User Concern in App Reviews, a study of perceived privacy violation among user sentiments and other contribute factors

by

Yue Cheng

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

User Concern in App Reviews, a study of perceived privacy violation among user sentiments and other contribute factors

by

Yue Cheng

B.Sc., Mount Royal University, Calgary, 2017

Supervisory Committee

Dr. Daniela Damian, Co-supervisor
(Department of Computer Science)

Dr. Neil Ernst, Co-supervisor
(Department of Computer Science)

ABSTRACT

Privacy, a significant factor in software usage, also provides software developers with additional insights into how applications can be improved. However, it is a delicate matter that peeks into user behaviour to the amount of information they are willing to share. With the rise of mobile applications, another concerning factor of user information collection also became prominent. The existence of user chatter on the google app store can help identify whether privacy concern is problematic or not. However, little research has been conducted to study privacy violations and their contributing factors. In this project, we proposed using an LDA based privacy identification model that assesses the factors relating to user concerns with privacy matters on the Google App Store user reviews. A total of 45,114,727 rows of data were scraped from the google play store, which was later filtered and processed into workable data. With the help of the Gensim LDA library, we can identify a coherence score of 0.604 and eight topics of various subjects. We later arranged these subjects into their corresponding categories, which could be used to analyze why specific privacy terms are more sensitive while others are not.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the beginning of time, software privacy has always been controversial. Until recently, various privacy laws set in place to protect the consumers have come into full force worldwide. Some of the more well-known privacy laws include the GDPR in the EU and privacy law in California(CCPA). They provide a basis for what a software company can and can not collect. This would influence both functional and non-functional requirements with regard to software engineering. When understood, privacy guidelines would ensure stakeholders' value is unimpaired. Thus, making an application efficient. Privacy within software engineering governs how the application will behave during its operation. Privacy also signifies the aim of the developers as to the delegation between functionality and intrusiveness. If we look at an app's usage, we can see where one user considers a specific action an invasion of privacy while another feels otherwise no so. There are many ways the app can violate a user's privacy, with the most common being collecting personal data without the user's knowledge or using system hardware without asking for permission. We are interested in the effect of these privacy laws on the end-users, mainly if they think and react towards privacy changes when using and reviewing apps. In an ideal world, when privacy becomes law in a jurisdiction, people in that region should scrutinize an organization to ensure that the app obeys privacy laws. However, there is still a disconnect between those who create the app that adheres to these rules and those who use them. With this in mind, we hope to uncover whether there is any indication of people talking about privacy in the app reviews and, if so, by how much.

Besides privacy, we also want to understand users' concerns about their apps. In the ever-expanding world of app development, the app store has become a platform to download and install mobile applications and for users to voice their opinions on

subjects regarding their experiences. This can provide detailed insight into comparing the users' sentiment towards an app, thus a better understanding of the trade-offs between acceptances of risks regarding privacy and functionality from an observer's point of view.

We hypothesize most users should have a level of privacy demands in the apps they use, but it is not clear whether users reference privacy in their reviews of apps. We also believe that there is a difference in user concern over application functionality and performance and that privacy would rank accordingly based on the number of user-produced reviews.

1.1 Research questions

RQ1: Do users discuss privacy in app reviews?

In looking into how many reviews are related to privacy issues, we can identify the aspect of privacy engineering in its current state within software engineering. From there, we can also peek into the nature of these discussions, as it will provide an understanding of the severity of privacy being mentioned. These would ultimately pave the way toward the root of the problem. Does privacy exist(or not) in the user discussions? For which we are expected to find clear and identifiable data from the experiment conducted.

RQ2: What does the user concern with besides privacy?

Generally, we hope to gain insights into the negative sentiment of user reviews. Identifying the factors/topics users are more concerned with can help app developers make better applications. This process would also demonstrate that we can delve deeper into what users perceive as priority factors in the current state of software development. In contrast, privacy constitutes a high level of importance in software development. However, other factors such as performance and functionality are equally as important, but how these factors are ranked presents a curious case for research. By examining the amount of each, we hope to understand the foremost concern of the user community.

Chapter 2

Related Work

We present the related work on analyzing user app reviews. The current literature comprises four major categories: sentiment analysis, NFR classification, app improvement, and forum question analysis. Sentiment analysis has always been an integral part of ML-related software development strategies.

2.1 Sentiment analysis

Since sentiment usually involves the polarity of a word or phrase, it is possible to use it to help understand whether a review is considered privacy-related or not [3]. Emotional context also plays a part in understanding how users react to bugs and issues regarding functionality in the apps [10].

2.2 NFR classification

Lastly, in comparing different classifiers of app reviews, we can correctly identify what needs to be used and for what purpose in terms of this experiment [9]. NFR classification [7] helps distinguish the bulk of the reviews into their own unique types as to how the user feels something that's privacy-related rather than simply a usability issue.

2.3 App improvement

Most papers aim to refine feature suggestions, classify reviews, and even detect fake reviews for app improvement. Using reports generated from the ML technique applied to user reviews, the developers can build better apps by identifying interesting patterns and automatically patching them [8, 5, 14, 6, 16]. In contrast, others are focused on dealing with issues that exist within the app via user response [12]. Another interesting take was on the detection and identification of fake use reviews. It does so by comparing real reviews with alleged fake ones [11]. The researchers also provided questionnaires to the fake review companies to see how they give this type of service. This type of research helps in the development of a simple classifier to automatically detect fake reviews in app stores.

2.4 Forum question analysis

As for asking technical questions to get effective feedback [4], the study provides a closer look as to why some StackOverflow questions are successful while others are not so much. This was conducted via a conceptual framework for ranking the successful SO questions and understanding the internal reason of those who participate in the communities.

Chapter 3

Methodology

We selected a cohort of app reviews from the Google Play store for our investigation of user app reviews regarding user privacy concerns and discussion. We first filtered the app reviews by the number of privacy keyword mentions. This step allowed us to collect a set of privacy-leaning reviews. We manually labelled a sample of these privacy-leaning reviews and produced a Naive Bayes classifier to automatically classify reviews as either privacy-related or not using machine learning. Our identification of privacy-related app reviews allowed us to answer our RQ1. The process is done via the development of our Naive Bayes classifier for labelling app reviews. This then sets the basis for creating another set of identifiable review data ranges that helped answer RQ2. By selecting the data as a whole, we can apply Latent Dirichlet allocation algorithms to the data range to derive valuable topics. In cataloguing these topics, we can provide a detailed conjecture as to which topics are most concerned and least concerned besides the information obtained from the ML classified data.

3.1 Selecting and scraping the apps

We acquire our user app reviews from the Google Play Store. Unlike the Apple App Store, where each app has a separate version for each country and whereby app reviews are separated by country, an app's reviews on the Google Play Store are not separated by country. In other words, an app on the Google Play Store has a combined set of app reviews across all country variations of the app, regardless of country of origin. So for example, the Facebook Messenger App is listed as a popular app in Canada and the US. Still, the app reviews collected from Canadian and US

users are combined to form Facebook Messenger's complete list of app reviews.

The dictionary terms were carefully selected to represent the definitive term relating to privacy. We furthermore conducted a keyword search on the text of each review, trying to identify privacy-related terms. The dictionary terms were carefully selected from several sources, including a paper on identifying NFRs by Williams et al. [13] as well as a few other sources. The overall process involves running a regex keyword search to determine how many instances of the privacy terms exist in each review. The same term can be counted multiple times. Then we break the dataset into subsets based on the number of privacy terms in a review.

[autostyle]csquotes

Table 3.1: keyword search term bucket

category	terms [1][13][15]
privacy	'health','protected','entity','disclose' ,covered',use','disclosure',individual' ,'such','purpose','law','permit','section' ,'require','plan','person','paragraph','care','request'
security	'cookie','encrypted','ephi','http','predetermined','vulnerability', 'username','inactivity','portal',ssl','deficiency','uc3','authenticate', 'certificate','session','path','string','password','incentive'
access_control	'administrator','personal','access'
legal	'infeasible','custodian','hipaa','breach','dua','discovery','iihus', 'Publication','iihi',recipient',delay','secretary',definition', 'harm','scope','jurisdictional','affect','derive','vocabulary','reuse','right to', 'gdpr', 'ccpa', 'pipeda'
personal_data	'personal','data','ID', 'location', 'biometric', 'account'
other	'accountability','consent','violate','unlawful','obligations','obligate', 'unprotected','leak','usage','codes of conduct','location','exploit', 'cybersecurity', 'disclosure', 'collect', 'monitor', 'encryption', 'theft', 'identity', 'backdoor', 'certification', 'hack', 'authentication'

3.2 Pre-processing app reviews

The reviews were preprocessed via python; firstly, duplicate reviews were removed due to the complication of redundancy. Since the third-party library would collect the same review multiple times, this proposed a serious concern in the analysis phase. Other techniques of text normalization were also applied to the data set, including

Stemming, punctuation removal, and Lemmatization. We also aim to minimize textual noise by removing stop words during the last data processing stage. Stop words can be found in abundance in many human languages, which can be reflected in the user review themselves. By removing these words, we can remove low-level or useless information from our text to give more focus to the critical information. Next, we aim to remove common words that often occur within the reviews, provide no meaningful feedback, and affect the classifier negatively. An example of this can be "app," "star," and even "reviews." Then we perform stemming on the tokenized data, removing all punctuation from the text.

3.3 Manual labelling

After obtaining the google app store data, we conducted a manual label procedure to check and see the keyword search performance. A subset of 250 reviews was used for this task. This was done by taking 50 reviews from 0, 1, 8, 9 and 10 privacy term buckets. Manually label each review whether it is about privacy or not via 0 and 1. 0 is not privacy-related, and 1 if the text mentions privacy and how it affects the user. Each row of the data was examined and labelled by two reviewers. The final result was then cross-checked for alignment and a calculated kappa value of 0.879. We are almost in perfect agreement that the data contains privacy mentions relative to the research focus.

3.4 Classifier development

The classifier was built utilizing Multinomial Naive Bayes from the Scikit-learn python library. The classifier was implemented with the idea that textual data can be better calculated using a bag of words approach. Thus, the classifier assumes that each feature is independent of the other. At the same time, it might be naive to assume that is the case; the classifier does perform supprising well under these circumstances. Since word frequency and count matter, Multinomial distribution can be seen as a generalization of the binomial distribution. Compared to other toolkits in the Scikit-learn package, Multinomial Naive Bayes is more intuitive in word frequency counts. Gaussian Naive Bayes and Bernoulli Naive Bayes were also considered for this study. However, since we are not specifically looking into the feature set of the review texts, it would be less relevant. In terms of Gaussian Naive Bayes, it focuses on data with

features that have continuous values. Since these classifiers were not as effective as Multinomial Naive Bayes, they were not selected from the initial stages.

With the classifier in place, the previously labelled text data was selected for a training set of the classifier. It contains 250 rows of data that were split into 60/40, with 40 being the training set and 60 being the testing set. The accuracy of the classifier was favourable at 69%, with a null accuracy of 53%.

Figure 3.1: confusion matrix of preliminary labeled reviews

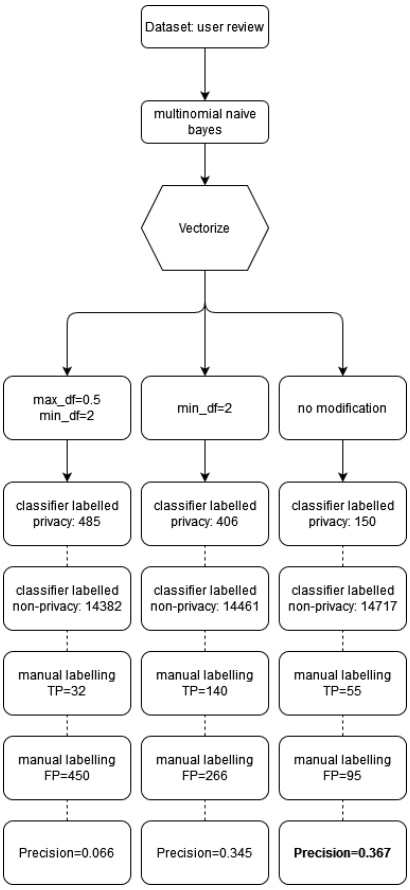
	Predicted Class Privacy yes(1)	Predicted Class Privacy no(0)
Actual class Privacy yes(1)	TP 24	FN 40
Actual class Privacy no(0)	FP 2	TN 71

Further manual data labelling was done to better train the classifier in identifying correct privacy reviews. The classification process was prepared using 1200 randomly selected reviews as the training set. Out of the selected reviews, only 89 rows of data were labelled as privacy-related. This became the basis for the training set of the Multinomial Naive Bayes.

These rows of processed data were then fed into the classifier for classification. During this stage, we applied a few tweaks to the classifier with the goal of improving accuracy. Yet to no avail; it became challenging to increase accuracy any further. We first applied a term exclude code on the "CountVectorizer" of the feature extraction package. Implemented as: `vect = CountVectorizer(max_df=0.5, min_df=2)` The tweak will ignore terms that appear in more than 50% of the documents. This tweak effectively eliminates the amount of common and recurring terms present for calculation. 482 rows of data were labelled as privacy, but with manual checking, only 32 rows were correctly labelled by the system. This promoted another tweak aimed at improving the classifier. In order to provide an efficient calculation of posterior likelihood, the prior probability can be raised. This comes in the form of `vect = CountVectorizer(min_df=2)` Where we only keep terms within the body of all text that appears in at least two documents, which returned 140 correctly labelled rows out of 406.

The final classifier was selected without minor tweaks; it uses the original settings for the class CountVectorizer within the code. Since there are several tradeoffs be-

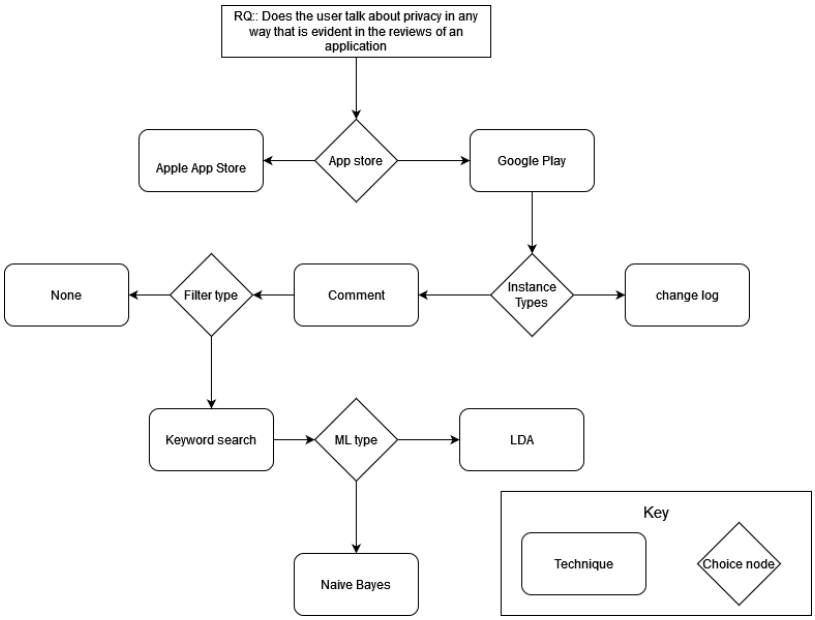
Figure 3.2: classifier selection process



tween false positive and true positive, the precision of the `vect = CountVectorizer(max_df=0.5, min_df=2)` and `vect = CountVectorizer(min_df=2)` falls behind the stock setting. After a manual review of each output, the classifier, with no further modification, could label the true positive more closely to user sentiment than the other two. For example, the app "Nextdoor: Local Updates, Recommendations and Deals" is a community-based business app that provides users with deals and information about their neighbourhood. As with such an app, user permission and location information would need to be delegated. One such user provides a concerning review after the usage of the app.

> "Security breach and intrution on privacy, collects data and gives out personal information. Not happy about security used on this app as it breaches data protection. GDPR. UK. Also breaches us data handling laws. Do not use."

Figure 3.3: protocol map of possible research aspects



The review showed that the user not only has knowledge of GDPR and expresses dislike for the acts committed by the app during usage. Many other studies were also labelled closer in sentiment to this example. Thus, the CountVectorizer was initiated with no modification to analyze the total reviews.

3.5 LDA development

For the latter part of the experiment, we use Latent Dirichlet Allocation (LDA). It is a generative probabilistic model of a corpus taken from the document. The notion behind this is that every text document is made of a mixture of smaller topics [2]. By analyzing these topics, we can have a firm understanding of the general consensus users have. LDA is a helpful method for identifying topics within the documents and mapping documents to those topics via an unsupervised natural language processing algorithm. In a way, LDA can best be explained as retrieving the underlying information in a particular document, as for this experiment, we are retrieving these topics on an immense scale. Most of the data obtained from the scrapping were unstructured; with more than 40 million entries, it became apparent that noise removal was needed to purify the data. Since the data set is extensive, sampling techniques were applied to reduce it into smaller parts. This was done by randomly sampling

the entire data at 50%; the subset would provide an unbiased representation of the total population while retaining valuable feature sets. Next, since we are interested in only the entries containing negative feedback on the app store platform, we can use the rating system to help our task. All reviews that are more than three stars are removed. This was due to users being more inclined to rate their interaction negatively if they voiced a concern. By dividing up the reviews into positive(5-4 stars), neutral(3 stars), and negative(1-2 stars), we can investigate further the difference in opinions of users about their experience. Moreover, all entries containing less than 15 characters are removed so the leftover entries can better derive helpful information for the reviews. For example, if an entry only contains 'app bad, I hate it,' then there is nothing definitive to be gathered for topic modelling. A bad application can be due to serval reasons since the study aims to explore users' concerns. Entries like these needed to be cleaned for added reliability of data output.

LDA algorithms were imported from the Gensim library, a highly efficient open-source Python library to train vector embeddings. The multicore variation uses all of the CPU cores to parallelize and speed up model training, which was used for this study. The number of topics is selected via a quantitative approach. The Gensim LDA library contained a call function for topic coherence as a score(C_v). It measures the score of a single topic using the degree of semantic similarity between high-scoring words it possesses. These measurements aim to differentiate between specific topics that are semantically interpretable and ones that are statistical inference in nature. Since C_v measures the relative distance between words within a topic, the score ranges between 0 to 1; a higher number would indicate that the distance is relatively far and preferred. A baseline was first set up to find a relevant standard which, with finer tuning, a more favourable topic coherence can be obtained. Latent Dirichlet Allocation can be mathematically represented in the figure 4.1.

Figure 3.4: Latent Dirichlet Allocation

$$p(\theta, \mathbf{z}, \mathbf{w} \,|\, \alpha, \beta) = p(\theta \,|\, \alpha) \prod_{n=1}^{N} p(z_n \,|\, \theta) p(w_n \,|\, z_n, \beta),$$

$$p(\theta \,|\, \alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1},$$

Where alpha and beta are hyperparameters, each represents different input values used to calculate $p\,(\Theta|\alpha)$, the Dirichlet distribution. Alpha represents document-topic

density; a bigger α value would mean the document comprises more topics and vice versa. As for β, it represents topic-word density. A higher beta value would mean topics are composed of many words in the corpus, and inversely, the lower the beta, the smaller the number of terms are made up in the topic. Lastly, N is the number of topics that can be generated. Via fine-tuning these hyperparameters, we hope to identify the suitable coherence score from the baseline model that allows us to develop the right amount of topics.

Chapter 4

Experiments

The experiment was conducted rigorously and concisely. During the initial stages of the process, 45,114,727 rows of data were scraped from the google play store. We aim to gather as much data as possible so the experiment can provide a broader range of information. The filtering critical was very rigorous to provide a consistent and clean result. We select the language of English for the basis of detecting language that can be understood later on if we want to look into the data manually. For this task, five countries were picked: Australia, Canada, the United Kingdom, the United States, and New Zealand. We accounted for the 200 most popular apps from those five countries; this brings us to a total of 893 applications. Since some apps occur across different stores, we had to remove the duplicate ones using the app ID. Selenium was used with "google-play-scraper" to help scrape data. During this process, we noticed that some apps only contain at most a few thousand reviews while others might have more than millions.

4.0.1 Privacy classification

The reviews were then processed and prepped for classifier labelling, including the usual language analysis techniques. During context processing, we removed punctuation so the words could be isolated. As for the next stage, most of the common words were also drawn to improve information quality. As common words usually cloud the classifier's calculations. The same is valid for stop words since they do not contribute to the meaning of a sentence. These usually can be safely ignored without changing the sentiment of the text. The result can be seen in table 4.1. An example of this would happen to a single text before and after the textual preparation stage.

Table 4.1: data process example

original content	processed content
"Guys, dont install and register. After my sister and brother install, in night the hacker will hack your account and chance the password, the hacker try to hack my sister another thing. Be careful"	guy do install register sister brother install night hacker hack account chance password hacker try hack sister another thing careful

This process took a week to complete. Then the data was fed through a keyword search that consisted of various terms containing privacy related. The systems used were mac OS Mojave with a six-core Intel i7 processor at 2.2 GHz. The classifier was developed and labelled with a windows 8 OS that contains an Intel i7 GPU running at 2.2 GHz and 16 GB of RAM.

In the next stage, we used a Multinomial Naive Bayes classifier to label all data rows that contain privacy keywords. This narrows down the dataset from over 45 million to 14,870 from keyword search. We also noticed that foreign language was also persistent in the data rows; this prose a determent to the accuracy of the classifier. A language detection tool was used to remove any row that contained a foreign language; it was called 'langdetect'. It is a port of Google's language-detection library into the python environment. This detection package is very useful in detecting large text and provides a high accuracy compared to other detection tools. Out of 14,870 data rows, only 14,367 were kept, with a high textual concentration of English. The finalized result was then relabeled manually to see the possible discrepancy between the classifier's ability to identify the correct content and evaluate performance. At this stage, the labelled data was able to identify the amount of user engagement, out of the total data set, regarding privacy. These procedures provide a basis for answering RQ1 and RQ2.

4.0.2 Topic modeling

As for RQ 2, first, only a subset of the entire set was selected. The criteria for this was a review that had to be lower than three stars. Since we wish to only look at data that provides details regarding user concerns, any positive or neutral entries need to

Table 4.2: sample data row

appName	appId	userName
WhatsApp Messenger	com.whatsapp	Sistar KenyaSue

content
No stars what so ever, the app keeps crashig, stalling Stopping. Further to this . I DO NOT AUTHORIZE wats app to collect my personal data. GDPR Applies. I do not give permission for wats app developers/Company to collect my personal preferences for marketing purposes & otherwise General Data Protection Regulations apply GDPR A opt out option should be provided to give customers the choice to decline from the above mentioned updates scheduled for 2021.

score	thumbsUpCount	reviewCreatedVersion
1	1	2.20.206.24 8

createDate	replyContent	repliedAt
2021-01-27 16:53:16		

reviewId
gp:AOqpTOEofgdD0gXLgSi 5oRYEHTDz9tUFNAcv9J5_ A6tOmVx6QzS6d1O5aVbsqs1w 1ntmu2wedy8xNLtqXDhpM68

be dropped, this brings the total down to 6,974,117 reviews. For the next level of filtering, we removed any data mentioned above that was used in the privacy keyword search and then randomly sampled it at 50%. Since the existence of any data relating to privacy would contaminate the set, we have to remove them to answer RQ2 better. The aim of randomly sampling half of the data was to reduce workload while retaining the proper amount of information needed for machine learning and textual analysis. If a topic is found profoundly in half of the set, it would have a high probability of existing in the entirety of the collection. Then all null entries and those less than 15 characters were dropped, which returns 2,763,712 rows. Data processing of lemming, stemming, removal of stop words and punctuations were also applied. Bigram was later created to increase the model's consistency of the user input understanding in their reviews wording. Thus begins the topic modelling phase. To derive the optimal number of topics, we must first set up a baseline to measure the topic's relevancy.

Table 4.3: C_v generated for N=2

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.3825686	0.3755446	0.3772087	0.3694499	0.3887601	0.3887601
	0.1	0.3825686	0.3772087	0.3634631	0.3780580	0.3887601	0.3887601
	0.19	0.3825686	0.3772087	0.3710237	0.3809641	0.3887601	0.3887601
	0.28	0.3907279	0.3766577	0.3766577	0.3662299	0.3804616	0.3854621
	0.37	0.4068879	0.3766577	0.3766577	0.3662299	0.3804616	0.3854621
	0.46	0.4207604	0.3936339	0.3766577	0.3626504	0.3696711	0.3862300

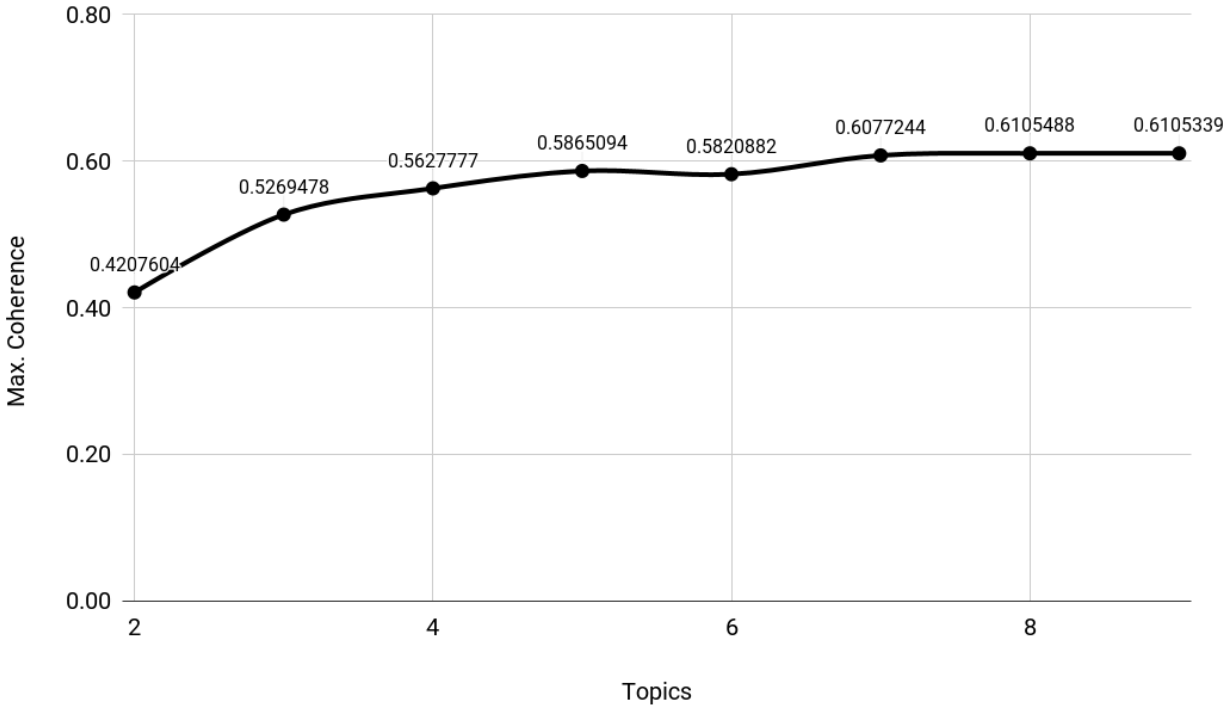
Table 4.4: C_v generated for N=9

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5919314	0.5974084	0.5932387	0.5948049	0.5877500	0.5901709
	0.1	0.5953114	0.5999266	0.5965943	0.5963547	0.5881545	0.5896488
	0.19	0.5966718	0.5982092	0.6055478	0.5968584	0.5945625	0.5917045
	0.28	0.5787489	0.6009189	0.5971506	0.5976029	0.5949348	0.5964456
	0.37	0.5791769	0.5714869	0.6006076	0.5885314	0.5936458	0.6105339
	0.46	0.5670401	0.5628591	0.5885737	0.5990512	0.5938184	0.6036319

The number of topics extracted from the corpus can be evaluated via coherence score(C_v) using the Gensim topic modelling library in conjunction with fine-tuning hyperparameters N, Alpha, and Beta. Considering Alpha controls the topic sparsity and uniformity, a lower α value would assume that the document contains fewer topics or at least one. This resembles the nature of user reviews. Usually, most reviews are simply about a singular opinion of their application. As for Beta, it represents topic-word density. A smaller value would mean the topic is more likely to be made up of fewer words in the corpus. Lastly, N would represent the number of topics. We start with a baseline of N=5, α=0.1 β=0.1. This yields a C_v of 0.56. To improve upon the baseline values, an iterative approach was taken. A small loop program was written to find the optimal topic number as well as Alpha and Beta. The program starts from N=2 and ends with N=9. Alpha and Beta were increased from 0.01 to 0.5 with a step of 0.09. The corpus was limited to 50% to reduce computational time. Still, the program ran for more than 100 hours. This generated eight tables representing the maximum obtainable C_v for alpha and Beta. An example can be seen below with N=2 and N=9. Since we only need the maximum C_v, a line graph containing all contact points between the topic and coherent score was generated from the tables.

This helps us in determining which C_v is most suitable.

Figure 4.1: Max coherence score at 50% corpus vs number of topic



From the derived coherent score, we can find the optimum topic at N=8, with $\alpha = 0.37$ and $= 0.46$. This yields C_v=0.60, an improvement of 11% over the baseline. The keywords for each topic generated are presented as a word cloud in figure 4.2 the size of the word is correlated to their weighting in the corpus. The keywords are extremely helpful in understanding the meaning behind each topic for the last stage of the study. We can provide an in-depth analysis of the underlying relationships between the application and user concerns.

Furthermore, pyLDAvis was used to generate the graph to visualize the LDA model. Each bubble of the chart represents a topic. The bubble size represents the percentage of the document in the corpus about that topic. The distance of each bubble also represents the amount of difference in topics. For example, topic two is more similar to topic three than topic six. We can provide a deeper evaluation and answer both research questions in the next section with these derived statistics.

Figure 4.2: Words cloud of generated topics

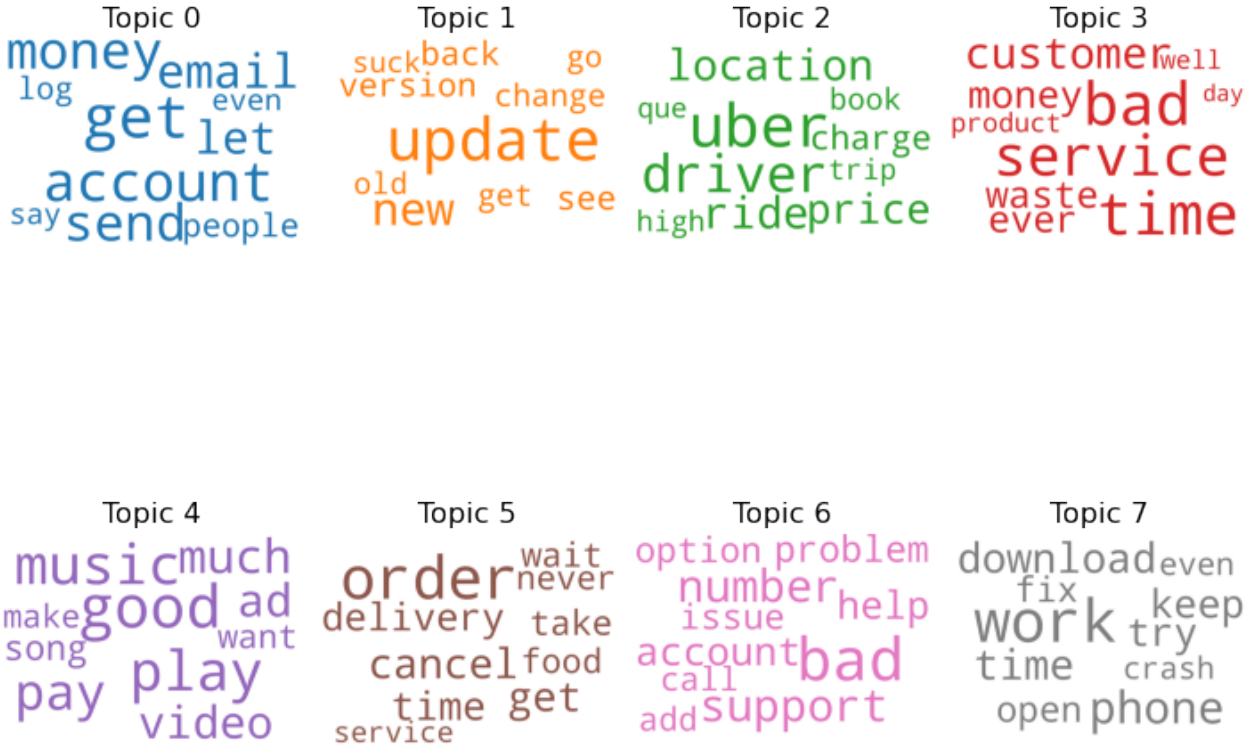
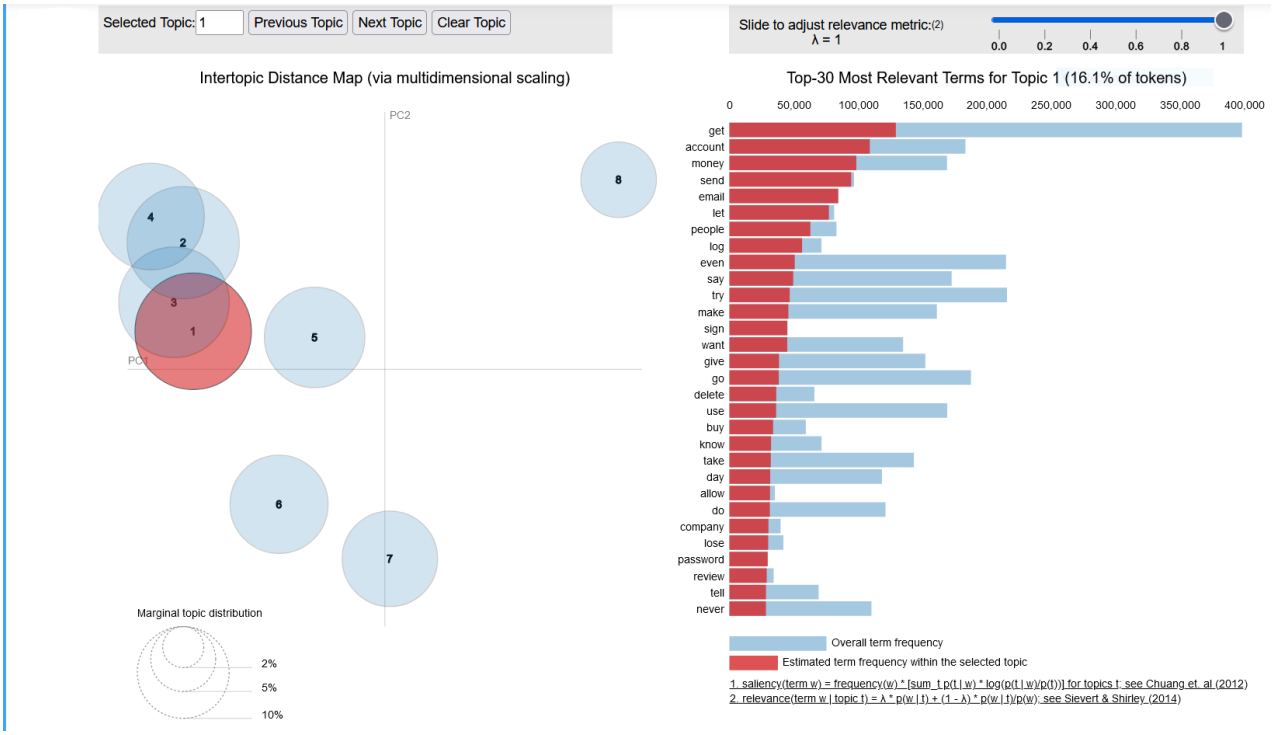


Figure 4.3: pyLDAVis graph generated from LDA model



Chapter 5

Evaluation and Analysis

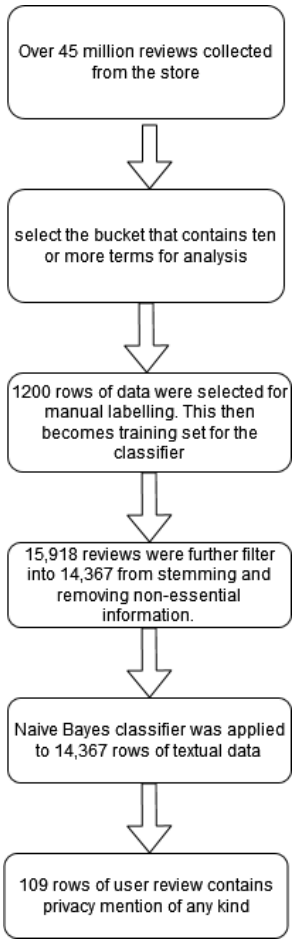
Of the 1200 manual labelled data, it was clear that there was a precise disproportionate amount of reviews that contained privacy mentions. This was further translated into the larger dataset used for the preliminary test. We selected 200 reviews from each level of keyword search buckets. From a minimum of five privacy, then six, seven, eight, and up to 10+ terms. Out of 1200 randomly chosen rows, only 89 contain actual privacy mentions or close to privacy mentions by the user. This was then used as training data for a more comprehensive set.

The selected data contains about 14,367 rows of unique data; each includes the user reviews of applications in the store. This was selected based on the number of privacy terms contained in the document. There was a good balance of review amount and term mentioned in the ten or more privacy terms subset. Thus it was chosen. Each row contains an average of 92 words with 1,326,206 words in total. There are various types of apps ranging from 122 unique apps. These include Tiktok, Facebook, Instagram, and WhatsApp. Out of 14,367, only 109 rows contain user reviews regarding labelled by the classifier.

According to the labelled text, there is only about 0.7587% of the users on the app store had a concern over the privacy-related aspects of the app they are using. These numbers are significantly small compared to the overall data set at 40 million.

In answering RQ1, users are proactively discussing their concerns over privacy violations. These are evident from the labelled texts by the classifier. Users are aware of the infractions against them by the application; this was gathered from the repetitive manual labelling process. However, the number of privacy mentions is insignificant compared to the total user reviews that were being gathered. This can be directly seen as most users are indifferent to the privacy aspects of an application.

Figure 5.1: filtering process



As for RQ2, we can first classify the generated topics from the keywords they contain. This is reflected in table 5.1 as with each topic, and the keyword provides crucial information in determining the type of concern a user has. For instance, topic 0 relates to e-transfer and moving money between accounts; the users are concerned with sending and receiving money on either end. Therefore, for the following process of answering RQ2, we also need to know the most discussed topics in the documents. This was done by measuring the number of documents attributed to each topic. More importantly, the weight contribution between the topic and documents. First, we identify the number of documents for each topic by assigning the document to the topic that has the most weight in that document. Next, calculate the number of documents for each topic by summing up each topic's actual weight contribution to respective documents. Both are presented as bar graph in figure 5.3 and A.1 below.

Topic 7 "app performance" is the most concern by the users, based on the number

Figure 5.2: privacy mention of user review contains privacy keywords

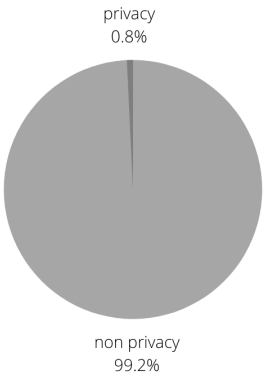


Table 5.1: topic classification

topics	classification
0	e-transfer
1	version conflict
2	uber functionality
3	payment user support
4	music app paywall
5	food delivery problem
6	account login issues
7	app performance

of documents(523027) and their weightage(441489.30). This provides definitive evidence that many users on the app store platform are more interested in voicing app performance than privacy violations. Based on our experiments, we can recognize LDA topic method result clearly shows the number of user input containing privacy keywords pales compared to the number of user complaints about app performance and usability. Even those documents that contain privacy terms, the ones that talked about privacy were less than one percent.

Figure 5.3: Number of documents by dominant topic

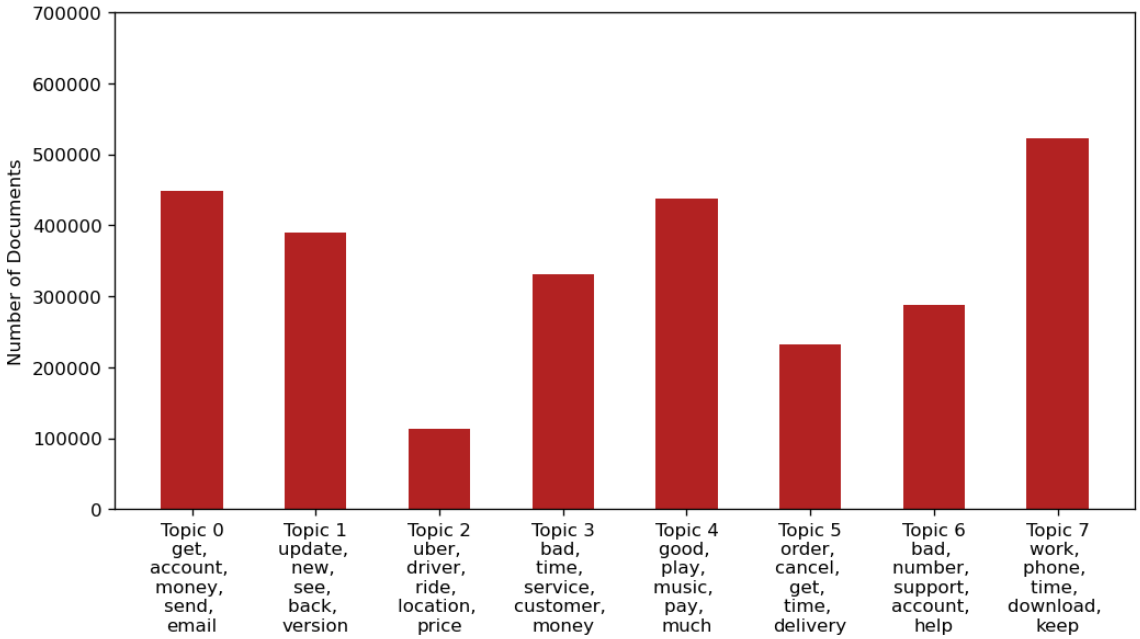
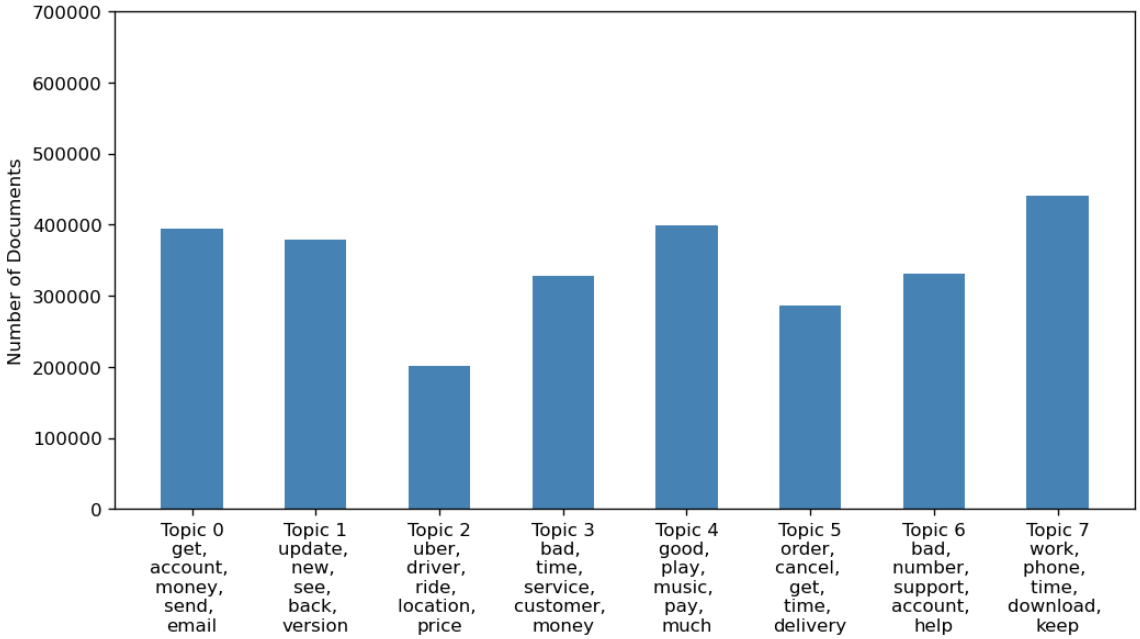


Figure 5.4: Number of documents by topic weightage



Chapter 6

Conclusions

With the increasing focus on privacy engineering for mobile applications today, we often wonder about the amount of user concern with app usage. The google app store is an excellent platform to investigate deeper into the behaviour of the users in an effort trying to find applicable information about specific topics or objects. The challenge with inferring topics and identifying privacy concerns resides in the overwhelming amount of data and its noise. In applying ML techniques, NLP process, and LDA modelling, we can derive practical knowledge and overcome such a problem since it is considered a powerful technique that can assist in detecting and analyzing important textual content in the reviews. Conversely, it is not done without substantial challenges relating to run time and scarcity of training data.

This paper delved into a detailed examination of how much weight users place on privacy concerns or perceived violations focusing on understanding the status of privacy engineering in the digital era. In addition, we want to understand the most addressed topic that user voices on the platform. Our evaluation used textual datasets from 839 applications, totalling more than 45 million rows of raw data. The performance achieved by LDA and naive Bayes classifier was measured in precision, coherence score, and other standard metrics. We also defined secondary output data to identify well-organized and meaningful topics of user concerns. As a result, we found that many users on the app store are more interested in improving application performance and have fewer issues with functionality. This type of user behaviour was reflected in minimal privacy mentions in the portion of the user reviews compared to other topics. Privacy is still very relevant to the user and developers alike despite these outcomes in ensuring valuable information isn't being wrongfully taken. The work presented in this paper can be a vital reference for researchers on review analysis

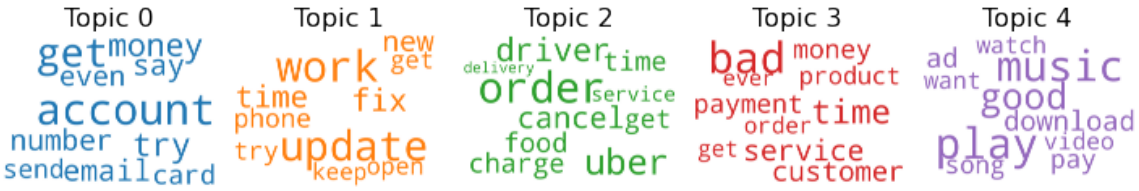
and topic classification.

Appendix A

Additional Information

As for the baseline LDA model, the topics generated are somewhat different compared to the finalized result. Since it was not as precise in comparison to the eight topics one, this was put in the appendix. Some of the information was not counted into the overall document; this was due to the number of wording that is counted more repetitive due to fewer topics.

Figure A.1: words cloud of base LDA model



The following are rest of the coherent score tables calculated from different α and β during the experiment phase. Since we are more interested in obtaining the maximum coherent score and the hyperparameter value that comes with it, it was in favour to show figure 4.1.

Table A.1: C_v generated for N=3

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.4915804	0.4948104	0.5053588	0.5107244	0.5008609	0.4916288
	0.1	0.4995749	0.4912300	0.5038443	0.5106457	0.5091579	0.4971691
	0.19	0.5027093	0.5096265	0.5136656	0.5219971	0.5147012	0.5080247
	0.28	0.5098127	0.5074697	0.5074697	0.5117567	0.5195908	0.5097192
	0.37	0.5065849	0.5138075	0.5141403	0.5225880	0.5204272	0.5148849
	0.46	0.5062146	0.5005439	0.5139680	0.5269478	0.5204272	0.5209117

Table A.2: C_v generated for N=4

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5319969	0.5292210	0.5246070	0.5075326	0.5306518	0.5484535
	0.1	0.5245469	0.5337066	0.5383673	0.5138092	0.5279528	0.5594056
	0.19	0.5306619	0.5245868	0.5387969	0.5250763	0.5331958	0.5627777
	0.28	0.5233498	0.5248577	0.5354303	0.5237279	0.5314398	0.5469267
	0.37	0.5224551	0.5282903	0.5331673	0.5308653	0.5390756	0.5392769
	0.46	0.5335339	0.5321392	0.5526425	0.5362569	0.5352923	0.5453970

Table A.3: C_v generated for N=5

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5546566	0.5651485	0.5693951	0.5661280	0.5767650	0.5676145
	0.1	0.5631628	0.5663232	0.5688333	0.5663362	0.5751206	0.5704520
	0.19	0.5601419	0.5597034	0.5582068	0.5720041	0.5749519	0.5840999
	0.28	0.5568423	0.5590793	0.5678794	0.5658852	0.5698578	0.5865094
	0.37	0.5719156	0.5752017	0.5762544	0.5719823	0.5782319	0.5753118
	0.46	0.5864620	0.5802174	0.5792454	0.5727832	0.5799999	0.5839352

Table A.4: C_v generated for N=6

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5537870	0.5537870	0.5517525	0.5582225	0.5635678	0.5485145
	0.1	0.5569294	0.5333987	0.5421407	0.5477820	0.5547387	0.5512285
	0.19	0.5486853	0.5381456	0.5443049	0.5476331	0.5548771	0.5452725
	0.28	0.5435587	0.5423327	0.5401748	0.5512070	0.5534352	0.5480644
	0.37	0.5583505	0.5745469	0.5546008	0.5513176	0.5553932	0.5556489
	0.46	0.5706314	0.5800197	0.5820882	0.5519432	0.5556440	0.5467411

Table A.5: C_v generated for N=7

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5534884	0.5644873	0.5718322	0.5744646	0.5503115	0.5606180
	0.1	0.5965394	0.5863530	0.5997855	0.5910787	0.5553826	0.5755605
	0.19	0.5891563	0.5950158	0.5999884	0.5870291	0.5557749	0.5755605
	0.28	0.5925708	0.5932822	0.5959939	0.5997500	0.5598591	0.5796258
	0.37	0.5907415	0.5910563	0.6077244	0.5886205	0.5553589	0.5901369
	0.46	0.5871681	0.5861932	0.5950643	0.5808615	0.5514293	0.5863616

Table A.6: C_v generated for N=8

		Alpha					
		0.01	0.1	0.19	0.28	0.37	0.46
Beta	0.01	0.5780003	0.5905442	0.5889357	0.5925661	0.5849365	0.5840374
	0.1	0.5910045	0.5952186	0.5905999	0.5949389	0.5899465	0.5848416
	0.19	0.5851652	0.5969692	0.5924111	0.5989210	0.5982347	0.5880475
	0.28	0.5703398	0.5859052	0.5905401	0.6028368	0.5957242	0.5887614
	0.37	0.5719764	0.5874001	0.5910847	0.6064960	0.6063057	0.6032019
	0.46	0.5763595	0.5885175	0.5940000	0.6040892	0.6105488	0.6037243

Bibliography

[1] Vanessa Ayala-Rivera and Liliana Pasquale. The grace period has ended: An approach to operationalize gdpr requirements. pages 136–146, 08 2018.

[2] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. volume 3, pages 601–608, 01 2001.

[3] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. Sentiment polarity detection for software development. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, page 128, New York, NY, USA, 2018. Association for Computing Machinery.

[4] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. How to ask for technical help? evidence-based guidelines for writing questions on stack overflow. *Information and Software Technology*, 94:186–207, 2018.

[5] Maria Gomez, Bram Adams, Walid Maalej, Martin Monperrus, and Romain Rouvoy. App store 2.0: From crowdsourced information to actionable feedback in mobile ecosystems. *IEEE Software*, 34(2):81–89, 2017.

[6] Marlo Haering, Christoph Stanik, and Walid Maalej. Automatically matching bug reports with related app reviews. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021.

[7] Zijad Kurtanović and Walid Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495, 2017.

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

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

[10] Daniel Martens and Timo Johann. On the emotion of users in app reviews. In *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 8–14, 2017.

[11] Daniel Martens and Walid Maalej. Towards understanding and detecting fake reviews in app stores. *Empirical Software Engineering*, 24(6):3316–3355, 2019.

[12] Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. Crowdsourcing user reviews to support the evolution of mobile apps. *Journal of Systems and Software*, 137:143–162, 2018.

[13] John Slankas and Laurie Williams. Automated extraction of non-functional requirements in available documentation. In *2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, pages 9–16, 2013.

[14] Christoph Stanik, Marlo Haering, and Walid Maalej. Classifying multilingual user feedback using traditional machine learning and deep learning. *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019.

[15] Christina Tikkinen-Piri, Anna Rohunen, and Jouni Markkula. Eu general data protection regulation: Changes and implications for personal data collecting companies. *Computer Law Security Review*, 34(1):134–153, 2018.

[16] Huayao Wu, Wenjun Deng, Xintao Niu, and Changhai Nie. Identifying key features from app user reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 922–932, 2021.