Reproducibility Study of "*On the automatic classification of app reviews*" by W. Maalej and H. Nabil (2015)

DATA 553 Final Project

KT Hobbs, Binal Patel, Yuening Li, Sofia Bahmutsky

1. Introduction

Mobile applications (apps) became a feature of smartphones starting in 2008, where they began as useful applications available for mobile users to check email, calendar, and other basic applications. Their appearance and usage became very popular and in demand, so much so that currently there are millions of paid and free apps available on the App Store (Apple platform), Google Play (Android), and other common operating systems. These apps vary between categories, from games to productivity, health, news, and many others. Apps are developed and released for public use, and give an option for the user to provide ratings and feedback on their experience using the app. These reviews can be very useful for developers, and it has been shown that the rating score relates to the app popularity with customers, thus motivating development of useful, functioning, and engaging apps (Finkelstein et al., 2014).

Given the importance of developing useful and functioning apps, the reviews provided from users are very important and contain a lot of essential information which developers can use in order to improve apps, correct bugs, and add features that are requested by users. Since the volume of reviews provided by users is enormous, and multi-faceted, it is imperative for analysts and app developers to utilize the information and sift through the very prevalent useless information as well (Pagano and Maalej, 2013). The type of review provided by the users review can be categorized into categories, for example: bug reports (reviews which generally describe a problem that occurred using the app, such as crash or strange behaviour), feature request (reviews which request an additional missing feature to be added to the app, perhaps a new language availability), user experience (reviews which explain how the app was used, the general experience, or reasons which the customer found the app to be useful), and finally ratings (reviews which are relatively simple reiterations of the star rating, generally have little additional information) (Maalej and Nabil, 2015). The task of classifying and reading reviews is extremely tedious, therefore machine learning models have become very popular to accomplish this task of classifying app reviews into categories for easier utilization of informative reviews which can be beneficial to app developers.

The concepts of reproducibility and replicability are well-rooted in natural and experimental sciences, with methodology and practices for researchers to abide by. However, in computational sciences, especially data science, there is a growing problem regarding the inability of producing reproducible, replicable, and robust models (Pineau, 2020). Reproducibility is defined as the ability to recompute results, given an observed data set and knowledge of the methodology (Peng, 2015). Pineau explains that it may be easy to find reasons why reproducibility may be difficult to achieve in the natural sciences, but strange that is such a large problem in computer sciences, where in theory, running the same algorithm on a different machine should provide the same output, but this is often not the case. The root cause of poor reproducibility in computational sciences stems from poor methodology, misleading results, or lack of details in publications.

The purpose of this report is to show the possibility of reproducibility of a very popular machine learning model which automates the classification of app reviews, based on the 2015 paper by Maalej and Nabil, *'On the automatic classification of app reviews'*. We used the authors training data and created our own code for the classification project. Our project constitutes a reproducibility study because we are using the authors data and methodology, and the goal of our project is to observe whether or not our classification metrics of the reviews is the same as the classification scores obtained by the authors in their results.

2. Methodology

2.1 Pre-processing

The pre-processing steps we used for

our dataset were the same as the steps we used in DATA 542 for the final

project. These steps were completed in python using several packages (os, glob,

pandas, nltk, langid, re, unicodedata, and others). First, records of all

reviews were concatenated into one large data frame, then duplicate rows were

dropped. The number of entries in the data frame was originally 2715303. Review

text was processed by use of regex to remove multiple characters from the

review if they occurred in sequence 3 or more times. Nltk corpus 'words' was

used to remove non-English text from the reviews. UnicodeData package was used to remove non-ASCII characters from the reviews. Punctuation was also removed from reviews. In addition, reviews that contained two or fewer words were entirely removed from the dataset. Upon all pre-processing, the remaining number of reviews was 1829048.

2.2 Sample size & Labelling

Our testing dataset sample size was calculated by sampling the pre-processed review data frame with confidence level 95% and confidence interval 5. This resulted in the calculated sample size of 384, using an online tool (https://www.surveysystem.com/sscalce.htm). The dataset of size 384 was seen by 2 group members, who were tasked with manually labelling the reviews and classifying into one of the four categories (bug reports, feature requests, user experience, or ratings) using a binary classification method. We decided to use the binary classification scheme because it was preferred by the authors of the paper which we were attempting to reproduce results for. A third group member was tasked to check the assigned labels, to determine if there were any disagreements, and in the case of a disagreement between the labels, to discuss and reach consensus on selecting an appropriate category.

2.3 Building the Classification Model

We implemented several classification techniques alone and in combination to evaluate the best model. We used sklearn's

GaussianNB module for binary classification and MultinomialNB for multi-classification (see binary classifer.ipynb and multi-classifier.ipynb, respectively). It was decided that the classifier method would be blocked in this study to ensure robust assessment of the effect of natural language processing (NLP) on classification accuracy utilized by the researchers. Both classifiers were trained on the investigated paper's processed data according to the applied text processes; for example, classifiers categorizing lemmatized reviews from our app review dataset were trained on the lemmatized output from the author's review dataset.

2.4 NLP Techniques

This study focussed on three NLP techniques: term frequency–inverse document frequency (TF-IDF), stop word removal, and lemmatization. Since vectorizing text is required for classifiers, TF-IDF serves as a control (no processing).

2.5 Study Design

Our study is split

into 2 main parts, the first in which we developed our own code from sklearn library and used our code to train the dataset provided from the authors and test it on our own testing data set. The second part of the study involved using the authors publicly available code, using the authors training data and testing the model on the authors testing data. We did this in order to investigate true reproducibility of their study using all the components which they used and checking the metrics we obtained. In theory, we expect that all metrics would be the same as in their paper given that we are using all the exact same data and classification code.

3. Results – Part I

3.1 Metrics –

our code, author training data, our test data

Based on the predicted labels, we computed precision, recall, and

F1-score metrics of the sample dataset. These results are presented below.

Table 1. Summary of binary classifier metrics for each natural language processing technique. Since the data is asymmetric, secondary metrics were used to evaluate classifier accuracy.

NLP Technique	Precision	Recall	F1_score
TF-IDF	0.3015604	0.28924868	0.24562
TF-IDF and Stopword Removal	0.01953125	0.25	0.036232
TF-IDF and Lemmatization	0.01953125	0.25	0. 036232

Table 1a. Binary classifier metrics for Bug Report

reviews.

NLP Technique	Precision	Recall	F1_score
TF-IDF	0.04	0.104167	0.057803
TF-IDF and Stopword Removal	0.01953125	0.25	0.036232
TF-IDF and Lemmatization	0.01953125	0.25	0. 036232

Table 1b. Binary classifier metrics for Feature Request

reviews.

NLP Technique	Precision	Recall	F1_score
TF-IDF	0.0285714	0.14	0.047458
TF-IDF and Stopword Removal	0.01953125	0.25	0.0362319
TF-IDF and Lemmatization	0.01953125	0.25	0.0362319

Table 2. Summary of multiclass classifier metrics for each natural language processing technique. Since the data is asymmetric, secondary metrics were used to evaluate classifier accuracy.

NLP Technique	Precision	Recall	F1_score
TF-IDF, Stopword Removal, Lemmatization, Bigram	0.365691	0.286458	0.247592
TF-IDF	0.311680	0.260417	0.203163
TF-IDF and Lemmatization	0.645778	0.271507	0.225073

Table 2a. Multiclass classifier metrics for Bug Report

reviews.

NLP Technique	Precision	Recall	F1_score
TF-IDF, Stopword Removal, Lemmatization, Bigram	0.030469	0.162500	0.051316
TF-IDF	0.035102	0.179167	0.058703
TF-IDF and Lemmatization	0.032959	0.183333	0.055873

Table 2b. Multiclass classifier metrics for Feature

Request reviews.

NLP Technique	Precision	Recall	F1_score
TF-IDF, Stopword Removal, Lemmatization, Bigram	0.033333	0.153333	0.054762
TF-IDF	0.027184	0.186667	0.047458
TF-IDF and Lemmatization	0.025926	0.186667	0.045528

3.2 Comparison of Results

Comparing our results with the equivalent
combination of classification techniques from the authors paper shows that
overall our classifiers metrics are much lower than the ones produced in the authors
results. In our study we used three combinations of classification techniques:
1) TF-IDF,
Stopword Removal, Lemmatization, Bigram, 2) TF-IDF, 3) TF-IDF and
Lemmatization. As seen below in Table 3, the precision, recall, and F1 scores
from the authors paper are generally between 0.5 to 0.9, which are quite high
and indicate good classification of the reviews by the algorithm. The authors
used more combinations of classification; however, we will focus mainly on the combinations
which highlighted in yellow since they best represent the equivalent to our
combination of classification techniques. Our results tend to be much lower,
the highest value we measured was 0.645 for the multiclass classifier, using
the combination of TF-IDF and
Lemmatization. The majority of the metrics we obtained from our algorithm for
our testing data were quite low, typically 0.2 or less. This indicates a poor
classification ability of the algorithm, which could be due to many factors

which will be further discussed in the following sections of this report.

Table 3. Accuracy of the classification techniques

using Naive Bayes on app reviews from Apple and Google stores (mean values of the 10 runs, random 70:30 splits for training:evaluation sets). Taken from Maalej et al.

Part II

3.3 Alternative Results - (using author data on author code,

using our data on author code)

NLP Technique	Precision	Recall	F1_score
TF-IDF, Stopword Removal, Lemmatization, Bigram	0.101562	0.101562	(0.101562,0.101562,0.101562, None)
TF-IDF	0.111979	0.111979	(0.111979,0.111979,0.111979 ,None)
TF-IDF and Lemmatization	0.114583	0.114583	(0.114583,0.114583,0.114583, None)

4. Discussion

Based on the results above, we were able to achieve relatively high precision for classification on the “Ratings” category of reviews, however for the other three categories, our classification metrics are much lower than the metrics which were achieved by the authors. We were unable to achieve the same accuracy or precision as the original paper for ‘Bug report’, ‘Feature Request’, and ‘User Experience’ categories.

According to the authors, they stated that the binary classification protocol was more appropriate, and they achieved better results using it, however according to our findings the opposite is true. This is seen comparing Table 1 and 2 metrics. The values are larger in the multiclass table, with F1 scores ranging between approximately 0.2 and 0.65, as compared to very low F1 scores from the binary classifier.

Based on the talk from keynote speaker Dr. Pineau, we would say that from our experience, this library is not robust or reproducible. Reasons for this stem from the problems with the dataset which we used, and the randomly selected sample which was used for the testing of the classifier. There are many differences between our data and the data, which was used by the author, in addition our pre-processing seems to be more in-depth than the pre-processing used by the

authors. Since this project asked us to use our pre-processed dataset from Data 542, it may be that the pre-processing was over-processed, and the classifiers were not able to ‘read’ the processed text. This can also be seen simply by reading some of the review data provided by the authors: their reviews are generally legible and in sentence format, whereas our data has many reviews which we could not even understand, since they were processed too much, or were simply very poorly written by the reviewer. Further sources of error and differences are discussed below.

4.1 Additional Thoughts, Sources of Error

This study has many potential sources of error, mostly since we were trying to reproduce a prior study and ran into issues with code, packages, etc. This resulted in the use of other packages and writing of our own code for many steps of the experiment. Doing so introduces sources of difference and error that are difficult to account for.

Perhaps the largest problem we encountered was that the reviews were often illegible, nonsensical, or did not belong in any of the four categories. The reading difficulty may have occurred from our processed text, since using the pre-processing steps actually resulted in text that was ‘overly processed’, we couldn’t properly read the reviews for manual labelling, nor could libraries and functions properly utilize the text data. For reviews which we could not easily understand and classify into a category, we had to make judgement on very little information, and this resulted in us often selecting ‘user experience’ or ‘rating’ category. This source of error introduced bias into our manually labeled dataset.

A large source of error which was discovered was the problem with sentiment. The sentiment package in python from nltk required a large training set of words, excerpts, and sentences with

punctuation in order to properly function and determine sentiment of our test data. Given that the preprocessing step involved removal of all punctuation, our processed text was in the form of just words and phrases, therefore was not valid input for sentiment validation. In addition, we would have been incorrectly using our own reviews as training data for sentiment analysis, and subsequently including the sentiment on the same 384-row data frame which is our testing data. This means that in order for the sentiment analysis to work correctly we would have required additional training data full of sentences, which was not available, therefore we decided to omit sentiment analysis from our machine learning model.

Something we noticed which may be an additional source of bias is that the authors had collected app reviews from different platforms, Apple Store and Google Play, and as noted in their paper, the reviews from Apple Store had much more sample size than the sample size of reviews from Google Play. This is an issue because the authors paper therefore would have the potential to bias the results to reviews which were from Apple

Store and not be representative of Google Play users. In our dataset, we only had reviews from the App Store (Apple Store) thus our experiment would also be biased on not representative of Google Play users at all. Although it is speculation, there may be societal differences between people who prefer Apple products to those who prefer Android products. For example, Apple products tend to be pre-programmed , formatted and in the high-end price range, making them easy to use, but also have little flexibility in terms of customization and extended features. On the other hand, android products have much more winder range of affordability, and apps and they allow usage of third-party apps (which is very popular amongst developer communities). Based on this knowledge, it may be that reviews from Apple Store come predominantly from ‘average users’, or people who are more likely to use Apple products thus lead to fewer reports of Bug Reports and Feature Requests. Perhaps the android developer community would be more motivated to fix bugs and add features to their apps since they have a more open platform which allows third party apps, as well as more users who are familiar with more types of apps in general.

Based on the general sources of error

which were described in this section, we didn't test the authors combination of features for classification: bigram - stop words + lemmatization + rating + 2x sentiment scores + tense. Our attempt to reproduce the authors study was limited due to the fact that we could not reproduce their best combination of features, as well as we had to use our own code for the classification. A future study which would be beneficial to investigate the reproducibility of the 2015 paper by Maalej and Nabil

should consider using app reviews from android platforms, and to use the authors code without modification.

5. References

Finklestein, A., M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang. (2014). App Store Analysis: Mining App Stores for Relationships between Customer, Business and Technical Characteristics. *Research Note RN/14/10, UCL Department of Computer Science.*

Gibney, E. (2020). This AI researcher is trying to ward off a reproducibility crisis. *Nature* 577: pg. 14.

Maalej, W., and H. Nabil. (2015). Bug report, feature request, or simply praise? On automatically classifying app reviews.

Pagano, D., and W. Maalej. (2013) User feedback in the Appstore: an empirical study. *Proceedings of the international conference on requirements engineering--RE'13, pp 125---134*

Peng, R. (2015). The reproducibility crisis in science: A statistical counterattack. *Royal Statistical Society.*