# Analysis and Prioritization of App Reviews

**Conference Paper** · December 2019

DOI: 10.1109/ICAC347590.2019.9036801



**5 authors**, including:

Tejas Hirave

**16** PUBLICATIONS **18** CITATIONS

SEE PROFILE

Saurabh Malgaonkar

University of Otago

**37** PUBLICATIONS **124** CITATIONS

SEE PROFILE

Mostafa Alwash

University of Oslo

**8** PUBLICATIONS **18** CITATIONS

SEE PROFILE

# Analysis and Prioritization of App Reviews

Tejas Hirave
Computer Engineering Department
Shah and Anchor Kutchhi Engineering College
Mumbai University
Mumbai, India
tejas1mumbai@gmail.com

Saurabh Malgaonkar, Mostafa Alwash, Jithin Cheriyan
Department of Information Science
University of Otago
Dunedin, Otago, New Zealand
saurabhmalgaonkar@gmail.com,
malwash@gmail.com,
jithinraj85@gmail.com

Sakshi Surve
Computer Engineering Department
Thadomal Shahani Engineering College
Mumbai University
Mumbai, India
geetams24@rediffmail.com

***Abstract*— Smartphone apps market is a billion dollar industry and is growing rapidly. Thus, app developers are constantly on the lookout for efficient and reliable data analytics and prioritization tools that analyze crucial feedback present in app reviews and at the same instance prioritize the reviews for remedial actions. Such analysis and prioritization tools significantly assist app developers to identify and address requirements raised in the app reviews by the app's customers (end-users). This, in general, aids the app maintenance and evolution cycles. In this study, we propose and develop analytics and prioritization methods and represent the methods in the form of a software tool to support app developers of the MyTracks app towards the app's maintenance and evolution cycles. Our findings through the means of system usability scores reveal that the app developers had significantly agreed with the usability, reliability, and efficiency of our tool.**

***Keywords—App reviews, Data mining, Natural language processing, Data visualization, Sentiment analysis, Multinomial Naïve Bayes, Requirements prioritization, System usability scale***

## I. INTRODUCTION

In recent times, almost every enterprise in the world is offering its business or product services to their customers through the means of an app [1]. The app is made available by the enterprises and is installed by the customers on their smart devices (phones, tablets) via Online Application Development Platforms such as Google Play or Apple's App Store [2]. These business or product services range over a wide variety of applications related to communications, medical, education, entertainment, finance, farming, robotics, gaming, transportation etc. ultimately making the app a crucial medium of product service delivery. That said, the app industry has been continuously evolving and growing due to the enormous increasing popularity of smart devices worldwide [3]. Subsequently, enterprises are constantly analysing the feedback of their apps shared by the customers that exists in the form of app reviews, and these enterprises aim towards the conversion of the gathered feedback into actionable knowledge [4]. This task is primarily undertaken by enterprises to address the requirements of the customers present in the feedback (app reviews) for the app's maintenance and evolution cycles [5, 6]. These requirements reflect the request for new features, issues pertaining to the app, or suggestions for app improvements. The addressal of such requirements, in general, enables the enterprises to launch a new version of the app with customer-solicited requirements and thus making the usage of the app more functional and user-friendly for ultimately increasing the app's overall popularity in the app market and gain monetary profits [7]. Previously, researchers have utilized various approaches to analyse and gain insights on the feedback mentioned in app reviews [8, 9]. However, these approaches analyse the feedback from a single dimension (e.g., analysing frequency of entities of interest or ratings only), a research gap that needs to be addressed. Along with this, to our best knowledge, only one study addressed the prioritization of reviews [10]. Therefore, we further investigate and address the issue of ranking numerous app reviews by taking inspiration from the requirements prioritization domain, domain knowledge made available by app developers and the machine-learning field [11, 12].

Furthermore, in this study, we explore, develop and utilize data analytics methods to gain insights on the reviews of MyTracks app [13] and implement a method to prioritize its reviews. The prioritization process significantly assists the app developers in resolving the issue of prioritizing app reviews [14]. This process is crucial when the app developers are dealing with numerous reviews and want to decide on which review to address first in the app maintenance and evolution cycle. Addressing of reviews based on their severity or importance ultimately leads towards the rectification of critical customers' requirements present in those reviews [6].

Thus, the prime contributions of this study are:

1. Data analysis and visualization methods to assist app developers in gaining useful insights on the performance of their app in the market and subsequently identifying the requirements of their customers.

2. A method to prioritize the reviews so that the app developers can initiate the necessary remedial actions for app maintenance and evolution cycles.

We evaluated the usability, reliability and efficiency of our methods incorporated in a tool and results generated by the tool at an enterprise level through the app developers using the System Usability Scale [15].

The further sections of this study are as follows: In section II, we report the related studies and mention the crucial research gaps. In section III, we describe the methods that assist in the data analysis, visualization, and reviews prioritization. In Section IV, we provide the details regarding the experimental setup and the process related to the evaluation of our tool performed by the app developers. We report our results pertaining to the methods in section V and document our discussions based on the results in section VI. We provide our concluding remarks in the Conclusion section followed by the future work.

## II. Related Works

Prior works has performed analysis of app reviews but have utilized only one dimension. For instance, Guzman et al. [8] have used frequency analysis to find keywords pertaining to an app that depict app concerns. In another example, Fu et al. [9] have used ratings as a means to analyse reviews that required attention. Similarly, Parrot [16] has introduced the concept of emotion (sentiments) analysis to understand 'embedded' customer's emotions related to product usage. However, such approaches being operational from a single dimension miss on the other information that exists in other dimensions and potentially fail to provide the meaningful interpretations by not performing the analysis of an entity of interest from a multidimensional perspective. Additionally, single dimensional analysis may be error-prone due to the presence of false-positives or false-negatives in the reviews analysis process [17]. For example, analysing a product feature from only a single rating perspective.

Further, we investigate the issue of prioritizing the reviews so that the enterprises can address them based on their importance or severity. To our best knowledge, only one study has managed to prioritize app reviews. Chen et al. [10] have developed AR-Miner which prioritizes the groups' app reviews. The authors' initially extract a group of informative reviews and later classify them using the unsupervised Latent Dirichlet Allocation (LDA) algorithm. Finally, the authors use time-series patterns, ratings, and volume of reviews present in each group to prioritize the unlabelled groups of reviews. However, this study has limitations in its prioritization application. Due to the unsupervised clustering, the dynamically created groups hold duplicate reviews, thus questioning the priorities of the groups. More to this, the groups are unlabelled, hence, they do not depict generalized information of the reviews present in it. Additionally, the prioritization process does not consider the enterprises' inputs (priority preferences) on the priority of the groups. Enterprises' inputs depicting the priority of each review is of key significance as the enterprises possess the necessary domain knowledge pertaining to their developed app, and are in a position to understand what app concerns need to be addressed first for the app to sustain and grow in the competitive market [18].

In this study, we address the gaps mentioned above. Initially, we develop and evaluate data analysis and visualization methods that operate on multiple dimensions to provide fine-grain insights into the customer feedback present in the app reviews. Next, we develop a method to prioritize the reviews based on the priority levels submitted by the app developers (enterprise personals). Furthermore, utilizing a suitable machine-learning algorithm (Multinomial Naïve Bayes) we reduce the app developers' efforts towards labelling the priority level of each review and automate the reviews prioritization process. In the next section, we describe these methods accordingly.

## III. Methods

In this section, we propose the methods that drive the data analysis, visualization and prioritization process.

### A. Text Preprocessing

Initially, we perform preprocessing of the reviews by converting the reviews into the relevant word vectors [19]. This, in general, is achieved by the removal of: unwanted whitespaces, special characters, numbers and punctuations before converting the words in lower case. After which stop-words are removed and there is stemming of the final set of words [20]. Thus, the outcome of this process leaves us with the 'keywords' of interests that can be used for analysis and prioritization purposes [21].

### B. Frequency and Word Cloud

Firstly, we utilize frequency information of keywords and represent this information in the form of a word cloud [22]. A word cloud is a visual representation for text data, typically used to depict keyword-frequencies extracted from the text corpus. Keywords are single words and the importance of each keyword is shown with font size. Word cloud is implemented using a session variable that is created for a matrix and data frames. Every filtered keyword is analysed and the frequency of each descriptive keyword in the review is calculated. The frequencies are placed in a matrix in a decreasing order while the keywords are placed in a data frame. The frequencies in the matrix are mapped to the corresponding keywords in data frames. The analysed keywords are then flushed, and the new keywords are analysed thereby incrementing the frequency counter as per the occurrence. The keywords along with their frequencies are stored in the form of a list and can be individually accessed by means of indices. Word clouds offer a good structure to perform visualization of the data.

### C. Association Mining based on Correlation

Next, we develop a correlation method to find keywords that are associated with a specific keyword in terms of their contextual semantic similarity [23]. The objective of this process is to achieve word sense disambiguation as it is observed in the case of app reviews, app's customers tend to mention keywords within close proximity to each other that are similar in terms of their contextual meaning [24, 25]. For instance, consider the review, "*phone screen keeps crashing*". In this review, keywords such as 'phone', 'screen', 'crashing' appear in close proximity to each other which in turn reflects the fact that the app causes the phone screen to crash. Such associations are of significant importance to app developers as they tend to highlight issues or requests pertaining to the app features. That said, the aim of this method is to find keywords that are closely associated with the specific keyword of interest (mostly app features) based upon correlation to understand what exactly the specific keyword commonly specifies. The method is elaborated as follows:

Consider the preprocessed review 'r' of length 'n' in an array r[0], r[1],.....,r[n-1]. The keywords 'k' occurring in the preprocessed review are seen as k[0], k[1],.....,k[n-1]. For each keyword k[i], a vector of length N-1 is defined:

X[i] = array ()

The vector X[i] is calculated as:

**Data:** $X$ as a co-occurrence matrix with keyword proximities ranging from 0-1
**Result:** $Z$ as a vector-space of keyword-pairs with corelations of 0.5-1

```
for i ← 0 to X[n] by 1 do
    for j ← 0 to X[n] by 1 do
        if X[i] >= 0.5 OR X[j] >= 0.5 OR X[i + j] >= 0.5 then
            Z[i][j] ← i&j
        else
            Z[i][j] ← 0
```

The $i^{th}$ data of the vector is 1 if the keyword is the i or the i+1 keyword, and 0 in the other case. In actual operations, the correlation coefficient between two keywords k[a] and k[b] is computed first, by performing a dot product between X[a] and X[b] (represents the number of times the words are adjacent) which is then divided by the lengths (square root of the number of frequency of the keyword). In this case, the correlation coefficient is given as input, hence associated keywords could be found out. Furthermore, correlation (X[a], X[b]) is larger if k[a], k[b] have more degree of adjacency.

### D. Sentiment Analysis

Sentiment analysis is a method often used by enterprises to help them understand if their product is liked or disliked by customers [26]. However, many works on sentiment analysis tend to focus only on one dimension, i.e., performing analysis of individual reviews to generate the sentiment scores reflecting satisfaction or dissatisfaction of a product or service [27-29]. This approach, in general, raises the question:

"*Is it possible to predict the positivity or negativity of customers' opinions just by counting words?*"

In this study, we address this issue by investigating and evaluating a specific sentiment analysis method, and later check to see the extent to which the proposed method can predict a customer's rating based on their written opinion. In addition, we investigate and analyse the discrepancies between customer ratings and the computed sentiment scores of the keywords. For performing sentiment analysis on each review, the support of a lexicon [30] and its subsequent word library is required, which provides a score for each word from most negative to most positive. For example, the word lists provided by the utilized lexicon library indicate word 'renounce' with a score of -2 (negative), word 'hate' with a score of -3 (more negative) and the list goes on for the rest of the words present in the English dictionary. The main objective of this method is to analyze and evaluate the sentiments of the customers based upon the reviews and ratings the customers provide. Hence, for the sentiment analysis method, the data belonging to the preprocessed reviews and ratings are considered. Keywords are first classified as 'positive' or 'negative', then the value of each keyword is averaged to categorize the entire review set. For the dataset used in the study, there is a collection of many reviews, each accompanied by a 1-5-star rating submitted by the app's customers. The final objective is to validate the app customers' ratings based upon the written reviews and visualize the keywords that are associated with the average ratings of the reviews.

### E. Prioritization of App Reviews

Our final objective is to prioritize the reviews based on human judgements of domain experts or enterprise personals i.e., app developers, product managers or other stakeholders. Human judgements obtained through domain experts or enterprise personals are crucial towards the prioritization process as the domain experts or enterprise personals have the necessary domain knowledge that indicates which review needs to be addressed first based on its importance or severity. The importance can be determined based on time or resource constraints, budget, market or business values etc. However, as the app reviews are numerous in nature, it is not possible for the domain experts or enterprise personals to prioritize each review manually based on a Likert scale [31]. The Likert scale helps to determine the degree of importance or severity of a review [32]. For instance, a review labelled as '1' is of most important than a review labelled '10' which has no importance at all. Manually labelling each review can be a laborious, arduous and time-consuming process, and may potentially introduce errors. We automate the reviews priority labelling process using the Multinomial Naïve Bayes which is a well-known machine learning algorithm used in software engineering applications and outperforms other machine learning algorithms when the data under processing is textual data i.e., data existing in natural language form [33, 34]. Initially, the app developers label the reviews based on a 10-point Likert scale that ranges from levels '1' to '10'[11]. The review labelled '1' is termed as 'most important', whereas as the review labelled '10' is termed 'unimportant'. The degree of the importance of the reviews decreases as the labelling range increases from 1 to 10. After the app developers label a substantial amount of reviews, these reviews along with their associated labels are given as input in the form of learning data to Multinomial Naïve Bayes algorithm to predict the priorities of the remaining reviews. For the given application, the aim of Multinomial Naïve Bayes is to predict the priority ($p_n$) of a review which is given as:

$$P(p_n) = \text{Number of reviews having priority } p_n \text{ / Total number of reviews} \quad (1)$$

where p = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Next task is to calculate the maximum likelihood estimation which is given as:

$$P(k_i|p_n) = \text{Total number of times a keyword } k_i \text{ occurs for a priority } p_n \text{ / Total number of keywords k in the reviews having priority } p_n \quad (2)$$

Finally using equations (1) and (2) the priority of an unknown review R can be calculated using:

$$Pmap = \text{argmax}_p\, P(p) * \Pi i\, P(k_i|p) \quad (3)$$

In equation (3), the most probable priority for a review is determined as the arguments of the maxima over all the priorities of the priori times the maximum likelihood estimates. That said, we tweak the equation (2) to incorporate Laplace smoothing to avoid the situation of conditioning away of any zero probabilities [35]. This, in turn, increases the overall accuracy of Pmap in determining the priority of a review. Thus, we subject equation (2) to Laplace smoothing

that adds a count of 1 to its numerator and denominator to handle zero counts of keywords that it does not encounter in its learning phase. This allows Multinomial Naïve Bayes method to not lose counts of the keywords in determining the priority of a review. The learning phase of the Multinomial Naïve Bayes is as follows:

1. From the manually prioritized pre-processed reviews, extract keyword information (I)
2. Calculate $P(p_n)$ terms
   2.1 For each $p_n$ in p do:
      2.1.1 Get all the reviews with priority $p_n$
      2.1.2 Calculate $P(p_n)$
3. Calculate $P(k_i|p_n)$ - maximum likelihood estimate i.e., for every keyword ki, given each priority $p_n$ :
   3.1 Create keywords space containing keywords belonging to reviews with priority $p_n$
   3.2 For each keyword $k_i$ in I do:
      3.2.1 Calculate the total occurrence of $k_i$ in the keywords space
      3.2.2 Calculate $P(k_i|p_n)$ using Laplace smoothing

In the next section, we highlight the details regarding our experimental settings.

## IV. Experimental Setup

To conduct the experiments, we utilized the dataset pertaining to MyTracks app [13]. This app is useful for tracking routes and is commonly used by app customers for outdoor activities such as jogging, walking, trekking, etc. The dataset of MyTracks was made available to us by the MyTracks app developers. The dataset consisted of 18,289 reviews, each accompanied by a customer rating in range 1-5. Furthermore, to test the accuracy, precision, recall, and F-measure [36] of Multinomial Naïve Bayes, the app developers had already prioritized the reviews present in the dataset using the Likert scale which was in range 1 to 10. Next, we explain our metrics. Accuracy indicates the correctly prioritized reviews among the total number of reviews (i.e., true positive entries + true negative entries / total entries). Precision identifies the reviews that were correctly predicted to have the specific priority level among the total number of reviews that were predicted to have the specific priority level (i.e., true positive entries / true positive entries + false positive entries). Recall indicates the reviews that were correctly predicted to have the specific priority level among the total number of reviews that had the actual specific priority level (i.e., true positive entries / true positive entries + false negative entries) and F-measure is the harmonic mean of precision and recall (i.e., (2 * precision * recall) / (precision + recall)) that reflects the robustness of Multinomial Naïve Bayes. The already prioritized dataset (reviews labelled with priority levels) provided by the app developers acted as a 'ground truth' to evaluate the validity of the priorities of the reviews predicted by the Multinomial Naïve Bayes machine learning algorithm. That said, we implemented and evaluated the methods mentioned in Section III in the form of a tool using the R software with the support of additional libraries such as text mining, AFFIN lexicon, e1071 and Shiny [37-40]. Furthermore, for validating the application of the Multinomial Naïve Bayes we randomly split the collection of pre-processed reviews into a training set which was used to learn the Multinomial Naïve Bayes for MyTracks reviews with their associated priority levels and a testing set which was used to evaluate its performance in predicting the priorities of unknown reviews. Each experiment was repeated 10 times using the stratified 10 fold cross-validation mechanism to obtain average scores of accuracy, recall, precision, and F-measure [36].

Additionally, we evaluated the usability, reliability and efficiency of the tool through the System Usability Scale (SUS) [15]. The prime objective of this evaluation was to understand how intuitively easy it was for the app developers to use the tool and interpret the results, and along with this, determine if the results generated by the tool were reliable and efficient. Firstly, we briefed three app developers of our methods and tool. Next, we allowed the app developers to use our tool and interpret the generated results. Finally, the app developers were requested to answer 10 questions pertaining to SUS on a five-point Likert scale of 'Strong Agreement' to 'Strong Disagreement'. These scores were later averaged to determine the usability, reliability and efficiency of our tool. The SUS questionnaire consisted of the following questions mentioned in Table I whose responses from app developers were measured using the Likert scale mentioned above.

TABLE I. SUS Questionnaire

| ***ID.*** | ***Question*** |
|---|---|
| 1. | Do you think that you would use the developed tool frequently? |
| 2. | Did you find the results generated by the tool easy to analyze? |
| 3. | Did you think the tool was easy to use? |
| 4. | Do you think that you would not need any assistance to use the tool? |
| 5. | Did you find the results generated by the tool easy to interpret? |
| 6. | Do you think the results generated by the tool were reliable and efficient? |
| 7. | Do you think that most app developers would not require any prior training to understand this tool and the results it generates? |
| 8. | Did you find the results practical and useful for app maintenance and evolution cycles? |
| 9. | Did you feel confident using the tool? |
| 10. | Do you think you do not need to learn a lot of things before you could get along with the tool? |

Next, we report our results in the next section.

## V. Results

To begin, we performed the pre-processing of reviews mentioned in Section II.A. before running any data analytics or prioritization method. This process is central to avoiding the garbage in (non-useful input) and garbage out (non-useful results) conditions while conducting experiments that involve natural language processing [41]. Firstly, we initiated our tool and performed analysis of the reviews based on the first method followed by the others. Finally, we performed the prioritization of reviews and later allowed the app developers to utilize and evaluate our tool based on SUS. In this section, we report the results generated by our tool and

the tool's evaluation outcome submitted by the app developers.

### A. Frequency and Word Cloud analysis

Firstly, we sorted the keywords based on the decreasing order of their frequency of occurrences. Next, we generated a word cloud of the top 100 frequently occurring keywords. Figure 1 reflects the word cloud wherein the size of the keywords is set according to their frequency of occurrences.

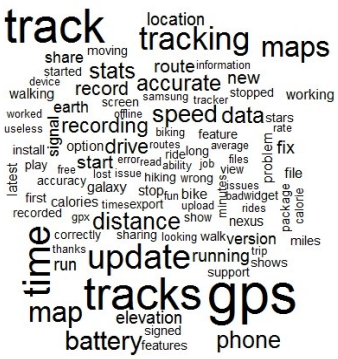


Fig. 1. Word cloud analysis based on frequency of keywords

Such analysis is crucial when the app developers are trying to figure out the most frequently mentioned app features that raise certain problems or requests. Both app features or the problems or requests can be identified using such analysis however the relationship among them can be identified using the correlation method mentioned in Section II.C to gain insights on what do the product features or the problems or issues reflect.

### B. Association Mining based on Correlation Analysis

The frequency based word cloud analysis provided insights on the keywords that occurred most frequent throughout the reviews. However, to gain further insights on what the keywords actually portray, we perform the correlation analysis on the keywords of interest. We selected 'GPS' and 'Battery' as the app features for correlation analysis. Our selection was further supported by app developers. Table II indicates the top 5 keywords that correlated with 'GPS' in terms of their contextual similarity whereas its second entry indicates the keywords that correlated with 'Battery'. For both cases, the lower limit of correlation was set to 0.5 which is a standard threshold limit [23].

TABLE II. ASSOCIATION OF GPS AND BATTERY WITH OTHER KEYWORDS

| *Keyword of interest* | *Keyword 1* | *Keyword 2* | *Keyword 3* | *Keyword 4* | *Keyword 5* |
|---|---|---|---|---|---|
| GPS | drops (0.9) | lost (0.8) | fading (0.8) | error (0.8) | losing (0.7) |
| Battery | drains (0.9) | draining (0.9) | usage (0.8) | force shut (0.8) | kills (0.6) |

Initially, we interpret the results of Table II. From the given correlation analysis, it is evident that the GPS signal in the app keeps fading or losing thus the app's customers are generally complaining about its regular unavailability or errors caused in positioning due to its irregularity. Similarly, it can be concluded that the app generally tends to consume the phone's battery, a concern that needs immediate attention from the app developers. Subsequently, other specific keywords of interest can be targeted for further investigation with the assistance of word cloud analysis, and their associations with other relevant keywords could be found out. Additionally, we extracted the top most frequent keywords and initiated a correlation analysis. In the process, the individual top keywords were combined with the respectively associated keyword producing a paired term, which was analysed through word cloud. Figure 2 reports the analysis of such operation and indicates that the first two can be combined for a generalized analysis.


Fig. 2. Strongest word pair visualization among the top 10 frequent words

From figure 2, it is evident that the battery issue persists as a major level of concern. The other keyword word pairs can be accordingly interpreted by the app developers as they possess the necessary domain knowledge.

### C. Sentiment Analysis

Next, based on the method mentioned in Section II.D an average sentiment alongside the ratings is calculated which predicts the user opinions towards the MyTracks app. Figure 3 shows the correlation of sentiment scores generated from the reviews with those of ratings. It was observed from figure 3 that the sentiment scores correlated with the ratings.

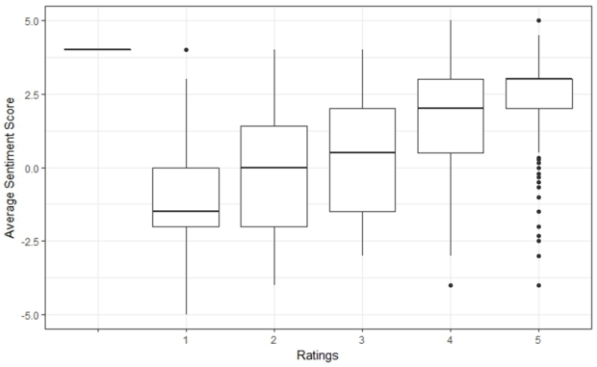


Fig. 3. Box plot representation of sentiment scores correlating with ratings

Further, to find out the keywords that appear in the positive or negative reviews, it was necessary to create a per keyword summary, which required more grouping iterations. Figure 4 indicates the keywords that occur in the reviews subjected to specific ratings. Such analysis can pinpoint product features or problems that are rated low by the customers and requires attention from the app developers.

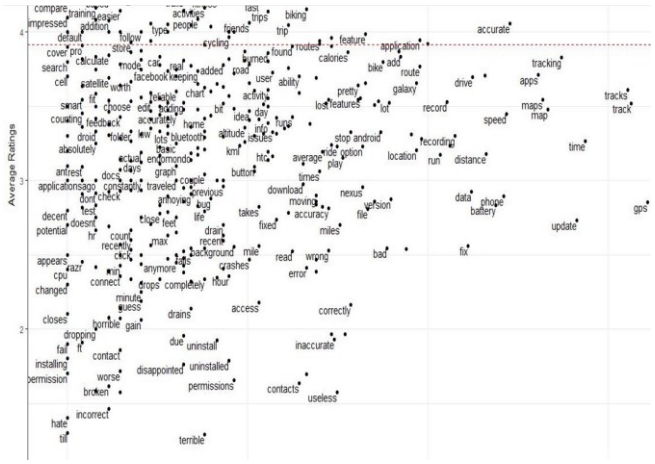


Fig. 4. Preview of plot depicting actual keywords against average ratings

To check the authenticity of the computation of a positive or a negative review and to measure the discrepancy between the ratings and customers sentiments, the outcome of the sentimental analysis method is subjected to an accuracy test. Figure 5 indicates the alignment of customers' keywords having positive, negative or neutral sentiments with the sentiment scores assigned by the AFFIN sentiment lexicon.

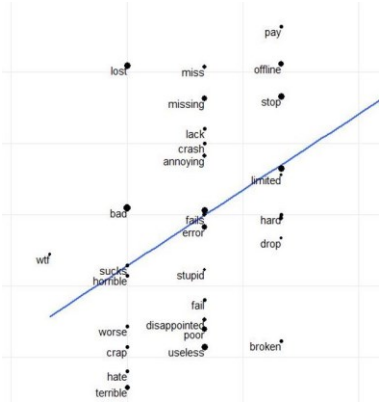


Fig. 5. Preview of plot depicting accuracy of sentiment analysis based on ratings and the extracted keywords (sentiment score vs average ratings)

Regarding the sentiment accuracy, average standard deviation error was found to be 0.37 (out of 1) which suggests that the majority of the keywords portraying sentiments of customers correlated with the assigned ratings and the sentiment scoring mechanism of AFFIN. The mismatched alignments indicate discrepancies between the customer's rating and review. For instance, an interesting observation is made from the above analysis, consider the word 'stop', having a negative sentiment score but has more than a 3 rating from the app's customers for their particular reviews. This case was further investigated, an association analysis was done and it was found out that this word correlated with 'accidental - 0.9', 'guaranteed - 0.8', 'resulting - 0.8', 'accidentally - 0.8', 'recording - 0.7', 'clicks - '0.7', 'printing - 0.7','accident - 0.7', 'hour - 0.6', 'accidently - 0.6', 'occasionally- 0.6'. It seems that the app's customers do complain about some accidental stops in some functionality (recording/printing) of the app or the app itself might be hanging (freeze) but still the customers have given a good rating (above 3) to the app.

### *D. Prioritization of App Reviews Results*

Next, we reveal the results of the 10 times stratified 10 fold cross validation experiment in Table III.

TABLE III. AVERAGE RESULTS OF 10 TIMES STRATIFIED 10 FOLD CROSS VALIDATION EXPERIMENT

| *Priority Level Label* | *Accuracy (%)* | *Precision (0-1)* | *Recall (0-1)* | *F-Measure (0-1)* |
|---|---|---|---|---|
| 1 | 87.48 | 0.81 | 0.92 | 0.85 |
| 2 | | 0.81 | 0.91 | 0.86 |
| 3 | | 0.81 | 0.90 | 0.86 |
| 4 | | 0.81 | 0.92 | 0.86 |
| 5 | | 0.80 | 0.91 | 0.86 |
| 6 | | 0.81 | 0.91 | 0.86 |
| 7 | | 0.80 | 0.90 | 0.85 |
| 8 | | 0.80 | 0.91 | 0.85 |
| 9 | | 0.81 | 0.91 | 0.86 |
| 10 | | 0.81 | 0.92 | 0.86 |

From the findings reported in Table III, it is evident that the average accuracy of review priority prediction was found to be 87.48%. Next, we report the precision, recall and F-measure of the Multinomial Naïve Bayes in predicting the 10 priority levels. For each priority level, the precision of Multinomial Naïve Bayes was found to be in the range of 0.80-0.81. Similarly, the recall was found to be in range 0.90-0.92. Subsequently, F-measure was in the range of 0.85-0.86. These readings show that the results generated by Multinomial Naïve Bayes are consistently reliable and definitive, thus making the application of Multinomial Naïve Bayes suitable for review priority prediction purposes.

### *E. Usability of Analytics and Prioritization Methods*

Next, we allowed the app developers to evaluate our analytics and prioritization tool, and later we provided our SUS questionnaire to three app developers to determine the usability of our methods and the results generated by those methods. The average score assigned by three app developers to the SUS questionnaire was found to be '4.3' which indicates a significant level of agreement, thus confirming the usability, reliability and efficiency of our tool.

In the next section, we provide discussions pertaining to the results.

## VI. DISCUSSIONS

To begin, our proposed analytics methods (refer to section II.B, II.C, and II.D) can be combined in different ways to perform analysis of app reviews from multiple dimensions. For instance, as observed from our results section word cloud can be combined with correlation analysis or correlation analysis can be combined with sentiment analysis method. Such analysis approaches based on the methods extract meaningful insights from the reviews

to app developers to fix the requirements of the app. Additionally, such approaches reveal crucial interrelated information that is not fully revealed by traditional single dimension analysis methods. The revelation of such information provides in-depth details of the requirements of the customers that majorly reflects the requested or problematic app features. Subsequently, in our study through the means of sentiment analysis, we showed that customers majorly tend to be true to their assigned app ratings, however, in such analysis certain outlier cases are also observed (refer to Section V.C).

Next, we performed the prioritization of reviews based on the truth set provided by the app developers. Our primary objective was to evaluate the accuracy of Multinomial Naïve Bayes in correctly predicting each of the ten priority levels of the respective reviews using the accuracy, precision, recall and F-measure metrics. It was observed that the machine learning algorithm delivered an excellent performance in terms of prediction. Given the number of app reviews to prioritize in a short period of time, on an average basis, the Multinomial Naïve Bayes was 0.81 precise in predicting the specific priority levels, 0.91 accurate in terms of recall, and 0.86 accurate in terms of F-measure.

That said, the app developers of MyTracks were satisfied with the usability, reliability and efficiency of our tool based on the SUS score. The evaluation results suggested that the app developers found the tool usable for performing the necessary analytics and prioritization of app reviews. Additionally, SUS seems like a valid approach in evaluating the usability of any developed software artefact.

## VII. Conclusion And Future Work

To conclude, in this study, we identified the research gaps pertaining to the analysis and prioritization of app reviews. Based on which, we proposed and implemented analytics and prioritization methods. Further, we incorporated these methods in a tool and evaluated the usability of the tool through app developers using the System Usability Scale approach. Based on the SUS score the MyTracks app developers found the tool usable.

Our next objective is to explore additional semantic similarity concepts for discovering associations among the keywords and empirically evaluate them to validate their application in our tool. Next, we plan to investigate the semi-supervised Expectation Maximization of Multinomial Naïve Bayes machine learning algorithm which demands less labelling efforts from app developers to label the priority levels of reviews for training purposes and is claimed to be more accurate than supervised Multinomial Naïve Bayes machine learning algorithm. Finally, we plan to investigate more supplementary analytics methods to incorporate in our tool and check any unexplored dimensions. Furthermore, we plan to evaluate the application of our tool on reviews pertaining to other apps. We also believe the application of our tool can be subjected to other domains such as e-commerce, hotel reviews etc.

## Acknowledgment

We sincerely thank the app developers of MyTracks app for their continuous support and collaboration. Additionally, this work was funded by a University of Otago Research Grant (UORG) – accessed through the University of Otago Research Committee.

## References

[1] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010: ACM, pp. 179-194.

[2] C. Clifford. "By 2017, the App Market Will Be a $77 Billion Industry." https://www.entrepreneur.com/article/236832 (accessed.

[3] Statista, "Number of apps available in leading app stores as of March 2017," ed, 2017.

[4] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? Classifying user reviews for software maintenance and evolution," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sept. 29 2015-Oct. 1 2015 2015, pp. 281-290, doi: 10.1109/ICSM.2015.7332474.

[5] R. P. d. Oliveira and E. S. d. Almeida, "Requirements Evolution in Software Product Lines: An Empirical Study," in *2015 IX Brazilian Symposium on Components, Architectures and Reuse Software*, 21-22 Sept. 2015 2015, pp. 1-10, doi: 10.1109/SBCARS.2015.11.

[6] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward Data-Driven Requirements Engineering," *IEEE Software,* vol. 33, no. 1, pp. 48-54, 2016, doi: 10.1109/MS.2015.153.

[7] W. M. F. S. Y. J. Y. Z. M. Harman, "A Survey of App Store Analysis for Software Engineering," *IEEE Transactions on Software Engineering,* vol. 43, no. 9, pp. 817 - 847, 2016.

[8] E. Guzman and W. Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 25-29 Aug. 2014 2014, pp. 153-162, doi: 10.1109/RE.2014.6912257.

[9] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: making sense of user feedback in a mobile app store," presented at the Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, Chicago, Illinois, USA, 2013.

[10] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: mining informative reviews for developers from mobile app marketplace," presented at the Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 2014.

[11] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. r. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology,* vol. 56, no. 6, pp. 568-585, 2014/06/01/ 2014, doi: https://doi.org/10.1016/j.infsof.2014.02.001.

[12] A. McCallum and K. Nigam, *A Comparison of Event Models for Naive Bayes Text Classification*. 2001.

[13] "My Tracks." https://play.google.com/store/apps/details?id=com.zihua.android.mytracks (accessed 2018).

[14] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Information and Software Technology,* vol. 43, no. 14, pp. 883-890, 2001/12/15/ 2001, doi: https://doi.org/10.1016/S0950-5849(01)00194-X.

[15] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction,* vol. 24, no. 6, pp. 574-594, 2008.

[16] W. G. Parrott, *Emotions in Social Psychology: Essential Readings*. Psychology Press, 2001.

[17] N. Jindal and B. Liu, "Review spam detection," in *Proceedings of the 16th international conference on World Wide Web*, 2007: ACM, pp. 1189-1190.

[18] P. Berander and A. Andrews, "Requirements Prioritization," in *Engineering and Managing Software Requirements*, A. Aurum and C.

Wohlin Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 69-94.

[19] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM,* vol. 18, no. 11, pp. 613-620, 1975.

[20] C. Aggarwal, & Zhai, C., *Mining Text Data*. Springer Science Business Media, 2012.

[21] R. S. Chea, J. Morris, and G. Prabhu, "System for dynamic product summary based on consumer-contributed keywords," ed: Google Patents, 2009.

[22] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu, "Context preserving dynamic word cloud visualization," in *2010 IEEE Pacific Visualization Symposium (PacificVis)*, 2010: IEEE, pp. 121-128.

[23] A. Ceglar and J. F. Roddick, "Association mining," *ACM Computing Surveys (CSUR),* vol. 38, no. 2, p. 5, 2006.

[24] Y. Karov and S. Edelman, "Similarity-based word sense disambiguation," *Comput. Linguist.,* vol. 24, no. 1, pp. 41-59, 1998.

[25] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 15-19 July 2013 2013, pp. 125-134, doi: 10.1109/RE.2013.6636712.

[26] X. Fang and J. Zhan, "Sentiment analysis using product review data," *Journal of Big Data,* vol. 2, no. 1, p. 5, 2015.

[27] V. K. Singh, R. Piryani, A. Uddin, and P. Waila, "Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification," in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, 2013: IEEE, pp. 712-717.

[28] W. Kasper and M. Vela, "Sentiment analysis for hotel reviews," in *Computational linguistics-applications conference*, 2011, vol. 231527, pp. 45-52.

[29] C. W. Leung, "Sentiment analysis of product reviews," in *Encyclopedia of Data Warehousing and Mining, Second Edition*: IGI Global, 2009, pp. 1794-1799.

[30] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics,* vol. 37, no. 2, pp. 267-307, 2011.

[31] I. E. Allen and C. A. Seaman, "Likert scales and data analyses," *Quality progress,* vol. 40, no. 7, pp. 64-65, 2007.

[32] P. Voola and A. V. Babu, "Comparison of requirements prioritization techniques employing different scales of measurement," *SIGSOFT Softw. Eng. Notes,* vol. 38, no. 4, pp. 1-10, 2013, doi: 10.1145/2492248.2492278.

[33] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," presented at the Proceedings of the 23rd international conference on Machine learning, Pittsburgh, Pennsylvania, USA, 2006.

[34] T. Wang and W.-h. Li, "Naive bayes software defect prediction model," in *2010 International Conference on Computational Intelligence and Software Engineering*, 2010: Ieee, pp. 1-4.

[35] D. Lowd and P. Domingos, "Naive Bayes models for probability estimation," in *Proceedings of the 22nd international conference on Machine learning*, 2005: ACM, pp. 529-536.

[36] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information retrieval,* vol. 1, no. 1-2, pp. 69-90, 1999.

[37] "R." https://www.r-project.org/ (accessed 2018).

[38] D. Meyer, K. Hornik, and I. Feinerer, "Text mining infrastructure in R," *Journal of statistical software,* vol. 25, no. 5, pp. 1-54, 2008.

[39] K. Z. Aung and N. N. Myo, "Sentiment analysis of students' comment using lexicon based approach," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017: IEEE, pp. 149-154.

[40] C. Beeley, *Web application development with R using Shiny*. Packt Publishing Ltd, 2013.

[41] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science,* vol. 349, no. 6245, pp. 261-266, 2015, doi: 10.1126/science.aaa8685.