE, 2020.

[5] A. Amaar, W. Aljedaani, F. Rustam, S. Ullah, V. Rupapara, and S. Ludi. Detection of fake job postings by utilizing machine learning and natural language processing approaches. *Neural Processing Letters*, pages 1–29, 2022.

[6] BBC. The BBC Standards and Guidelines for Mobile Accessibility, 2017.

[7] M. Brijain, R. Patel, M. Kushik, and K. Rana. A survey on decision tree algorithm for classification. 2014.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[9] A. Ciurumelea, A. Schaufelbuhl, S. Panichella, and H. C. Gall. Analyzing reviews and code of mobile apps for better release planning. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 91–102, Klagenfurt, Austria, Feb. 2017. IEEE.

[10] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[11] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang. Efficient knn classification algorithm for big data. *Neurocomputing*, 195:143–148, 2016.

[12] M. M. Eler, L. Orlandin, and A. D. A. Oliveira. 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*, pages 1–11, 2019.

[13] J. L. Fleiss, B. Levin, M. C. Paik, et al. The measurement of interrater agreement. *Statistical methods for rates and proportions*, 2(212-236):22–23, 1981.

[14] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[15] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[16] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[17] S. Levin and A. Yehudai. Towards software analytics: Modeling maintenance activities. *arXiv preprint arXiv:1903.04909*, 2019.

[18] X. Li, Z. Zhang, and K. Stefanidis. Mobile App Evolution Analysis based on User Reviews. page 14, 2018.

[19] B. Omar, F. Rustam, A. Mehmood, G. S. Choi, et al. Minimizing the overlapping degree to improve class-imbalanced learning under sparse feature selection: application to fraud detection. *IEEE Access*, 9:28101–28110, 2021.

[20] M. Owhadi-Kareshk, S. Nadi, and J. Rubin. Predicting merge conflicts in collaborative software development. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. IEEE, 2019.

[21] K. Park, T. Goh, and H.-J. So. Toward accessible mobile application design: Developing mobile application accessibility guidelines for people with visual impairment. In *Proceedings of HCI Korea*, HCIK '15, page 31–38, Seoul, KOR, 2014. Hanbit Media, Inc.

[22] R. Patel, P. Breton, C. M. Baker, Y. N. El-Glaly, and K. Shinohara. Why software is not accessible: Technology professionals' perspectives and challenges. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–9, 2020.

[23] A. Rodrigues, H. Nicolau, K. Montague, J. Guerreiro, and T. Guerreiro. Open challenges of blind people using smartphones. *International Journal of Human–Computer Interaction*, 36(17):1605–1622, 2020.

[24] A. S. Ross, X. Zhang, J. Fogarty, and J. O. Wobbrock. 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*, ASSETS '18, page 119–130, New York, NY, USA, 2018. Association for Computing Machinery.

[25] N. Safdari, H. Alrubaye, W. Aljedaani, B. B. Baez, A. DiStasi, and M. W. Mkaouer. Learning to rank faulty source files for dependent bug reports. In *Big Data: Learning, Analytics, and Applications*, volume 10989, page 109890B. International Society for Optics and Photonics, 2019.

[26] S. Yan and P. Ramachandran. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)*, 12(1):1–31, 2019.

[27] S. Yan and P. G. Ramachandran. The Current Status of Accessibility in Mobile Apps. *ACM Transactions on Accessible Computing*, 12(1):1–31, Feb. 2019.

[28] Y. Zhou and A. Sharma. Automated identification of security issues from commit messages and bug reports. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 914–919, 2017.