8-2020

Accessibility in User Reviews for Mobile Apps: An Automated Detection Approach

Murtaza Tamjeed
mt1256@rit.edu

Accessibility in User Reviews for Mobile Apps: An Automated Detection Approach

by

Murtaza Tamjeed

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Software Engineering

Supervised by
Dr. Mohamed Wiem Mkaouer

Department of Software Engineering
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

August 2020

The thesis "Accessibility in User Reviews for Mobile Apps: An Automated Detection Approach" by Murtaza Tamjeed has been examined and approved by the following Examination Committee:

Dr. Mohamed Wiem Mkaouer
Assistant Professor, RIT
Thesis Committee Chair

Dr. Yasmine Elglaly
Assistant Professor, RIT

Dr. Zhe Yu
Assistant Professor, RIT

Dr. J. Scott Hawker
Associate Professor
Graduate Program Director, RIT

Dedication

To my parents who brought me to the earth, to my brothers who supported me emotionally, and to my teachers who instilled the love of learning in me.

Acknowledgments

Many people and programs had important roles in supporting me through my Master's studies. Foremost, I would like to thank my advisor, Dr. Mohamed Wiem Mkaouer for his immeasurable encouragement, advice, and guidance throughout the work on my thesis. His advice, assistance, and expertise were invaluable to me, and without him, this thesis would have been an impossibility. Besides, I would like to thank Dr. J. Scott Hawker, Dr. Yasmine El-Glaly, and Dr. Zhe Yu, for being part of my thesis committee and for providing invaluable feedback on my thesis. I would also like to thank Dr. Eman Abdullah Alomar for her comments and technical support throughout my work.

Meantime, I am grateful to the Fulbright program for providing me amazing opportunities for growth and for sponsoring me throughout my program in the United States. Finally, I am incredibly grateful to my family for their constant and continuous emotional and spiritual support. They have inspired me throughout my entire academic career and never once failed to remind me that they believed in me. I am grateful to everyone that has supported me during this journey and express my apologies for not being able to mention them each by name.

Abstract

Accessibility in User Reviews for Mobile Apps: An Automated Detection Approach

Murtaza Tamjeed

Supervising Professor: Dr. Mohamed Wiem Mkaouer

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 restrictions. Recent work demonstrated that 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 thesis is to support the automated identification of accessibility in user reviews, to help practitioners 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 5326 mobile app reviews. The findings show that (1) our approach can accurately identify accessibility reviews, outperforming two baselines, namely keyword-based detector and a random classifier; (2) our model achieves F1-measure of 90.7% with relatively small training dataset; however, F1-measure value improves as we add to the training dataset.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Users are allowed to search and download apps for mobile devices from application distribution platforms or app stores such as App Store[1], Google Play[2] and Amazon Appstore[3]. Users are also allowed to write reviews in these platforms in the form of text or star ratings, known as *user reviews*. Developers, app designers and analysts can study their users' experience and identify issues with their apps using the user reviews [28][11][27]. Reviews that users put can be feature requests, identification of issues, praises, complaints, and dissatisfaction, among others. The reviews can also be classified on higher levels, for example, on how good or bad a feature was implemented.

Moreover, this higher level classification of app reviews can also be about design and usability. It can also be about accessibility — whether apps and webs are accessible to people with disabilities when they are using mobile phones and other devices [21]. However, accessibility in user reviews is rarely studied especially for mobile applications [15]. Right now, accessibility-related research and associated guidelines are more towards websites rather than mobile apps [40], while mobile app development has greatly increased lately. This dramatic increase in the development of mobile apps invites a deeper understanding of how accessibility concerns are discussed in user reviews; however, it's not necessarily easy to tackle.

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

[2] https://play.google.com/store

[3] https://www.amazon.com/mobile-apps/b?ie=UTF8node=2350149011

1.2 Problem Statement

There are multiple challenges associated with studying accessibility in user reviews. First, app stores contain a vast amount of reviews that require lots of effort to analyze. Developers and analysts normally suffer from the burden of analyzing this enormous reviews. Second, the manual process of identifying accessibility-related reviews (for sake of simplicity, we call them *accessibility reviews*) is error-prone and it is subject to reader bias. A preliminary work studying accessibility in user feedback [15], is entirely depending on manual inspection of app reviews. Third, identification of user reviews pertaining to accessibility can be hard to accomplish through current state-of-the-art approaches (i.e. string-matching) due to the lack of a set of standard keywords or phrases to filter those reviews. The current state-of-the-art approach [15] relies on 213 keywords extracted from 54 British Broadcasting Corporation[4] (BBC) recommendations for mobile accessibility [6] to identify accessibility reviews. Ignoring 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. Finally, the presence of certain keywords in a review does not necessarily mean the review is about accessibility. For example, consider the following reviews from Eler et al. dataset:

> *”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 its not an accessibility review. In this review, the word ”*old*” is referring to a device rather than a person. Both the words ”*blind*” and ”*sight*” refer to knowledge of goods in the game rather than describing a player. Similarly, in the reviews:

[4] https://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile

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

Users use keywords like "*accessible*" and "*blind eyes*" but they do not really express any accessibility concern. Thus, these limitations in the identification of accessibility reviews make the current state-of-the-art approach practically unreliable.

1.3 Proposed Approach

For reducing the efforts developers and analysts require in identifying and understanding accessibility from user reviews, we use supervised learning to formulate the identification of accessibility reviews as a binary classification problem. This approach takes a set of accessibility reviews obtained by manual inspection in a previous study [15] as input for training several classifiers.

The advantages of using a binary classifier, over the keyword-based detector, are at least twofold. First, our approach automatically identifies new accessibility reviews which were simply missed by the keyword matching approach. The reason why the new accessibility reviews were identified is that our model depends on learning algorithms rather than a set of predefined keywords. Second, in contrast to relying on words derived from guidelines, our approach extracts features (i.e. patterns) from actual user reviews and learns from them. A pattern refers 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 with their apps. The patterns can be about assistive technologies that help people with disabilities (e.g. "*text to speech*", "*word prediction*" or "*voice over*"); about usability feature of the apps that helps people with disabilities (e.g. "*font customization*", "*page zooming*" or "*speed control*"); as well as about disability comments (e.g. "*low vision*", "*handicapped*", "*deaf*" or "*blind*").

We test our approach using a dataset of 5326 user reviews, provided by a previous study [15]. We compare our approach against two baselines; (1) keyword-based approach and (2) random classifier approach. Our approach provides a significant improvement in the F1-measure score for identification of accessibility reviews, outperforming the baseline-1 (keyword-based approach) by 1.370 times, and surpassing the baseline-2 (random classifier) by 39.434 times.

Since we also wanted to find the size of the training dataset needed to effectively identify accessibility reviews, we incrementally added batches of 100 reviews to our training dataset until we reached 500 reviews. Then, we kept adding batches of 500 reviews until we used all of our training data (5326 reviews). Our results show that the highest F1-measure (i.e. 0.907) was achieved with 5326 reviews (our total training dataset) and the lowest F1-measure value (i.e 0.635) was achieved with 100 reviews. We report in more details in chapter 5.

The contribution of this study is multi-fold. In the first place, we propose an approach to automating the detection of accessibility reviews with high performance. This approach can help developers automatically detect accessibility-related reviews and filter out irrelevant reviews. Second, we show 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. Third, we make our model and datasets publically available[5] for analysts and developers to extend our work and gain further understanding of the topic.

The rest of this thesis is organized as follows. Chapter 2 describes key information about software accessibility, user reviews analysis, various approaches in the classification of text documents, metrics for evaluating the performance of classifiers, and other important background knowledge to better understand this work. Chapter 3 reveals related studies that motivated this work. Chapter 4 reviews our data preparation process; how we preprocess data; how we extract features; how we train and build our models; as well as how we select and evaluate various classifiers. We then discuss our experiments and findings in Chapter

[5] https://gallery.cortanaintelligence.com/Experiment/Automating-Detection-of-Accessibility-Reviews

5. We examine threats to the validity of the study and motivate future work in Chapter 6 before concluding in Chapter 7.

Chapter 2

Background

In this chapter, we review key information about software accessibility, user review, classification of text, and other information central to a thorough understanding of the problem space.

2.1 Software Accessibility

About a billion people in the world have some form of disability, and this number is rising with advances in medical care and population growth [9]. In the United States only, 26% of individuals (1 in 4) above the age of 18 have some disability [34]. Since, a broken arm, low vision, hearing loss, limited movement, stroke, aging, losing glasses, to name just a few, can render someone disabled, thus, anyone can become disabled. Furthermore, everyday life circumstances can mimic a disability, forcing someone to depend on the solutions built for people with disabilities. For example, you must depend on your keyboard if your mouse stops functioning. Similarly, audio transcripts become important for someone who is blind. While, captions become necessary for someone with hearing impairments. Thus, when tools and technologies are designed with accessibility in mind, people with disabilities can use them.

Making technologies and tools accessible benefits individuals, society, and businesses [49]. Today technologies such as web and mobile applications are increasingly important in all walks of life. It's also legally required that these technologies are accessible in order for people with various abilities to get equal access and equal opportunities. Access to information and communications technologies is considered a basic human right in the United Nations Convention on the Rights of Persons with Disabilities [3]. Furthermore, accessibility encourages inclusion not only for people with disabilities but also for minorities [49].

There is also a strong business case for accessibility. It can enhance branding, drive innovation, and boost marketability [49]. An accessible and inclusive app cannot only improve user experience and satisfaction but also increases the user base. Thus, it is valuable for developers to make their apps accessible and inclusive. In contrast, failing to account for accessibility can not only cause inconvenience to users, but it could also shift a large user base to alternative apps. Currently, many websites and mobile apps have accessibility issues that make them too hard to use.

2.2 User Review

Smart devices such as smartphones, tablets and various wearable devices are ubiquitous; so are applications for them. According to a survey, there are more than two billion smartphone users, and over twelve million app developers [4]. More than five million apps have been developed and put on the app stores recording more than 204 billion downloads [1].

Digital application distribution platforms allow users to get apps for mobile devices, also allow them to describe their experience about the apps as in the form of reviews. Current research indicates that user reviews provide valuable information for analysts and developers to identify user experience with their apps, issues, requests, and even the rationale for their likes and dislikes [29][13][24]. There are many types of reviews that the reviewers write, such as feature requests, praises, complaints, dissatisfaction, and identification of issues, among others. It is important for developers to address issues if any, in a timely manner [32], otherwise users could simply choose to use different app [47].

Another study by Pagano and Maalej [35] shows that mobile apps receive about 23 reviews each day, while popular apps, receive 4,275 reviews every day. App developers consistently spent considering the amount of time in exploiting and classifying these reviews to improve user satisfaction. Many previous studies [10][8][35] report that only one third of these reviews are helpful for developers.

2.3 Classification of Text

In this section, we discuss various techniques that can be performed on text to predict their types.

2.3.1 String Matching Approach

A keyword-based approach, also known as pattern-based approach or string matching, relies on a set of predefined keywords to categorize text into classes. We can first determine a set of keywords that we look to find in text documents (e.g. user reviews). Then check if any of the keywords are available in the text. For this work, we can use regular expressions or SQL queries.

2.3.2 Machine Learning Approach

Classification systems are built using Machine Learning approaches. These approaches are divided into supervised and unsupervised techniques. Both learning techniques classify numbers and text using a previously defined set [37]. Machine learning algorithms, without detailed domain-specific knowledge, are proven to solve problems from many fields such as ASR (Automated Speech Recognition), computer vision, document classification, among others [7].

In the supervised learning, algorithms are provided a set of labeled *training* examples. Then, the algorithm uses the training examples to develop a classification rule that will perform well over new data. The main characteristic of machine learning is learning from one set of data in order to perform well on new data it has not seen yet. This key feature distinguishes it from other algorithmic tasks.

In this thesis, We focus on binary classification where documents are classified into either positive class or negative class. Specifically, we are working on a binary classification to automatically categorize app reviews into accessibility-related reviews (i.e. *accessibility reviews*) and reviews that are not related to accessibility (i.e. for sake of simplicity, we call them *non-accessibility reviews*).

2.4 Evaluation Metrics

To measure how well a classification model performs, the following metrics are needed to be reviewed.

2.4.1 Accuracy

It's the accuracy metric that measures the overall performance of a classification model. Accuracy of a classification model is calculated as follow:

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn} \tag{2.1}$$

True positives (tp) are the number of items correctly classified for the positive class, and true negatives (tn) are the total correct classifications for the negative class. Similarly, false positives (fp) are the number of incorrect classifications for the positive class; and false negatives (fn) are the number of incorrect classifications for the negative class. To simply put it, total correctly classified items are divided by the entire number of classifications — correct and incorrect.

2.4.2 Precision

The precision metric measures accuracy of the positive class predictions. Specifically, it shows the number of correct classifications out of the positive class predictions. Precision can be presented as follows:

$$Precision = \frac{tp}{tp + fp} \tag{2.2}$$

2.4.3 Recall

Recall calculates the accuracy of a classification model for true positive cases. Specifically, how many were correctly classified as positive out of all of the positive cases. This intuition is captured by the following:

$$Recall = \frac{tp}{tp + fn} \tag{2.3}$$

2.4.4 F-score

Finally, F-score (i.e. also known as F-measure or F1-score) is the harmonic mean of precision and recall that provides a trade-off between them; and allows mentioning one number to represent both measures. This means that the f-score measure is a mean for us to determine how well our model is performing on both precision and recall as seen in the following:

$$F - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

It's worth noting that we mainly rely on the F-measure score to determine how effective our machine learning algorithm is when comparing our approach with the state-of-the-art.

Chapter 3

Related Work

In this chapter, we highlight several previous works that profoundly influenced our approach. We split our related work into three sections: user review, which briefly highlights the role of user reviews in app evolution; accessibility in user reviews, 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.

3.1 User Review

Many researchers concluded that reviews and ratings posted by users on app store platforms can play an essential role in the evolution of mobile apps because a lot of developers consider reviews when working on new release [11][27][36][39]. Maalej et al. [28] introduces user review as the first means when eliciting requirements for software development. Similarly, Vu et al. [48] emphasize on the users' role in the software process by developing an approach to identify useful information from users' review. Moreover, Seyff et al. [44] proposes the use of mobile devices to continuously gather end-user feedback. Since user reviews are essential to mobile app evolution, we are looking into whether we can effectively identify accessibility-related reviews from users' feedback. In a highly competitive market, it is necessary to identify accessibility-related issues from user reviews and address them in order to build satisfaction for diverse users.

3.2 Accessibility in User Review

Even though user reviews can be considered a powerful tool in the evolution of mobile apps, and that even well-grown apps have many accessibility issues [16][51], only 1.24% users report such issues to application distribution platforms [15]. In other words, 98.76% mobile app users do not post accessibility issues in the form of reviews to app stores. In an effort to find if mobile app users post accessibility-related issues to the platforms, Eler et al. [15] investigate 214,053 mobile app reviews using a string-matching approach. They report that in total 5076 reviews were identified as accessibility reviews. However, through a manual inspection later, the researchers found that only 2663 of the reviews were really about accessibility. We used these 2663 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.

3.3 Classification of Text Documents

Many previous work classify app reviews by different taxonomies [11][14][20][31][38][39], such as bug reports, complaints, praises, feature requests, and even user interfaces; however, none of them even mention accessibility [15].

A previous work [22] automated classification of user stories and identified 6 communication and clarification patterns with regards to requirements. They relied on NB (Naïve Bayes) algorithm in their classification. Their work is similar to ours in the use of a supervised machine learning approach, however, the focus of both works is different. We work to classify user reviews into accessibility and non-accessibility categories.

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. [15] relies on 213 keywords to identify accessibility-related reviews. Another study [45] detects commits related to refactoring using one keyword, "*refactor*".

Similarly, Ratzinger et al. [41] used 13 keywords to detect refactorings. Then, Murphy-Hill et al. [33] repeated Ratzinger's approach in two open-source projects using the 13 keywords he used. Interestingly, they negated their finding that commit messages in version history of software projects present refactoring activities. Their findings show that developers not usually report refactoring activities since such activities can simply be considered a different activity by developers such as building new features. Similarly, we believe users can express accessibility concerns without explicitly using any accessibility keyword presented section 4.2.1.

In contrast to the keyword-based approaches, we use an automated machine learning approach since learning approaches outperform the keyword-based approach by at least 1.6 times [5]. 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*.

Chapter 4

Approach

In this chapter, we first introduce our overall approach to the detection of accessibility reviews. We then elaborate on technical details of our adopted classification technique, in the following subsections.

4.1 Overall Framework

The goal of this work is to automate the detection of accessibility-related reviews in a large dataset of app reviews. Our approach receives 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 into *accessibility reviews* and *non-accessibility reviews*). To be able to do so, this work goes through two main phases: (1) building a model and (2) prediction. In the first phase, we build a classification model using a corpus of reviews and current classification techniques. In the second phase, we use the created model in predicting types of new app reviews.

Our framework, (1) uses a dataset of app reviews along with their ground truth categories previously identified through a manual approach [15] for learning purposes. (2) Then, we apply 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) Then, we extract features from the preprocessed review text to build a feature space. (4) Next, a training set was built using the extracted features for the model to learn from. (5)

Then, we examined a total of nine classification algorithms to evaluate the performance of the model for prediction. These classifiers are chosen because they are commonly used for classification of text such as app reviews [20][22]. After training and evaluating the model, we use test data to challenge the performance of our model. Since the model has already learned from the N-Gram vocabulary and their weights discussed in Section 4.2.3 from the training dataset, the classifier will finally output predicted-labels and probability-scores for the testing dataset. Figure 4.1 provides an overview of the process used in the detection of accessibility reviews.

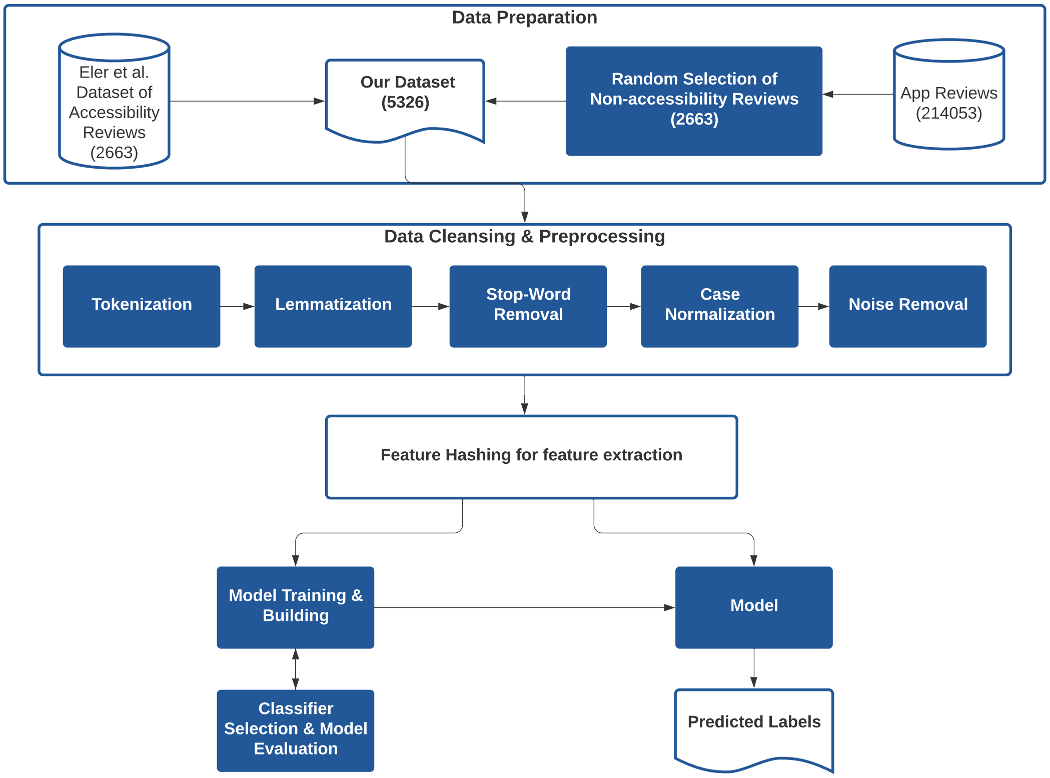
Data Preparation
Eler et al. Dataset of Accessibility Reviews (2663)
Our Dataset (5326)
Random Selection of Non-accessibility Reviews (2663)
App Reviews (214053)
Data Cleansing & Preprocessing
Tokenization
Lemmatization
Stop-Word Removal
Case Normalization
Noise Removal
Feature Hashing for feature extraction
Model Training & Building
Model
Classifier Selection & Model Evaluation
Predicted Labels

Figure 4.1: Overall Classification Framework.

4.2 App Review Classification

As mentioned, our classification method has five main phases. (1) Data preparation, (2) data cleansing, (3) feature extraction using Feature Hashing (4), model training and building, and (5) selecting classifier and evaluating the model.

4.2.1 Data Preparation

To prepare training data for the binary classification of app reviews we need to create two groups of app reviews i.e. (1) reviews indicating accessibility and (2) reviews not related to accessibility. Since a large number of app reviews had to be manually analyzed to obtain sufficient reviews at least for the first group (i.e. *accessibility reviews*), we decided to choose the set of reviews (i.e 2663) previously identified and validated as accessibility reviews through manual inspection by Eler et al. [15].

The second group of the training data contains the negative set representing the non-accessibility reviews, which can be created by randomly choosing reviews that simply did not contain any of the accessibility keywords. However, we wanted to challenge and improve the decision making of our model by adding reviews that are closest to neighboring regions between the two groups. To do so, for the second group we added an equal number of reviews that did contain one or more of the accessibility keywords (presented in 4.1 and Appendix A), but, they were really excluded from the set of accessibility reviews after manual verification. 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 [25][26], we considered an equal size of reviews for both groups of the training dataset.

In summary, we used a total of 2663 accessibility reviews for the first group and added another 2663 non-accessibility reviews to the training dataset as the second group. To challenge and improve the decision making of the model, the second group is created from two sub-groups of reviews; (1) 1331 reviews that contain one or more accessibility keywords but are previously excluded from the list accessibility reviews through manual inspection, (2) and 1332 reviews that are not related to accessibility and they do not contain

Table 4.1: Eler et al. List of keywords having accessibility reviews.

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

any accessibility-related keywords. Meanwhile, we wanted to provide the model enough reviews that could speak for all possible accessibility reviews. Thus, unlike the thresholds used in many previous text classification studies, our training data is consisting of 5326 reviews. Table 4.2 explains our training dataset.

Table 4.2: Training dataset.

Accessibility	**Non-accessibility**		**Total**
	With keywords	Without keywords	
2663	**1331**	**1332**	**5326**

To decide on the number of reviews necessary for training purposes, we analyzed previous text classifications works to find the thresholds. The average number of items used in comparable studies was close to 2000. 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 5326 reviews for training purposes. 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 5.3.

4.2.2 Data Cleansing and Preprocessing

After data preparation, we used a common approach detailed in [23] for preprocessing. For a model to classify text documents correctly, the text needs to be cleaned and preprocessed. To preprocess the app reviews text, we use some techniques in Microsoft Azure ML [2] such as tokenization, lemmatization, capitalization, and stop-words removal.

Tokenization

This technique is the process of splitting natural text data into tokens (i.e., words, phrases, and other elements) that contain no white space. In this work, we tokenize each review by breaking them into their constituent set of words.

Lemmatization

Lemmatization is about getting the base form of words by either removing the suffixes of the words or replacing the suffixes. It's also the process of reducing the number of unique occurrences of similar words. We use this preprocessing technique to represents words in their canonical form in order to reduce the number of unique occurrences of similar text tokens.

Stop-Word Removal

Stop-Word removal is about removing 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 letter 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 to ignore words that differ just in letter case to have different features.

avoid having repeated features differing only in the letter case.

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.

4.2.3 Feature Extraction Using Feature Hashing

After cleansing and preprocessing the reviews text, we extract features from the preprocessed text that matter the most in distinguishing between the two classes in classification. Particularly, we use the Feature Hashing technique for feature extraction. The Feature Hashing transforms text documents into a set of features and represent as numeric vectors.

Internally, the Feature Hashing creates a dictionary of N-Grams. We use 2-Gram in our classification since it greatly improves the performance of text classification [46]. Generally, N-Grams have more meaning and semantic than any isolated word. 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*" could indicate accessibility reviews.

4.2.4 Model Training and Building

We built a model on the Azure ML platform to automatically classify app reviews as accessibility pertaining or not. During this phase, we applied a ten-fold cross-validation technique to evaluate the reliability and variability of our classifier. We split our data set into ten different folds each containing the same size of app reviews. Then, we performed 10 evaluations with various testing datasets. Nine folds were used as a training dataset and the other fold was used as a testing dataset in each evaluation. We analyze the results of all evaluations and report the average stats tested with multiple classifiers.

4.2.5 Classifier Selection and Model Evaluation

Choosing an appropriate classification technique for optimal performance is hard by itself [17]. In this work, we are tackling a binary classification problem as we are categorizing app reviews into two groups, accessibility and non-accessibility. Our approach relies on the supervised machine learning algorithms to assign each review into one of the two categories since we already have a predefined set of classes.

Since knowing which classification algorithm works better is important, several studies have compared many classifiers, namely Naive Bayes Multinomial (NBM), K-Nearest Neighbor (KNN), Gradient Boosting Machine (GBM), and Random Forest (RF) [26][25]. They reported that Random-Forest works better than others. However, we tested nine different classification algorithms ourselves as to see which one provides the best results in the context of accessibility and app reviews classification. While testing the nine classifiers, we compared them based on their common statistical measures (i.e. precision, recall, and F1-Score). The classifiers we tested are: Decision Forest (DF)[12], Logistic Regression (LR), Boosted Decision Tree (BDT)[2], Support Vector Machine (SVM)[50], Neural Network (NN)[18], Averaged Perceptron (AP)[2], Bayes Point Machine (BPM)[19], Decision Jungle (DJ)[2], and Locally Deep SVM (LD-SVM)[2]. 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 5.1.

Chapter 5

Experimental Results and Evaluation

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

5.1 Classifier Evaluation

RQ1: Is it possible to accurately perform two-class classification of app reviews (i.e. into accessibility reviews and non-accessibility reviews) using a machine learning technique?

To a certain extent its possible to manually identify accessibility reviews from a given set of user reviews as done in a previous work [15]. However, manually reading a large number of reviews is challenging. The manual process of filtering accessibility reviews from other reviews is human-intensive, error-prone, and could be subject to reader bias. Developers and analysts frequently face the burden of analyzing large user reviews. Considering the lack of an automated approach to detect accessibility-related reviews and filter out irrelevant reviews, we performed an experiment as to figure if the automatic classification of user reviews using machine learning techniques can be performed with high accuracy. Answering this research question is important as can help determine the limitations and opportunities of the machine learning technique in the automated identification of accessibility reviews.

A comparison between different classification algorithms tested in this study is presented in Table 5.1. This table shows that the accuracy and F1-measure of the Boosted Decision Tree (BDT) are clearly higher than its competitors for the classification of app reviews. The BDT with the accuracy of 90.6% and F1-measure of 90.7%, outperforms other classification algorithms. The table 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 Tree. The BDT, with the best performance, is selected to classify the test data.

The fact that BDT achieves 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 is fixed by the third, and so on. In this method, the entire ensemble trees together form the prediction.

Table 5.1: Performance of binary classification techniques.

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-measure**
Logistic Regression	0.881	0.920	0.834	0.875
Decision Forest	0.880	0.914	0.838	0.874
Boosted Decision Tree	0.906	0.898	0.916	0.907
Neural Network	0.854	0.825	0.901	0.860
Support Vector Machine	0.869	0.892	0.840	0.865
Averaged Perceptron	0.886	0.903	0.864	0.883
Bayes Point Machine	0.889	0.906	0.869	0.887
Decision Jungle	0.870	0.914	0.818	0.863
Locally Deep SVM	0.873	0.882	0.860	0.871

Summary. We found that our approach with the accuracy 90.6% and F1-measure of 90.7%, can accurately identify accessibility reviews and filter out irrelevant reviews.

5.2 Approach Comparison

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

Our goal is to recommend 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) [15] and Random Classifier [30]. 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. If a learning algorithm does not perform better than a String-matching algorithm, then there is no need to propose such a framework. To examine if the hypothesis holds true, we chose to investigate the following two baselines similar to [5] and [30], and compare them with our learning approach.

5.2.1 Baseline 1 (Keyword-based Approach)

The keyword-based (string-matching) approach for identifying accessibility reviews is suggested by Eler et al. [15]. In their work, they inspected 214,053 user reviews to identify the reviews pertaining to accessibility. Their String-matching approach classified a total of 5076 reviews as accessibility reviews. However, manual verification of the 5076 reviews later found that only 2663 of the reviews were correctly identified.

For calculating statistical metrics for baseline 1, we randomly selected 1000 reviews from the list of accessibility reviews and non-accessibility reviews provided by [15] with their ground truths. Then, we inspected these reviews to determine true positives (*tp*), true negatives (*tn*), false positives (*fp*), and false negatives (*fn*) by comparing the classification (labels) of the random reviews with the manual classification results performed by Eler et al. [15]. If a review had the same classification as the manual classification, we will assign a true positive (*tp*) or true negative (*tn*). True positives are when the keyword-based approach correctly detected accessibility reviews, and true negatives are when non-accessibility reviews are correctly identified. Likewise, if the classification in the random reviews is different from the manual classification, we assign false positive or false negative. 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* as shown below.

tp	243
tn	509
fp	248
fn	0

After having the values for *tp*, *tn*, *fp* and *fn*, we could easily calculate the precision ($P = \frac{tp}{tp+fp}$), recall ($R = \frac{tp}{tp+fn}$), and F1-measure ($F = \frac{2 \cdot P \cdot R}{P+R}$) of the keyword-based approach. The results are presented in Table 5.2, however, we only use F1-measure score for comparing different approaches. F1-measure being the hormonic mean of precision and recall, provides a good trade-off between them and allows mentioning one number that represents both measures.

5.2.2 Baseline 2 (Random Classifier)

Like Da Maldonado et al. [30], we considered Random Classifier as a baseline to compare our approach with. The precision of this technique is calculated by dividing total accessibility reviews by total user reviews (i.e. $\frac{2663}{214053} = 0.012$). When it comes to the recall, there is only 50% probability for a review to be classified as accessibility review since there are two possible classifications available. Finally, the F1-measure of baseline 2 is calculated as $2 * \frac{0.012*0.5}{0.012+0.5} = 0.023$.

Table 5.2 shows the standard statistical measures of the three approaches, also the improvements in our approach compare to the other approaches. As we can see, F1-measure obtained by the machine learning approach is much better compared to other approaches. F1-measure achieved by the machine learning approach is 90.7, while the F1-measure values using the keywords and the random classifier are 0.662 and 0.023 respectively. Figure

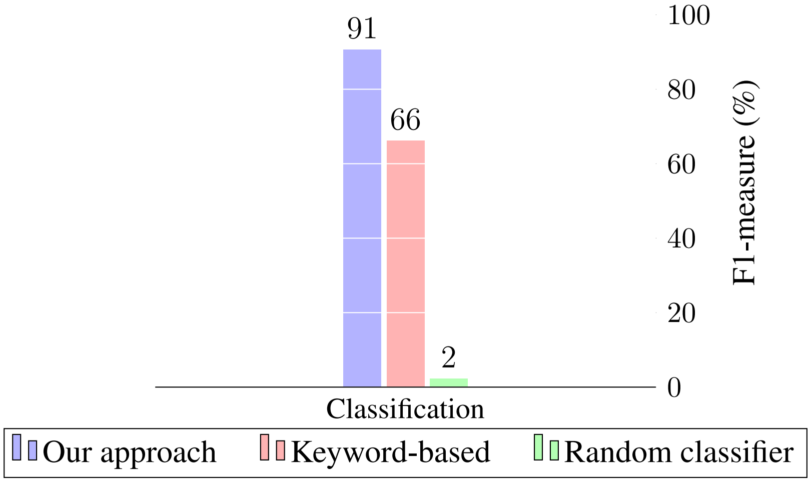

Figure 5.1: F1-score for different approaches.

5.1 presents a visual comparison of the three approaches in terms of F1-measure. Both Figure 5.1 and Table 5.2 show that our approach outperforms the keyword-based approach by 1.354x and the random classifier by 39.434x when identifying accessibility reviews.

Table 5.2: Comparing performance of different approaches.

	Our approach			**keyword-based**			**Random classifier**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Classification	0.898	0.916	0.907	0.490	1	0.662	0.012	0.500	0.023
Improvement	–	–	–	1.832 x	0.916 x	1.370 x	74.833 x	1.832 x	39.434 x

It is worth knowing that the keyword-based approach has low precision but high recall. The high recall is due to the fact that accessibility reviews make only a small subset of app reviews. The number of accessibility reviews for mobile apps is around 1.24% of all reviews available on app distribution platforms [15]. On the other hand, low precision is because the presence of a keyword (derived from guidelines) in a review does not necessarily mean the review is about accessibility.

Summary. Our machine learning approach can effectively outperform 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 1.370x and 39.434x over the keyword-based and random classifier approaches respectively.

5.3 Training Size Impact Evaluation

RQ3: How much training dataset is needed for the classification to effectively identify accessibility reviews?

So far, we saw 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 dataset. Also, creating a good training dataset is a tedious and challenging task. Thus, the question is: how much training data is needed to effectively classify user reviews? If our approach requires a very large training dataset than our approach 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 extended and applied with little efforts. That being said, we hypothesize that the performance of a classifier improves as we add to the training dataset.

In order to answer this RQ, we add reviews to the training dataset incrementally then evaluate the performance of the classification. We begin by creating a large training dataset that contains equal size of accessibility reviews and non-accessibility reviews as explained in Table 4.2. Then, we divide the dataset into 10 folds making sure they contain equal size of both the classes. Next, we test our approach using a 10-fold cross-validation technique, 9 folds for training and 1 fold for testing. Since we want to monitor the performance of our classifier as training dataset gets increased, we incrementally add batches of 100 reviews until we reach 500 reviews, then we start adding batches of 500 reviews until we use all of our training data (e.g. 5326 reviews). It's important to note that we consider equal size of accessibility reviews and non-accessibility reviews even with batches incrementally added to the training dataset. Next, we compute the F1-measure value of each iteration (e.g. after adding batches of new reviews to the training set) and record the average F1-measure. Then, we report the number of reviews needed to achieve at least an F1-measure of 80% to 90%.

Figure 5.2 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 5326 reviews (our total training dataset) and the lowest F1-measure value (i.e 0.635) 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.

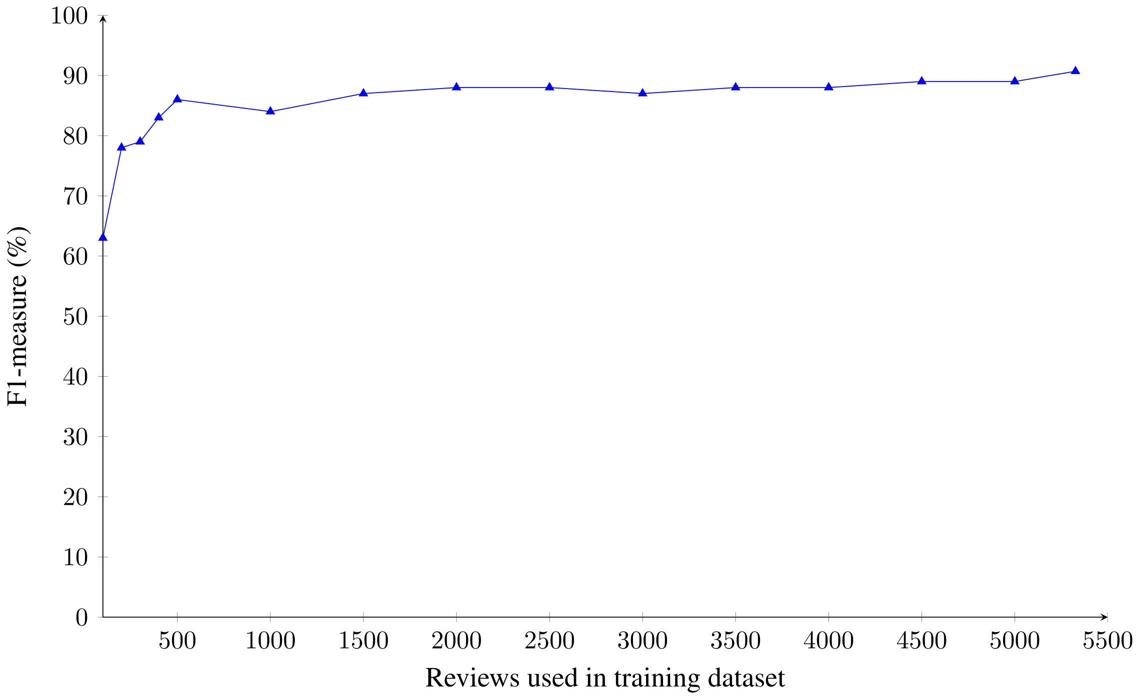

Figure 5.2: F1-score received by incrementally increasing the training data size

Summary. We find 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.

Chapter 6

Threats to Validity and Future Work

In this chapter, we identify several threats to the validity of our study and identify future work for this topic. We discuss threats in two groups, construct and external threats. Then, we review a list of potential areas of future work that could prove valuable to this study going forward.

6.1 Threats to Validity

Construct Threats relate to the appropriateness of our dataset and accuracy of the previous work [15]. A potential threat is related to creating a training dataset or the manual classification. Since developing a training dataset is typically a tedious job also subject to reader bias, we mitigated this by choosing a dataset of accessibility reviews as our training data that were previously identified and validated [15]. Additionally, we are not certain if the manually identified accessibility reviews cover all possible accessibility topics. To reduce this risk we used all of the identified reviews as training input rather than choosing a sample set of reviews. A total of 2663 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 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. We leave these tasks as our future work.

External Threats relate to the generalizability of our findings for this evaluation. We evaluate and test our findings on a dataset collected by previous researchers [15]. The dataset is from android open source applications. Obviously this dataset cannot represent entire mobile apps on the App stores. Also, commercial and open source apps may be different. Still, we believe the size of the dataset makes it inclusive and diverse. Meanwhile, we removed reviews written in any language other than English during our reprocessing phase. Since we only analyzed the reviews written in English, our approach is not applicable to non-English reviews.

6.2 Future Work

We can at least explore three areas for future work — exploring the suitability of different learners; performing a multi-class classification on the accessibility reviews; and introducing a set of accessibility keywords that users use in their reviews.

6.2.1 Active learning

Typically, the first step towards the automatic classification of app reviews is creating a training dataset. The training dataset has to be manually labeled by developers or subject matter experts. As our results show, having an adequately large training size is essential for optimal performance. The training process for an adequate training set can be cumbersome and laborious since there a large number of reviews available on app stores. In order to further reduce the efforts needed in creating a training data, we are planning to explore *Active Learning* [42] [43], a well-known machine learning paradigm for classification.

6.2.2 Multi-class classification of accessibility reviews

Another potential future direction is performing a multi-class classification on the identified accessibility reviews — dividing them into categories such as text, audio, video, UI, gestures, and others.

6.2.3 Introducing accessibility keywords that users use in their reviews

Another good contribution can be the introduction of a set of keywords that reviewers use to express accessibility concerns. The current set of keywords available is driven from guidelines. Keywords used in guidelines can differ from the keywords used by users in their reviews.

Chapter 7

Conclusion

Developers can use user reviews to improve their apps. This thesis 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. We conduct an evaluation of nine different two-class classifiers — presented in table 5.1. 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.

Bibliography

[1] Annual number of mobile app downloads worldwide 2019. Library Catalog: www.statista.com.

[2] Azure Machine Learning | Microsoft Azure. Library Catalog: azure.microsoft.com.

[3] Convention on the Rights of Persons with Disabilities (CRPD) | United Nations Enable.

[4] Evans Data Corporation | Worldwide Developer Population and Demographic Study 2019 Volume 2.

[5] Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni. Toward the Automatic Classification of Self-Affirmed Refactoring. page 46.

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

[7] Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. Cambridge University Press, 1 edition, January 2020.

[8] Laura V. Galvis Carreno and Kristina Winbladh. Analysis of user comments: An approach for software requirements evolution. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 582–591, San Francisco, CA, USA, May 2013. IEEE.

[9] Dr Margaret Chan and Mr Robert B Zoellick. World Report on Disability. page 24.

[10] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-miner: mining informative reviews for developers from mobile app marketplace. In

Proceedings of the 36th International Conference on Software Engineering - ICSE 2014, pages 767–778, Hyderabad, India, 2014. ACM Press.

[11] Adelina Ciurumelea, Andreas Schaufelbuhl, Sebastiano Panichella, and Harald 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, February 2017. IEEE.

[12] A. Criminisi and J. Shotton, editors. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer London, London, 2013.

[13] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, pages 499–510, Seattle, WA, USA, 2016. ACM Press.

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

[15] Marcelo Medeiros Eler, Leandro Orlandin, and Alberto Dumont Alves 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, Vitória Espírito Santo Brazil, October 2019. ACM.

[16] Marcelo Medeiros Eler, Jose Miguel Rojas, Yan Ge, and Gordon Fraser. Automated Accessibility Testing of Mobile Apps. In *2018 IEEE 11th International Conference*

on Software Testing, Verification and Validation (ICST), pages 116–126, Vasteras, April 2018. IEEE.

[17] Manuel Fernandez-Delgado, Eva Cernadas, Senen Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? page 49.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *J. Mach. Learn. Res.*, 1:245–279, September 2001.

[20] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 41–44, San Francisco, CA, USA, May 2013. IEEE.

[21] W3C Web Accessibility Initiative (WAI). Mobile Accessibility at W3C. Library Catalog: www.w3.org.

[22] Eric Knauss, Daniela Damian, German Poo-Caamano, and Jane Cleland-Huang. Detecting and classifying patterns of requirements clarifications. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 251–260, Chicago, IL, USA, September 2012. IEEE.

[23] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown. Text Classification Algorithms: A Survey. *Information*, 10(4):150, April 2019.

[24] Zijad Kurtanovic and Walid Maalej. Mining User Rationale from Software Reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 61–70, Lisbon, Portugal, September 2017. IEEE.

[25] Stanislav Levin and Amiram Yehudai. 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, pages 97–106, Toronto, Canada, 2017. ACM Press.

[26] Stanislav Levin and Amiram Yehudai. Towards Software Analytics: Modeling Maintenance Activities. *arXiv:1903.04909 [cs]*, March 2019. arXiv: 1903.04909.

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

[28] Walid Maalej. When users become collaborators: towards continuous and context-aware user input. page 9.

[29] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331, September 2016.

[30] Everton da Silva Maldonado, Emad Shihab, and Nikolaos Tsantalis. Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt. *IEEE Transactions on Software Engineering*, 43(11):1044–1062, November 2017.

[31] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21(3):1067–1106, June 2016.

[32] Itzel Morales-Ramirez, Denisse Munante, Fitsum Kifetew, Anna Perini, Angelo Susi, and Alberto Siena. Exploiting User Feedback in Tool-Supported Multi-criteria Requirements Prioritization. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 424–429, Lisbon, Portugal, September 2017. IEEE.

[33] Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering*, 38(1):5–18, January 2012.

[34] Catherine A. Okoro. Prevalence of Disabilities and Health Care Access by Disability Status and Type Among Adults — United States, 2016. *MMWR. Morbidity and Mortality Weekly Report*, 67, 2018.

[35] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 125–134, Rio de Janeiro-RJ, Brazil, July 2013. IEEE.

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

[37] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. page 94.

[38] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 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)*, pages 281–290, Bremen, Germany, September 2015. IEEE.

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

[40] Christopher Power, André Freire, Helen Petrie, and David Swallow. Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 433–442, 2012.

[41] Jacek Ratzinger, Thomas Sigmund, and Harald C Gall. On the Relation of Refactoring and Software Defects. page 4.

[42] Burr Settles. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, June 2012. Publisher: Morgan & Claypool Publishers.

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

[44] Norbert Seyff, Florian Graf, and Neil Maiden. Using Mobile RE Tools to Give End-Users Their Own Voice. In *2010 18th IEEE International Requirements Engineering Conference*, pages 37–46, Sydney, Australia, September 2010. IEEE.

[45] Konstantinos Stroggylos and Diomidis Spinellis. Refactoring–Does It Improve Software Quality? In *Fifth International Workshop on Software Quality (WoSQ'07: ICSE Workshops 2007)*, pages 10–10, Minneapolis, MN, USA, May 2007. IEEE.

[46] Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. The use of bigrams to enhance text categorization. *Information Processing & Management*, 38(4):529–546, July 2002.

[47] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, pages 14–24, Austin, Texas, 2016. ACM Press.

[48] Phong Minh Vu, Tam The Nguyen, Hung Viet Pham, and Tung Thanh Nguyen. Mining User Opinions in Mobile App Reviews: A Keyword-based Approach. *arXiv:1505.04657 [cs]*, October 2015. arXiv: 1505.04657.

[49] w3c_wai. Introduction to Web Accessibility. Library Catalog: www.w3.org.

[50] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, January 2008.

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

Appendix A

All Accessibility Keywords

Table A.1: Eler et al. List of all accessibility keywords.

Keywords
accessibility
accessible
actionable
adaptability
adaptable
adjust color
adjust colour
adjust font
adjust size
adjustability
adjustable
alert dialog
alternative format
alternative text
animated content
aria label
assistance
assistive
audible content
audio conflict
audio cue
audio description
auto-play
autoplay
background color
background colour
background image
black mode
black theme
blind
button order
can’t read
can’t see
cannot read
cannot see
captcha
change focus
change font
change language
choose language
clear error message
clear label
colour coding
complex control
component order
consistency
consistent label
container
content description
content order
content resizing
contentDescription
contrast
control focus
control key
customization
dark mode
dark theme
decorative background
decorative content
decorative element
decorative image
default input format
default language
difficult to read
difficult to see
disabled person
disabled user
disorientating
disorientation
distinguishable
dizziness
duplicate link
epilepsy
explicit labels
eyestrain
fatigue
flicker
focus order
focusable
font size
font sizing
grouped
haptic
hard to see
have focus
headache
header
heading
hearing loss
hidden text
images of text
impaired
impairment
impossible to read
impossible to see
inactive space
input method
input sign
input type
keyboard language
keyboard trap
label absent
label place
landmark
large font
larger font
leave focus
light mode
light theme
link description
logical order
logical sequence
magnify
metadata
migraine
minimum space
missing image
missing label
missing layout
mute
narrative audio
nausea
navigable
navigational aids
night mode
no label
non actionable
non executable
non text
non visual
non-actionable
non-executable
non-text
non-visual
older device
operable
overlap
page refresh
page title
pause button
perceivable
personalization
photosensitive
play automatic
plays automatic
properly grouped
readable
repeated link
scaling
screen glare
screen reader
screen title
screenreader
seizures
select language
self voicing
sign language
small button
small control
small font
small widget
spacing
speech synthesizer
stop button
strain
subtitle
suplementary information
svg image
switch device
talkback
target area
text alternative
text equivalent
text reflow
text resize
text resizing
text size
text-to-speech
timeout
timing out
tiny font
too bright
tooltip
touch size
touch target
transcript
understandable
unexpect open
unexpected audio
unexpectedly opening
unfamiliar
unique description
unique label
unsupported style
verbose
verbosity
vertigo
vibration alert
vibration feedback
visible focus
visible label
vision impairment
visual cue
visual formatting
voice command
volume
wcag
white mode
white theme
widget order
zoom