Mobile App Evolution Analysis based on User Reviews

Conference Paper · September 2018

CITATIONS
12

READS
676

3 authors:

Xiaozhou Li
Tampere University
27 PUBLICATIONS **61** CITATIONS
SEE PROFILE

Zheying Zhang
Tampere University
45 PUBLICATIONS **270** CITATIONS
SEE PROFILE

Kostas Stefanidis
Tampere University
123 PUBLICATIONS **1,478** CITATIONS
SEE PROFILE

September 2018

Mobile App Evolution Analysis based on User Reviews

Xiaozhou LI [a,1], Zheying ZHANG [a] and Kostas STEFANIDIS [a]
[a] *University of Tampere, Finland*

Abstract. The user reviews of mobile apps are important assets that reflect the users' needs and complaints about particular apps regarding features, usability, and designs. From investigating the content of such reviews, the app developers can acquire useful information guiding the future maintenance and evolution work. Previous studies on opinion mining in mobile app reviews have provided various approaches to eliciting such critical information. A particular update of an app can provide changes to the app that result in users' reversed opinions, as well as, specific new complaints or praises. However, limited studies focus on eliciting the user opinions regarding a particular mobile app update, or the impact the update imposes. In this paper, we propose a method for systematically studying and analyzing the evolution of the users' opinions taking into consideration a set of mobile app updates. For doing so, we compare the topics appearing in the users' reviews before and after the updates. We also validate the method with an experiment on an existing mobile app.

1. Introduction

The increasing number of smartphone users has led to a continuous increase in the number of mobile apps and their overall usage. Users browse and download apps via different digital distribution platforms, and these platforms also provide an important channel for users to provide feedback to the app developers. Users can rate a release and express their opinions on the release they are using at the time of submitting a review, as in traditional recommender systems [13, 16]. Along with frequent release updates through the distribution platforms, the reviews accumulate the bugs, desired features, the comparison and users' attitude toward the app informally, and form a useful resource for analyzing change requests in the app maintenance phase.

Existing studies have proposed different approaches to identifying changes [22, 26]. One particular example is to identify change requests from user reviews for mobile app maintenance. With various opinion mining techniques, such as natural language processing (NLP), sentiment analysis (SA) and supervised learning, many studies have been conducted regarding the classification of reviews towards different issue perspectives [19, 21]. Other perspectives, such as user preferences, entities analysis, app evaluation, user satisfaction, the relation between download and rating, feature extraction and review prioritization, have also been widely studied [5, 7, 8, 12, 14, 15, 18]. However, lim-

[1] Corresponding Author: University of Tampere, Finland; E-mail: xiaozhou.li@uta.fi

ited studies focus on the use of such methods in opinion mining on particular updates of a mobile app and the impact on the app's updates in the following releases, despite the importance of such information. It is unclear how bugs and requests in user reviews are addressed in the follow-up releases and how users' attitude towards a particular issue changes when new updates are released.

In this paper, we investigate the correlation between users' positive and negative reviews before and after a sequence of apps' releases. We consider two dimensions: time and sentiment. Specifically, we divide user reviews based on a set of major updates and distinguish them between those precede and follow the particular updates. Each group of reviews is further divided into positive and negative ones using sentiment analysis. We propose a way to measure the similarity of each group of reviews. The measurement reveals the similarity and changes between different groups of reviews and helps to gain insight into the app's change requests in the maintenance phase. Besides understanding the main issues in user reviews, it helps developers be aware of the users' opinions on particular updates and guides them proactively to address the most important issues early.

2. Method Description

2.1. Preliminaries

This method aims to detect the evolving topic trend regarding a particular app by investigating the correlation between its users' positive and negative reviews before and after each major update within a sequence of updates through the app maintenance life cycle. Such correlation shall show the degree to which the users' comments are reflected in the sequence of updates and such updates are accepted by the users. The factors that influence and reflect such correlation include the main topics of the reviews between every two updates, the sentiment of those reviews, and the topics similarities before and after each update. Next, we illustrate how to detect the correlation via investigating these factors.

Let R be a collection of user reviews for a particular mobile app A, covering the time period $T_{time} = [ti_s, ti_e]$ (where ti_s, ti_e are two different time points with $ti_s < ti_e$). Therein, each review $r_i \in R$ is associated with a particular time when r_i is published. Let U be the set of updates released by the app developers within the time period T_{time}. Meanwhile, each update is also associated with a time point t_i, reflecting the time u_i is announced and delivered. Furthermore, we consider the review set R_i, of which each r_i is published within $[t_i, t_{i+1}]$, as the collection of reviews regarding the update u_i. Hence, for the n updates $\{u_i | i \in N, k \leq i < k+n\}$ where $k \geq 1$ and $n > 0$ for the app A within period T, the review set R can be divided into $n+1$ subset where each $R_i \subseteq R$ is the set of reviews commenting on the according u_i. R_0 is the review set given before the first update of A. For each review sentence $r_j \in R_i$, a sentiment score shall be calculated and assigned to r_j, whose sentiment is either *positive* or *negative*. In this way, by identifying the sentiment of each individual review in R_i, we can divide R_i into a positive review set R_i^+ and a negative one R_i^+, where $R_i^+ \cup R_i^- = R_i$ and $R_i^+ \cap R_i^- = \emptyset$.

When modeling topics of each review set, we assign T_i^+ and T_i^- as the topic set for the positive and negative reviews. Therefore, we investigate the merits and issues of a particular update u_i by compare the similarities and changes between T_{i-1}^+, T_{i-1}^-, T_i^+, and

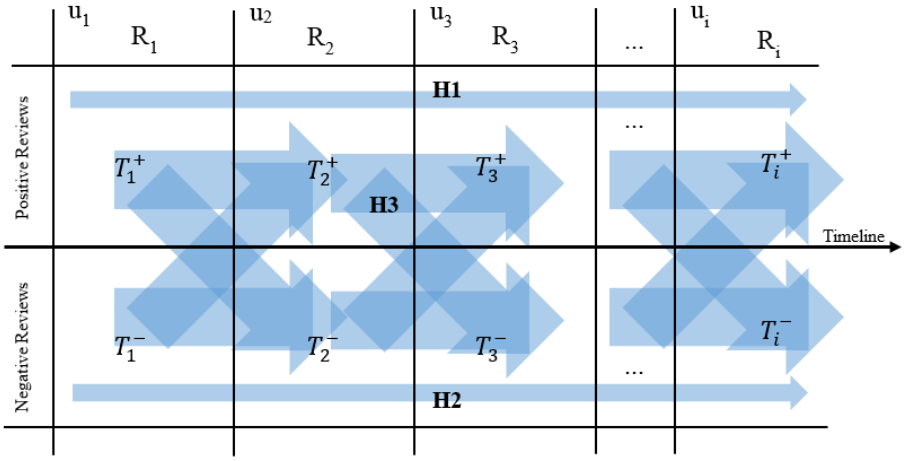

Figure 1. Hypotheses for the updates and the topic change of user reviews.

T_i^-. Furthermore, by investigating such topic similarity and changes regarding each u_i, we shall be able to observe the users' opinion change on the merits and issues within a given period of time. Therefore, given a set of updates U = $\{u_1, u_2, ... u_n\}$ with the according review sets $R_1^+, R_1^-, R_2^+, R_2^-, ... R_n^+, R_n^-$, finding the topic similarity correlations throughout $T_1^+, T_1^-, T_2^+, T_2^-, ... T_n^+, T_n^-$ shall reflect the evolving user opinions concerning various aspects of the app within a particular period of time. Therefore, provided a set of topic $\{t_1, t_2, ... t_i ... t_n | t_i \in T_i\}$, where each t_i is similar to t_{i+1} $(1 \leq i < n)$, then such topic set is defined as a *similar topic chain*. We furthermore propose three hypotheses as follows.

- H1. The similar topic chains through $T_i^+, T_{i+1}^+, ... T_n^+$ reflect the merits and users' praise regarding a sequence of app A's updates $u_i, u_{i+1}, ... u_n$.
- H2. The similar topic chains through $T_i^-, T_{i+1}^-, ... T_n^-$ reflect the issues and users' complaints regarding a sequence of app A's updates $u_i, u_{i+1}, ... u_n$.
- H3. The similar topic chains containing topics from both $T_i^+, T_{i+1}^+, ... T_n^+$ and T_i^-, $T_{i+1}^-, ... T_n^-$ reflect the changing user opinions (positive to negative, or negative to positive) regarding particular aspects of app A.

Thus, the results obtained from these hypotheses for a certain period of updates u_i, u_{i+1}, ... u_n provide the following information: 1) the merits and users' praise for the app A, 2) the issues and users' complaints for the app A, and 3) the evolving of users opinions regarding particular aspects of the app A.

2.2. Sentiment Classification

The aim of sentiment classification in this method is to classify each review set R_i into two subsets, i.e., R_i^+ and R_i^-. Herein, R_i^+ denotes the set of positive reviews from R_i, and R_i^- denotes the set of negative reviews. Therefore, each r_j in R_i shall be determined whether it is positive or negative.

To do so, we assign a sentiment score to each review by exploiting a robust tool for sentiment strength detection on social web data. As each r_j can be seen as a list of words W_j, we first select a lexicon that will determine the sentiment score of each word w_z in W_j. The lexicon for sentiment analysis is a list of words used in the English language, each of which is assigned with a sentiment value in terms of its sentiment valence (intensity) and

polarity (positive/negative). To determine the sentiment of words, we assign a rational value within a range to a word. For example, if the word "okay" has a positive valence value of 0.9, the word "good" must have a higher positive value, e.g., 1.9, and the word "great" has even higher value, e.g., 3.1.

Furthermore, the lexicon set shall include social media terms, such as Western-style emoticons (e.g., :-)), sentiment-related acronyms and initialisms (e.g., LOL, WTF), and commonly used slang with sentiment value (e.g., nah, meh).

With the well-established lexicon and a selected set of proper grammatical and syntactical heuristics, we shall then be able to determine the overall sentiment score of a review. Namely, the sentiment score of a review r_j is equal to S_j, where $S_j \in (-1,1)$. The grammatical and syntactical heuristics are seen as the cues to change the sentiment of word sets. Therein, punctuation, capitalization, degree modifier, and contrastive conjunctions are all taken into account. For example, the sentiment of "The book is EXTREMELY AWESOME!!!" is stronger than "The book is extremely awesome", which is stronger than "The book is very good.".

With both the lexicon value for each word of the review and the calculation based on the grammatical and syntactical heuristics, we can then assign unique sentiment values to each review. That is, each review r_j is classified into positive, neutral or negative, as follows:

$$r_j \text{ is } \begin{cases} \text{positive, if } 0 < S_j < 1, \\ \text{neutral, if } S_j = 0, \\ \text{negative, if } -1 < S_j < 0. \end{cases}$$

Overall, each review set R_i is divided into R_i^+, R_i^0, and R_i^-, denoting the positive, neutral and negative review sets. By experimentally observing, R_i^0 contains a large number of reviews with also useful information. Therefore, we further classify R_i^0 into positive and negative using the Naive Bayes Classifier with the training data from R_i^+ and R_i^-. R_i^0 is classified into R_i^{0+} and R_i^{0-}, which in turn, are added to R_i^+ and R_i^-, respectively. The reason to perform supervised classification after sentiment analysis instead of directly applying classification is twofold. Firstly, manually creating training data is time-consuming and less accurate than using existing sentiment analysis methods. Secondly, training the sentiment classified reviews will provide domain specific and reliable results.

2.3. Topic Analysis

After dividing the review sets R_i into R_i^+ and R_i^-, we elicit the main topics from each of the classified review sets by exploiting the Latent Dirichlet Allocation (LDA) method [3]. First, we consider each review sentence r_j in a particular set of reviews as a list of words W_j, where the sequence of the words is not recorded. The number of topics in this review set is set as t. Presumably, there is a distribution for the probability of a particular word appears in a particular topic, when there is also one for that of a particular review of a topic. We build the set of $Review-Topic$, where each word of each review is assigned with a topic out of the t topics. As preparation, we define $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ denoting the number of occurrence of each topic in each review, the number of occurrence of each

word in each topic, and the number of words in each topic, respectively. For example, $Review-topic-numbers(r_j,k)$ denotes the number of occurrences of topic k in review r_j. Then, we randomly assign each word w_z of each review r_j with a topic t_k. Accordingly, the $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ will be updated as the referencing weight of the distribution of the words for each topic. Then iteratively, for each word, we assign a new topic based on such weight of distribution and adjust the weight with the $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ for the next iteration. After a given number of iteration, t topics will be determined by the $Topic-words-numbers$, which is the number of occurrences of the words in each topic. Each t_k is then denoted by the most common keywords used in this topic. Then, the set of topics T is returned as a result. For the review sets R_i^+, R_i^-, R_{i-1}^+ and R_{i-1}^-, we will have the topics sets T_i^+, T_i^-, T_{i-1}^+ and T_{i-1}^- accordingly.

2.4. Calculating Topics Similarities

Based on the topic sets T_i^+ and T_i^- elicited from the review sets R_i^+ and R_i^-,we further analyze the similarities between the individual topics between each pair of the topic sets. As the result from the previous topic analysis, each topic set T encompasses k topics, each of which is represented by the list of the most possible appearing keywords. Thus, each topic set T with k topics each of which is represented by w keywords, can be denoted as:

$$T = \begin{bmatrix} kw_{1,1} & kw_{1,2} & \dots & kw_{1,w} \\ \dots & \dots & \dots & \dots \\ kw_{k,1} & kw_{k,2} & \dots & kw_{k,w} \end{bmatrix}$$

with each $t_i \in T$ can be denoted as $[kw_{i,1}, kw_{i,2}, ...kw_{i,k}]$. To compare the similarity between two topic sets, each consisting of k topics, we compare all pairs of topics. Due to the fact that each topic is represented as a set of keywords, the similarity of two topics shall be denoted by the common keywords of these topics. Hence, an easy way for calculating the similarity between any two topics t_i and t_j is by using the Jaccard similarity. This similarity function reflects the percentage of the common keywords of the two sets in the whole keywords set of the two: $J(t_i,t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}$.

On the other hand, the probability (p_{ij}) of each keyword kw_{ij} appearing in topic t_i is different. Hence, despite J(t_i, t_j) = J(t_i, t_k), when the probability value of the common keywords varies, the topic similarities for t_j, t_k towards t_i can still differ. Therefore, we propose a modification of the previously described Jaccard similarity calculation taking into account the probability of the keywords, named as the Jaccard Extended Similarity. The c common keywords of t_i and t_j include $[kw_{ij,1}, kw_{ij,2}, ...kw_{ij,c}]$, when the according probability list in t_i and t_j is $[p_{i,1}, p_{i,2}, ...p_{i,c}]$ and $[p_{j,1}, p_{j,2}, ...p_{j,c}]$. The Jaccard Extended Similarity is then calculated as follows:

$$JE(t_i,t_j) = \frac{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}}{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2} + \sum_{x=c+1}^{k} p_{i,x} + \sum_{x=c+1}^{k} p_{j,x}}.$$

The probability for each keyword of any topic belongs to (0,1). Hence, for this formula, when t_i and t_j contain more common keywords, the numerator increases mono-

tonically, and the denominator decreases monotonically. Therefore, $JE(t_i,t_j)$ increases when t_i and t_j have more keywords in common. In addition, when the probability of the common keywords increases, $\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}$ increases. Because the denominator is greater than the numerator, and both are greater than 0, $JE(t_i,t_j)$ increases when the probabilities of the common keywords of t_i and t_j increase. In this way, JE takes into account both the number of common keywords and the probabilities of such keywords.

2.5. Identifying Matching Topics

After computing similarities between pairs of review topics, we shall also identify which are the matching topics when cross-comparing the topics of the topics sets T_i^+, T_i^-, T_{i-1}^+ and T_{i-1}^-. Hence, to identify the matching topics between two review topic sets T_a and T_b, the aim is to identify all the topic pairs (t_{ai}, t_{bj}), $t_{ai} \in T_a$ and $t_{bj} \in T_b$, that have the highest similarity, and contain the unique t_{ai} and t_{bj} which are both different from those of the other selected pairs. We set a threshold similarity value, so that only the topic pairs that have a similarity value greater than it can be considered as similar. Given that several topics are similar to multiple other topics, we apply an algorithm to select the unique similar topic pairs when comparing two review topic sets, T_a and T_b.

Algorithm 1 Matching Topics Identification

procedure MATCHING TOPICS IDENTIFICATION
$k \leftarrow$ number of topics
$Ta \leftarrow \{t_{a1}, t_{a2}, ..., t_{ai}, ..., t_{ak}\}$
$Tb \leftarrow \{t_{b1}, t_{b2}, ..., t_{bj}, ..., t_{bk}\}$
$s_{ij} \leftarrow Similarity(t_{ai}, t_{bj})$
For $i, j \in \{1, 2, ..., k\}$
    if $s_{ij} \geq$ threshold value **then**
        $S \leftarrow s_{ij}$
While $len(S) > 0$
    if Only 1 Max(S) in S **then**
        P.append$((t_{ax}, t_{by}))$ where $Similarity(t_{ax}, t_{by}) = Max(S)$
        S - (S $\cap$ ($\{s_{x1}, s_{x2}, ..., s_{xk}\} \cup \{s_{1y}, s_{2y}, ..., s_{ky}\}$))
    if More than 1 Max(S) in S **then**
        P.append $((t_{ax}, t_{by}))$ where $Similarity(t_{ax}, t_{by}) = Max(S)$ & $Sum(\{s_{x1}, ..., s_{xk}, s_{1y}, ..., s_{ky}\})$is the minimum
Return P

The aim of Algorithm 1 is to select only the similar topic pairs with the highest similarity value. When a particular topic pair is selected, the other pairs which either of these two selected topics is also pairing with will be ignored. With this pair and the ignored ones deducted from the original selected similar topic pairs, we continuously select the one with highest similarity value, until all selected pairs are unique without any selected topic being simultaneously similar to more than one topic. Especially, if at a certain selecting iteration, there is a tie on the highest similarity value, we select the pair of topics that are the least similar to the rest of the topics, by calculating the sum of all the similarity values with either topic and selecting the one with the minimum sum value.

3. Case Study

3.1. Preprocessing

Before starting the experiment with the proposed method, preprocessing on the raw review data is required. The whole preprocessing work can be divided into four individual steps as follows.

Filtering non-English reviews. The raw review data may contain a number of review items that are not written in English, which needs to be filtered out. Also, similar to social media text, user reviews usually contain many commonly used slurs that are not regular English vocabularies. Our goal is to not filter out these words, as they likely contain sentiment related information, without which shall influence our experiment results. Overall, we screen out the non-English review sentences using Langdetect[2], a convenient language detecting package for Python language. Compared with PyEnchant[3], another language detecting package, Langdetect enables determining the language of text on sentence level. It shall remain the review data containing such English slurs.

Focusing on sentence-level granularity. Because each user review can contain more than one sentence, a multi-sentence review can contain multiple meanings, one for each sentence. Thus, we divide each review from a review set R_i into individual sentences. Hereafter, we use r_j to denote a review sentence in R_i. We use the sentence tokenizer feature from the NLTK[4] python package, with a further checking on the legitimacy of the sentences.

Sentiment Classification with VADER. To perform sentiment analysis on the collected app reviews, we select the Valence Aware Dictionary for sEntiment Reasoning (VADER) approach [10]. Compared with other sentiment analysis tools, VADER has a number of advantages regarding this study. Firstly, the classification accuracy of VADER on sentiment towards positive, negative and neutral classes is even higher than individual human raters in social media domain. In addition, its overall classification accuracies on product reviews from Amazon, movie reviews, and editorials from NYTimes also outperform other sentiment analysis approaches, such as SenticNet [6], SentiWordNet [1], Affective Norms for English Words [4], and Word-Sense Disambiguation [25], and run closely with the accuracy of an individual human. On the other hand, VADER approach is integrated into the NLTK package, which can be easily imported and performed using Python.

Filtering stop-words and non-verb-or-nouns and Stemming. In addition, the collected English review sentences are also transformed into lower cases, screened with stop-words, and stemmed before topic modeling. In order to obtain more meaningful topic modeling results, we add the words that connect to only general information but have a significant appearing rate in the reviews to the list of stop words. For example, the name of the app and the word app are of neither help towards topic modeling nor towards sentiment analysis. Furthermore, because the sentiment analysis is done previously, we select only the nouns and verbs in the review sentence sets to enhance the analysis result.

[2] https://pypi.python.org/pypi/langdetect
[3] https://pypi.python.org/pypi/pyenchant/
[4] http://www.nltk.org

3.2. Datasets

In this case study, we use the reviews submitted between 2016-09-13 and 2017-08-31 on the mobile app Whatsapp[5] on Android platform. We collected 1,148,032 reviews, which are then tokenized into 1,327,504 individual sentences. After eliminating the non-English review sentences, we have 1,012,088 English review sentences as experiment data.

We investigate the merits and issues concerning the major updates of Whatsapp released on Android platform. Within this period, 46 updates were released. On average, the app has been updated nearly every 7 - 8 days with one major update in every 50 days. By observing the content of each update from the given information on Google Play, we find that some consecutive updates contain the same content based on their descriptions. Therefore, we consider the first update of a set of updates which contain same descriptions as a major update, with the rest of the updates as minor updates. Accordingly, seven major updates are identified during the given period. They divided the review dataset into seven parts. Each sub review set is seen as the reviews regarding one particular update. Amongst the major updates of Whatsapp during this period, each update $\{u_1, u_2 \ldots u_7\}$ provides unique changes in UI design and user experiences. By classifying all selected review sentences into positive and negative using sentiment analysis and supervised classification with Naive Bayes Classifier, the number of review sentences in each segment is listed in Table 1. Accordingly, the number of reviews for each review sets of $R_1^+ \sim R_7^+$ and $R_1^- \sim R_7^-$, are shown in the figure below. Overall, the total number of positive reviews around the particular update is bigger than the number of negative reviews. Meanwhile, the monthly review number increased sharply after this particular major update.

Table 1. The number of Positive and Negative reviews after each Update

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	
Release Date	16.9.13	16.10.11	16.12.1	17.3.14	17.4.18	17.5.17	17.7.13	
Positive	39241	92222	246384	99889	50205	89766	72960	690667
Negative	15238	31674	165723	38174	18254	27646	24712	321421
Sum	54479	123896	412107	138063	68459	117412	97672	1012088

After sentiment analysis and classification, we perform the LDA topic analysis to identify the topics of the review set $R_1^+ \sim R_7^+$ and $R_1^- \sim R_7^-$, in order to investigate the users' opinions concerning the update and further verify the previously proposed hypothesis. To train the LDA topic models, we need to set the number of topics k. Based on an experimentally study regarding the quality of the topics produced for different k values, and select $k = 10$. Overall, for the collected 1,012,088 review sentences on Whatsapp, we perform an LDA topic analysis on the review set. For each of the 14 sets of review data, we use the Gensim[6] topic modeling toolbox to train the 10-topic LDA models. For the 14 LDA models, each individual topic is represented by 20 keywords. Then, we assign each review sentence in the LDA models to the topic to which it has the highest probability to belong.

[5]https://www.whatsapp.com/

[6]https://radimrehurek.com/gensim/

3.3. Topic Evolution Analysis with Similar Topic Chains

Each topic contains 20 keywords. For every major update u_i, there are four sets of associated reviews which results in four sets of topics, i.e., T^+_{i-1}, T^-_{i-1}, T^+_i, and T^-_i. We assess the similarity of topics after the update u_i, which provides 400 topics pairs. The number of common keywords for those 400 pairs range from 0 - 14. Since it is difficult to identify the topic content based on less than five keywords, we set an initial threshold equal to 0.143 for the Jaccard Extended similarity value, meaning that we consider two topics as similar only when they have five or more common keywords with the probability of these keywords above average. Based on this threshold, we apply Algorithm 1 to select the unique similar topic pairs in each of the four comparison sets (before and after each update, positive and negative reviews). Part of the results are shown in Figure 2. In Figure 2, positive topics are illustrated as blue circle while negative topic as red square. For example, $t^+_{1,7}$ is denoted as a blue circle marked "7" in column "T1" when $t^-_{2,10}$ is denoted as a red square marked "10" in column "T2".

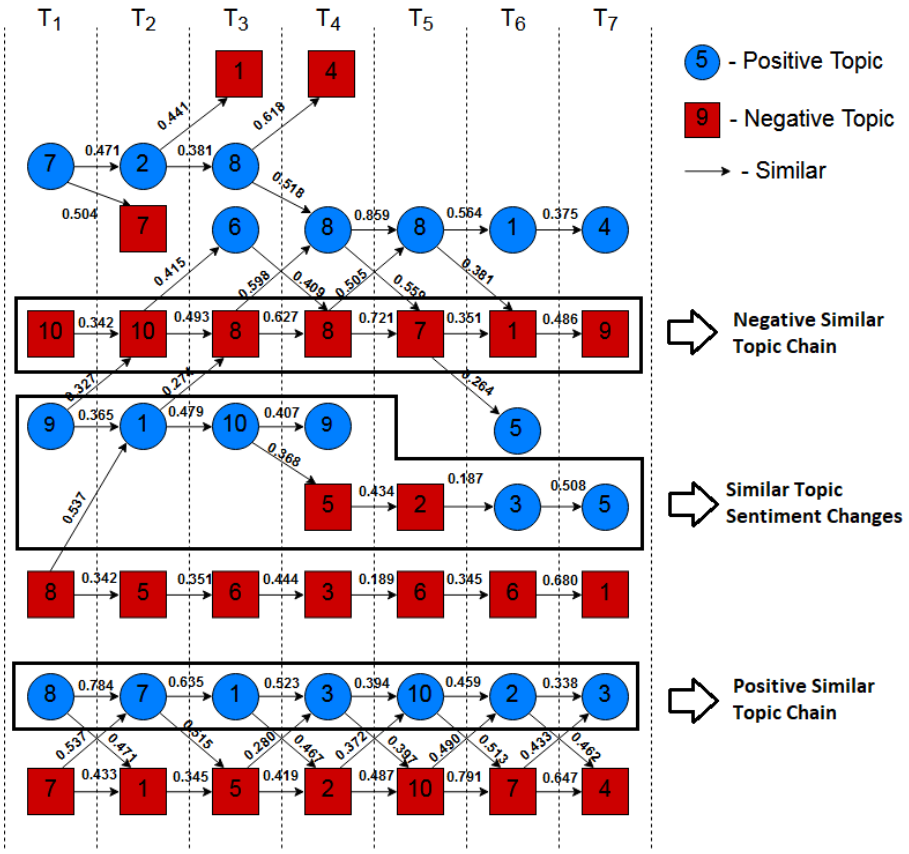

Figure 2. A Segment of the Topic Similarity Graph

We can firstly analyze the topic similarity between two sets of reviews. Based on the common keywords of each identified similar topic pairs, we could extract the users'

opinions regarding the particular update. For example, in Figure 2, we can identify that $t^+_{1,7}$, $t^+_{1,9}$, and $t^+_{1,8} \in T^+_1$ are similar to $t^+_{2,2}$, $t^+_{2,1}$, and $t^+_{2,7} \in T^+_2$ respectively. The common keywords for these three topic pairs are listed in Table 2. Therefore, these similar topic pairs suggest that the users reflect positively regarding the four topics both before and after Update 2 (Version 16.10.11), including the quality of video and voice call, asking for fixing and updating themes and emoji feature, and, contacts and social group option with picture sending feature. Similarly, by identifying the similar topic pairs ($t^-_{1,10}$, $t^-_{2,10}$), ($t^-_{1,8}$, $t^-_{2,5}$), and ($t^-_{1,7}$, $t^-_{2,1}$), we can also conclude that the users are still not happy regarding the function of contact group and its options, message sending and receiving, and the quality of video calls after Update 2. Similarity, by analyzing the common keywords between positive topics and negative topics, such as, ($t^+_{1,9}$, $t^-_{2,10}$) and ($t^-_{1,7}$, $t^+_{2,7}$) in Figure 2, we can also identify the users' opinion changes after Update 2. The common keywords of topic pair ($t^+_{1,9}$, $t^-_{2,10}$) reflect that the users start to complain negatively concerning the feature of sending image and picture in chat group after the update. The common keywords of ($t^-_{1,7}$, $t^+_{2,7}$) then reflects users become more satisfied with the general call feature as well as the video and voice quality.

Table 2. Common keywords of Similar Topic Pairs Regarding Update 2

Topic Pairs	Common Keywords
($t^+_{1,7}$, $t^+_{2,2}$)	['chang', 'emoji', 'fix', 'plea', 'pls', 'theme', 'updat', 'want']
($t^+_{1,9}$, $t^+_{2,1}$)	['contact', 'group', 'option', 'photo', 'pic', 'pictur', 'send', 'share', 'want']
($t^+_{1,8}$, $t^+_{2,7}$)	['ad', 'add', 'call', 'facil', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic']
($t^-_{1,10}$, $t^-_{2,10}$)	['add', 'contact', 'group', 'list', 'option', 'want']
($t^-_{1,8}$, $t^-_{2,5}$)	['come', 'get', 'messag', 'open', 'phone', 'read', 'receiv', 'see', 'seen']
($t^-_{1,7}$, $t^-_{2,1}$)	['call', 'featur', 'option', 'qualiti', 'video', 'work']
($t^+_{1,9}$, $t^-_{2,10}$)	['contact', 'group', 'imag', 'list', 'option', 'pictur', 'send', 'status', 'want']
($t^-_{1,7}$, $t^+_{2,7}$)	['call', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic', 'work']

Furthermore, by investigating the similar topic pairs throughout the evolution timeline of a particular app, we can detect the evolution trend of the app. For example, from Update 1 to Update 7, the positive opinion regarding each update can be reflected by the common keywords of T^+_1 and T^+_2 to T^+_6 and T^+_7. Therefore, the merit of the app throughout these updates can be reflected by the similar topic chain shown in Table 3.

Table 3. Common keywords in a Positive Similar Topic Chain

Topic Pairs	Common Keywords
($t^+_{1,8}$, $t^+_{2,7}$)	['ad', 'add', 'call', 'facil', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic']
($t^+_{2,7}$, $t^+_{3,1}$)	['call', 'facil', 'make', 'need', 'qualiti', 'vedio', 'video', 'voic']
($t^+_{3,1}$, $t^+_{4,3}$)	['call', 'improv', 'make', 'need', 'qualiti', 'video', 'voic']
($t^+_{4,3}$, $t^+_{5,10}$)	['call', 'data', 'improv', 'make', 'qualiti', 'send', 'video', 'voic']
($t^+_{5,10}$, $t^+_{6,2}$)	['call', 'group', 'improv', 'make', 'qualiti', 'send', 'video', 'voic']
($t^+_{6,2}$, $t^+_{7,3}$)	['call', 'featur', 'group', 'make', 'send', 'video', 'voic']

From the similar topic pairs identified in Figure 2, we could claim that topic $t^+_{1,8}$, $t^+_{2,7}$, $t^+_{3,1}$, $t^+_{4,3}$, $t^+_{5,10}$, $t^+_{6,2}$, and $t^+_{7,3}$ are all identified as similar to one another. The similarity is also reflected by the common keywords listed in Table 3, based on which we could conclude

from Update 2 to Update 7, the users constantly share positive opinions regarding the video and voice call quality of the app, which verifies Similarly, the constant negative opinions can be reflected by the common keywords of the similar negative topic chains throughout updates. For example, topic $t^-_{1,10}$, $t^-_{2,10}$, $t^-_{3,8}$, $t^-_{4,8}$, $t^-_{5,7}$, $t^-_{6,1}$, and $t^-_{7,9}$ are identified as similar (shown in Table 4). Such negative topic similarity chain reflects the users' request for features regarding options of contact and status.

Table 4. Common keywords in a Negative Similar Topic Chain

Topic Pairs	Common Keywords
$(t^-_{1,10}, t^-_{2,10})$	['add', 'contact', 'group', 'list', 'option', 'want']
$(t^-_{2,10}, t^-_{3,8})$	['add', 'contact', 'list', 'option', 'pictur', 'remov', 'see', 'status', 'want']
$(t^-_{3,8}, t^-_{4,8})$	['add', 'featur', 'need', 'option', 'photo', 'put', 'remov', 'status', 'stori', 'text']
$(t^-_{4,8}, t^-_{5,7})$	['add', 'featur', 'option', 'plea', 'remov', 'status', 'stori', 'text']
$(t^-_{5,7}, t^-_{6,1})$	['featur', 'option', 'peopl', 'status', 'want']
$(t^-_{6,1}, t^-_{7,9})$	['contact', 'featur', 'list', 'peopl', 'see', 'seen', 'status', 'view']

By investigating the update history of Whatsapp, we we can observe that the "*video calling*" feature was added to Whatsapp at Update 3 (Version 16.12.1). This can explain why the common keyword '*call*' became the highest probable common keyword from $(t^+_{2,7}, t^+_{3,1})$. On the other hand, Whatsappp added one update on "*WhatsApp Status: Post photos, videos, and GIFs to your status and share with your contacts what's going on throughout your day. Status updates from your contacts appear in the Status tab, and they'll disappear after 24 hours. Long press on a contact's name in the Status tab to mute their updates.*" in Update 4 (Version 17.3.14) and another update "*You can now send multiple contact cards at once*" in Update 5 (Version 17.4.18). This partially proves the developers noticed the constant negative opinions of users shown from Table 4

Moreover, in Figure 2, we can also identify particular topics are considered similar to one positive topic and one negative topic from the next review set simultaneously. For example, $t^+_{3,10}$ is similar to both $t^+_{4,9}$ and $t^-_{4,5}$. From the common keywords of topic pair $(t^+_{2,1}, t^+_{3,10})$, we can observe that the users share positive opinion regarding the feature of contacting and chat groups. Then topic pair $(t^+_{3,10}, t^+_{4,9})$ contains common keywords of ['add', 'featur', 'group', 'make', 'option', 'person', 'provid', 'show'], when $(t^+_{3,10}, t^-_{4,5})$ contains common keywords of ['ad', 'contact', 'group', 'list', 'peopl', 'see', 'seen', 'show']. These keywords suggest that after Update 4, a number of users are satisfied with the chat group feature and options, when some users are not happy with the contact and chat list. After Update 5, topic pair $(t^-_{4,5}, t^-_{5,2})$ suggests that the complaints regarding contact and chat list continues until Update 6 where $(t^-_{5,2}, t^+_{6,3})$ suggests that the complaints regarding contact and chat list is not significant any more. By investigating the update history of Whatsapp, we can observe that in Update 6 (Version 17.5.17) the developers added a feature "*Pin chats to the top of your chat list, so you can quickly find them. Just tap and hold on a chat and tap the pin icon at the top of your screen*", which results in the reduce in negative opinions regarding the chat list feature. It also verifies the existence of the significant complaints before this update regarding this matter.

4. Related Work

The spontaneous feedback from the users, i.e., the user reviews, helps effectively the evolution of the target system, where a key to enable such feedback is to ease the users' effort in composing and uploading it [23]. For the contemporary mobile apps, the app distribution platforms, e.g., Apple AppStore, and Google Play, enable such spontaneous reviews of the users and facilitate the developers in terms of maintaining and evolving mobile apps [20]. Such feedback and reviews contain helpful information for developers in terms of identifying missing features and improving software quality [9]. From those review information, user requirements can be elicited continuously and, in a crowd-based fashion [11, 24].

The reviews of the mobile app users reflect a variety of topics, which majorly cover the perspectives of bug reports, feature requests, user experiences, solution proposal, information seeking and giving, and general ratings [17, 19, 21]. For such purpose, natural language processing (NLP), sentiment analysis (SA), and supervised learning are the common techniques used to classify user reviews [19, 21]. According to recent studies, the combination of NLP and SA techniques has high accuracy in detecting useful sentences [21]. These techniques are also used in many other studies in terms of review opinion mining [7, 9, 15].

Many studies have contributed to the user opinion mining of mobile apps. Fu et al. [8] proposes WisCom, which can detect why the users like or dislike a particular app based on the users' reviews throughout time and provide insights regarding the users' concerns and preferences in the app market. Similar studies regarding mining information from app stores data focus on different perspectives of the issues, e.g., the correlations between ratings and download rank [14], ratings and API use [2], review classification and useful sentences detection [21]. On the other hand, Chen et al. [7] provides the AR-miner framework, which facilitates the informative reviews extraction, review grouping based on topics, review prioritization and informative reviews presentation with visualization. Guzman and Maalej [12] proposes an approach focusing on feature extraction and sentiment analysis, which facilitates the evaluation of individual app features. Similarly, Iacob and Harrison [15] also focuses on the feature requests extraction but via means of linguistic rules and LDA topic modeling.

Compared to the previously mentioned approaches in user review opinion mining, our method aims towards the similar topic detection and analysis concerning not only the trend in users' opinions in general but also the specific users' opinion changes at particular updates. Furthermore, we focus on the topic similarity of data segments, classified by sentiment analysis and supervised learning, which is different from the methods mentioned above. It enables the developers to acquire the overall perspective regarding the users' opinions throughout the mobile app evolution phase, but also the information regarding each individual update providing insights on the future updates.

5. Conclusion

In this paper[7], we propose a method for analyzing the mobile apps user reviews for a series of updates. After classifying the reviews before and after a particular update by

[7]The work was partially supported by the TEKES Finnish project Virpa D.

positive and negative sentiment, we extract the topics of each segment. By comparing the similarities of these extracted topics, we identify not only the positive and negative issues reflected by these reviews regarding particular updates, but also the overall trend in users' opinions regarding the app in general as well as regarding specific aspects. Overall, this study is an exploratory investigation on using user review opinion mining techniques in detecting update-specific issues and user opinion evolution. The future studies shall extend the use of this method by taking into account other factors, e.g., the different app categories, different platforms, etc. And different mining and analysis techniques will also be taken into account. Furthermore, a mobile app evolution recommend system can be developed based on this method to provided maintenance and evolution suggestions to the developers regarding the continuous user opinion mining result.

References

[1] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, 2010.

[2] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk. The impact of api change-and fault-proneness on the user ratings of android apps. *IEEE Trans. on Soft. Engin.*, 41(4):384–407, 2015.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] M. M. Bradley and P. J. Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer, 1999.

[5] P. Fafalios, V. Iosifidis, K. Stefanidis and E. Ntoutsi. Multi-aspect Entity-centric Analysis of Big Social Media Archives. In *Proceedings of TPDL*, 2017.

[6] E. Cambria, R. Speer, C. Havasi, and A. Hussain. Senticnet: A publicly available semantic resource for opinion mining. In *AAAI fall symposium: commonsense knowledge*, volume 10, 2010.

[7] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[8] 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. In *KDD*, 2013.

[9] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In *ICSE*, 2013.

[10] C. H. E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, 2014.

[11] E. C. Groen, J. Doerr, and S. Adam. Towards crowd-based requirements engineering a research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2015.

[12] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *RE*, 2014.

[13] I. Ntoutsi, K. Stefanidis, K. Rausch and H. P. Kriegel. Strength Lies in Differences - Diversifying Friends for Recommendations through Subspace Clustering. In *Proceedings of CIKM*, 2014.

[14] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: Msr for app stores. In *MSR*, 2012.

[15] C. Iacob and R. Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Mining Software Repositories*, 2013.

[16] K. Stefanidis, H. Kondylakis, and G. Troullinou. On Recommending Evolution Measures: A Human-aware Approach. In *Proceedings of ICDE Workshops*, 2017.

[17] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan. What do mobile app users complain about? *IEEE Software*, 32(3):70–77, 2015.

[18] K. Stefanidis, E. Pitoura and P. Vassiliadis. Managing Contextual Preferences. *Information Systems*, 36(8):1158–1180, 2011.

[19] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *RE*, 2015.
[20] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *RE*, 2013.
[21] S. Panichella, A. Di 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 *ICSME*, 2015.
[22] Y. Roussakis, I. Chrysakis, K. Stefanidis, G. Flouris, and Y. Stavrakas. A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets. In *Proceedings of ISWC*, 2015.
[23] K. Schneider. Focusing spontaneous feedback to support system evolution. In *RE*, 2011.
[24] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *RE*, 2010.
[25] M. Stevenson and Y. Wilks. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265, 2003.
[26] Y. Roussakis, I. Chrysakis, K. Stefanidis, and G. Flouris. D2V: A Tool for Defining, Detecting and Visualizing Changes on the Data Web. In *Proceedings of ISWC*, 2015.