11-2017

How Should You plan Your App's Features? Selecting and Prioritizing A Mobile App's Initial Features Based on User Reviews

Rebaz Saber Saleh
rss1803@rit.edu

How Should You plan Your App's Features? Selecting and Prioritizing A Mobile App's Initial Features Based on User Reviews

by

Rebaz Saber Saleh

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

Supervised by

Dr. Daniel Krutz

Department of Software Engineering

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

November 2017

The thesis “How Should You plan Your App’s Features? Selecting and Prioritizing A Mobile App’s Initial Features Based on User Reviews” by Rebaz Saber Saleh has been examined and approved by the following Examination Committee:

Dr. Daniel Krutz
Assistant Professor
Thesis Committee Chair

Dr. Christian Newman
Assistant Professor

Dr.Hawker Scott
Associate Professor

Dedication

This thesis and all of my academic achievements are dedicated to my mother, Dulbar Tofiq, my father, Saber Saleh and my beloved wife, Zhela Rashid

Acknowledgments

I would like to thank my committee members, Dr. Daniel Krutz and Dr. Christian Newman. In particular, I would like to thank my adviser, Dr. Daniel Krutz, who provided me with the opportunity and necessary guidance to pursue this research.

I would like to thank the Higher Committee for Education Development in Iraq(HCED), for sponsoring my study.

Finally, I would like to thank all my family and friends, without whom I would not be where I am today.

Abstract

How Should You plan Your App's Features? Selecting and Prioritizing A Mobile App's Initial Features Based on User Reviews

Rebaz Saber Saleh

Supervising Professor: Dr. Daniel Krutz

The app market is extremely competitive, with users typically having several alternative app possibilities. To attract and retain users, it is imperative for developers to consider the ratings and reviews their apps receive. App reviews frequently contain feature requests, sometimes hidden among complaints. Developers use these complaints and requests to improve their apps, thus increasing their rating which is incredibly important for attracting new users. Unfortunately, developers of new apps are at a severe disadvantage: They do not have the benefit of existing reviews, with only the reviews of similar apps to potentially rely upon. To address this problem, we conducted a study and developed a novel technique that extracts feature requests from similar, existing apps to help prioritize the features and requirements important in an initial app release.

We compared different classification models in order to identify most appropriate classifier for classifying reviews category-wise. We found that there is not one single classifier that could have a higher accuracy than others for all categories.Our study also involved extracting features from user reviews in the Google Play store. The features were presented to 17 Android developers twice; once without applying our technique and once after applying our technique. Our proposed technique created a 48% reduction in the number of features

considered high priority by participants; helping developers focus on what features to consider for their apps. We surprisingly found that the frequency of requested features did not impact the developer's decisions in prioritizing the features in the inclusion of new apps.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to the advance in phone technology, smart phones are used in the majority of our daily tasks, such as email, chat, banking, reminders, etc. These tasks are performed using mobile application which is also called an “app”. These apps are ranging from performing a simple task such as operating the flashlight of apps to apps that are specific to games such as “Candy Crush” and social apps such as Facebook and Twitter.

These apps are developed either by an individual developer or a team of developers in a company or an open source community. They are making money from either paid apps, in-app purchases or ads that are integrated into the apps. As of now, there are 3,287,763 apps in Play Store[4]. The number of apps continues growing and new apps released frequently. In Sep-2016 play store contained 2,419,362 while play store currently contains are 3,287,763 apps which increased by almost 1/3 comparing to last year.

While this large number of apps seems to be an advantage for the users, because they have more options and alternatives, but it is not the case with developers, because now the market is very competitive, the team has to spend more time and even money in marketing and developing high quality apps in order to attract more users and keep them use the app and its services, in order to grow their business and also maintain the apps while making a profit for the developer as well.

In this study, our goal is to help developers to publish feature rich application, so that their apps can emerge to the market as soon as possible and also compete with the rival apps. To do so, we are getting benefit from the reviews that users post on other similar apps in the same category. We mine the feature requests from the user reviews, and them

associate them with real functionality inside the apps, then we propose a framework that the developer can use in order to integrate these features rapidly in their apps.

1.1 Background

Mobile apps have been increasing popularity due to the advances in mobile technology. Nowadays, mobile apps provide most of the daily needs of users, from calendars, reminder, calculator, note taking to word and excel processing. In addition, Users are now able to play video games and use videos and photo editing tools in mobile phones. In third quarter revenue report 2016, Facebook announced that 90% of the daily active users are from mobile phones which was 80% in 2014 [16]. Most of Twitter's active users are also from smartphones, according to the company 83% of monthly active users are from smartphones [17]. Although the type of application significantly impact the users to mobile phones, however as compared to last years of company data, the user base of mobile phones for Facebook has increased significantly.

There are several mobile platforms including Android, iOS and Windows which each being developed and maintained by a different company. Users can install new apps through the marketplace that is provided by the company or some external app stores if the platform allows installing apps from third-party markets. Furthermore, these companies provide tools in order to make the development and publishing new apps easier for the developers. For instance, Apple provides Xcode which is an IDE for iOS app development and Google has released Android Studio to facilitate developing Android apps. These platforms are also providing tools and APIs in order to integrate ads and in-app purchases to the apps by the developers. These tools allow developers and owners of the apps making money either by selling the app, ads or in-app purchases through their apps. However, It is reported that only 7.0% of apps in play store are paid-for app [3]. Most of mobile apps depend on ads in order to generate revenue. Facebook mobile advertising revenue represented approximately 84% of all advertising revenue [16] of the company and Twitter reported that 90% of the advertising revenue was from its mobile app. Thus, the more users

the mobile apps attract, the more money they make. For this reason, many advertising technology companies provide APIs for developers in order to integrate ads to the mobile apps easily.

Ads in mobile apps are added as banners, videos, and images to the apps or within the app structure as it can be seen in Facebook's news feed. In some apps for example in games users are rewarded with game specific coins or other types of boosters when they watch or click on an ad that is provided by the app. Therefore, most developers prefer integrating ads and provide in-app purchases rather than paid apps. In addition, depending on the type of app, some users prefer ads rather than purchases, and not all users want to pay for the services provided by the app, but they can watch and click on the ads. According to Appbrain [3] there are 2,953,661 free apps and 218,481 paid apps. That means approximately 93% of the apps are free apps.

The availability of different tools and APIs makes it easy for both companies and individual developers to create and distribute their apps through the app marketplace. This leads to releasing several apps each and every day. App marketplaces might have some rules and regulations, but several apps are approved to be published through the marketplace without considering the quality of the app. But due to the expansion in the number of mobile apps, it is now more competitive and more difficult to increase the user base of the mobile app than before. Additionally, the rating and number of downloads of an app have significant effect on the popularity of the apps since they show how users are reacted and interested in using the app and the services or the tools provided by the app [21].

1.2 Motivation

After installing the app, users can post their feedback and experience in the app store. Users may also rate the app. When other users try to install the app, they usually read the average rating and some reviews of other users on the app and read number of downloads if available, which determines popularity of the app in order to have an insight about the quality of the app. Later if the user satisfied with the ratings and user reviews, he/she installs

the app otherwise they avoid installing the app and look for an alternative due to the high number of available apps. Therefore, user rating and reviews and number of downloads are very important measurements of the app popularity and quality and success of the app. Developers have to take into account users reaction to the apps, how is their experience, what are the users major interests in order to maximize it, and what are their main concerns and issues in order to eliminate them to increase user satisfaction which will eventually lead to the increase in the average rating of the app [22] and then popularity of the app as the number of downloads increase[5].

According to a study by AppBoy [2] which was conducted in order to examine the reason behind removing the apps by the users. The participants where 22 of their employees. They found that 25.9% of the participants remove apps due to problems that exist in the apps core functionality such as crashing and lack of update [2]. In addition, they found that 14.8% of participants were uninstalling apps due to existing of competitor apps or switching to an alternate app[2].For this reason, developers must now take into account the quality of the app. They must keep testing and maintaining the app and periodically update the app, in order to provide the users with the latest features and requirements that makes them keep using the app otherwise the user might switch to a rival app [2].

The reviews that users post in different app stores contains their experience, opinions, reaction, issues, feature request and etc. Developers are able to get useful information from the users feedback and make some improvements and fixes in the app. But not all user reviews are useful. For instance, if it is a gaming app, the user does not like the game, that does not mean the app quality is bad, but instead the user is not interested in this kind of game. Contrary, there are other reviews that request features in the app, if the developer add the feature of the app, it might lead to increase number of users, and might also lead to get high rating as a reward from the users. Because when developers respond to the need and feature requests of users, other users will also benefit the feature and might recommend the app to friends and family.

When the number of ratings and reviews of an app increases, it is difficult for the developers to separate the informative from non-informative reviews manually or go through them one by one. Developers now can analyze the user reviews and ratings through Google Play Console. It allows developers identify common themes and topics used in the reviews and how the average rating and reviews might affect rating [26]. Besides that, several research studies [8][10] have been conducted in order to help developers extract the informative reviews as well as grouping them, and help prioritizing app features and issues. This way developers can make improvements to their apps, in order to get positive feedback from users for next-release of the app which might lead to the success of the app [22].

Apps in the app stores distributed under several categories such as education, music, games and sub-categories such as action and adventure in games. Most apps under same categories share some common features in the app. For instance, social media login is a feature that is available in most of the gaming apps. User can sign up and login with their social accounts and share their scores and activities with friends and family. This feature allows direct sign up through social APIs instead of providing a separate sign up and login form for the app.

When users use an app, they might post their feedback or comments on the app which might be in the form of a feature request in the app [20][22][13]. The app developers that address user requests by either including the feature requested by the user or fixing a bug report, are rewarded with high rating and positive feedback and reviews from users which leads to increasing its popularity [5]. Other developers that have similar app under same category might be able to get benefit from other apps user reviews and integrate the same feature to attract more user. The user has requested a feature in an app that has been used for a while, and gained some popularity. In contrary, developers that have released new app, require sometime in order to engage more users to the app and get the user feedback. Moreover, since it is a new app, it might lack some features that are considered essential features in the app domain. Therefore, it requires sometime until the app merges into market and gets popularity.

In this study, our goal is to examine how having insights of features of other apps can help developers get to know and familiarize themselves to the most trending features and then prioritize the features for the release of the app so that they deliver the app in a shorter period of time enriched with features that are most preferred by users of similar apps under same category.

1.2.1 Motivating Example

To give a context for our work, we will now provide a specific instance where a developer of a new app could be helped by feature information from an existing, related app. In this example, a developer wants to create a music app similar to Spotify. The key differences between the new app and Spotify is that in the new app, users can invite their friends to chat and listen to the same song at the same time as well as customizing the background of the chatting interface. These two features might seem attractive to users, but they might not use the app if the app lacks some important features that are in existing music apps such as 'Playlist' and 'Using offline'.

The developers of the new app can copy some features from Spotify and other similar apps. But how do they decide which features are the most important features to users? How do they know if they are still missing some important feature? Additionally, how does the developer prioritize features? Through our study, we propose an approach to familiarize app developers with the top requested features in the music category as well as the frequency of the request for these features. Once this information is presented to the developer, they can make informed decisions on which features to include and implement first.

1.3 Research Objectives

The primary objective of this work is to study how extracting features from reviews of users would introduce and prioritize the features and requirements for initial release of an app. The target developers are those developers who are new to Android app development and

those developers who are working on a new app in a different category. However, even the developers that have experience with app development and have already published apps can also get benefit if they are looking to add more features to their apps and want to be familiar with trending features in other apps. The proposed method is intended to be used by any android developers, but due to the fact that it is hard to collect the reviews, it would be better to be used by vendors that analyze and provide user reviews for app owners such as Appbot and Appannie.

In this study we address three questions

- **RQ1: Which classifier is the most appropriate for classifying the reviews of apps category-wise?**

  In this study, we use different classifiers on the reviews of different categories and compare the performance of each classifier. The classifiers include Naive Bayes and Decision Tree. We found that, Naive Bayes performs better and has a higher accuracy in two categories(tools and music) while DT has higher accuracy in a single category(social) than Naive Bayes.

- **RQ2: What is the impact of extracting features on informing developers about features they are unaware of?**

  When developers create apps based on the ideas that they have, they usually collect information from similar apps or surf Internet in order to find some features that are common in those kind of apps and then implement it inside their apps. For example, sharing score with friends or on social apps or using leader board to compare and compete with their friends. Thus, we try to find out how does extracting features and present them to developers allow them to have an insight on the most trending features that are mentioned or requested in the user reviews.

  After extracting the features from the user reviews and presenting them to the developers, we found that, it has a significant impact on developers decision to include

these features in the next release of their apps. There was an increase by 12 times of using the features after the developers see the users requests for these features.

- **RQ3: How does extracting features of apps in same category help developers in prioritizing the features for the initial release of the apps?**

  Sometimes developers might have listed all features that they want to include in their app either in the current release or future releases of the app. However, they might change the order of these features in order to deliver in time, or have their own preferences in the feature implementations or any other reasons. Now that we have listed all features and sorted them by most requested ones, we want study how the user requests effect the developers plan in order to make some change to implement user requested features in the app prior to other feature that she/he has planned before.

  Based on the data collected from our survey, while developers want to include these features in their apps, the frequency of user requests did not impact their decision of prioritizing the features. Therefore, it help them replacing some old features with the new ones, while still making their own decision to which features to include instead of the frequency of users request.

Chapter 2

Related Work

There are several reviews posted frequently by the users after they install the app which expresses their impressions and issues about the app. But, not all these reviews are informative to the developers. Due to the large number of reviews, it is also difficult for the developers to analyze the reviews and get most informative data from them manually. To overcome this issue several mining and feature analysis research studies have been done on the user reviews to assist the developers filter the most informative reviews from all the user reviews. The majority of these studies have been conducted in order to see how it will help the developers with the reviews of their own apps. However, we are trying to extract and mine these feature from a broader area, which is category. That means instead of analyzing single app under a category, we analyze the features of a number of apps which are most popular apps. This way, any developer can get benefit from these features that are planning to publish an app under a specific category. Also companies and open source community can get benefit our finding to create recommender that suggest top features included in the app.

Chen et al. [8] propose a framework in order to automate the process of extracting useful information from the app reviews in app marketplace. They use mining and topic modeling technique in order to analyze the reviews and group similar reviews and rank them, in order to help developers identify most requested features by the users. The researchers aim to answer three main question, first the performance of the framework, secondly are the ranked topics are really informative and useful to developers, and finally what would be the advantage of using the framework over manual inspection. They have worked on

four different apps from different categories in play store and have distributed the review data into three different datasets which are Training, Unlabeled and Test dataset. The researchers assert that by using their framework there will be less necessity for man power, and that their framework would analyze the reviews much faster than manual inspection of the reviews. While, this study is very close to our study, but as we mentioned before, we are working on a broader area which is category instead of individual apps.

Villarroel et al. [29] are proposing an approach to help developers prioritize the bug fixes and feature requests in planning for the next release of the app. For this reason they develop a tool code CLAP (Crowd Listener for Release Planning). They are using some mining tools and techniques in order to mine the app store reviews and ratings, These tools include Weka and clustering algorithms such as DBSCAN and then use the data to train CLAP in order to make better recommendation for prioritizing process. The difference with this tool and our study is that, they are working on the reviews of the same app, however we are collecting reviews from other apps than original apps main reviews in addition we are specifically working on extracting only features, while they have also worked on bug fixes too. The researchers in this study[29] first process the data so that it can be classified into bug reports, features, and other. There focus is mainly on the bug reports and feature requests. They then cluster these data so that they can group similar bug reports and feature requests, then they help the developer to prioritize them, in order to decide which part to do first in the next release either bug reports, or feature requests, and among them which are low and high priority.

In order to measure the tools accuracy, the researchers are doing some manual evaluation of the data. They assign two different developers in order to analyze and decide about the reviews and make plan on the analyzed data manually. They found out that the accuracy of the tool is 76%. They also tested the tool with three different software companies in Italy in order to evaluate the results and get user feedback about the tool, and they claim that they got positive feedback about the tool and the users asserted that the tool will help them in future planning of their apps release.

Vu et al. [30] are using some mining techniques to analyze the reviews and group similar reviews and visualize the change of the reviews over time, in order to help developers identify the trends in the apps growth. In addition, they are mapping the user reviews to keywords so that the developer can retrieve reviews that are most relevant to the keyword. They evaluated the study on 95 different apps of 2,106,605 reviews from Google Play. They found out that the accuracy of grouping similar reviews is 83% and 90% of returned results are relevant to the searched keyword. Finally, the researchers assert that the tool(MARK) will save a lot of time and effort for the developers that are concerned about specific feature or issue from user reviews and analyze the change of their concern through time. While they study changes in the apps over time, we are working to help them to decide which features to include before actually tracking how it impacted the users perspective about the app.

Similarly, Vu et al. proposing a framework in order to extract the useful information from user reviews [31]. They implement a **phrase-based** technique when filtering informative from non-informative sentence in the reviews and then cluster the reviews. After extracting the information, they also monitor how the user reviews change through time in order to make it easier for the developer to see how their modification in the code is reflected with the users. The researchers collected three million reviews from 120 different apps within a period of time between January, to September 2015.They also evaluated their approach on two different case studies in order to show the ability of the tool to automatically detect major user opinions and support developers in understanding them.

Iacob et al. created a tool MARA (Mobile App Review Analyzer) which automates the process of downloading the user reviews and separating features requests in the user reviews from other reviews. They then used the LDA model to identify topics that can be associated with the feature requests. They conducted the study on 136,988 user reviews and found out that nearly 23% of user reviews contains feature requests. The feature request includes adding new levels to games, re-design a feature in the app that already exists, and also more customization in the app[13].

Palomba et al. [22] examined how addressing the feature requests posted by the users in the reviews of an app, would lead to the success of the app in the future. Furthermore, they propose a framework in order to automate the process of finding the feature/bug fix request coverage. They are linking the reviews of the users with the issues and commits of the app and make a comparison between two different versions of the app in terms of the apps' rating. They use AR-Miner [8] in order to filer non-informative reviews. They perform the study on 100 android apps and they found out that 49% of the feature requests are addressed by the developers and following up users requested lead to the success and increase of the user ratings in the future.

Guzman and Maalej [10] have used a Natural Language Processing, and Data Mining techniques to filter the non-informative reviews. They have also extracted the information about sentiments and opinion associated with the features. They performed the study on the reviews of seven different apps on app store and play store. As a result, they found that their approach would help developers identify the requirement and would help the would release-planning of the app. In addition, Guzman et al. created a tool (DIVERSE) in order to simplify the process of extracting informative reviews from user reviews in different app stores. In addition, DIVERSE groups related reviews, as well as classifying them into Positive, Negative, and Neutral reviews. They collected the data for seven most popular apps on play store. They evaluate DIVERSE in order to calculate it is efficiency and it is usefulness among developers. Eighteen developers participated in the evaluation, four of them which worked in industry and the other fourteen were graduate students. Consequently, they found out that their algorithm is much faster than the baseline. In addition, the developers that use DIVERSE need less time in order to get the informative reviews and similar user opinions from the user reviews.

Hoon et al. conducted two research studies on the structure of user reviews. They study how the rating effect on the length of the user review and the vocabularies used in the user review. They collected and analyzed 8.7 million reviews from 17,330 apps on the App Store for both studies. They found that users tend to leave short and informative reviews,

and users that rate the app poorly tend to leave longer reviews and also app category affects length of the reviews. Furthermore, the range of words of is much higher in negative opinions that the positive reviews [12].

The author of previous study, performed another study on user reviews. Hoon et al. conducted a research in order to find the consistency of user reviews to their rates on the apps [11]. They classify user reviews under Positive, Negative, Inconsistent Positive, Inconsistent Negative, Neutral, and Spam based on the consistency of the review to the app rating by the user. The researchers have analyzed 29,182(27.4% of the total) reviews which is from 25 Free Health & Fitness apps in the App Store and found that "27,685 (94.9%) of the dataset consistent in content and rating to communicate Positive satisfaction". As a result, they claim that the user reviews are very consistent to user rating when the review is limited to 5 words, In addition they report strong correlation between the aggregated user rating and the user reviews from which the developers and users can depend on when installing the app.

Shaw et al. [25] performed a research study on the correlation between the traditional software quality metrics and its effect on the success of the apps. They conducted a research on the source code of 1000 high rated apps 1000 low rated apps in order to identify the software quality metrics in those apps. They found that there is not a high correlation between the traditional software quality metrics and success of the app. They introduce a set of new metrics that would lead to the success of the mobile apps. Ruiz et al. [24] also scanned the source code of several apps in order to find the impact of the number of ads on the app rating. As a result, they claim that there is no indication of relation between the number of ads and the rating. But they assert that some type of ads can negatively impact the app rating. Additionally, Bavota et al. [6] apps that use API's which is either change-proneness or fault-proneness are more likely to get lower rating than the apps that use API's which is not prone to fault and change.

Panichella et al. [23] created a framework in order to help developers extract the information that help them specifically with the app maintenance and development. In addition,

They combined multiple mining techniques in order to find out if combining them would filter user reviews faster than applying each mining technique individually. The mining techniques include Natural Language Processing, Text analysis and Sentiment analysis. The researchers use reviews of seven top ranked apps from both play store and apple store from different categories and evaluate their approach. As a result, the researchers assert that, applying their technique would help developers classify the user reviews to app maintenance and evolution and that the precision is 75.2% and recall is 74.2%. They also conclude that the precision and recall can by increased by increasing the training dataset.

Previous research studies show the importance of the user reviews, and it is impact on the app success. They also propose some techniques and frameworks in order to help developers extract and analyze useful information from the reviews. These studies are conducted on the **existing reviews** for the apps. However, new apps have **none or very few reviews**, which means developers cannot collect sufficient information from the user reviews. Therefore, there is a need for some studies and tools that would help developers with new apps to get benefit from the reviews of apps in **similar category** to extract feature that can be used in the initial release of an app. This way the developers can integrate features before releasing the app so that it attracts more users, and increase the likelihood of getting high rating without waiting a long time to increase the number of users and get their reviews.

Chapter 3

Approach

To find the feature requests in the reviews, the review has to go under some mining process in order to classify the review type and then separate the reviews that contains the feature from other reviews. After classifying the reviews, we need to associate a feature or group of features to a specific requirement in the app.

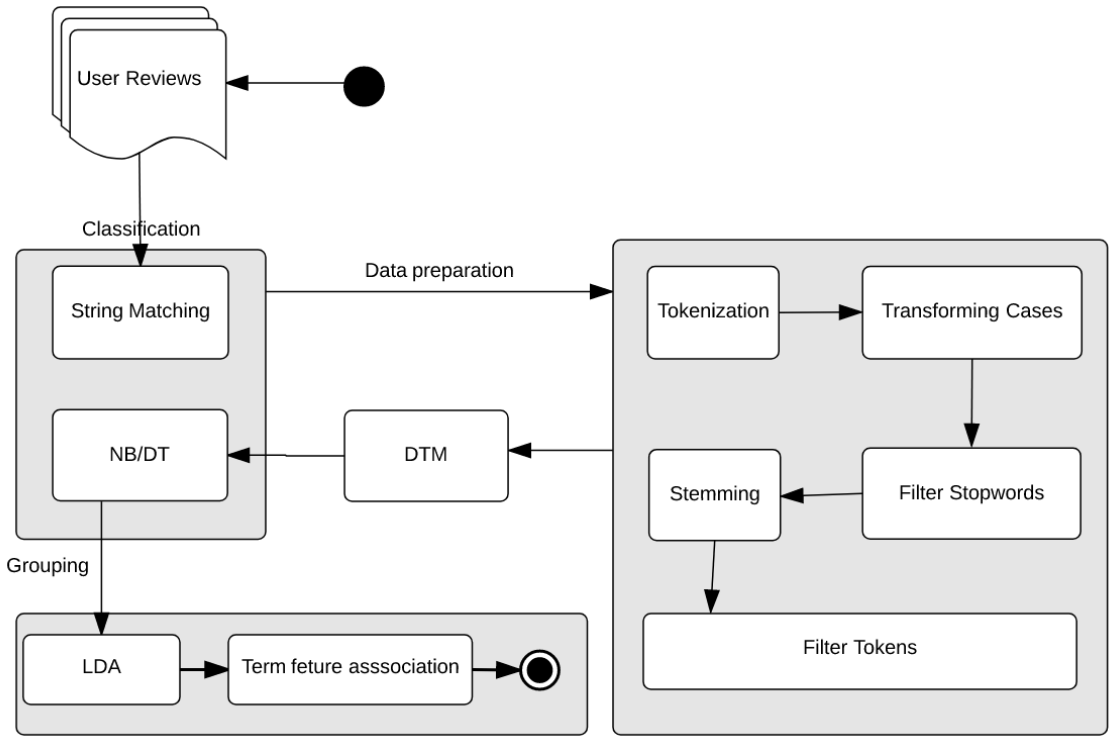

Figure 3.1: Approach overview

We started with collecting user reviews. Then the data undergoes cleaning in order to

transform the data into matrix. Then we constructed our classification model in order to classify the data and separate feature requests from others. After that we applied LDA in order to find common themes in the reviews, and finally associated these common themes manually with app features3.1.

For classification process, we have followed same steps described by Maalej et al. [20] when they worked on classifying the user reviews. However, we have selected a different model for classification after comparing the performance of different classification models. While Naive Bayes had higher accuracy in both Tools and Music category, DT had a higher accuracy than Naive Bayes in social category.

3.1 Raw Reviews

For the raw reviews data and their meta data we used two different datasets one that contained the raw reviews [30] and the second one [18] which contained information about apps meta data. The first dataset is a Postgresql database which contains several different tables. There are only two tables that is relevant to our study which are reviews and apps table. We compose our dataset from these two tables. The attributes which we selected are (appid, reviewid, raw_text) from reviews table and (appid, appname) from the apps table. We constructed a dataset from these two tables, however we needed one more attribute which was not exist in this database which is category. Therefore, in this step we have appid, appname, reviewid, raw_text. But we are missing the category attribute.

Next, we needed to match each app with its category. For this reason, we used the second dataset which is a database of 400,000 apps. We needed this dataset in order to match the names of the apps, and then fetch it is category from Google Play. This dataset contained these attributes(name, datePublished, numDownloadsMin, fileSize, packageName, price, aggregateRating, softwareVersion, ratingCount, dateCrawled, url) [18]. But the only attributes that we needed in this dataset are (name, url). The name of apps is to associate with our data with the app name, in other words is to make a relationship with our table, and the url is to use it in order to scrape the page url to fetch the apps category. We then

Table 3.1: Sample user reviews.

Review	**Type**
People who complain need to get better phones and tablets	Opinion
Please provide me with chat heads like facebook	Feature request
it always crashes! i always get an error that says try again. fix it please!	Bug report

wrote a scraper to fetch the category.

During the process of getting category of the apps, some of the apps were not available in play store anymore. Whenever, the script tried to fetch information about these apps, there was a 404 error. Since the category of such apps were not found, their reviews were automatically discarded in order to make sure that we only get the reviews that belongs to one of the categories.

User reviews contain many different form of data that would not be useful for the purpose of our study. For instance, the review might be an opinion or it might be in the form of bug fixes or in the form of feature request as shown in 3.1. There are also some other forms of noise, such as length of the token. For instance a user has posted a review and wrote "Aaaaaawwwwweeesssssssooooooommmmmmmeeeeee" and another user have written "Looooooooooooooooooooooovvvvvvvvvvvveeeeerrrr". These kind of reviews does not serve any purpose. One way for handling these types of reviews or terms in the review is by eliminating them after tokenizing the reviews because they will be in a form of single token. In other words the token that are very long can be discarded. These lengthy tokens originated either from the user review itself, or during data collection when the words shrink and become one single word.

The total number of apps is 137 from all different categories with 3,390,526 reviews.

Table 3.2: List apps and reviews for each category.

	#Apps	#Reviews
Social	9	643332
Music&Audio	8	89189
Tools	18	506423

We select three top categories for the purpose of this study. The top categories in which top rated apps belong to them according to Appbrain[3]. Therefore, after separating the reviews that belong to the top categories from others, the total number of remaining reviews is 1,238,944, which is 36% of the original number of reviews.

3.2 Data Preprocessing

The user reviews are in the form of sentence or a short paragraph. This data is noisy, it is incomplete and an organized. We want to transform these reviews to an understandable form through a process which is called data preprocessing in data mining. It is an important step in data mining, and it significantly impacts the quality of the output of mining process. Tasks in preprocessing includes Data cleaning, transformation, reduction, feature extraction, etc. Once the data passes this stage it will be ready to further apply other mining techniques such as classification, top modeling, etc.

3.2.1 String Matching

One of the simplest and most efficient ways to categorize a document is by searching for category specific keywords in the text. It is required that you have a list of keywords that would possibly identify that the document belong to a specific category. According to previous research [13] if a document contains one of these keywords (**add, please, could,**

would, hope, improve, miss, need, prefer, request, should, suggest, want, wish) it is most likely identifies the document (user review) as a feature request.
There are many different techniques that would help simplifying this process. For instance, using regular expression tools such as grep and providing it with the list of the keywords that we look for in the document. While previous process works well, but we had the data in a Postgresql database. Therefore, we used SQL keyword "LIKE" in order to match the keywords.

In order to further investigate whether the keywords are accurate or not, due to the time that takes to manually analyze the reviews, we examined 2000 reviews and found that only on review was a feature request. After applying this technique (String Matching) the number of reviews significantly reduced from 3,390,526 to 354,613 which is 10.45% of total number of reviews for all apps in the data including apps from different categories other than the categories that we have selected for our study. From the basic classification we can conclude that as compared to other types of reviews such as bug fixes, only few number of reviews are feature requests (if we assume that all of the reviews are counted as feature request regardless of noise data). In addition, the sql query on the raw reviews would match any reviews that contains the keywords or might be a part of a word. For instance, these words (address, addict) are starting with the word "add" and it is obvious that these words does not have the same meaning as "add" but the basic classifier would still retrieve them. However, we cannot remove them from feature request, as it might contain other keywords that would match feature request keyword list.

As it can be seen in Table 3.3 the least feature requests in apps belongs to Tools category, and the most features request belong to Music category. This might be due to the fact that there is more interaction with music apps than tools apps. Users usually stay with using music apps much more than tool apps[15]. According to a report by comSource Music apps(cumulative) are ranked second among the apps that are most used daily by smartphone users, which comes directly after social apps [1].

Table 3.3: Number of reviews before and after String Matching.

	#Before	#After	%Remaining Reviews
Social	643332	73361	11.4%
Music & Audio	89189	13622	15.27%
Tools	506423	26293	5.19%

3.2.2 Natural Language Processing

In previous steps, we filtered the reviews in order to reduce the huge number of reviews, so that it can be more manageable for further processing. However, we have not worked on the raw reviews yet. That means the reviews are still as same as the time it was posted by the user. We need to prepare the data for further processing such as classification and topic modeling such a way that the mining models can process it. For example, in mining we need to feed the model with a set of documents(corpus). Here in our study user each user review is a document, that will be the input to the mining models used in the study. However, the documents need to be processed in order to produce an organized form of data and feed to the classification model known as corpus. We apply natural language techniques in order to produce the corpus.

Natural language is a term used to distinguish formal languages such as computer languages(Java, C++, etc..) and mathematical expressions from human language[14]. NLP is a term used to describe the function of any hardware or software part computer that can manipulate human languages and process it such a way that it can be understandable by computing machines. There are many different NLP techniques and each of them are used for a different purpose they can also be combined in order to manipulate a text depending on what the text is needed for. Some NLP techniques are (Sentence delimiters, Tokenizers, Stemmer, Taggers etc.) [14].

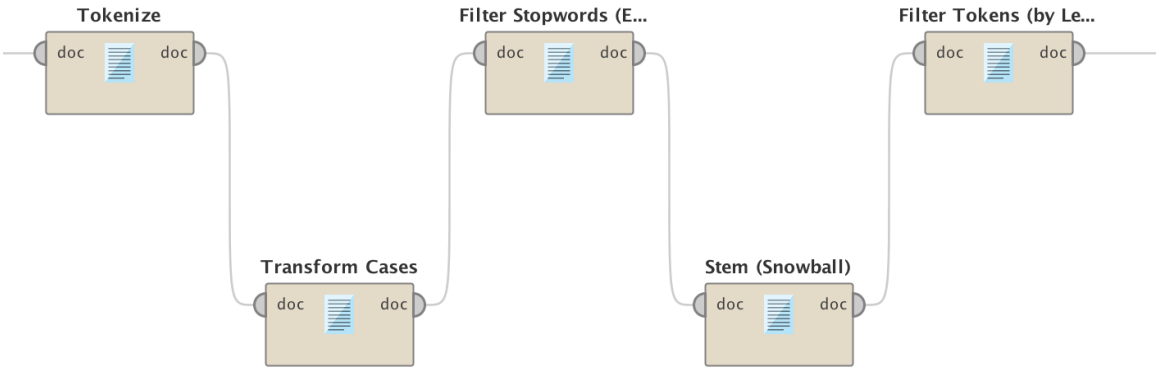

Figure 3.2: Processing reviews for vector generation

In this study we have used the techniques that would syntactically help improving the accuracy of our classification method. The NLP techniques we used in our study are tokenization, stopword removal and stemming which is used in preprocessing technique in similar work [28]. The only difference is that we have incorporate transforming letter cases to small in order to make sure the words match regardless of their cases.

Tokenization Users express their opinions or post the feedback in a sequence of (characters or symbols or phrases) that eventually form a sentence or paragraph. In text processing, these reviews has to be segmented in to linguistic units that would help the mining process better understand the entity. The linguistic units include words, punctuations, numbers, alpha-numeric, etc. The process of segmenting text into linguistic units is called Tokenization [27].

In this step we input the data which user reviews, and it will generate the tokens for each review. We have used non-letter as the tokenization mode for the reviews. It will mostly split the words by the space as no other non-letter characters are used in reviews as space.

Stopwords are some English words that are repeated and used frequently in sentences. They are used to connect words, sentence, etc. between a group of person communicating with each other. It would be hard for audience to understand and follow the sentence without the stopwords. However, the existence of those words in user reviews does not

serve the process of classification. That means removing these words will not affect the outcome of classification. However, since we are specifically looking for feature requests, and we previously have mentioned that we have made the string matching on a set of words, therefore we made sure that those keywords have not been removed from reviews. We have customized the stopword list such that it does not remove our given words. In addition, we added other keywords that is not formal English language words but still used in online chatting or reviews such as “urs”.

In a text or a user review a word can be used in different forms within different reviews. In one review a word might be used as the present form of the verb, and in other might be used as past form or a noun form. For instance, the base of add, added, addition are all “add”. Therefore, unifying different forms of words **(Stemming)** into their base would give a higher frequency of the term or word usage. There the purpose of using stemmers is “to associate variants of same term with a root form. The root can be thought of as the form that would be normally be found as an entry in a dictionary” [14].

Stemming is the process of reducing all words that have the same stem or root to a common form [19]. The difference arises due to Morphological Variants which is the most common. For instance (computer, computational, computers, computing) [33]. Besides that, there are some other reason for the difference, such as mis-spelling, abbreviation, and valid alternative variant (e.g ‘apologise’, ‘apologize’, ‘analyse’, ‘analyze’).

The purpose of using stemmers in this study is mainly to increase the frequency of the words used in the reviews. In addition, it will significantly decrease the dictionary size because the large number of reviews as well as different expressions lead to the generation of large dictionary, this is because users are free to write what every they want, there are not any restriction on the writing, or preexisting list of choices for feedback.

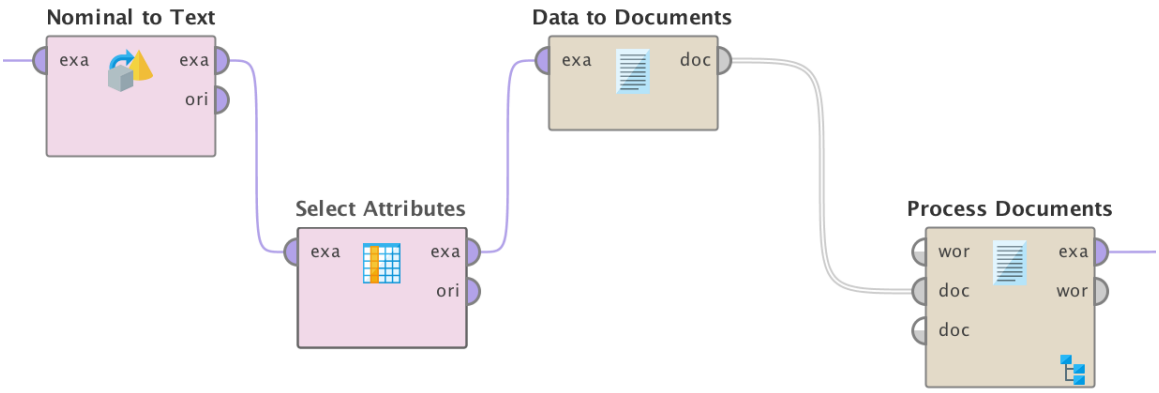

Figure 3.3: Document matrix process

3.3 Document Term Matrix

There are two types of matrix in terms of the organization. The first one is "Document Term Matrix" in which the attributes represent the terms and the tuples represent the document. The second one is "Term Document Matrix" which is opposite to first, in which the attributes represent the documents and the tuples represent the terms. Due to the large number of reviews and small number of attributes as well as the mining techniques used in our study, we generate DTM instead of TDM. As a result, the attributes in the generated matrix will represent the terms used in the user reviews and the tuples represent the user reviews.

There are also different types of vector creation methods which are Term Frequency, TF-IDF, Term Occurrences, Binary Term Occurrences. We have applied the term frequency to create the vector. In addition, we made some tweaks to the methods by ensuring that the most frequent or infrequent terms are ignored from the word list. The values are 3 for infrequent words and 500 for frequent word list. We also kept the 'reviewid' with the matrix for future reference whenever needed.

3.4 Classification

It is a two-step process which is used to predict class labels for a given data. It starts with the learning step where the classifier (classification model) is constructed and then the next step is classification, which basically is using the constructed model in first step to predict class labels for a given data.

One of the examples that used to describe classification, is to predict whether an email is a spam or not. Is the email legitimate or it is a junk email. The classifier immediately does some calculation in order to determine the type of the email. There are some keywords and symbols used in the spam emails, whenever the classifier identifies these keywords in the email it will predict it as a spam email. There are many applications of classification such as target marketing, fraud detection, manufacturing and medical diagnosis.

In this study, we use classification to predict the review type similar to the study that have been conducted by Maleej et.al [20]. We are specifically looking for feature requests in the reviews. Therefore, any other type of reviews will be discarded. This process will further break down the huge amount of review data. As the data size reduced it will be easier to control and faster when further processing is done on the data.

3.4.1 Learning Phase

In the first step or classification, the learning step, or the training phase, we build the classifier which describes a predetermined set of data classes. A tuple, X, is represented by an n-dimensional attribute vector, X = (x1, x2,..., xn), depicting n measurements made on the tuple from n database attributes, respectively, A1, A2,..., An.1 Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute.[Book]. This method of training is called **Supervised Learning** because we have provided the dataset with the class label attribute which basically determines whether the given tuple is a feature request or not. The classifier will then form the rules based on the attributes and the provided weights for each attribute and whether it is a feature or not. On the other hand, in **unsupervised learning** the class label is unknown and the number or

youtub	youv	yrs	zenfon	zero	zone	reviewid	feature_request
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	1
0	0	0	0	0	0	gp:AOqpTO...	0
0	0	0	0	0	0	gp:AOqpTO...	0
0	0	0	0	0	0	gp:AOqpTO...	0
0	0	0	0	0	0	gp:AOqpTO...	0
0	0	0	0	0	0	gp:AOqpTO...	0
0	0	0	0	0	0	gp:AOqpTO...	0

Figure 3.4: A portion of labeled data

type of classed might not be determined prior to the mining process. This method is also called clustering, in which the model tries to identify the commonality between the tuples and group them together.

Sampling

According to Maalej et al. [20] 150-200 reviews is adequate for the training set for each type of review [20]. Since we are working on more than one app, and since each app has different specification and features, we chose the top three apps from the top three categories of apps according to Appbrain. For each app we have labeled reviews until we got to a point that each app has at least 150 features. That means for each category the minimum number of reviews that is labeled as feature request is 450. Because we are working on only feature requests, therefore the ratio of feature requests to others is small. We then selected the reviews to label after ”String Matching” in classification in order to make sure that the review contains words that could possibly classify the review as feature request[20]. In other words, the reviews are still raw reviews without any preprocessing. Figure 3.4 shows a portion of the sample data it does not contain all attributes that defines the feature request due to the large number of attributes. We have set the “feature_request”

as the label of the dataset, and the "reviewid" as the id of each tuple in the dataset for future reference if needed.

Classification Model

We have applied different algorithms in order to build the model and then chose the one that would best fit our need for the feature selection. According to previous study [20] Naive Bayes is better than other classifiers which are Decision Tree (DT) and MaxEnt [20] when comparing the accuracy of these models. However, this was not the case with all categories in our study as the accuracy of models were slightly different in different categories. The reason might be due to the fact that we have combined the reviews of multiple apps. Therefore, we applied and tested both Naive Bayes and Decision Tree which are very popular binary classifiers and also have high accuracy to predict features as described in a previous study [20]. The output and details of both classifiers are discussed in this section.

Both Naive Bayes and DT are very popular classification models. Naive Bayes is based on Bayes Theorem which based on the assumption that predicators are independent on each other. In other words, it assumes that the presence of a particular features is independent on the presence of other features. Naive Bayes is simple and easy to build and does not require large training set. On the other hand, DT is incrementally developed by breaking down large sets of data into subsets and the result is a decision tree with two types of nodes which are decision nodes and leaf nodes. A decision node, has two or more leaf nodes which are either represent a decision or classification node.

We applied both techniques on the same dataset and then evaluated the results for each algorithms and compare the performance of the two different algorithms using standard metrics which are precision and recall. Precision is the fraction of user reviews that are correctly classified as feature request by the classifier while recall is the fraction of correctly classified reviews. The equation for precision and recall is as follows:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad F1 = 2{*}\frac{Precision * Recall}{Precision + Recall}$$

TP is the number of user reviews that are classified as feature request by the classifier.

Table 3.4: Accuracy of classification techniques using DT and Naive Bayes for each category

Music			
	Precision%	**Recall%**	**F1%**
DT	84.06	42.34	56.31
Naive Bayes	65.43	77.37	70.90
Social			
DT	59.3	95.93	73.29
Naive Bayes	65.33	79.67	71.79
Tools			
DT	77.55	37.25	50.33
Naive Bayes	67.37	62.75	64.98

FP is the number of reviews that are classified as feature request but they are not actually a feature request. FN is the number of reviews that are not classified as feature request, but they are a feature request. In addition to precision and recall, we calculated F-Measure, which is a measure of accuracy which considers both precision and recall.

Based on the results as shown in Table 3.4 we have applied different model for different categories. We can see that Naive Bayes performs better in both music and tools review however DT performs better in social reviews. Therefore, we applied different techniques for different categories depending on the algorithms performance.

There are many evaluation metrics used for the predictive accuracy of a classifier. These techniques include Holdout and random sub-sampling, cross-validation and bootstrap methods. In this study we have applied the cross-validation technique in which we split the data into two random tuples of total 0.75 for training and 0.25 for testing. Table 3.5 and Table 3.6 are the results for each algorithm for the reviews of social apps

Table 3.5: Confusion matrix of Tree of social reviews

	true 1	**true 0**
pred. 1	118	81
pred. 0	5	52

Table 3.6: Confusion Matrix of Naive Bayes of social reviews

	true 1	**true 0**
pred. 1	98	52
pred. 0	25	81

3.4.2 Prediction phase

Now that we have constructed the classifier in the first step of classification process, it is time to use the constructed model to predict the user reviews in order to determine if the user review contains a feature request or not. We have run the model on the three different categories separately as the model construction was also done for three different categories separately.

Figure 3.5 shows that the 'social' and 'tools' category have equal ratio of feature requests to the reviews which is 2% and music category reviews contains the most feature requests which is 8%. This shows that the result of the string matching of feature requests is directly proportional to the results of classification process. That means the difference in category does not necessarily affect the review types of users. However, if comparing reviews under different category, music contains most feature request than both social apps and tools apps.

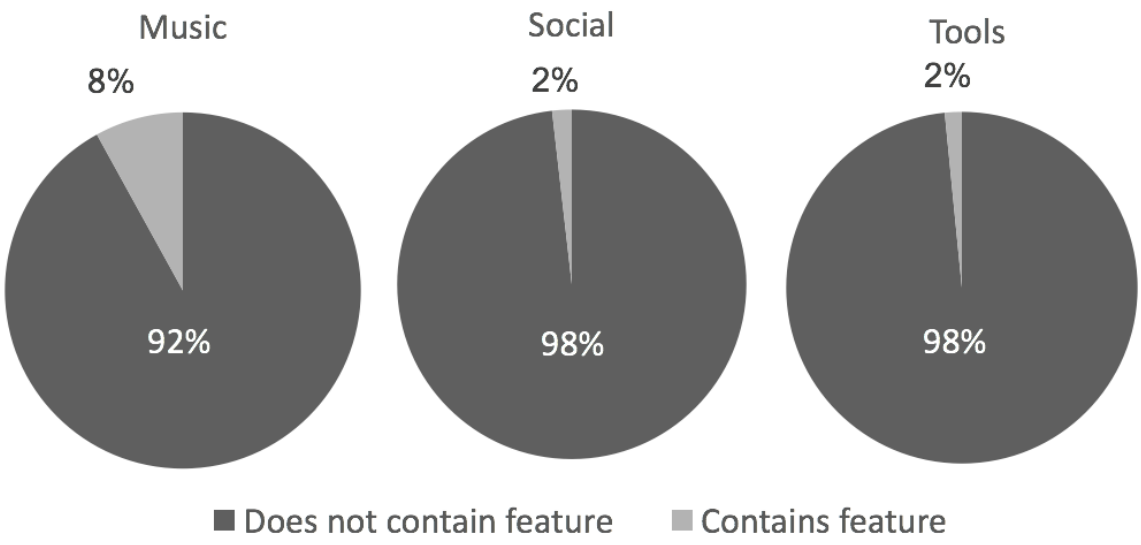

Figure 3.5: Predicted features request by the classifier

3.5 Extracting Features

In this step, we are discussing the way we want to extract the feature in the user reviews. In a previous step we worked on separating the reviews that contains feature requests from other types of reviews. From now on we are working on the reviews that are only predicted as "1" by the classifier. The difference in the reviews in this step from the raw reviews is that we have the matrix instead of the raw text. That means we do not need to repeat the same steps that we prepare the data for the classification.

Usually multiple users request the same feature in the app. However, they use different terms or statement to express their request. For example, some Spotify app users have requested adding a timer to the app so that the timer stops the music player after a specific period of time. Table 3.7 is a sample of different expressions used to ask for adding a timer to the app. It can be clearly seen that the length of reviews is different, and other terms used in the reviews in addition to the feature request. However, there are some similarities between the sentences that are used to state the feature request.

We use topic modeling methods in order to find topics (features) of the reviews. According to Zoe Borovsky topic modeling is "a method for finding and tracing clusters of

Table 3.7: Same feature request sample (Timer).

Cool but, needs a sleep timer so that when I wake up half way thru the night I don't have to go on my phone and turn it off.
It's good but you should make a sleep timer
I love it and I'm starting to prefer this over Pandora and this app needs a sleep timer that shuts the app down for those who live off of limited data
Can you guys add a sleep timer? And make it so you can listen to music while you sleep and it'll automatically turn off. That'd be amazing! I use the piano music to sleep and that'd make it better.
Please add a timer so I can sleep with music but don't kill my phone battery life span too fast!

words in large bodies of texts" [32]. In our study user reviews will represent the bodies of text and the words used to request the feature represents the terms that are used to form the topics. We use the LDA baseline model in order to identify topics in the user review corpus. LDA is a "generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document" [7]

LDA can also be used as the learning model when classifying documents. The model needs to be trained and tested same as other classification models. The reason why we wanted to use in Topic Modeling is to extract the requested features intuitively. We do not know all users feature request otherwise we would be able to list them all and there will be no need for mining the user reviews.

Table 3.8: Topics in music apps reviews

	Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**	**Topic 6**	**Topic 7**
1	miss	chromecast	hope	repeat.	cant	sleep	internet
2	replay	support	upload	edit	see	timer	version
3	album	artist	profil	sing	pls	list	cool
4	equal	line	chang	mobil	bring	guy	connect
5	possibl	devic	disappear	order	anymor	set	wifi
6	happen	someth	pictur	comput	cach	mode	put
7	art	direct	function.	hit	perfect	pandora	lot
8	definit	move	annoy	singl	otherwis	includ	pretti
9	help	android	widget	gave	read	amaz	cloud
10	info	know	mayb	fav	dont	enjoy	desktop

3.6 Term Feature Association

The last step in the approach is to associate the terms to the features. We need to match the words that users use in the reviews to a feature in the application. For example, “zoom” is a request to add zooming functionality to the photos and images.

We have manually associated the terms to the features. The automatic association is out of the scope of this research. Some features would replace an essential existing feature in the app, while others are an improvement to the basic feature in the app. For example, some users request material design, if applied it will replace the basic design or theme used in the app. However, if a user requests “Shuffle” feature it is an improvement to the current music or video player in the app, since we do not work with how they affect the app functionality in the app.

Table 3.9: Top requested features

Music & Audio	**Social**	**Tools**
Replay	Pause	Theme
Shuffle	Replay	Applock
Upload	Copy Post	Block
Sleep	Translation	Notification
Lyric	Zoom	Schedule
Theme	Material Design	Kill
Offline	Live	RAM-Clear
Widget	Repost	Backup
Premium	Rotate	Vault
	Unfollow	Fingerprint

3.7 Summary

In this chapter, we discussed the process of mining user reviews and extracting the feature requests. The user reviews undergo some process in order to prepare it for classification and topic modeling. We started first with basic classifier, which we used SQL queries in order to specifically filter the requests the contains the keywords that would possibly classify the review as a feature request [13]. Then we generated the matrix from those reviews.

We then trained and tested Naive Bayes and Decision Tree in order to choose that one that performs better for each category of the apps in order to use it in the classification process. We found that Naive Bayes performs better in music and tools category, and DT in social Category. We then applied topic modeling technique in order to find the common themes in the feature requests.

Chapter 4

Implementation

There are many frameworks and tools that help developers create apps. Each tool has different approach for its construction and using it for creating new apps. Depending on the developer's preference of language, paradigm, ease of use, community or sometimes joining an ongoing project help them decide which framework to choose. For example, some developers prefer separating HTML from JavaScript implementations rather than writing HTML code inside JavaScript, for this reason they pick VUE rather than ReactJS. Sometimes performance issues with some frameworks make developers migrate to other framework. But in general, most frameworks for a specific language can handle all tasks that is handled by frameworks of same language.

So far, we have been able to identify the features that are requested by users. But sometime finding the requests is not enough for developers, as they might lack experience to integrate the feature to the app or it might take a long time to implement it. Furthermore, new developers do not have enough information about the packages that are available for the frameworks that they use. In addition, there are usually more than one package available for a specific feature. They will have to spend some time, to see and compare which ones are better than the others.

There are some metrics that would help developers on which libraries they should use in their apps. For example, if the package is open source and hosted on Github, they can check number of stars, watchers, and number of open issues as well as forks to decide which package to use. If it is hosted on npm, number of download helps a lot. There are also other package managers, for instance composer in php. However, when developers work on the

apps, it would be better for them to focus more on development rather than these secondary time consuming tasks. Therefore, if the process of associating the requested features to the libraries automated it will save a lot of time for developers.

When the framework or the tool that they use, automatically suggest the libraries that they need to include in order to integrate the feature to the app, then the developer can start immediately diving into the library api and learn how to use it inside their apps. It also makes much easier for a team of developers to collaborate development. For example if they use npm as their package manager, when the framework that they work on, automatically includes the package in the app, it would write to the package.json file the included library, then this file can be shared with other developers to start working on developing immediately.

4.1 Challenges

There are some challenges with completely automating the process of including the features inside the apps. Some of the challenges includes:

- Getting all functions of a package can be cumbersome. There is not specific guidelines about describing the tasks that each package can perform. Some developers use bulletin to list all functions. Others use one paragraph to describe them, some other reference to an external link to browse the functionalities. Meanwhile, the functionalities cannot be extracted through scanning the code as developers follow different naming convention and have different level of programming skills. For example, a package is used in order to integrate authentication to an app. Some package provide authorization with it, while others provide only authentication and requires another package to provide authorization.

- Some open source packages are discontinued after a while. Although, they usually provide links to alternative packages, however it is not always the case.

- Some libraries need to be extended in order to provide all functionalities. Same as the previous example, some packages are very popular and are very stable, however they do not provide all functionalities. For example, if it provides only authentication, there must be a way to extend or create a separate module for authorization. Therefore, the framework should contain all these details so that the developer does not spend time checking for all package functionalities and dependencies.

- Providing updates to the packages is also one of other challenges. For current project this might not be a big deal, however when they switch to another app they might face some issues because the package api might have been changed. Some functions are renamed, others are added and some others are removed. Therefore, the framework should be able to track these changes and inform the developer about changes or display the change log.

  There are also other challenges that arise from using package managers to use packages inside an app. For example, how to check for security issues and vulnerabilities inside a library and some issues with performance as well as detecting malicious libraries.

4.2 Tools

The framework is composed of several tools but the main tools are described below.

- **VUE**: is one of the popular JavaScript framework for building user interface. It is lightweight and very fast mainly because it interacts with DOM. It separates the views or template from the js implementation which makes is easier to read and maintain the code. It gained popularity after it was introduced as the primary UI framework used by Laravel (PHP framework). We used VUE in order to have some interaction with the pages or part of the application that we will be creating and because of the packages that have created by its community. It is all related to

experience and preference of the developers, otherwise other tools or packages or frameworks or even native js script can be used.

- **Babel:** As web applications are getting popular, more application are either migrating the entire application or a part of the application to web. The native JavaScript code is hard to maintain due its readability, especially for large projects when it is required to write thousands of lines of code. For this reason, there are some tools or packages that help writing js scripts in a more readable and understandable fashion. Although this is helpful for developers, but they are not browser-friendly tools. To solve this problem, transpilers come to place. Transpilers basically transforms the code to native js script so that they can be interpreted by browsers.

  Babel is one of the most popular transpilers that converts JavaScript to JavaScript that are supported by most browsers. The output generated by Babel is human readable thats one of the reason for its popularity. The main reason we have used Babel is because of the VUE. Otherwise, if it is a simple application these transpilers are not needed.

- **Webpack:** When developing web apps, many packages and libraries are used. Each library also might have some dependencies. Therefore, it makes a daunting task and hard to include all required files and images in the application and sometimes the application needs to make multiple requests to include all files. Webpack bundles all js modules and their dependencies with all static files and puts them in a dependency graph which allows accessing the files dynamically through the code.

- **Gulpjs:** There are some repeated tasks in web application development which is time consuming and tiring, because it has to be repeated several times. For example, refreshing browser on making changes, compiling sass and less, module bundling, and copying the files to and output directory. These tasks and some others can be automated by using gulp. Beside the common operation with gulp we also use to copy the main js file to the mobile project, so that the mobile project has latest version

of the js file.

- **Cordova:** Apache cordova is a platform that helps building native mobile apps by using web technology such as html and js. Web pages cannot access device functionalities unless through an interface when building native apps. Thus, with Cordova developer get access to the device functionalities and the native app is generated by Codova.

In addition to these tools, we have also used vue-cli and vue-awsome. We used vue cli in order to generate apps for both mobile and web using our template. vue cli generates vue projects based on the templates with predefined features with customized functionalities. We have created our own template in order to load the features that we collected from user reviews.

Vue-awesome is a Github repository that contains list of the most popular vue packages and their description. This is specifically important for including the packages based on the user requests. For instance, many users have requested material design in the apps. The most popular package for material design is listed in this repository and it can easily be integrated into the project.

Each of these tools has different purpose and performs a different task. However, once they are configured to work together, they can help development much faster and help reduce the repeated development tasks.

4.3 Components

We divide the system into two major components as shown in Figure 4.1. The first component which is “Review Mining” handles all mining process while the second one is responsible for generating the app and including the packages in the apps.

The user reviews are fed in to the “Mining Reviews” component. Then it will undergo the mining process in order to extract the user reviews that contains feature request. It will then separate the feature request from the rest and discards other reviews. Then it will find

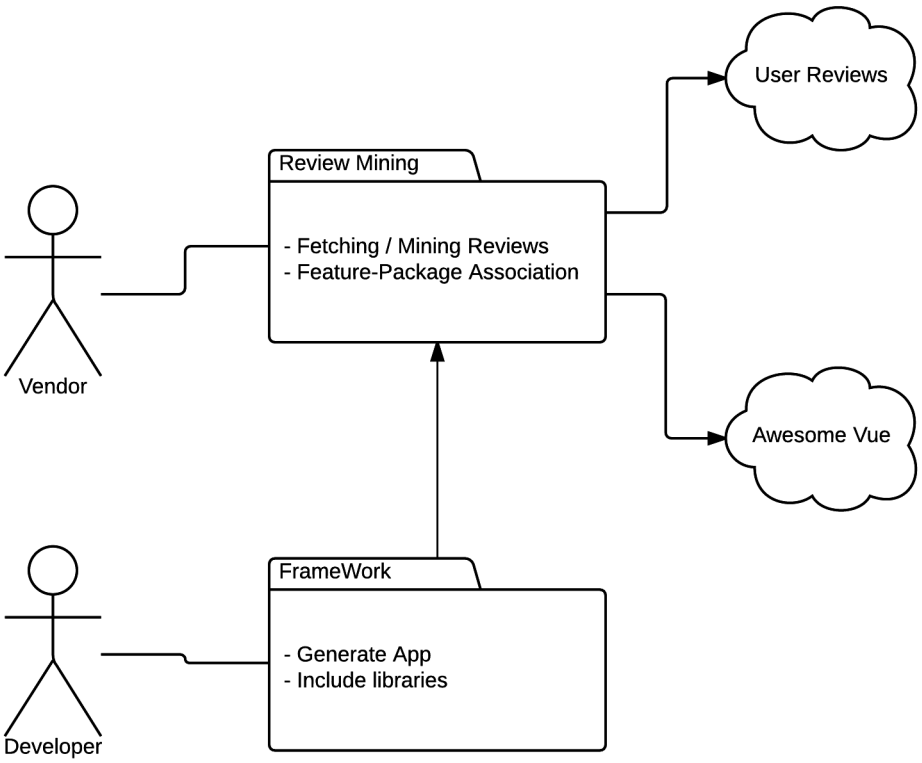

Figure 4.1: System user case diagram

the topic of most requested features. The details of the operation are described in Chapter 3. In this chapter we will focus mainly on the "Framework" component of the entire system.

The framework handles all app related tasks. For example, generating projects for both Android and iOS using Cordova. Then gulpjs will handle automating all the other tasks such as packing using webpack, copying new generated files into the cordova project. The framework is based on vue-cli which is a command line tool for vue framework.

By using the framework, the developer has just to focus on the app content rather than searching for packages in spending time with configuration and doing repeated tasks.

4.4 Workflow

The command line interface will walk the developer through steps needed to generate the app and directly include the features inside the app as shown in Figure 4.2. The steps for creating a new project is as follows:

1. Developer runs the command line interface with our webpack template to create the project. The developer needs to provide name of the project and author name.

   ```
   thesis <template-name> <project-name>
   ```

2. Then the user selects the platform that they want to create the app for. Since it generates a hybrid app the supported platforms for now are Android and iOS.

3. Developer will be prompted to select app category of the app. currently there are only three app categories which are music, tools and social.

4. After selecting the app category, the cli will show the features that are most requested by users and sort them by most requested feature. Then the developer selects the features that they want to include directly in their project.

5. Once these features selected, the framework will automatically fetch the packages that are associated with features through Awesome-Vue repo. Then the packages and

their dependencies will be installed and compiled through npm and webpack using gulp. Then it creates project for the selected platforms.

6. Whenever the developer writes code, he/she just need to compile the project by running gulp. It is configured so that it performs all required operation to compile and copy necessary js file to the correct path. If the developer does not want to repeat this operation, he/she can just run gulp watch so that is watches for the changes in the directory and performs operations in the background.

4.5 Summary

In this chapter we propose a framework in order to integrate the extracted features from the user reviews to a real functionality in the app. The framework is based on a command line tool which is combines the functionality of a number of tools that helps the app development process.

When the developer wants to start a new app, the command line tool prompts the developer for the features that he/she wants to include in the app. After choosing the features, the framework will directly create a hybrid app with the features selected previously selected by the developer. This way the developer do not have to spend time looking for trending features as well as looking for particular packages that helps in development process.

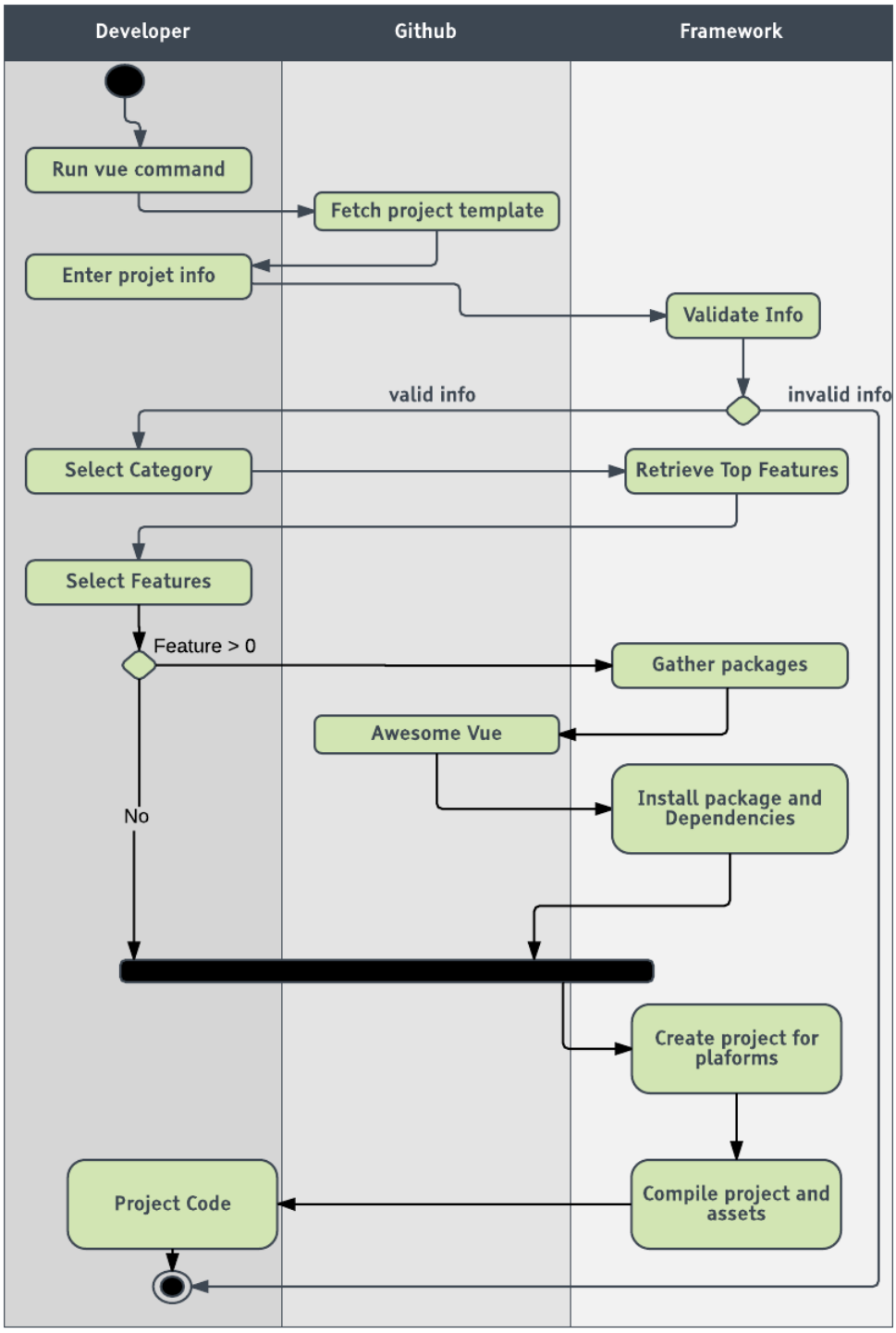

Figure 4.2: Activity diagram for creating a new project

Chapter 5

Analysis

The results of the approach will be presented in this section. Firstly, we explain the details of the survey and then present the results which is essentially the data collected from the participants during the survey. Then we discuss the results and analyze the collected information from the participants.

5.1 Survey

After extracting the features from each category (Music, Social, Tools), we presented these features to a group of developers that had different level of Android app development expertise. Here we discuss the structure of the survey and how we performed it.

5.1.1 Structure of Survey

The survey consists of three parts:

1. We asked the participants in order to create a social app because social apps have highest unique visitors [1]. Therefore, there is a higher chance of using social apps by participants than music and tool apps. We described the app we were looking to create and then listed some basic functionalities in the app such as login and posting. Then we asked them in order to list the five features that they wanted to add to the app starting from most important to list important.

2. The participants were redirected to second part after finishing the first part. On the left side of the survey, it contains the features that are most requested by user of social

apps from which we extracted. On the right side, it contains the features that they filled out before. The purpose is to show them their previous input, if they forget it, or if they still wanted to use exact same features.

3. After finishing the second step, the participants were asked to fill out a simple questionnaire in order to collect information about their experience with app development.

5.1.2 Participants

The total number of developer participated in the survey were 17. We are confident in our number of participants because this is a similar number as have been included in an analogous work [9]. The participants had different background with app development from beginner to intermediate and even expert that have published several apps and has high number of installs in play store. 7 of the participants were graduate students who had little background in Android development by either taking a course of involving in an app development. The remaining 10 participants had intermediate to an expert level in app development.

In terms of the experience and number of published apps based on the participants response, 7 of them have not published any apps. While 3 of the participants have published 1-3 apps and the other 7 participants have published 4+ apps. In other words, 59% of the participants have published at least one app in play store. We also asked participants specifically about the time they have been developing or working on Android apps. 4 of them has 5+ years experience, while 6 of them has less than one year experience. As demonstrated in Table 5.1, the rest of participants have 1 to 5 year of experience.

The participants were also asked to provide some information about how they collect information about features and how they decide on including the features. 82% of the participants said that, they are referring to other apps whenever they want to include features. Meanwhile, 88% of the participants claim that they include features in their developing apps based on their usage experience. For example, if the developer wants to create a social app, they try to copy some features from other social apps, such as Facebook and

Table 5.1: #Published apps by participants

#Developers	#Published apps
7	0
3	1-3
2	3-5
5	5+
	#Years of experience
6	¡1
7	1-5
4	5+

Instagram. 7 of the participants also surf internet and 8 of the participants decides on which features to include in the app based on the easiest one to implement.

Finally, 76% of the participants claim that they add features to the app, if the features are requested by users while the rest of the participants claim that they do not respond to the user requests and the user requests do not impact how they implement the features in their apps.

5.1.3 Results

In this section, the discussion will be mainly focused on the second and third research questions, because the results of first question have been merged into Chapter 3.

Firstly, we want to study how developers react to the features requests by users of similar apps to see whether developers are willing to include the latest features, or they stick with the features that they previously had in mind. The reason we want to do that is to introduce a new way such that it would help developers directly include these features in the app. In the second step, we will study how these features help developers prioritize the features. When the developers were asked to write the features, we also asked them to write them in order from most important to least important to include in the release of the application.

In the first step of the survey, we asked developers to write down five most important features that they would include in a social app. In the second step we asked same question, but this time, we have showed the developers the list of features that was most requested by users of similar apps. We report the results of the survey in both in terms of features usage, and feature priority which will address the second and third research questions as discussed below.

1. **What is the impact of extracting features on informing developers about features they are unaware of? (Feature Usage Frequency)**

   The list of features that is provided to the users composed of nine features. In the first step of the survey the list was not visible. As it can be seen in Table 5.2, in the first step, these features were mentioned only 4 times, while in the second step it raises to 48 which is 12 times more than the first step.

   Regarding the usage by participants, from the total of 17 participants, only three of the participants include the features in the first step that was with 1,1,2 features respectively which is 17%. On the other hand, only 11% of the participants stick with the same features, or did not include the features even after seeing the features requested by the users.

2. **How does extracting features of apps in same category help developers in prioritizing the features for the initial release of the apps? (Prioritizing)** In previous section we discussed how frequent these features are integrated to the apps by developers. As we have mentioned before, the features that are most requested by users are presented to the developers from most requested features to least requested. There are three cases to consider when studying how developers deal with these features in terms of priority. First comparing new features to old features based on the frequency of usage before and after seeing the list of requested feature. Second, comparing similar feature from the list of old and new features. Third comparing new features among each other based on the frequency of requests. The comparisons are

Table 5.2: Feature usage before and after showing list of users requested features

Features	Step One	Step Two
Pause Video	0	4
Replay Video	0	5
Copy Post	0	3
Translation	1	9
Zoom	1	5
Material Design	1	9
Live Video	0	5
Repost	0	6
Rotate Image	1	2
	4	48

described as follows:

(a) **New features with old features based on usage frequency** The developers provided 5 features based on the priority of the features as we asked them to write them in order from most to least important. In first step, participants included the features that were requested by users only 4 times, while this number increase by 12 times in second step as shown in Table 5.2. Since in both steps, participants were allowed to provide only 5 features, and as we have seen that the change is 12 times, that means they have replaced 53% of the features with new features. Therefore, we can conclude that these new features will be among the developers' priorities.

(b) **Priority of features with respect to frequency of user requests** As we mentioned before, the list of feature in Table 5.2 has been sorted from most requested to least request features. For example, "Pause Video", was most request feature and "Rotate Image" was least requested feature. The features that are participants priorities are "Material Design, Translation, Zoom, Replay Video,

Table 5.3: Feature priority list

Features	**1**	**2**	**3**	**4**	**5**
Pause Video	0	0	1	1	2
Replay Video	1	0	0	3	1
Copy Post	1	1	1	0	0
Translation	1	5	2	0	1
Zoom	1	0	2	1	1
Material Design	6	1	0	1	1
Live Video	0	1	1	2	1
Repost	1	1	1	2	1
Rotate Image	0	1	1	0	0

and Pause Video" respectively. For instance, Material design was mentioned 6 times as the first features by participants, and "Translation" comes as the second most important feature. Surprisingly, while "Pause Video" was most requested by users, it was least important to the developers in terms of both priority and frequency as it is listed in that last feature.

5.2 Threats to Validity

Several threats to validity exist in the way data is collected and analyzed. We were only able to obtain moderate results using our classifiers. This indicates that the features we suggested to participants may have come from reviews incorrectly categorized as containing a feature request. This is mitigated by the fact that LDA attempts to cluster by semantic similarity. That is, LDA cluster documents by common topic (i.e., a feature), meaning that the minority of documents that were mis-classified will either simply not cluster or will form a singular topic (in the case where they are similar). Either way, it is not likely our results will be thrown off by this. Additionally, our human participants would likely ignore any suggestion we gave that did not sound like a feature (i.e., a bad feature suggestion

generated by LDA).

The participants in our study were not experts in designing social apps. This makes it difficult for us to generalize the results to those with well-established apps. However, determining what users want is a large problem for both those entering a market (i.e., with a new app) and for seasoned developers (i.e., with a currently available app). Our proposed technique lowers the barrier of entry to help new app designers discover what their competitors lack or to assist seasoned developers determine what future features they can focus on developing.

We only performed the survey portion of our study on social media apps because they have the highest unique visitor count [1], so we assumed our participants would most likely have more experience with social media apps than with apps of other types. Additionally, we were only able to find 17 suitable participants for our study. While more participants would have optimal, we are confident in our number of participants because this is a similar number as have been included in an analogous work [9].

Chapter 6

Conclusions

In this study we have analyzed the reviews posted by users in Play Store. The goal of this study is to help developers plan the features they include in the next release of their apps. In addition, they can use our proposed framework, to directly include the most requested features in the app. In terms of the venders, they can get benefit from our findings about which mining technique and methods would work for specific category in play store. They can follow the approach in order to build their custom recommenders that recommends the inclusion of specific features for a specific app.

We started with collecting the raw reviews. Then applied preprocessing techniques to prepare the data for applying other mining techniques. We applied to different mining techniques which are Naive Bayes and Decision Tree which are two of the most common classifiers. We found that while Naive Bayes have a higher accuracy than Decision Tree, but it was not the case with all different categories, as Decision Tree had higher accuracy in Social review. After applying these classifiers and separating the reviews that contains feature request from the others, we applied topic modeling techniques to find common themes inside these reviews.

We then conducted a survey and presented these common themes in the form of features requested to the participants in order to see how it would help them plan and prioritize these features in their apps. We found that, while most of the participants included the feature in their apps, the frequency of the requests did not impact the decision of developers to include the most requested ones.

In order to further help developers with developing their apps, we introduced a new

framework that embeds the most requests features directly inside the framework. Now, with the framework the developers can directly include these features, and the framework will take care of including all the required packages and their dependency inside the app.

6.1 Current Status

Until now, we have constructed and trained the classification model as well as the topic modeling technique. We have used two different classification model which are Naive Bayes and Decision Tree. As of the topic modeling technique we have used LDA. Using these mining techniques we were able to extract the features from raw reviews and present them in a complete set of comprehensive list of features, which encompasses the first component of the entire automation process of extracting and using these features in the app.

As the second part of the proposes technique, which is the app framework, we have integrated the features for the three different categories used in this study. There are about 10 features in each category. The developer can now use and navigate through the features used in the framework and generate apps for both Android and iOS apps that contains the features which are selected by the developer.

6.2 Future Work

1. **Increasing number of categories**

   Due to large amount of reviews and limited space and time on the machines we have used in this research, we have worked on only 3 categories. It would be interesting to also study the reviews of apps of other categories, and see how possibly they affect our findings. Because although the categories that we selected are top categories, however there are thousands of apps under other categories.

2. **Collecting reviews of apps in other app markets**

   All of the reviews that is used in our study is from Play Store. However, there are other app markets that have a high number of apps and downloads such as app store.

In the future, we will expand our study to include reviews from other app stores as well, and study how the ratio of feature requests might change from a platform to another platform as compared to bug requests, or even the number of feature requests for the same app in different platforms.

3. **Group the features based on their functionality**

   In our study we have used the features in general. For example, we have showed the list of feature based on the number of requests originally requests by the user regardless of the relationship between the features. In the future, we will group these feature based on their functionality. For example, rotation and zoom belong to photos, while live and replay belong to video. This way, when the features are presented to the developers, they will be able to directly select the features that are more relevant to their apps, such as image manipulation. In other words, if their apps work mostly on images, they do not have to check the feature requests for video.

4. **Increase the number of feature in the framework**

   The framework currently supports four features. These features are fully functional and can be directly included and used in the apps. However, the framework is still in its early stages and needs more feature in order to work as a fully functional framework. Meanwhile, the features need to be updated frequently, therefore, in the future connecting and fetching these features from cloud will help the framework stay up to date with the latest features.

5. **Completely separating the components of the framework so that the mining operation can be applied on a different platform and then integrated to the framework.**

   Although the process of mining and the applying features in the framework is performed in two different steps, however they are completely coupled and the data of the framework completely depends on the mining process. There is another way of

performing these two different operations such that the mining can be done on a separate platform and the frameworks updates itself periodically fetching the data from the mining platform. This way each component can work independently, and can be used for a purpose while sharing information between each other.

6. **Automate the whole mining process including association of the topics to features.**

   The process of associating the features to the terms and topics is performed manually in this study. In the future, this process can be automated in order to make the entire extraction and association automatic. One way to do that, is to use n-grams model in order to derive a sequence of words instead of one single word. Or an alternative approach, is to create a model that can automatically parse the description for each package in the awesome vue github repository.

Bibliography

[1] Ben Martin Adam Lella, Andrew Lipsman. The 2015 U.S. Mobile App Report. https://www.comscore.com/Insights/Presentations-and-Whitepapers/2015/The-2015-US-Mobile-App-Report, 2015. [Online; accessed 08-August-2017].

[2] Appboy. App Uninstalls: A Totally Unscientific Look at What Makes Customers Pull the Plug. http://bit.ly/2iVdlcp, 2016. [Online; accessed 09-August-2017].

[3] Appbrain. Number of available Android applications. https://www.appbrain.com/stats/free-and-paid-android-applications, 2012. [Online; accessed 19-July-2017].

[4] Appbrain. Android Operating System Statistics. https://www.appbrain.com/stats, 2017. [Online; accessed 09-August-2017].

[5] Appbrain. Ratings of apps on Google Play. https://www.appbrain.com/stats/android-app-ratings, 2017. [Online; accessed 09-August-2017].

[6] Gabriele Bavota, Mario Linares-Vasquez, Carlos Eduardo Bernal-Cardenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. The impact of api change-and fault-proneness on the user ratings of android apps. *IEEE Transactions on Software Engineering*, 41(4):384–407, 2015.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[8] Ning Chen, Jialiu Lin, Steven CH 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*, pages 767–778. ACM, 2014.

[9] Emitza Guzman, Omar Aly, and Bernd Bruegge. Retrieving diverse opinions from app reviews. In *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*, pages 1–10. IEEE, 2015.

[10] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162. IEEE, 2014.

[11] Leonard Hoon, Rajesh Vasa, Gloria Yoanita Martino, Jean-Guy Schneider, and Kon Mouzakis. Awesome!: conveying satisfaction on the app store. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, pages 229–232. ACM, 2013.

[12] Leonard Hoon, Rajesh Vasa, Jean-Guy Schneider, and Kon Mouzakis. A preliminary analysis of vocabulary in mobile app user reviews. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, pages 245–248. ACM, 2012.

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

[14] Peter Jackson and Isabelle Moulinier. *Natural language processing for online applications: Text retrieval, extraction and categorization*, volume 5. John Benjamins Publishing, 2007.

[15] Simon Khalaf. Seven Years Into The Mobile Revolution: Content is King Again. `http://flurrymobile.tumblr.com/post/127638842745/`

seven-years-into-the-mobile-revolution-content-is, 2015. [Online; accessed 08-August-2017].

[16] Orges Leka. Facebook Reports Third Quarter 2016 Results. https://investor.fb.com/investor-news/press-release-details/2016/Facebook-Reports-Third-Quarter-2016-Results/default.aspx, 2016. [Online; accessed 02-February-2017].

[17] Orges Leka. Q3 2016 Letter to Shareholders. http://files.shareholder.com/downloads/AMDA-2F526X/5305357775x0x913983/81893241-7B36-4E80-8CED-454808BEC856/Q3_16_ShareholderLetter.pdf, 2016. [Online; accessed 02-February-2017].

[18] Orges Leka. Database of Android Apps. https://www.kaggle.com/orgesleka/android-apps, 2017. [Online; accessed 15-September-2017].

[19] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31, 1968.

[20] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 116–125. IEEE, 2015.

[21] ALEX WALZ. MAY. The Mobile Marketers Guide to App Store Ratings & Reviews. https://www.apptentive.com/blog/2015/05/05/app-store-ratings-reviews-guide/, 2015. [Online; accessed 02-February-2017].

[22] Fabio Palomba, Mario Linares-Vásquez, 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 *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 291–300. IEEE, 2015.

[23] 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 *Software maintenance and evolution (ICSME), 2015 IEEE international conference on*, pages 281–290. IEEE, 2015.

[24] Israel J Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst, and Ahmed E Hassan. Impact of ad libraries on ratings of android mobile apps. *IEEE Software*, 31(6):86–92, 2014.

[25] Eric Shaw, Alex Shaw, and David Umphress. Mining android apps to predict market ratings. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 166–167. IEEE, 2014.

[26] Google Support Team. View & analyze your app's ratings & reviews. `https://support.google.com/googleplay/android-developer/answer/138230?hl=en`, N.A. [Online; accessed 09-August-2017].

[27] Craig Trim. The Art of Tokenization. `https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en`, 2013. [Online; accessed 26-July-2017].

[28] S Vijayarani, Ms J Ilamathi, and Ms Nithya. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

[29] 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*, pages 14–24. ACM, 2016.

[30] Phong Minh Vu, Tam The Nguyen, Hung Viet Pham, and Tung Thanh Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 749–759. IEEE, 2015.

[31] Phong Minh Vu, Hung Viet Pham, Tung Thanh Nguyen, et al. Phrase-based extraction of user opinions in mobile app reviews. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 726–731. ACM, 2016.

[32] Andy Wallace. Very basic strategies for interpreting results from the Topic Modeling Tool. `http://bit.ly/2ms2BHC`, 2012. [Online; accessed 26-July-2017].

[33] Peter Willett. The porter stemming algorithm: then and now. *Program*, 40(3):219–223, 2006.

Appendix A

Diagrams

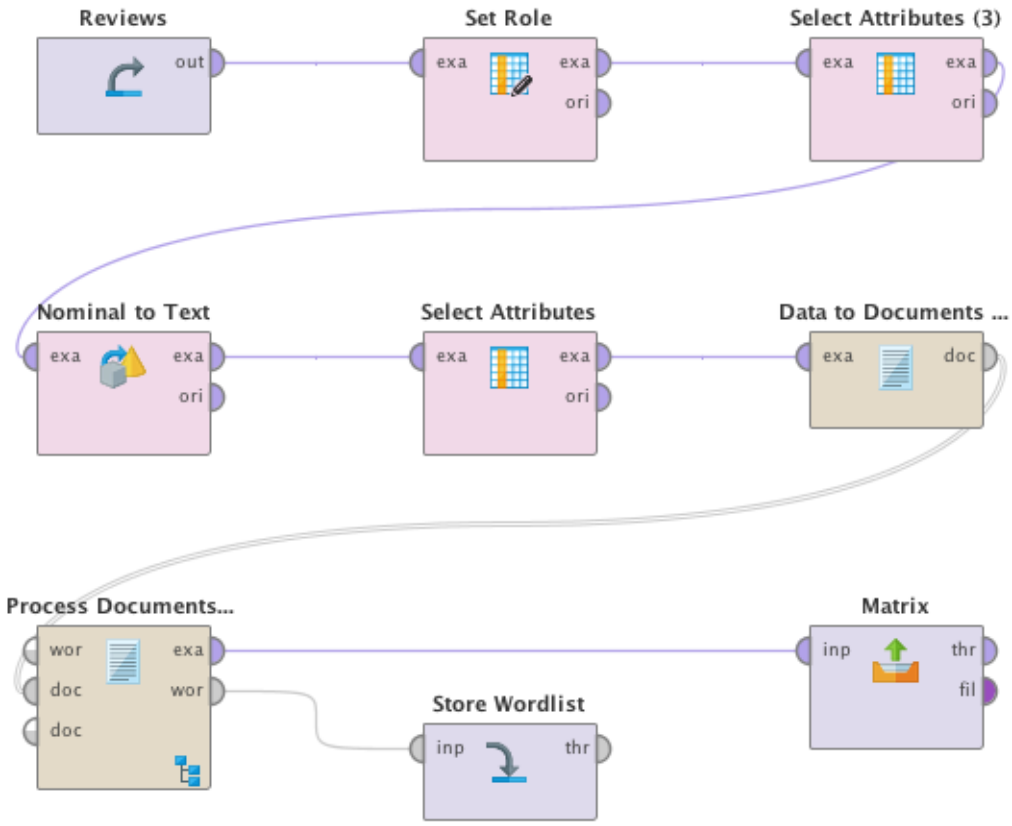

Figure A.1: Creating data matrix for classification model

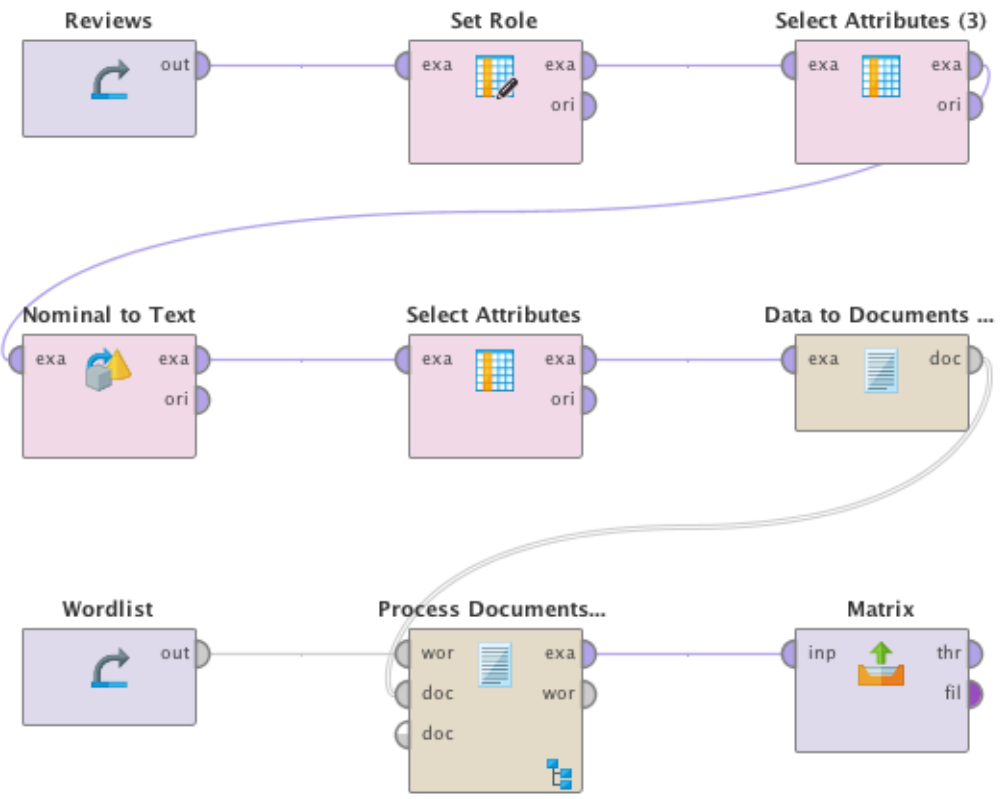

Figure A.2: Creating data matrix of all reviews

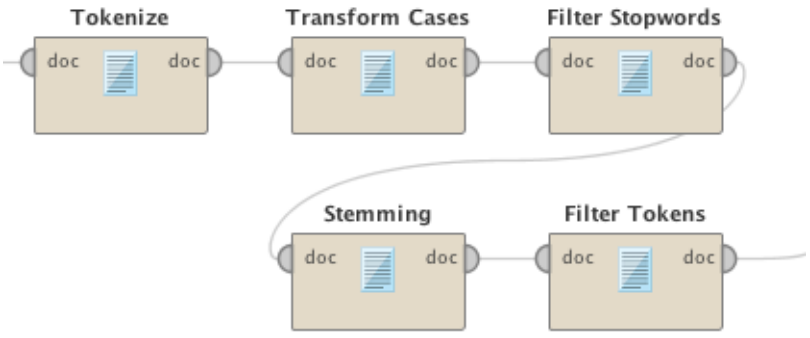

Figure A.3: Processing documents(reviews)

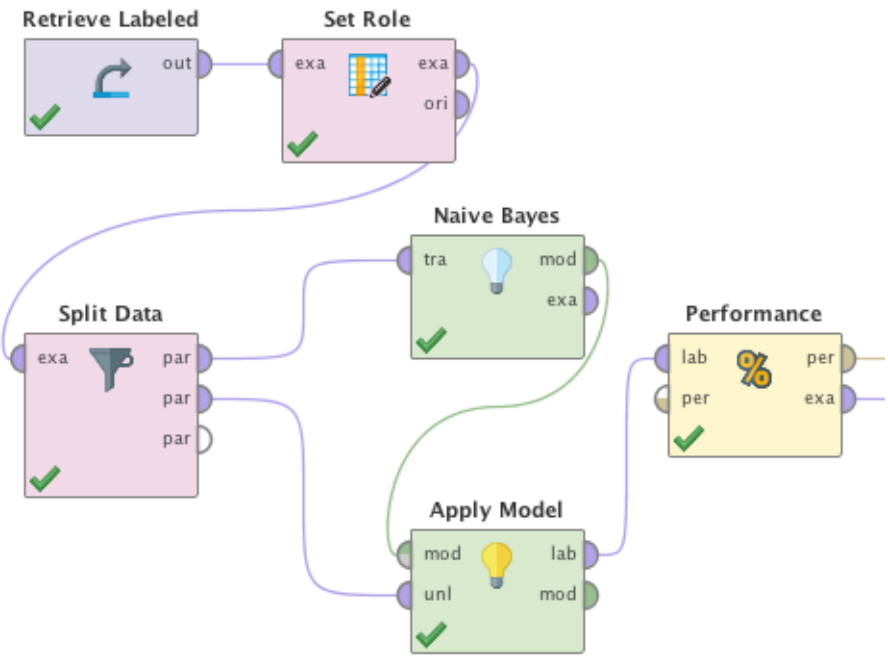

Figure A.4: Classification model construction

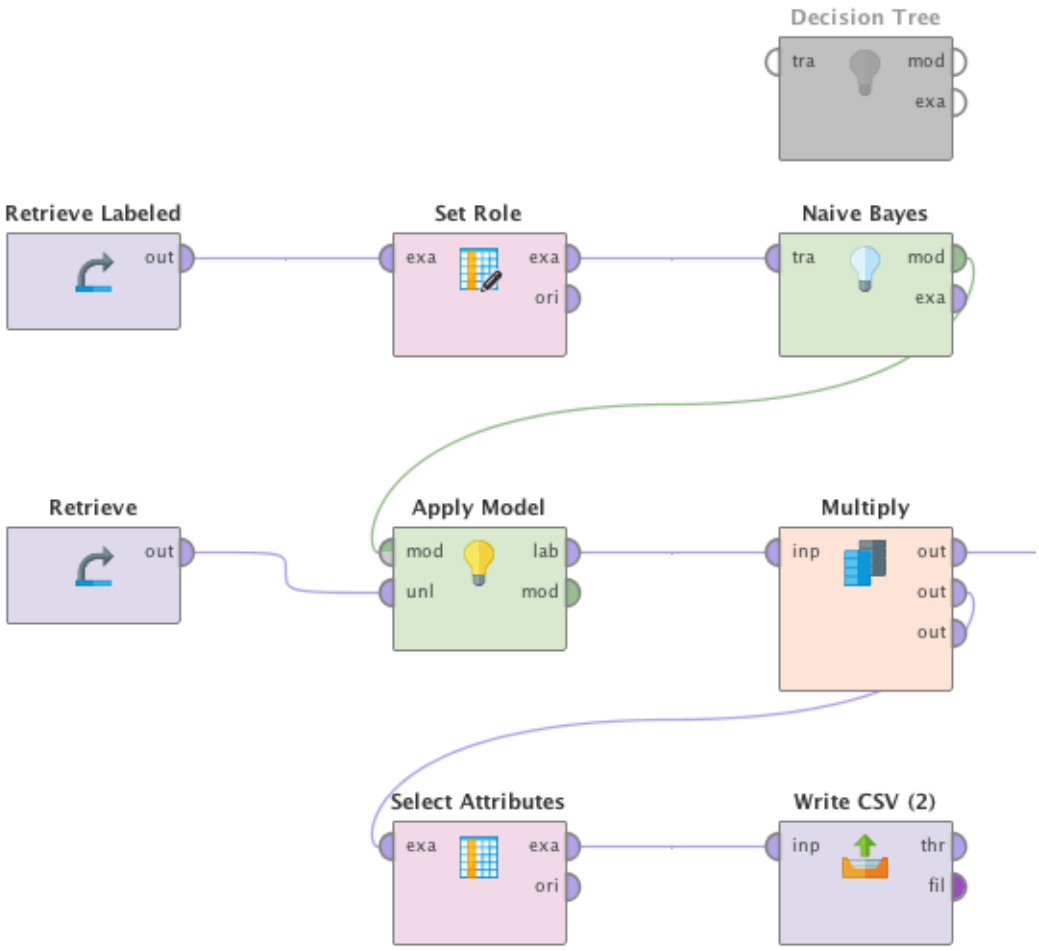

Figure A.5: Applying models

Appendix B

Survey

B.0.1 IRB Approval

R·I·T

Rochester Institute of Technology

RIT Institutional Review Board for the
Protection of Human Subjects in Research
141 Lomb Memorial Drive
Rochester, New York 14623-5604
Phone: 585-475-7673
Fax: 585-475-7990
Email: hmfsrs@rit.edu

Form C
IRB Decision Form

TO: Rebaz Saleh

FROM: RIT Institutional Review Board

DATE: **June 9, 2017**

RE: Decision of the RIT Institutional Review Board

Project Title – Mining User Reviews To Extract Features For Initial Release Of Mobile Apps

The Institutional Review Board (IRB) has taken the following action on your project named above.

☒ Exempt 46.101 (b) (2)

Now that your project is approved, you may proceed as you described in the Form A.

You are required to submit to the IRB any:

- **Proposed** modifications and wait for approval before implementing them,
- Unanticipated risks, and
- Actual injury to human subjects.

Heather Foti, MPH
Associate Director
Office of Human Subjects Research

Figure B.1: IRB approval

B.0.2 Request for participation email

Hello ***

I am writing to you to request your participation in a brief survey. Your responses to this survey will help us evaluate the effectiveness of mining user review in app stores and develop tools and packages that would help developers automate the process of including those features in their apps.

Please click the link below in order to access the survey.

http://www.se.rit.edu/ rss1803/survey/

email: ******@gmail.com password: *********

Please use Google Chrome

Thanks

B.0.3 Survey

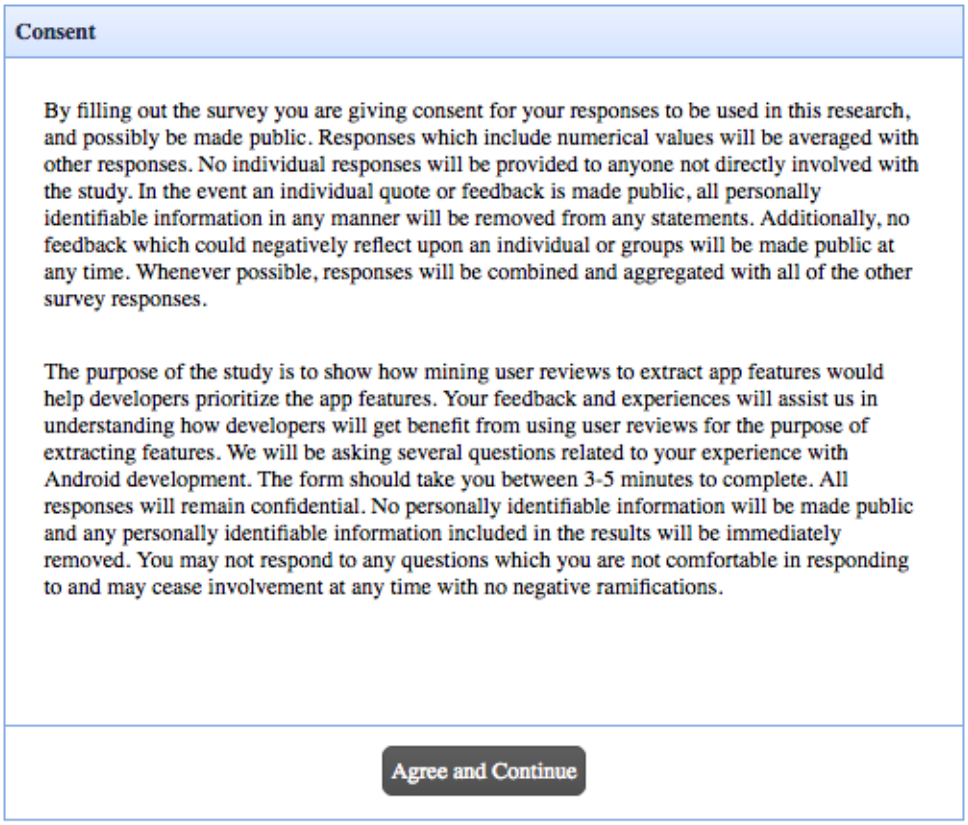

Figure B.2: Survey consent

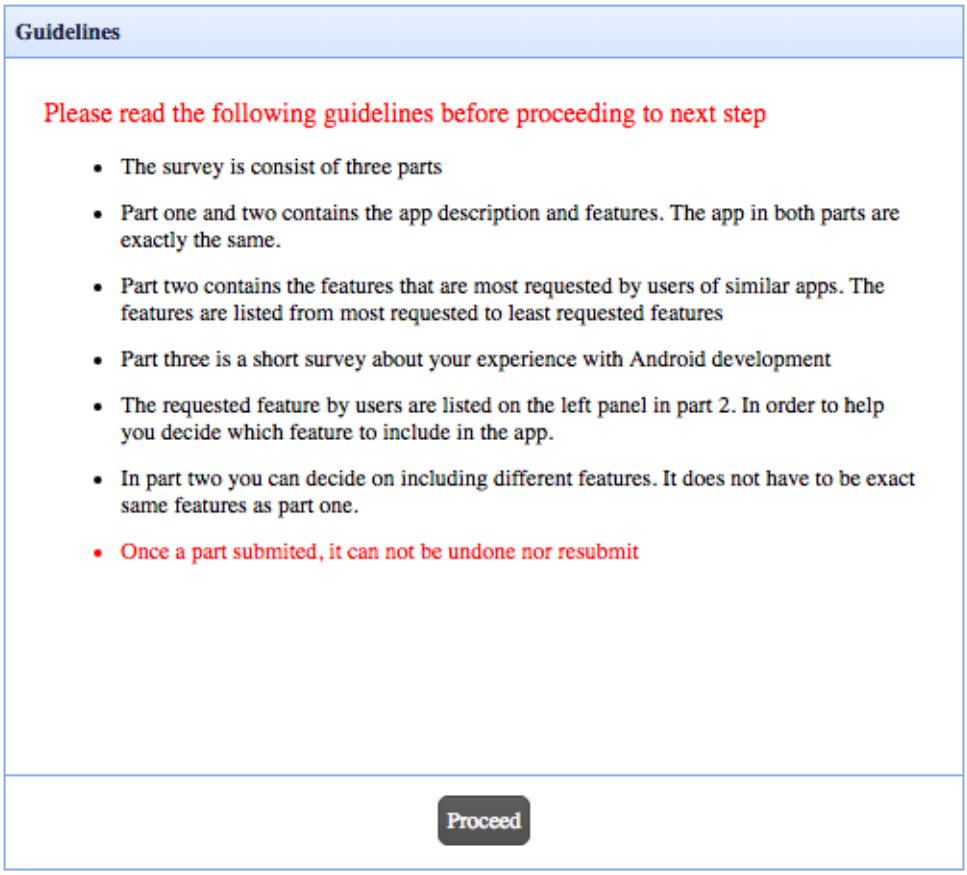

Figure B.3: Survey guidelines

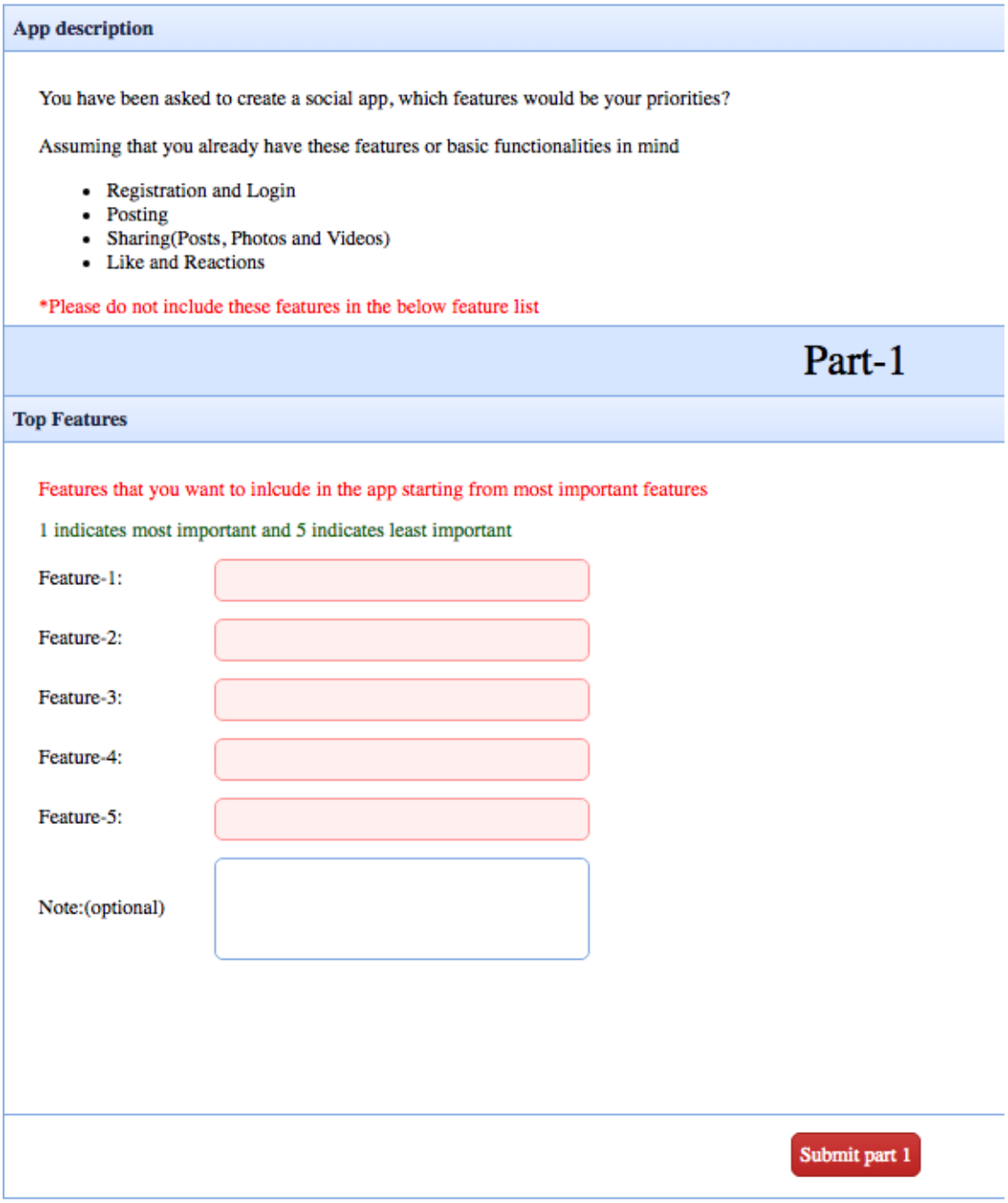

Figure B.4: Step one in the survey

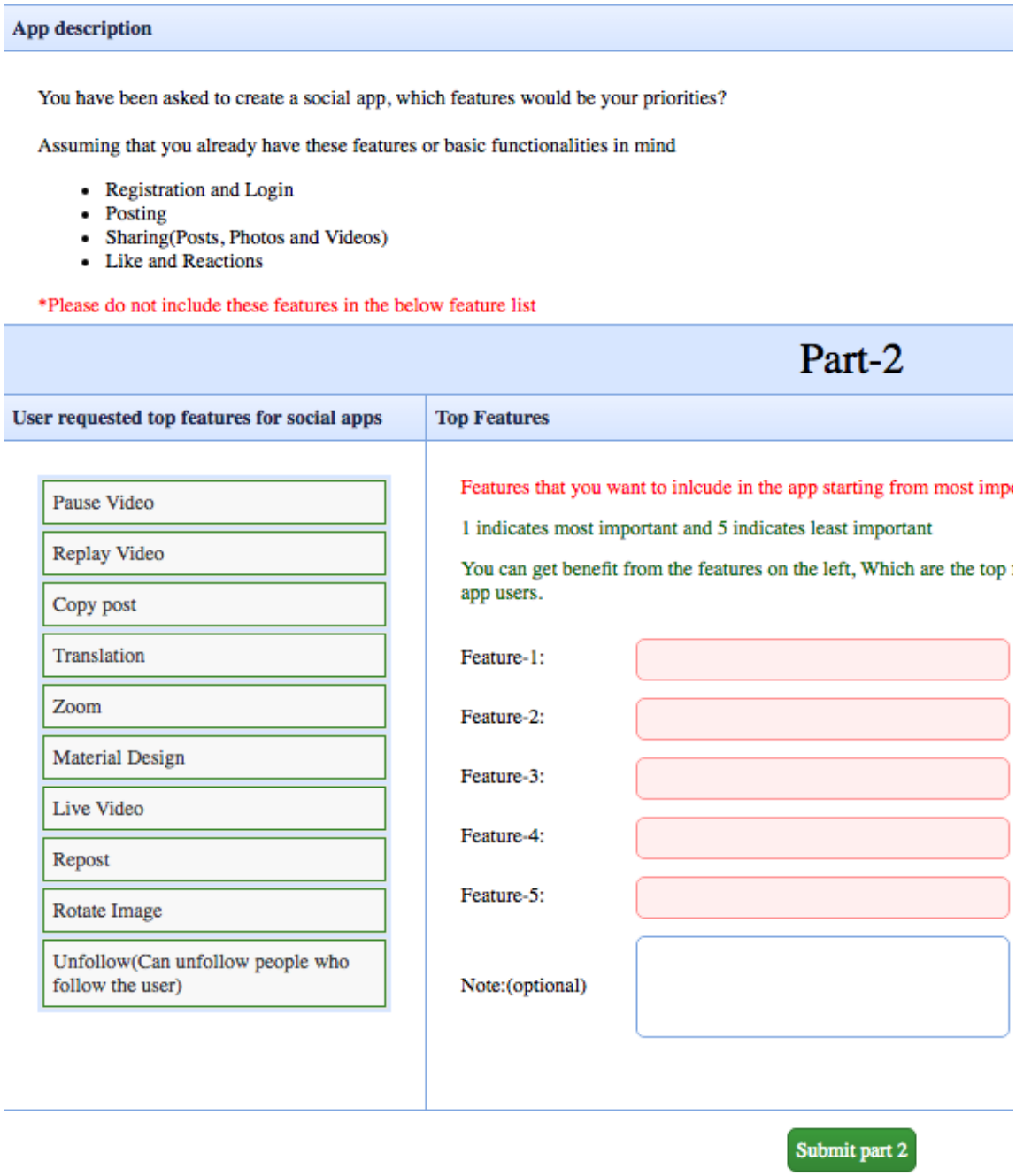

Figure B.5: Step two in the survey - A

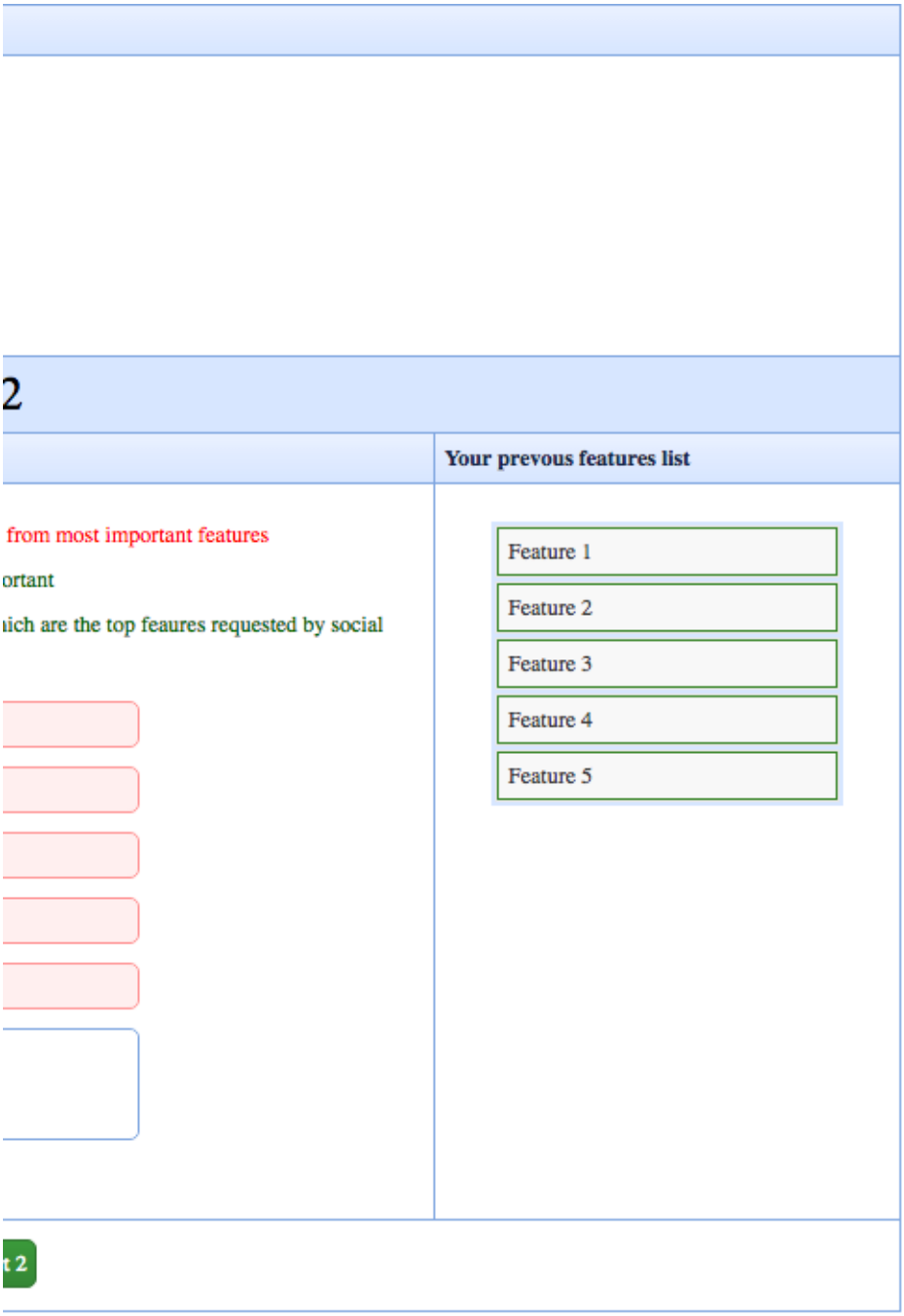

Figure B.6: Step two in the survey - B

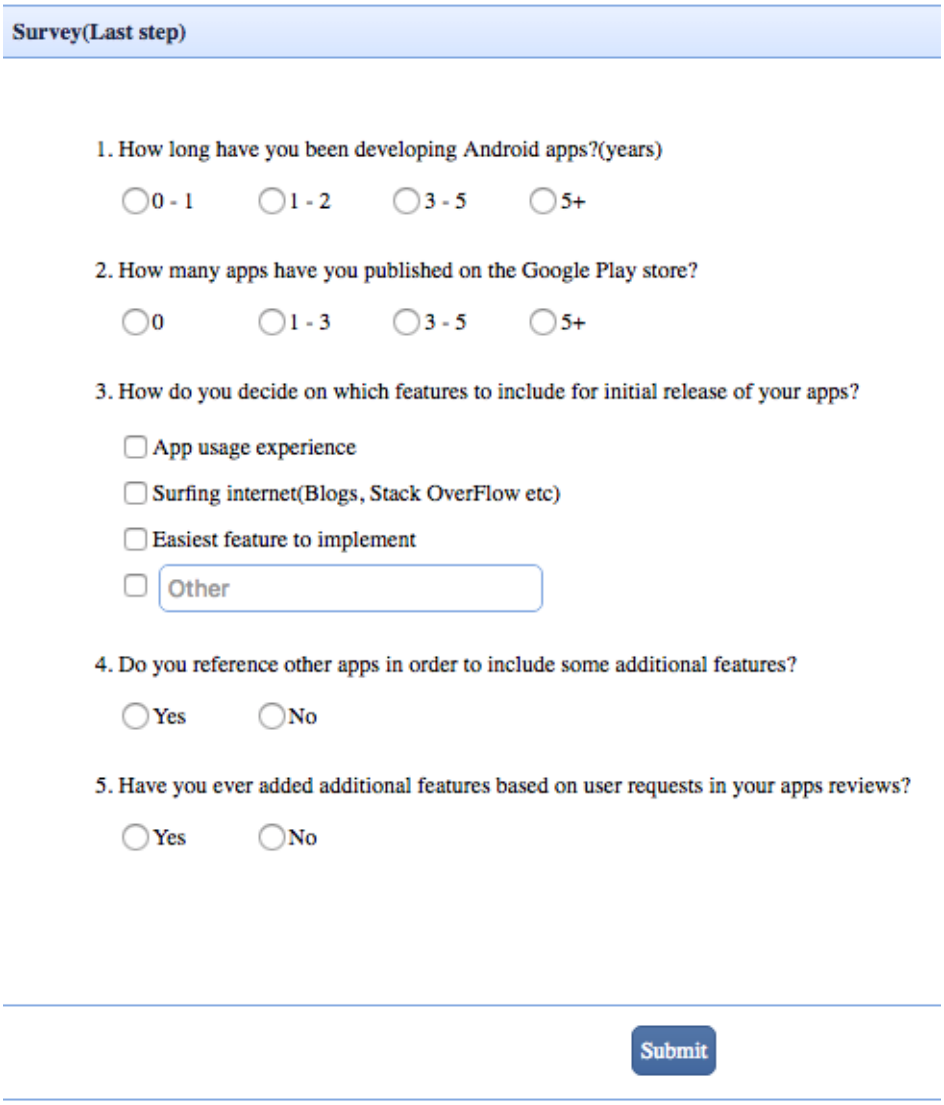

Figure B.7: Questionnaire

Appendix C

Tables

Social Reviews		Music Reviews		Tools Reviews	
App	#Reviews	App	#Reviews	App	#Reviews
Instagram	250762	SoundCloud	56816	Clean Master - Antivirus	209998
Facebook	237974	Spotify	14487	CM Security	103477
Snapchat	111467	Musicmatch	4519	DU Speed Booster	37731
Tango	22019	Shazam	4409	360 Security	36816
Pinterest	12450	Tunein	3080	Battery Doctor	24927
Badoo	5557	Amazon Music	2560	DU Battery Saver	24633
textPlus	1783	Melodis	1747	AVG Antivirus	14249
VK	1067	Piano Perfect	1571	Avast Antivirus	11871
KakaoStory	253			AppLock	11641
				Google Translate	11608
				Flashlight	5774
				Lookout	3120
				Brightest Flashlight	2932
				Adobe Air	2678
				Google Home	2158
				Torch Flashlight	1019
				Dr Web	945
				Smart Connect	846

Table C.1: Topics in social apps reviews

	Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**	**Topic 6**	**Topic 7**
1	paus	copi	zoom	usernam	design	effect	locat
2	default	caption	twitter	creat	lol	front	repost
3	tab	volum	roll	avail	materi	flash	frame
4	anyway	alot	includ	plzzz	taken	emoji	agre
5	main	translat	cover	complaint	select	live	insta
6	audio	solv	kind	invalid	icon	correct	rotat
7	fact	understand	multipl	switch	addit	write	whatev
8	press	system	recommend	simpl	googl	clear	map
9	replay	word	funni	entir	definit	sec	auto
10	hear	url	besid	size	month	super	yall

Table C.2: Topics in tools apps reviews

	Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**	**Topic 6**	**Topic 7**
1	theme	scan	password	notif	minut	automat	auto
2	show	card	comput	guy	includ	schedul	hide
3	wifi	block	background	reduc	disabl	type	kill
4	plz	put	turn	size	ram	cach	sms
5	applock	memori	temperatur	folder	miss	cleaner	blocker
6	look	abil	constant	panel	packag	booster	landscap
7	cool	locker	unabl	select	differ	data	mode
8	dark	choos	amaz	pictur	pro	fixer	manual
9	advertis	number	bad	etc	permiss	restart	filter
10	everth	feel	alright	avail	saver	transmiss	extra