Finding the Needle in a Haystack: On the Automatic Identification of Accessibility User Reviews

Eman Abdullah AlOmar
Rochester Institute of Technology
Rochester, New York, USA
eman.alomar@mail.rit.edu

Wajdi Aljedaani
University of North Texas
Denton, Texas, USA
wajdialjedaani@my.unt.edu

Murtaza Tamjeed
Rochester Institute of Technology
Rochester, New York, USA
mt1256@rit.edu

Mohamed Wiem Mkaouer
Rochester Institute of Technology
Rochester, New York, USA
mwmvse@rit.edu

Yasmine N. El-Glaly
Western Washington University
Bellingham, Washington, USA
elglaly@wwu.edu

ABSTRACT

In recent years, mobile accessibility has become an important trend with the goal of allowing all users the possibility of using any app without many limitations. User reviews include insights that are useful for app evolution. However, with the increase in the amount of received reviews, manually analyzing them is tedious and time-consuming, especially when searching for accessibility reviews. The goal of this paper is to support the automated identification of accessibility in user reviews, to help technology professionals in prioritizing their handling, and thus, creating more inclusive apps. Particularly, we design a model that takes as input accessibility user reviews, learns their keyword-based features, in order to make a binary decision, for a given review, on whether it is about accessibility or not. The model is evaluated using a total of 5,326 mobile app reviews. The findings show that (1) our model can accurately identify accessibility reviews, outperforming two baselines, namely keyword-based detector and a random classifier; (2) our model achieves an accuracy of 85% with relatively small training dataset; however, the accuracy improves as we increase the size of the training dataset.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in accessibility**; **Ubiquitous and mobile devices**.

KEYWORDS

Mobile application, user review, accessibility, machine learning.

ACM Reference Format:
Eman Abdullah AlOmar, Wajdi Aljedaani, Murtaza Tamjeed, Mohamed Wiem Mkaouer, and Yasmine N. El-Glaly. 2021. Finding the Needle in a Haystack: On the Automatic Identification of Accessibility User Reviews. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3411764.3445281

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI '21, May 8–13, 2021, Yokohama, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445281

1 INTRODUCTION

Many mobile applications (apps) have poor accessibility which makes it difficult for people with disabilities to use such apps [5, 53, 55, 71]. Researchers presented several methods, tools, frameworks, and guidelines to support developers in creating accessible mobile applications [9, 11, 19, 47, 54, 64]. However, many software developers and designers still do not incorporate accessibility into their software development process due to lack of awareness or lack of resources, e.g., budget and time, [15, 48, 51]. In this paper, we present a method that can help software developers to quickly become aware of specific accessibility problems with their apps that the users encountered. Our method is based on automatically identifying app reviews that users write on app stores, e.g., App Store[1], Google Play[2] and Amazon Appstore[3], where these reviews express an accessibility-related feedback.

Analyzing app reviews was used by technology professionals to identify issues with their mobile apps [12, 37, 39]. However, accessibility in user reviews is rarely studied especially for mobile applications [18]. Identifying accessibility-related reviews is currently done using two main methods: manual identification and automatic detection [18]. The manual identification approach is time consuming especially with the vast number of reviews that users upload to the app stores, and so it becomes impractical. The automated detection method employs a string-matching technique as a predefined set of keywords are searched for in the app reviews [18]. These keywords were extracted from the British Broadcasting Corporation (BBC) recommendations for mobile accessibility [10]. While this method sounds more practical than the manual one, it has its own drawbacks: the string-matching technique ignores that keywords derived from guidelines do not necessarily match the words expressed in reviews posted by users. This mismatch includes but not limited to situations when the keywords are incorrectly spelled by users. More importantly, the presence of certain keywords in a review does not necessarily mean that the review is about accessibility. For example, consider the following reviews from Eler et al. dataset [18]:

[1]https://www.apple.com/ios/app-store/
[2]https://play.google.com/store
[3]https://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011

> *"This is the closest game to my old 2001 Kyocera 2235's inbuilt game 'Cavern crawler'. Everything is so simple and easy to comprehend but that doesn't mean that it is easy to complete right off of the bat. Going into the sewers almost literally blind (sight and knowledge of goods in inventory) is a great touch too. Keep at it. I'll support you at least in donations."*

This review contains a set of keywords that could indicate accessibility (e.g., "*old*", "*blind*" and "*sight*") but it is not an accessibility review. In this review, the word "*old*" refers to a device rather than a person. The words "*blind*" and "*sight*" refer to knowledge of goods in the game rather than describing a player's vision. Therefore, the discovery of accessibility reviews heavily relies on the *context*, and so, simply searching for their existence in the review text is inefficient. Due to the overhead of the manual identification, and the high false-positiveness of the automated detection, these two methods remain impractical for developers to use, and so, accessibility reviews remain hard to identify and to prioritize for correction. To address this challenge, it is critical to design a solution with *learning capabilities*, which can take a set of examples that are known to be accessibility reviews, and another set of examples that are not about accessibility but do contain accessibility-related keywords, and learn how to distinguish between them. Therefore, in this paper, ***we use supervised learning to formulate the identification of accessibility reviews as a binary classification problem.*** This model takes a set of accessibility reviews, obtained by manual inspection, in a previous study [18] as input, we deploy state-of-the-art, machine learning models to *learn* the *features*, i.e., textual patterns that are representative of accessibility reviews. In contrast to relying on words derived from guidelines, our solution extracts *features* (i.e., words and patterns) from actual user reviews and learns from them. This is critical because there is a *semantic gap* between the guidelines, formally written on an abstract level, and technology-specific keywords. By features, we refer to a keyword or a set of keywords extracted from accessibility-related reviews that are not only important for classification algorithms, but they can also be useful for developers to understand accessibility-related issues and features in their apps. The patterns can be about an app feature that supports accessibility (e.g., "*font customization*", "*page zooming*" or "*speed control*"); about assistive technology (e.g., "*word prediction*", "*text to speech*" or "*voice over*") as well as about disability comments (e.g., "*low vision*", "*handicapped*", "*deaf*" or "*blind*"). Particularly, we addressed the following three research questions in our study:

RQ1: *To what extent machine learning models can accurately distinguish accessibility reviews from non-accessibility reviews?*
To answer this research question, we rely on a manually curated dataset of 2,663 accessibility reviews, which we augment with another 2,663 non-accessibility reviews. Then we perform a comparative study between state-of-the-art binary classification models, to identify the best model that can properly detect accessibility reviews, from non-accessibility reviews.

RQ2: *How effective is our machine learning approach in identifying accessibility reviews?*
Opting for a complex solution, i.e., supervised learning, has its own challenges, as models need to be trained, parameter tuned, and maintained, etc. To justify our choice of such solution, we compare the best performing model, from the previous research question, with two baselines: the string-matching method, and the random classifier. This research question verifies whether a simpler solution can convey competitive results.

RQ3: *What is the size of the training dataset needed for the classification to effectively identify accessibility reviews?*
In this research question, we empirically extract the minimum number of training instances, i.e., accessibility reviews, needed for our best performing model, to achieve its best performance. Such information is useful for practitioners, to estimate the amount of manual work needs to be done (i.e., preparation of training data) to design this solution.

We performed our experiments using a dataset of 5,326 user reviews, provided by a previous study [18]. Our comparative study has shown that the *Boosted Decision Trees* model (BDTs-model) has the best performance among other 8 state-of-the-art models. Then, we compared our BDTs-model, against two baselines: (1) string-matching algorithm and (2) a random classifier. Our approach provided a significant improvement in the identification of accessibility reviews, outperforming the baseline-1 (keyword-based detector) by 1.574 times, and surpassing the baseline-2 (random classifier) by 39.434 times.

The contributions of this paper are:

(1) We present an action research contribution that privileges societal benefit through helping developers automatically detect accessibility-related reviews and filter out irrelevant reviews. We make our model and datasets publicly available [4] for researchers to replicate and extend, and for practitioners to use our web service and filter down their user reviews.
(2) We show that we need a relatively small dataset (i.e., 1500 reviews) for training to achieve 85% or higher F1-Measure, outperforming state-of-the-art string-matching methods. However, the F1-measure score improves as we add to the training dataset.

2 RELATED WORK

It is crucial that mobile applications be accessible to allow all individuals with different abilities to have fair access and equal opportunities [27]. Prior studies investigated the accessibility issues raised in Android applications [5, 66], and others evaluated the accessibility of various websites [1, 17, 30, 69]. To the best of our knowledge, there is no study classifies user reviews in Android applications using machine learning.

In this section, we highlight several previous works that profoundly influenced our approach. We split the related works into three sections: user review, which briefly highlights the role of user reviews in app evolution; accessibility in user review, focuses particularly on detection of accessibility in user reviews; and classification of text documents, where we focus on current approaches in the classification of text such as user reviews by different taxonomies.

[4] https://smilevo.github.io/access/

2.1 User Reviews

Many researchers concluded that reviews and ratings posted by users on app store platforms can play an essential role in apps' evolution since most developers consider users' reviews when working on a new release [12, 37, 45, 49]. Maalej et al. [39] proposed to consider user-input as first means of requirements elicitation in software development. Similarly, Vu et al. [67] emphasized on the role of users in software lifecycle by developing an approach to identify useful information from users' review. Moreover, Seyff et al. [59] suggested continuous requirements elicitation from end-users' feedback using mobile devices.

Considering the fact that user reviews can be a powerful driver to mobile app evolution, we are looking into whether we can effectively detect accessibility reviews from users' feedback. This is important because in a highly competitive market, identifying accessibility issues from users' reviews can help developers improve their apps in order to attract more customers and provide better services to users with different abilities.

2.2 Accessibility in User Reviews

Even though user reviews can be a robust tool to mobile apps evolution, and that even mature apps have many trivial accessibility issues [19, 71], only 1.24% of mobile app users report accessibility issues to app stores [18]. In other words, 98.76% of mobile app users do not post accessibility issues in the form of reviews on app stores. In an effort to find whether mobile app users post accessibility-related issues to app stores, Eler et al. [18] investigated 214,053 mobile app reviews using a string-matching approach. They depend on a set of 213 keywords derived from 54 BBC recommendations [10] proposed for mobile accessibility. In their work, they inspected 214,053 user reviews to identify reviews pertaining to accessibility. Their approach classified a total of 5,076 reviews as accessibility reviews. However, through a manual inspection later, the researchers found that only 2,663 of the reviews were really about accessibility. We used these 2,663 identified accessibility reviews as one of the two groups in our training set required for a supervised machine learning. We created the second group (i.e., non-accessibility reviews) from their total dataset (i.e., 214,053). So far, this is one of the preliminary studies related to the accessibility in mobile app user reviews.

2.3 Classification of Text Documents

Many studies classify app reviews using different taxonomies [12, 16, 28, 41, 46, 49], for various purposes: detection of potential feature requests, bug reports, complaints, and praises, etc. Even though many of them identify reviews related to app usability, there is no explicit mention to accessibility related issues [18].

Unlike automatic approaches, classification of text documents using a set of ***predefined keywords*** has been vastly performed across different domains in software engineering. For instance, Eler et al. [18] relied on 213 keywords to identify accessibility-related reviews. Strogylos and Spinelles [62] identified refactoring-related commits using one keyword "*refactor*". Similarly, Ratzinger et al. [52] used 13 keywords to detect refactoring in commit messages. Later, Murphy-Hill et al. [43] replicated Ratzinger's work in two open-source software using the 13 keywords Ratzinger used. However, they disproved the previous assumption that commit messages in version history of programs are indicators of refactoring activities. The reasoning behind their findings is that developers do not always report refactoring activities as they might associate refactoring activities with other activities such as adding a feature. AlOmar et al. [2] have also explored how developers document their refactoring activities in commit messages using a variety of 87 textual patterns (i.e., keywords and phrases). Similarly, we believe users can express accessibility concerns without explicitly using any accessibility keywords from the BBC guidelines as assumed by Eler et al. [18].

In contrast to the keyword-based approaches, we used an automated machine learning approach since learning approaches outperform the accuracy of the keyword-based approach by at least 1.45 times [3, 40]. On the other hand, a keyword-based identification approach (i.e., relying on an existing set of predefined keywords) could generally miss certain reviews, not only because reviews left by users might not always use those keywords to express an accessibility concern, but also because a single word might not be enough to convey an accessibility message. For example, the review "*I hope someday we change size of the fonts*"; here the context provides an accessibility concern even though the user is not explicitly using keywords such as "*disabled*", "*blind*" or "*low vision*".

3 ACCESSIBILITY APP REVIEW CLASSIFICATION

The main goal of this work is to automatically identify accessibility-related reviews in a large dataset of app reviews. Our approach takes a set of reviews as input and makes a binary decision on whether the review is accessibility pertaining or not, i.e., classifying app reviews (for simplicity we refer to them as *accessibility reviews* and *non-accessibility reviews*). To be able to do so, we built a classification model using a corpus of reviews and current classification techniques. We then used the classification model to predict types of new app reviews. Figure 1 provides an overview of the process used in the detection of accessibility reviews. Our approach follows five main steps:

(1) **Data Collection:** We used a dataset of app reviews along with their ground truth categories previously identified through manual inspection [18] as input for training purposes.
(2) **Data Preparation:** We applied data cleansing and text preprocessing on this set to improve the *reviews text* for the learning algorithms. Some of the text preprocessing procedures we used are namely, tokenizing, lemmatizing, removing stop words, and removing capitalization.
(3) **Feature Extraction:** We used Feature Hashing [68] to extract features (i.e., words) from the preprocessed review text to create a structured feature space.
(4) **Model Selection and Tuning:** We examined a total of nine classification algorithms to evaluate the performance of the model for prediction. These classifiers were chosen because they are commonly used for classification of text such as app reviews [28, 31]. After training and evaluating the model, we used a testing dataset to challenge the performance of the model. Since the model has already learned from the N-Gram

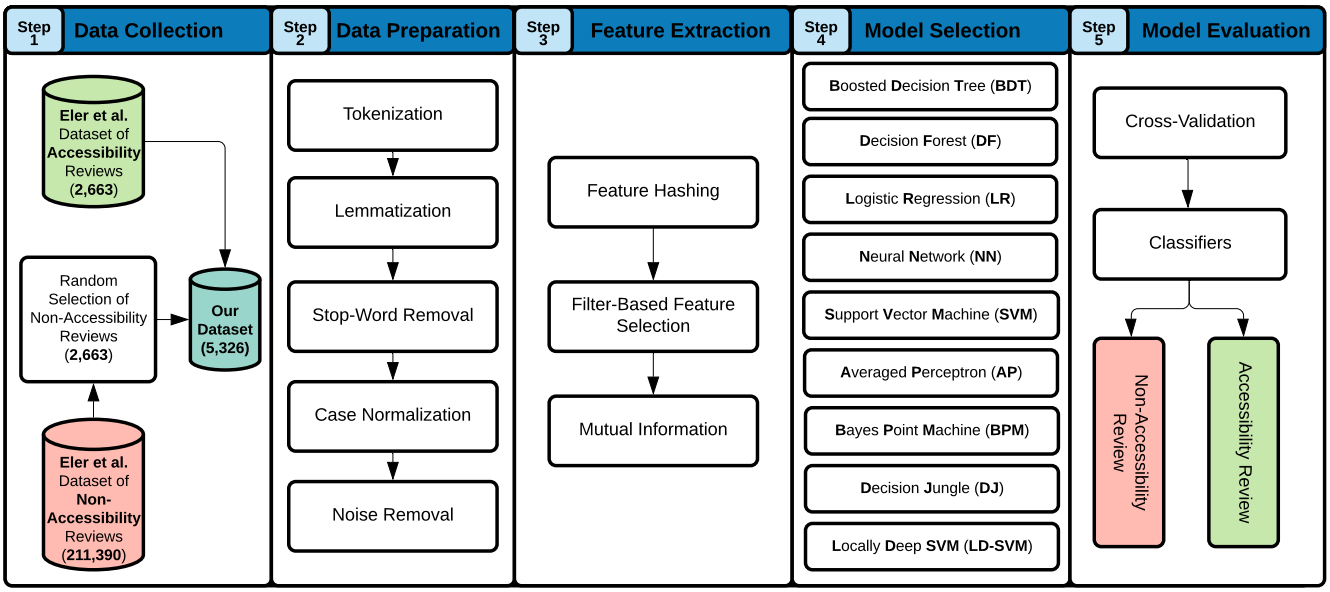

Figure 1: Accessibility app review classification process.

vocabulary and their weights discussed in Section 3.3 from the training dataset, the classifier output predicted-labels and probability-scores for the testing dataset. Since an app review is a plain text in our case, we follow the approach provided by Kowsari et al. [33] that discusses trending techniques and algorithms for text classification, similar to [3, 4].

(5) **Model Evaluation:** We built a training set using the extracted features for the model to learn from.

3.1 Data Collection

The dataset, used for this study, and shown in Table 1, is a collection of these 2,663 accessibility reviews, manually validated by Eler et al. [18]. The collected reviews are extracted from across 701 apps, belonging to 15 different categories, as shown in Figure 2. This dataset excluded all apps under the Theming and System categories, since they usually do not have any interface associated with them. Eler et al. [18] started with collecting 214,053 reviews, then they performed the string-matching using 213 keywords to filter down reviews and keep only those who potentially may contains information related to accessibility. These keywords are derived from 54 BBC recommendations proposed for mobile accessibility. The string-matching reduced the reviews from 214,053 to 5,076 candidate accessibility reviews. However, the manual inspection of these candidate reviews found that only 2,663 were true positives.

Table 1: Statistics of the dataset.

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

In order for us to verify the previous manual labeling of the reviews, we followed the process of Levin et al. [36] and randomly selected a 9% sample of reviews, i.e., 243 out of the 2,663 reviews. This quantity roughly equates to a sample size with a confidence level of 95% and a confidence interval of 6. Then we randomly added another 243 non-accessibility reviews, to end up with a total of 486 reviews. Afterward, one researcher labeled them. The selected data was not exposed to the researcher before. The review process was given a period of 7 days, to avoid fatigue, and the researcher had the opportunity to search online for any keywords they could not understand, during the labeling process. Once the data was labeled, we positioned our labeling against the original labeling of the reviews, from the dataset. We used Cohen's Kappa coefficient [13] to evaluate the inter-rater agreement level for the categorical classes. We achieved an agreement level of 0.82. According to Fleiss et al. [21], these agreement values are considered to have an almost *perfect agreement* (i.e., $0.81 - 1.00$).

To prepare training data for the binary classification of app reviews we created two groups of app reviews: (1) reviews indicating accessibility and (2) reviews not related to accessibility. For the accessibility reviews, we used the set of 2,663 reviews previously identified and validated as accessibility reviews through manual inspection by Eler et al. [18]. Since class starvation or an imbalanced training set (i.e., not having equal size of both groups) could decrease the performance of a classification model [35, 36], we need to select an equal number of non-accessibility reviews for the training. To efficiently train a classifier, it is important for the negative set to be as *close* as possible to the positive set. Therefore, we chose the negative set to be populated using the discarded reviews of the original authors, during their manual process. These discarded reviews tend to contain some keywords that are relevant to accessibility, but they were found to be conveying another meaning, and that is what we want our model to learn. Since the subset of discarded reviews was 2,413, we randomly selected reviews from the Eler et al. [18] remaining reviews dataset, so that these reviews

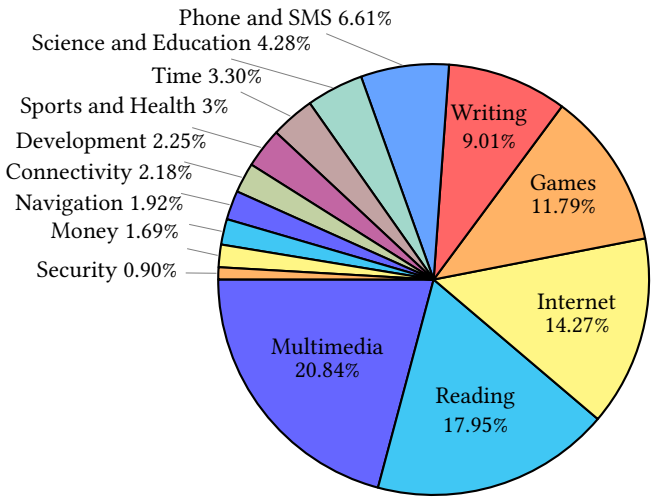

Figure 2: Distribution of accessibility reviews per app category.

are also extracted from the same apps, and most likely to contain some keywords that overlap with our true positive set.

To decide on the number of reviews necessary for training purposes, we reviewed the thresholds used in several text classification studies. The highest number of text documents used in comparable studies [3, 35, 36] was around 2000 text documents. Since our goal was to provide the model with sufficient reviews that could represent all possible accessibility topics, unlike existing works we chose a total of 5,326 reviews for the model creation and validation. However, we did evaluate our model with different sizes of training sets to understand the size of the training set that yields the best results. We report the results of our evaluations with regard to the testing of different training sizes in Section 4.

3.2 Data Preparation

Upon completion of the data collection phase, we applied a common approach explained in [33] for text preprocessing, similar to [3, 4]. For a model to classify text documents correctly, the text needs to be cleaned and preprocessed. To preprocess the app reviews text, we used natural language processing techniques, built-in the Microsoft Azure [7], such as tokenizing, lemmatizing, removing stop words, and removing capitalization.

Tokenization: is the process of splitting natural text data into tokens, or meaningful elements, that contain no white space. We tokenized app reviews by breaking them into their constituent set of words.

Lemmatization: is the process of getting the basic form of a word by either removing the suffix of a word or replacing the suffix of a word with a different one. It is also the process of reducing the number of unique occurrences of similar words. We used this preprocessing technique to represent words in their canonical form in order to reduce the number of unique occurrences of similar text tokens.

Stop-Word Removal: We removed words such as (*is, am, are, if, for, the*, etc.) that do not play any good role in classification.

Case Normalization: Since we wanted the same words with different font cases (e.g., "Accessibility" and "accessibility") to be treated as the same word, we converted original review texts to lower case. This type of text cleansing helps us avoid having repeated features differing only in the letter case. We realize that in some cases a user can identify themselves as 'Deaf' with uppercase 'D' to express their cultural identity in their review which is different from 'deaf'. However, as our classifier is a binary classifier that only distinguishes accessibility reviews from the rest, the words 'Deaf' and 'deaf' will yield the same classification result. Hence, case normalization in this context is safe and will not overrule users' expressions.

Noise Removal: We removed any noise that could deteriorate classification performance and confuse the model when learning. Examples of the noise we removed include removing special characters, numbers, symbols, email addresses and URLs.

3.3 Feature Extraction

After cleansing and preprocessing the reviews text, we extracted features from the preprocessed text that matter the most in distinguishing between the two classes in classification. Particularly, we used the ***Feature Hashing*** technique for feature extraction. Feature Hashing is a technique that operates on high-dimensional text documents used as input in a machine learning model, to map string values directly into encoded features and represent them as integers [60, 68]. This technique helps to reduce dimensionality and to make the feature weights lookup more efficient. Internally, the Feature Hashing technique creates a dictionary of N-Grams. We used bigrams in our classification since it greatly improves the performance of text classification [63]. Generally, N-Grams have more meaning and semantic than isolated words. For example, the word "*font*" does not provide enough information by itself. However, when N-Gram features extracted from reviews, e.g., "*small font*", "*font customization*", "*font size*", etc., the word "*font*" can indicate accessibility reviews. We discuss in details the features of our model (i.e., keywords and bigrams) in Section 4. We used Mutual Information filter-based feature selection. Mutual Information is a technique that measure how much a variable contributes towards reducing uncertainty about the value of another variable in order to identify features with the greatest predictive power. In fact, this feature set is the training set that the model learns from. In Figure 3, we illustrate how Feature Hashing applied to the text which was being transformed to a dictionary, as well as the process of the filter-based feature selection.

3.4 Model Selection and Tuning

Selecting an appropriate classifier for optimal classification is a challenging task by itself [20]. In this study, we are tackling a two-class classification problem as we are categorizing app reviews into two groups, accessibility and non-accessibility. Because we already have a predefined set of classes, our approach relies on supervised machine learning algorithms to assign each review into one of the two categories. We tested nine different classification algorithms as to see which one provides the best results in the context of accessibility and app reviews classification. The tested classifiers are: Logistic Regression (LR), Decision Forest (DF), Boosted Decision

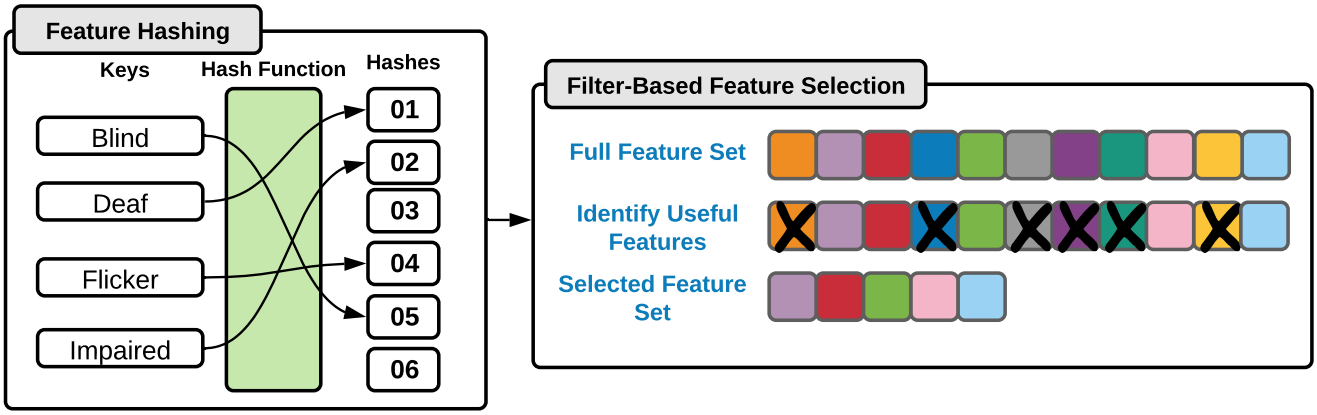

Figure 3: An example of feature hashing and feature selection process in feature extraction stage.

Tree (BDT), Neural Network (NN), Support Vector Machine (SVM), Averaged Perceptron (AP), Bayes Point Machine (BPM), Decision Jungle (DJ), and Locally Deep SVM (LD-SVM). We adopted these classifier algorithms because they are commonly utilized in the literature of software-related text classification [3, 23, 34, 38, 44, 72]. Below is a brief description of each of the classification algorithms used in this study.

- **Logistic Regression (LR)**[6] is a linear classifiers that predicts the probability of an outcome by fitting data to a logistic function.
- **Decision Forest (DF)**[50]: is a tree-based learner that builds many classification trees. A specific classification is associated with each tree produces. To classify a new object, DF chooses the classification that has the most votes over all other trees.
- **Boosting Decision Tree (BDT)**[22]: is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth. Predictions are based on the entire ensemble of trees together that makes the prediction.
- **Neural Network (NN)**[24]: is a set of interconnected layers. The inputs are the first layer that are connected to an output layer by an acyclic graph.
- **Support Vector Machine (SVM)** [70]: is a learner that constructs hyperplane(s) in n-dimensional space.
- **Averaged Perceptron (AP)**[14] is a simple version of Neural Network. The inputs are classified into several outputs based on a linear function, and then combined with a set of weights that are derived from the feature vector.
- **Bayes Point Machine (BPM)**[26]: is an algorithm that uses a Bayesian approach to linear classification called the "Bayes Point Machine". This algorithm approximates the optimal Bayesian average by choosing one "average" classifier, the Bayes Point.
- **Decision Jungle (DJ)**[61]: is a recent extension to decision forests. It consists of an ensemble of decision directed acyclic graphs (DAGs).
- **Locally Deep SVM (LD-SVM)**[29]: is a classifier that has been developed for an effencient non-linear SVM prediction.

We compared all the nine classifiers based on their common statistical measures such as precision, recall, accuracy, and F1-measure. These experiments were performed on the Azure ML platform because it provides a built-in web service once the classification model is deployed. We report the results of our classifier comparison and evaluation in Section 4.

We use grid search cross validation [56], a tuning method that performs exhaustive search over specified parameter values for an estimator, for tuning of our selected ML models. In order to facilitate the replication of our results, we provide the selected main parameters for ML techniques as shown in Table 2.

3.5 Model Evaluation

We assess the performance of our selected models based on the following four measurement aspects:

- **Precision** = $\frac{tp}{tp+fp}$: is a statistic that calculates the accurate number of correct predictions out of all the input sample.
- **Recall** = $\frac{tp}{tp+fn}$: is a statistic that calculates the accurate number of positive predictions that was actually observed in the actual class.
- **Accuracy** = $\frac{TP+TN}{TP+TN+FP+FN}$: is a statistic that calculates the accurate number of
- **F1-measure** = $\frac{2 \cdot P \cdot R}{P+R}$: is a a statistic that calculates the accuracy from the precision and recall.

Here TP denotes True Positive, TN denotes True Negative, FP denotes False Positive, and FN denotes False Negative. These metrics participation in measurement for a classifier's output.

- **True Positive (TP):** This parameter determines the predictions labeled correctly by the classifier as positive.
- **True Negative (TN):** This parameter determines the correct number of negative predictions.
- **False Positive (FP):** This parameter determines the number of instances (negatives) that were presumed as positive instances by the classifier by mistake.
- **False Negative (FN):** This parameter determines the number of positive instances that were falsely assumed to be as negative instances by the classifier.

Table 2: Summary of the hyperparameter in machine learning algorithm.

Classifier	Hyperparameter	Default	Description
LR	optimiz_tol	1E-07	Optimization tolerance
	1_weight	1	L1 regularization weight
	L2_weight	1	L2 regularization weight
	memory_L_BFGS	20	Memory size for L-BFGS
DF	n_estimators	8	Number of decision trees
	max_depth	32	Maximum depth of the decision trees
	n_samples_leaf	125	Number of random splits per node
	min_samples_split	1	Minimum number of samples per leaf node
BDT	max_n_leaf	20	Maximum number of leaves per tree
	min_samples_leaf	10	Minimum number of samples per leaf node
	learning_rate	0.2	Learning rate
	n_tree	100	Number of trees constructed
NN	n_nodes	100	Number of hidden nodes
	learning_rate	0.1	Learning rate
	n_learning_rate	100	Number of learning iterations
	learning_rate_weights	0.1	Initial learning weights diameter
	momentum	0	Momentum
SVM	n_iter	1	Number of iterations
	Lambda	0.001	Lambda
AP	learning_rate	1	Learning rate
	m_iter	10	Maximum number of iterations
BPM	n_training_iter	30	Number of training iterations
DJ	n_estimators	8	Number of decision directed acyclic graphs
	max_depth	32	Maximum depth of the decision directed acyclic graphs
	max_width	128	Maximum of the decision directed acyclic graphs
	n_optimiz	2048	Number of optimization steps per decision directed acyclic graphs layer
LD-SVM	max_depth	3	Depth of the tree
	lam_weight	0.1	Lambda weight
	n_theta	0.01	Lambda Theta
	n_theta_Prime	0.01	Lambda Theta Prime
	n_sigmoid	1	Sigmoid sharpness
	n_iter	15000	Number of iterations

Cross-Validation. We applied a 10-fold cross-validation technique to evaluate the variability and reliability of our models. For each model, we split our dataset into 10 folds containing the equal size of app reviews. Then, we performed 10 evaluations with various testing datasets wherein each evaluation 9 folds were used as a training dataset and the other fold was used as a testing dataset. Put differently, unlike other approach that is dependent on just one train-test split, when evaluating our model using 10-fold cross-validation, we train on multiple train-test splits in which one fold is left as a holdout data set, so it is unseen during the training. This approach is considered the preferred method as it gives us a better indication of how well our model performs on unseen data. We aggregated the results of the 10 evaluations and reported the average performance tested with multiple models.

4 EXPERIMENTAL RESULTS AND EVALUATION

In this section, we review the results of our experiments to evaluate the performance of our approach. For evaluating various accessibility classification models, we used standard statistical measures (*Precision*, *Recall*, *Accuracy*, *F1-measure*). Using the evaluation results, we provide answers to our research questions.

RQ1. To what extent machine learning models can accurately distinguish accessibility reviews from non-accessibility reviews?

We conducted an experiment to determine if the automatic classification of user reviews using machine learning techniques can be performed with high accuracy. We wanted to understand the opportunities and limitations of the machine learning technique in automatically detecting accessibility reviews.

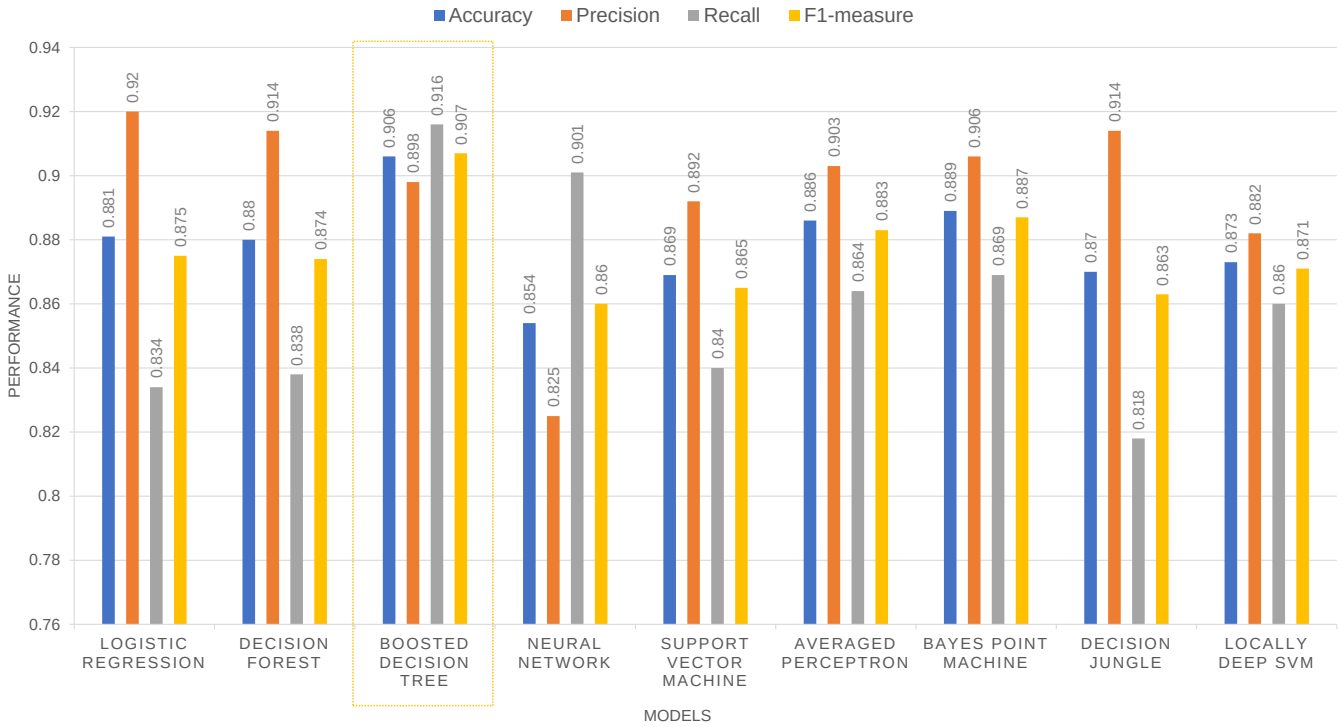

Figure 4: Comparison between binary classifiers, in terms of precision, recall, accuracy, and F1-measure.

We compared the nine classification algorithms tested in this study with respect to precision, recall, accuracy and F1-measure and reported the results as shown in Figure 4. The accuracy and F1-measure of the Boosted Decision Trees model (BDTs-model) is clearly higher than its competitors for the classification of accessibility reviews. The BDTs-model with the accuracy of 90.6% and F1-measure of 90.7%, outperformed other classification algorithms. Figure 4 also shows that the Bayes Point Machine (BPM) and Averaged Perceptron (AP) with F1-measure of 88.7% and 88.3% respectively, yielded higher predictive power after the Boosted Decision Trees.

The fact that BDTs-model achieved top performance rate can be explained by the fact that a boosted decision tree aggregates several learnings since it is an ensemble learning method. In the ensemble method, the errors of the first tree are fixed by the second tree, and the errors of the second tree are fixed by the third, and so on. In this method, the entire ensemble trees together form the prediction.

To further understand how these models distilled the text of the reviews into features, we extract keywords that were trending in our dataset, that we enumerate in Table 3. It is important to note that the majority of these keywords were identified by the BBC recommendations for mobile accessibility, however, not all of these keywords were found to be useful for our best performing classifier, i.e., BDTs-model. In Table 3, we report in bold, the features that were influential in increasing the accuracy of the trained Boosted Decision Trees. Such finding does not necessarily deny the relevance of the remaining keywords in describing accessibility related issues, but the fact that they were not selected, indicates their existence in non-accessibility related reviews. Keywords such as "*dark mode*" or "*mute*", while being used in the BBC guidelines, are also known to be used in general usability contexts. For example, the keyword "*mute*" tends to be frequently used in reviews related to media and video players, where sound is one of the main features of the app.

Further, on a more qualitative sense, we examine the set of frequently occurring bigrams for the keywords (reported in Table 3) that are strongly correlated to the accessibility review . Bigram corresponds to a sequence of two adjacent words in a sentence to help better understanding the context for the given terms. By analyzing the natural language in the accessibility review, we obtain more specific accessibility review-related terminology. Table 4 presents the frequently occurring bigrams in the review. Looking at these terms, we see that developers are either commenting on the features of the apps (e.g., "easily accessibile", "good text reflow", "great for visually impaired"), or they are discussing accessibility issues with their products pointing out that the apps need to be improved (e.g., "terribly hard to see", "no visual cue", "cant read").

The findings, illustrated in Tables 3 and 4 indicate a potential variation of how users typically state their accessibility needs. While it seems intuitive, there are no studies that focused on extracting such information in a structured manner to facilitate the identification of such accessibility problems by the app maintainers.

Although a high classification performance of our BDTs-model has been demonstrated in Figure 4, there are some limitations that lead BDTs-model to output some misclassified reviews as illustrated

in Table 5. According to our thorough analysis, we notice that the misclassification of our model can be related to:

- False positive instances caused by the format of reporting user perspective of the apps. The examples in the table show that different expression about the apps like "simple" or "headache" can be confusing to the classifier and hence it misclassified these reviews.
- False negative instances caused by the format of reporting a specific feature of the apps. As shown in the table, the users commented on a specific feature such as "functioning reader" and "caller ID". The BDTs-model will wrongly classify it because these could be seen as an accessibility-related features.

It is worth noting that the above misclassifications do not have a large influence on the overall performance of the BDTs model. Only a small number of reviews are wrongly classified by our model.

Summary. The Boosted Decision Trees model, with an accuracy of 90.6% and an F1-measure of 90.7%, is the best performing model in the binary classification of accessibility reviews.

RQ2. How effective is our machine learning approach in identifying accessibility reviews?

The main goal of this study is to propose an automatic approach for identification of accessibility reviews that can effectively outperform current state-of-the-art baselines: Keyword-based (i.e., also called pattern-based or string-matching) [18] and Random classifier [40]. Existing studies that have applied machine learning techniques in similar contexts (i.e., text classification) usually evaluate their approach using different classifiers. To compare their approach against others, they consider the keyword-based approach. To our knowledge, the only study that considers additional approach (i.e., random classifier) is the study by Da Silva et al. [40]. Thus, we consider keyword-based and random classifier to compare against our approach. Answering this question is important to understand if the detection of accessibility reviews is a learning problem. We hypothesize that learning algorithms can outperform string-matching algorithms. To examine if the hypothesis holds true, we chose to investigate the following two baselines, and compared them with our BDTs-model.

Baseline 1. Keyword-based Approach. The keyword-based (string-matching) approach for identifying accessibility reviews is suggested by Eler et al. [18]. In their work, they inspected 214,053 user reviews to identify the reviews pertaining to accessibility. Their string-matching approach classified a total of 5,076 reviews as accessibility reviews. However, manual verification of the 5,076 reviews later found that only 2,663 of the reviews were correctly identified [18].

To calculate statistical metrics for baseline 1, we used a set of 5,326 reviews (cf., set of 2,663 accessibility reviews, from Table 1, and another 2,663 non-accessibility reviews, selected from the same apps). Then, we manually inspected these reviews to determine true positives (*TP*), true negatives (*TN*), false positives (*FP*), and false negatives (*FN*). True positives are when the keyword-based approach correctly detected accessibility reviews, and true negatives are when non-accessibility reviews are correctly identified. False positives are the reviews identified as accessibility reviews while they are not; and false negatives are the reviews identified as non-accessibility reviews while they are accessibility reviews. Since we already had the reviews labelled, we were able to count *TP*, *TN*, *FP* and *FN*.

Baseline 2. Random Classifier. Similar to Da Maldonado et al. [40], we consider Random classifier as one of the baselines to compare our approach to. The precision of the random classifier technique is calculated by dividing the number of accessibility reviews by the total number of user reviews (i.e., $\frac{2663}{214053} = 0.012$). When it comes to recall, there is only 50% probability for a review to be classified as an accessibility review since there are two possible classifications available. Finally, the F1-measure of baseline 2 is calculated as $2 * \frac{0.012*0.5}{0.012+0.5} = 0.023$.

Using the values of *TP*, *TN*, *FP* and *FN*, we calculated the Precision, Recall, and F1-measure, for both baselines. Table 6 shows the standard statistical measures of the three approaches, also the performance improvements achieved by our BDTs-model compared to the other two methods.

As can be seen from Table 6, F1-measure obtained by the machine learning approach is much higher than the other methods. F1-measure achieved by the machine learning approach is 0.90, while F1-measure values using keywords and random classifier are 0.576 and 0.023 respectively. Table 6 shows that our approach outperforms the keyword-based approach by 1.574 times and the random classifier by 39.434 times when identifying accessibility reviews. To better understand the performance of the string-matching method, we have extracted examples reviews that were wrongly classified, as accessibility:

> *Review 1. "Good to have your files easily accessible. Would like integration of caldav/ carddav"*
>
> *Review 2. "Very useful application. Gmail users must go for it blind eyes"*

The existence of keywords such as "*accessible*" and "*blind eyes*", are string-matched to the keywords considered as accessibility by the guidelines, and so, the keyword-based approach will flag their corresponding reviews as accessibility. However, the first review (i.e., Review 1) refers to the new feature that allows user files to be accessible more efficiently and requests the integration of a protocol for the synchronization of calendars. Similarly, the second review (i.e., Review 2), is praising an app that synchronizes Gmail calendar with Outlook calendar, and the user's expression of "going with blind eyes", refers to their satisfaction, and not to what would be considered by the string-matching method as an accessibility issue.

To determine the different cases of when the keyword-based approach fails, we evaluated 592 reviews, a statistically significant sample with a confidence level of 99% and a confidence interval of 5%. By analyzing the selected reviews, we identified the following reasons behind the failure cases of the string-matching approach:

- **Keyword Misspelling.** This category depicts the case when accessibility aspects of the mobile application are addressed by the users using misspelled keywords. This case can be illustrated in the following example: *" Font size of lowercase letters is sooosmall! How to change it? It should be like*

Table 3: List of keywords trending in the 5326 reviews. Keywords in bold are found to be strongly correlated to accessibility reviews by our model.

Keywords				
(1) dark mode	(16) adjustable	(31) **voice command**	(46) colour coding	(61) captcha
(2) zoom	(17) **blind**	(32) **text-to-speech**	(47) **transcript**	(62) **audio description**
(3) customization	(18) **header**	(33) eyestrain	(48) default language	(63) container
(4) font size	(19) overlap	(34) strain	(49) older device	(64) distinguishable
(5) volume	(20) pause button	(35) background image	(50) **visual cue**	(65) input type
(6) **cannot see**	(21) **flicker**	(36) **screen reader**	(51) grouped	(66) keyboard language
(7) **accessibility**	(22) spacing	(37) change language	(52) seizures	(67) page refresh
(8) **readable**	(23) migraine	(38) small widget	(53) select language	(68) page title
(9) change font	(24) input method	(39) stop button	(54) understandable	(69) sign language
(10) **hard to see**	(25) autoplay	(40) **impaired**	(55) vibration feedback	(70) svg image
(11) background color	(26) metadata	(41) **text reflow**	(56) actionable	(71) switch device
(12) light mode	(27) too bright	(42) timeout	(57) audio cue	(72) touch target
(13) mute	(28) haptic	(43) consistency	(58) missing label	(73) adjust size
(14) contrast	(29) scaling	(44) epilepsy	(59) **navigable**	(74) adjust colour
(15) subtitle	(30) control key	(45) assistance	(60) verbose	

Table 4: A sample of frequently occuring bigrams for the keywords that are strongly correlated to accessibilty review by our model.

Bigram			
cannot see	**accessibility**	**readable**	**hard to see**
cannot see anything	easily accessible	readable text	very hard to see
cannot see worksheet	more accessible	document reader	too hard to see
cannot see number	great accessibility	easier reading	really hard to see
cannot see status	accessibility suite	can read	terribly hard to see
still cannot see	accessibility screen	cant read	hard to see theme
blind	**header**	**flicker**	**voice command**
blind user	theme header	screen flicker	voice command search
color blind	custom header	flicker taskbar	use voice command
supports blind	size header	flicker background	voice commands works
impaired / blind	adjust header	heavy flickering	simple voice command
totally blind	transparent header	constant flickering	custom voice command
text-to-speech	**screen reader**	**impaired**	**text reflow**
verbose text-to-speech	screen reader accessibility	visually impaired	text reflow feature
text-to-speech works	accessibility screen reader	vision impairment	activate text reflow
text-to-speech feature	talkback screenreader	visual impairment	good text reflow
text-to-speech news	small-screen reader	great for visually impaired	has text reflow
transcript	**visual cue**	**navigable**	**audio description**
transcript title	no visual cue	navigable bar	turns on audio description
recording / transcription	some visual cue	navigable button	
zooming and transcript	provide a visual cue	navigable app	
transcription not found		easily navigation	

on google keybord when you change capital/lowercase mode - lowercase letters have almost the same size as capital. It's much easier for your eyes!". The keyword matching approach can miss any word with a typo or with inproper spacing, such as "*keybord*" or "*sooosmall*". Misspellings are frequent in app user reviews, since mobile writing is known to be more prone to typos.

- **Keyword Variation.** This category shows the case in which users use different part-of-speech (POS) of the accessibility-related keywords reported in Table 3. As shown in the following review: *"very accessible as a blind user thank you"*, the user used the adjective form ("accessible") of the word accessibility.
- **Expression Variation.** This category represents cases in which users use different expressions of the keywords listed

Table 5: Examples of the misclassification case of our BDTs-model.

Type	Example
False Positive	*"Simple and easy to use"* *"This app works well - especially "lucid dream" - i still remember my dream last week. Amazing! But i dont like the side effects - like headache and other emotional thing."*
False Negative	*"Beautiful Functioning Reader"* *"Thank you for all your hard work in making this app for us to use. And to offer it to us for free is amazing. I use this app everyday, I got all my friends and family using it too. Thank you so much! I can only think of one thing that could make this app better, if you could add caller ID with name, and make it so users could turn it on or off, this would be great. Even without that, this app is great."*

in Table 3 to address accessibility aspects of the apps. This case is best illustrated in the following accessibility review: *"still getting responses from the wrong people and noticed that when in night mode with pure black background - when you try to delete a message the yes option is completely black so impossible to see"*. As can be seen, the expression "impossible to see" is used instead of the keyword "cannot see" to represent the user perspective on the problem.

Summary. The Boosted Decision Trees model outperforms the current state-of-the-art approaches in the classification of accessibility reviews. We obtained an F1-measure score of 90.7% with an improvement of 1.574x and 39.434x over the keyword-based and random classifier approaches respectively.

RQ3: What is the size of the training dataset needed for the classification to effectively identify accessibility reviews?

So far, we showed that our machine learning approach can accurately identify user reviews that pertain to accessibility. However, the performance of a classifier relies on the size of the training data. At the same time, creating a training dataset is a challenging and time-consuming task. Thus, the question is: What is the size of the training dataset needed to effectively classify user reviews? If an approach requires a very large training dataset than it will require a considerable time and effort to be applied to other similar contexts. However, if less training dataset is required to effectively classify accessibility reviews, then our approach can be applied and extended with little efforts.

To answer this research question, we incrementally added reviews to the training dataset and evaluated the performance of the classification. We began by creating a large training dataset that contains equal size of accessibility reviews and non-accessibility reviews. Then, we used cross validation technique, which is a technique that partitions the original dataset into a training set to train the model, and a test set to evaluate it using number of folds [32]. In this study, we divided the dataset into 10 folds making sure they contain equal size of both classes. Next, we tested our approach using a 10-fold cross-validation technique using 9 folds for training and 1 fold for testing. Since we wanted to monitor the performance of our classifier as the training dataset size increased, we incrementally added batches of 100 reviews until we used all of our training data (e.g., 5,326 reviews). It is important to note that we considered the equal size of accessibility reviews and non-accessibility reviews with batches incrementally added to the training dataset. We computed the F1-measure value for each iteration (e.g., after adding batches of new reviews to the training set). We recorded the number of reviews needed to achieve at least an F1-measure of 80% to 90%.

Figure 5 shows F1-measures calculated when detecting accessibility reviews, while incrementally adding batches of reviews to the training dataset. Our results show that the highest F1-measure (i.e., 0.907) was achieved with 5,326 reviews (our total training dataset) and the lowest F1-measure value (i.e., 0.630) was achieved with 100 reviews. Our results also show that 80 to 90 percent F1-measure is achieved with 400 to 5000 reviews in the training dataset. Such that, we need only 400 reviews to get around 80% F1-measure and we need at least 1500 reviews to get 85% or higher, while with 5000 reviews we got around 90% F1-measure. Finally, we found that the F1-measure score improves as we add to the training dataset.

Summary. We find that we need a relatively smaller training dataset (i.e., 1500 reviews) to get 85% or higher F1-measure. The F1-measure score improves as we add to the training dataset.

5 DISCUSSION

We presented a new approach that identifies app reviews with accessibility concerns. We compared our new approach to the current state-of-the-art methods. Based on these findings we discuss implications that can be theory-based and practice-based. Theory-based implications show how this study can further advance the research on accessibility reviews. Practice-based implications show how our model supports our community in building and maintaining accessible mobile apps.

Implication 1: App reviews are rich source of information that can be mined to identify specific accessibility problems with the mobile app. There are so many accessibility guidelines that developers and designers can find it difficult to test for all of these guidelines. Additionally, adhering to these guidelines does not necessarily guarantee the accessibility of the said app. Also, usability testing with different groups of people with disabilities, e.g., blind or deaf, can be infeasible especially for medium and

Table 6: Comparison in approaches used to the baselines in our study.

	Our approach			Keyword-based			Random classifier		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Classification	0.898	0.916	0.907	0.996	0.405	0.576	0.012	0.500	0.023
Improvement	–	–	–	0.901 x	2.261 x	1.574 x	74.833 x	1.832 x	39.434 x

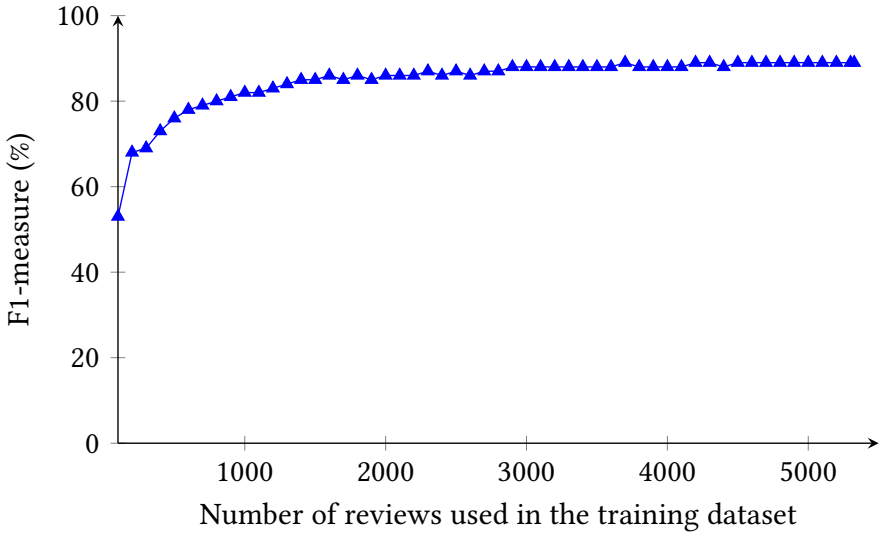

Figure 5: F1-measure achieved by incrementally adding training data size for binary classification.

small-scale companies. One way to discover accessibility problems which prior testing did not reveal is to listen to the users and learn from the reviews they wrote. Our approach can aid technology professionals to quickly spot accessibility problems with their app.

Implication 2: Accessibility as part of mobile apps maintenance and evolution. There exist accessibility testing tools and methods that are designed to support the implementation and testing phases of the software. However, there are no tools, to the best of our knowledge, that supports software accessibility in the maintenance phase. With changes made to an app, either for adding a feature or fixing a bug, accessibility can be at risk. Also, with updates made to the phone's operating system or the installed assistive technology, the accessibility of an app may deteriorate. We call for innovative methods that can support technology professionals in maintaining the accessibility of their app after its release. Our approach in analyzing app reviews offers an opportunity for developers and designers in detecting accessibility pitfalls based on their users' written feedback. However, with the tremendous number of reviews developers receive on a daily basis, it becomes impractical to manually read through them and identify potential issues related to their new release. Adding our model to the pipeline, will alleviate the manual overhead of looking up accessibility related reviews, and so developers can quickly locate their corresponding issues, and add them to their maintenance pipeline.

Implication 3: Understanding users' language in expressing their accessibility concerns. When we compared our BDTs-model to the keyword-based detector, we found that some accessibility reviews did not contain the accessibility keywords that were driven from accessibility guidelines [18]. This indicates that users voice their accessibility feedback using "user taxonomy" which may or may not echo the technical and professional terms used in accessibility standards. Further research is needed to understand how users describe mobile accessibility issues. By learning the accessibility "user taxonomy", we can improve our BDTs-model, which will lead to enhanced discovery of accessibility reviews.

Implication 4: The interplay between developers and designers, accessibility experts, and users. Accessibility experts establish guidelines and design methods in support of creating accessible software. Technology professionals often are not able to digest all these guidelines and often find existing resources lacking. This situation yielded to the existence of software products that are inaccessible to people with disabilities. The effective involvement of people with disabilities in this process can help bridging the communication gap between accessibility experts and developers and designers. By giving users the opportunity to lead the prioritization of accessibility issues based on their usage experience, mobile apps accessibility can be improved in a more meaningful way for people with disabilities. Analyzing app reviews is one way to give users the lead in determining which accessibility issue should be fixed in the next release. Analyzing app reviews can also offer insights to accessibility experts on users' accessibility needs right from the field, which will be more realistic than results collected from controlled lab studies.

Implication 5: Direct and immediate apps filtering benefit for end users. People find online reviews helpful in making purchase decisions [8]. Peer comments help users become aware

of the limitations of reviewed products [42]. Currently, on mobile applications stores, e.g., App Store and Google Play, users can read all reviews, sort them by most helpful or most recent. However, mobile application stores provide no means to filtering reviews based on relevance to specific quality metrics, e.g., accessibility. This lack of filtering pushes users to download the app first and then experience its accessibility, leaving no room for benefiting from peer comments. Sometimes, apps suffer from accessibility regression giving users an unpleasant surprise with an updated app that is less accessible than its former version [65]. We call on mobile application stores to take action and allow users to filter reviews based on relevance to accessibility.

Implication 6: Pushing the boundaries of Accessibility testing. Current accessibility testing strategies are human intensive, and therefore become expensive and impractical, as most developers struggle to find the appropriate testers who can evaluate the compliance of their apps to accessibility guidelines. Existing accessibility scanners are tailored for the web, and they cannot be applied to the mobile environment. In this context, online user reviews, offer a rich source of scenarios, which can be coupled with the app's current version, to create test cases of practically captured anomalies. Relying on this set of reviews, as a shared knowledge, developers can quickly identify potential test cases that they need to perform, in case they are incorporating a given accessibility tool in their app. Furthermore, as the mobile environment is extremely dynamic, recent user reviews can quickly reveal any appearing anomalies in the newer app releases.

6 THREATS TO VALIDITY

In this section, we identify several threats to the validity of our study. We group the threats to Construct Threats and External Threats to validity.

Construct Threats relate to the appropriateness of our dataset and accuracy of the previous work [18]. A potential threat is related to creating a training dataset or the manual classification. Developing a training dataset is typically a tedious job, also subject to reader bias. We mitigated this risk by choosing a dataset of accessibility reviews as our training data that were previously identified and validated [18]. Additionally, we used all of the identified reviews as training input rather than choosing a sample set of reviews. A total of 2,663 reviews were previously identified as accessibility reviews from 214,053 app reviews through manual inspections and validations.

Another potential threat relates to the keywords used for the identification of accessibility reviews through a string-matching approach. The string-matching approach relied on 213 keywords derived from 54 accessibility recommendations by BBC. The keywords and phrases users use in their reviews do not necessarily match the keywords available in the guidelines and recommendations. This mismatch includes but not limited to situations when keywords would be spelled incorrectly by reviewers. A related concern is whether the set of keywords is inclusive of all possible keywords that users use to express their accessibility concerns. To mitigate this threat, we used keywords defined by [18] in which the authors adopted variants for these keywords to ensure they would not miss any relevant review during their manual validation. This raised our confidence to use the dataset that has these keywords as a representative sample of accessibility reviews.

External Threats relate to the generalizability of our findings for this evaluation. We evaluated and tested our findings on a dataset collected by previous researchers [18]. The dataset was collected only from Android open-source applications. Therefore, the dataset did not represent the entire mobile apps on the App stores such as Apple store applications. Also, we only study mobile application reviews of open-source applications. Our results may not generalize to commercially developed projects or to other reviews that are written in other languages than English.

7 CONCLUSION

This study presents an approach that automates the classification of app reviews as accessibility-related or not so developers can easily detect accessibility issues with their products and improve them to more accessible and inclusive apps utilizing the users' input. As Hayes pointed out:"In Action Research, the goal is ultimately to create sustainable change. That is to say, once the research facilitators leave, the community partners should be able to maintain the positive changes that have been made." [25]. Our goal is to create a sustainable change, by including a model in developer's software maintenance pipeline, and raising awareness of existing errors that hinders the accessibility of mobile apps, which is a pressing need [48].

As we develop our model, we conducted an evaluation of nine different classifiers using an existing dataset of manually validated accessibility reviews. Our evaluation shows that the Boosted Decision Tree classifier offers higher accuracy than the other approaches in the classification of app reviews. Additionally, we compared our approach with two baselines, namely a keyword-based approach, and a random classifier. The results indicate that our approach outperforms the two state-of-the-art approaches with the F1-measure of 90.7%. Finally, we conduct an experiment to evaluate the impact of training data sizes on our classifier's accuracy. Our evaluation shows that we need a relatively smaller dataset (i.e., 1500 reviews) for training to get 85% or higher F1-measure. However, the F1-measure score improves as we add to the training dataset.

As our results show, having an adequately large training size is important for high accuracy in prediction. Given the millions of app reviews available on the app store platforms, the training process can be cumbersome and laborious. Additionally, it is necessary to obtain labels from multiple Subject Mater Experts (SMEs) to make the training dataset more reliable. In order to further reduce the efforts needed by developers and SMEs in creating a training data, we are planning to explore *Active Learning* [57, 58], a well-known machine learning paradigm for classification. We also plan to perform a multi-class classification on the accessibility reviews — dividing them into categories such as readability of text, audio, video, UI, gestures etc.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation, USA, under Grant No. 1757680.

REFERENCES

[1] Gaurav Agrawal, Devendra Kumar, Mayank Singh, and Diksha Dani. 2019. Evaluating accessibility and usability of Airline websites. In *International Conference on Advances in Computing and Data Sciences*. Springer, 392–402.

[2] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni. 2019. Can Refactoring Be Self-Affirmed? An Exploratory Study on How Developers Document Their Refactoring Activities in Commit Messages. In *2019 IEEE/ACM 3rd International Workshop on Refactoring (IWoR)*. 51–58. https://doi.org/10.1109/IWoR.2019.00017

[3] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni. 2020. Toward the automatic classification of self-affirmed refactoring. *Journal of Systems and Software* 171 (2020), 110821.

[4] Eman Abdullah AlOmar, Anthony Peruma, Mohamed Wiem Mkaouer, Christian Newman, Ali Ouni, and Marouane Kessentini. 2020. How we refactor and how we document it? On the use of supervised machine learning algorithms to classify refactoring documentation. *Expert Systems with Applications* (2020), 114176.

[5] Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. 2020. Accessibility Issues in Android Apps: State of Affairs, Sentiments, and Ways Forward. In *42nd International Conference on Software Engineering (ICSE 2020)*.

[6] Galen Andrew and Jianfeng Gao. 2007. Scalable Training of L1-Regularized Log-Linear Models. In *International Conference on Machine Learning* (international conference on machine learning ed.).

[7] Microsoft Azure. 2020. Azure Machine Learning. https://azure.microsoft.com/en-us/services/machine-learning/ Library Catalog: azure.microsoft.com.

[8] Hyunmi Baek, JoongHo Ahn, and Youngseok Choi. 2012. Helpfulness of online consumer reviews: Readers' objectives and review cues. *International Journal of Electronic Commerce* 17, 2 (2012), 99–126.

[9] Mars Ballantyne, Archit Jha, Anna Jacobsen, J Scott Hawker, and Yasmine N El-Glaly. 2018. Study of accessibility guidelines of mobile applications. In *Proceedings of the 17th international conference on mobile and ubiquitous multimedia*. 305–315.

[10] BBC. 2017. The BBC Standards and Guidelines for Mobile Accessibility. https://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile

[11] Sarah Chiti and Barbara Leporini. 2012. Accessibility of android-based mobile devices: a prototype to investigate interaction with blind users. In *International Conference on Computers for Handicapped Persons*. Springer, 607–614.

[12] Adelina Ciurumelea, Andreas Schaufelbuhl, Sebastiano Panichella, and Harald C. Gall. 2017. Analyzing reviews and code of mobile apps for better release planning. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, Klagenfurt, Austria, 91–102. https://doi.org/10.1109/SANER.2017.7884612

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

[14] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 1–8.

[15] Michael Crabb, Michael Heron, Rhianne Jones, Mike Armstrong, Hayley Reid, and Amy Wilson. 2019. Developing Accessible Services: Understanding Current Knowledge and Areas for Future Support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300446

[16] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Corrado A. Visaggio, and Gerardo Canfora. 2017. SURF: Summarizer of User Reviews Feedback. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, Buenos Aires, 55–58. https://doi.org/10.1109/ICSE-C.2017.5

[17] Trinidad Domínguez Vila, Elisa Alén González, and Simon Darcy. 2018. Website accessibility in the tourism industry: an analysis of official national tourism organization websites around the world. *Disability and rehabilitation* 40, 24 (2018), 2895–2906.

[18] 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*. ACM, Vitória Espírito Santo Brazil, 1–11. https://doi.org/10.1145/3357155.3358477

[19] Marcelo Medeiros Eler, Jose Miguel Rojas, Yan Ge, and Gordon Fraser. 2018. Automated Accessibility Testing of Mobile Apps. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, Vasteras, 116–126. https://doi.org/10.1109/ICST.2018.00021

[20] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research* 15, 1 (2014), 3133–3181.

[21] Joseph L Fleiss, Bruce Levin, Myunghee Cho Paik, et al. 1981. The measurement of interrater agreement. *Statistical methods for rates and proportions* 2, 212-236 (1981), 22–23.

[22] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[23] Katerina Goseva-Popstojanova and Jacob Tyo. 2018. Identification of security related bug reports via text mining using supervised and unsupervised classification. In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 344–355.

[24] Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 10 (1990), 993–1001.

[25] Gillian R Hayes. 2011. The relationship of action research to human-computer interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)* 18, 3 (2011), 1–20.

[26] Ralf Herbrich, Thore Graepel, and Colin Campbell. 2001. Bayes point machines. *Journal of Machine Learning Research* 1, Aug (2001), 245–279.

[27] Michael Heron, Vicki L Hanson, and Ian Ricketts. 2013. Open source and accessibility: advantages and limitations. *Journal of interaction Science* 1, 1 (2013), 1–10.

[28] Claudia Iacob and Rachel Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, San Francisco, CA, USA, 41–44. https://doi.org/10.1109/MSR.2013.6624001

[29] Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. 2013. Local deep kernel learning for efficient non-linear svm prediction. In *International conference on machine learning*. 486–494.

[30] Royce Kimmons. 2017. Open to all? Nationwide evaluation of high-priority web accessibility considerations among higher education websites. *Journal of Computing in Higher Education* 29, 3 (2017), 434–450.

[31] Eric Knauss, Daniela Damian, German Poo-Caamano, and Jane Cleland-Huang. 2012. Detecting and classifying patterns of requirements clarifications. In *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, Chicago, IL, USA, 251–260. https://doi.org/10.1109/RE.2012.6345811

[32] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Montreal, Canada, 1137–1145.

[33] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown. 2019. Text Classification Algorithms: A Survey. *Information* 10, 4 (April 2019), 150. https://doi.org/10.3390/info10040150

[34] Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch. 2008. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering* 34, 4 (2008), 485–496.

[35] Stanislav Levin and Amiram Yehudai. 2017. Boosting Automatic Commit Classification Into Maintenance Activities By Utilizing Source Code Changes. In *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE*. ACM Press, Toronto, Canada, 97–106. https://doi.org/10.1145/3127005.3127016

[36] Stanislav Levin and Amiram Yehudai. 2019. Towards Software Analytics: Modeling Maintenance Activities. (March 2019). http://arxiv.org/abs/1903.04909

[37] Xiaozhou Li, Zheying Zhang, and Kostas Stefanidis. 2018. Mobile App Evolution Analysis based on User Reviews. (2018), 14.

[38] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[39] Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. 2009. When users become collaborators: towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. 981–990.

[40] Everton da Silva Maldonado, Emad Shihab, and Nikolaos Tsantalis. 2017. Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt. *IEEE Transactions on Software Engineering* 43, 11 (Nov. 2017), 1044–1062. https://doi.org/10.1109/TSE.2017.2654244

[41] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering* 21, 3 (June 2016), 1067–1106. https://doi.org/10.1007/s10664-015-9375-7

[42] Susan M Mudambi and David Schuff. 2010. Research note: What makes a helpful online review? A study of customer reviews on Amazon. com. *MIS quarterly* (2010), 185–200.

[43] Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. 2012. How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering* 38, 1 (Jan. 2012), 5–18. https://doi.org/10.1109/TSE.2011.41

[44] Moein Owhadi-Kareshk, Sarah Nadi, and Julia Rubin. 2019. Predicting Merge Conflicts in Collaborative Software Development. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–11.

[45] Fabio Palomba, Mario Linares-Vasquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. 2015. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Bremen, Germany, 291–300. https://doi.org/10.1109/ICSM.2015.7332475

[46] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can i improve my app? Classifying user reviews for software maintenance and evolution. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE,

Bremen, Germany, 281–290. https://doi.org/10.1109/ICSM.2015.7332474

[47] Kyudong Park, Taedong Goh, and Hyo-Jeong So. 2014. Toward Accessible Mobile Application Design: Developing Mobile Application Accessibility Guidelines for People with Visual Impairment. In *Proceedings of HCI Korea* (Seoul, Republic of Korea) *(HCIK '15)*. Hanbit Media, Inc., Seoul, KOR, 31–38.

[48] Rohan Patel, Pedro Breton, Catherine M Baker, Yasmine N El-Glaly, and Kristen Shinohara. 2020. Why Software is Not Accessible: Technology Professionals' Perspectives and Challenges. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–9.

[49] Lucas Pelloni, Giovanni Grano, Adelina Ciurumelea, Sebastiano Panichella, Fabio Palomba, and Harald C. Gall. 2018. BECLoMA: Augmenting stack traces with user review information. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, Campobasso, 522–526. https://doi.org/10.1109/SANER.2018.8330252

[50] Anita Prinzie and Dirk Van den Poel. 2008. Random forests for multiclass classification: Random multinomial logit. *Expert systems with Applications* 34, 3 (2008), 1721–1732.

[51] Cynthia Putnam, Kathryn Wozniak, Mary Jo Zefeldt, Jinghui Cheng, Morgan Caputo, and Carl Duffield. 2012. How do professionals who create computing technologies consider accessibility?. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*. 87–94.

[52] Jacek Ratzinger, Thomas Sigmund, and Harald C Gall. 2008. On the relation of refactorings and software defect prediction. In *Proceedings of the 2008 international working conference on Mining software repositories*. 35–38.

[53] André Rodrigues, Hugo Nicolau, Kyle Montague, João Guerreiro, and Tiago Guerreiro. 2020. Open Challenges of Blind People Using Smartphones. *International Journal of Human–Computer Interaction* 36, 17 (2020), 1605–1622. https://doi.org/10.1080/10447318.2020.1768672

[54] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2017. Epidemiology as a Framework for Large-Scale Mobile Application Accessibility Assessment. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) *(ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/3132525.3132547

[55] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2018. 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* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 119–130. https://doi.org/10.1145/3234695.3236364

[56] Scikit-learn.org. 2006. Parameter Estimation Using Grid Search with Scikit-Learn. Available online:. https://scikit-learn.org/stable/modules/grid_search.html,. Accessed: 2020-04-01.

[57] Burr Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (June 2012), 1–114. https://doi.org/10.2200/S00429ED1V01Y201207AIM018 Publisher: Morgan & Claypool Publishers.

[58] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*. Association for Computational Linguistics, Honolulu, Hawaii, 1070. https://doi.org/10.3115/1613715.1613855

[59] Norbert Seyff, Florian Graf, and Neil Maiden. 2010. Using Mobile RE Tools to Give End-Users Their Own Voice. In *2010 18th IEEE International Requirements Engineering Conference*. IEEE, Sydney, Australia, 37–46. https://doi.org/10.1109/RE.2010.15

[60] Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. 2009. Hash kernels for structured data. *Journal of Machine Learning Research* 10, Nov (2009), 2615–2637.

[61] Jamie Shotton, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, and Antonio Criminisi. 2013. Decision Jungles: Compact and Rich Models for Classification. In *Proc. NIPS* (proc. nips ed.). https://www.microsoft.com/en-us/research/publication/decision-jungles-compact-and-rich-models-for-classification/

[62] Konstantinos Stroggylos and Diomidis Spinellis. 2007. Refactoring–Does It Improve Software Quality?. In *Fifth International Workshop on Software Quality (WoSQ'07: ICSE Workshops 2007)*. IEEE, Minneapolis, MN, USA, 10–10. https://doi.org/10.1109/WOSQ.2007.11

[63] Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. 2002. The use of bigrams to enhance text categorization. *Information Processing & Management* 38, 4 (July 2002), 529–546. https://doi.org/10.1016/S0306-4573(01)00045-0

[64] Garreth W Tigwell, David R Flatla, and Neil D Archibald. 2017. ACE: a colour palette design tool for balancing aesthetics and accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 9, 2 (2017), 1–32.

[65] Twitter. 2018. The new Skype is beyond frustrating. https://twitter.com/sinabahram/status/949000003451842560

[66] Christopher Vendome, Diana Solano, Santiago Liñán, and Mario Linares-Vásquez. 2019. Can everyone use my app? An Empirical Study on Accessibility in Android Apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 41–52.

[67] Phong Minh Vu, Tam The Nguyen, Hung Viet Pham, and Tung Thanh Nguyen. 2015. Mining User Opinions in Mobile App Reviews: A Keyword-based Approach. (Oct. 2015). http://arxiv.org/abs/1505.04657

[68] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*. 1113–1120.

[69] Brian Wentz, Dung Pham, Erin Feaser, Dylan Smith, James Smith, and Allison Wilson. 2019. Documenting the accessibility of 100 US bank and finance websites. *Universal Access in the Information Society* 18, 4 (2019), 871–880.

[70] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* 14, 1 (2008), 1–37.

[71] Shunguo Yan and P. G. Ramachandran. 2019. The Current Status of Accessibility in Mobile Apps. *ACM Transactions on Accessible Computing* 12, 1 (Feb. 2019), 1–31. https://doi.org/10.1145/3300176

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