Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates

Xiaozhou Li, Zheying Zhang, Kostas Stefanidis

Faculty of Natural Sciences, University of Tampere
Tampere, Finland
Email: xiaozhou.li@uta.fi, zheying.zhang@uta.fi, kostas.stefanidis@uta.fi

***Abstract*—The contemporary online mobile application (app) market enables users to review the apps they use. These reviews are important assets reflecting the users needs and complaints regarding the particular apps, covering multiple aspects of the mobile apps quality. By investigating the content of such reviews, the app developers can acquire useful information guiding the future maintenance and evolution work. Furthermore, together with the updates of an app, the users reviews deliver particular complaints and praises regarding the particular updates. Despite that previous studies on opinion mining in mobile app reviews have provided various approaches in eliciting such critical information, limited studies focus on eliciting the user opinions regarding a particular mobile app update, or the impact the update imposes. Hence, this study proposes a systematic analysis method to elicit user opinions regarding a particular mobile app update by detecting the similar topics before and after this update, and validates this method via an experiment on an existing mobile app.**

Keywords–Mobile app; review; sentiment analysis; topic modeling; topic similarity

I. INTRODUCTION

The increasing number of smart phone 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 (e.g., Apple app store, and Google Play). These platforms also provide an important channel enabling the users to provide feedback to the app. The ratings and comments given by the users at a particular time reflect their opinions regarding the overall app and the specific version of that app. While the app developers, continuously or sporadically, update their apps, the retrieved user reviews reflect not only their overall opinion changes throughout the evolution time but also their specific complaints and praises regarding the specific app version [1]. Such specific complaints, regarding various aspects [2], shall enable the developers to be aware of the issues and tackle them accordingly.

Existing studies have proposed different approaches to identifying 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 [3] [4]. Other perspectives, such as user preferences, app evaluation, user satisfaction, relation between download and rating, feature extraction, review prioritization and so on, have also been widely studied [5]–[9]. However, limited 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 users' attitude towards a particular issue changes when new updates are released and how the reviews have impacts on app's maintenance and evolution.

In this paper, we investigate the correlation between users' positive and negative reviews before and after an app's release. We consider two dimensions: time and sentiment. Specifically, we divide user reviews based on major updates, and distinguish them between those precede and follow the particular updates. Each group of reviews are further divided into positive and negative ones using sentiment analysis. We devise an approach to measuring 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 how to detect users' opinion changes regarding a particular update. Furthermore, detecting the users' update-specific opinions shall also help the developers be aware of the users opinion on a particular release and guides them proactively to address the most important issues early.

The remainder of this paper is organized as follows. Section II introduces the method with details. Section III presents a case study using this method. Section IV introduces the related works when Section V concludes the paper.

II. METHOD DESCRIPTION

In this section, we introduce our method with the main goal to detect the correlation between the content of app reviews before and after the app updates done by the app company through the app maintenance lifecycle. Such correlation shall show the degree in 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 each 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. Accordingly, Subsection A introduces the overall procedure of the method and brings forth the hypotheses it aims to verify. Subsection B and C introduce respectively how to analyze the sentiment and topics of user reviews. Subsection D introduces how to calculate the similarity between topics, when Subsection E presents how to identify the matching similar topics between review sets.

A. Preliminaries

Let R be a collection of user reviews for a particular mobile app A, covering a particular time period. Therein, each review $r_i \in R$ is associated with a particular time point, at which

the review r_i is published. Let also U be the set of updates released by the app developers within the time period with each update u_i is released at a particular time point as well. Therefore, we consider each review r_i, which is published after the release time of update u_i and before that of the next update u_{i+1}, as a review 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 a time period, 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 correlated with the last update before u_1 or the first version of A if u_1 is the first update of A. For each review $r_j \in R_i$, a sentiment score shall be calculated and assigned to r_j, whose the 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 positive review set R_i^+ and negative review set R_i^+, where $R_i^+ \cup R_i^- = R_i$ and $R_i^+ \cap R_i^- = \emptyset$ with an acceptable accuracy.

We further investigate the main topics for each of the positive and negative review sentence sets. 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 comparing the similarities and changes between T_{i-1}^+, T_{i-1}^-, T_i^+, and T_i^-. Specifically, the following hypotheses shall be verified.

- H1. The topic similarities between T_{i-1}^+ and T_i^+ reflect the merits regarding the app A in general.
- H2. The topic similarities between T_{i-1}^+ and T_i^- reflect the uncomfortable changes in the update u_i.
- H3. The topic similarities between T_{i-1}^- and T_i^+ reflect the improvement in the update u_i.
- H4. The topic similarities between T_{i-1}^- and T_i^- reflect the remaining issues regarding the app A.

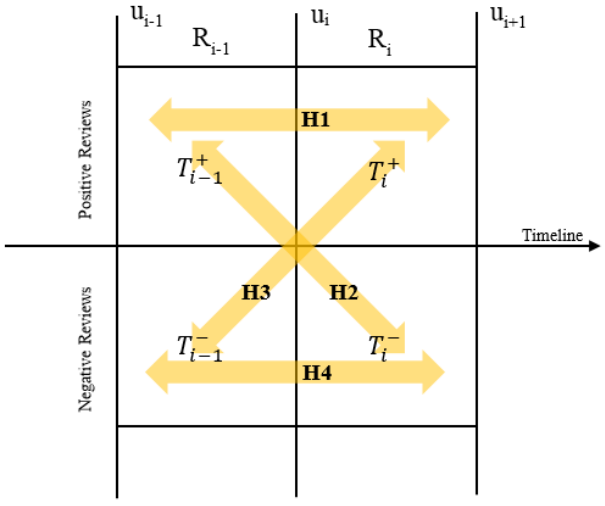

Figure 1. Relationship between hypotheses and updates.

Thus, the results obtained from these hypotheses for updating u_i provide the following information: 1) the merits and issues for the app A in general, 2) the merit and issues specific to the update u_i, and 3) the improvements and drawbacks of u_i compared with u_{i-1}. The merits and issues for app A and those for each $u_i \in U$ shall be recorded as the evolution status of app A throughout period T, which can be used as the reference to guide planning the following updates. Figure 1 visually depicts the focus of our hypotheses for updating u_i.

B. 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 [10]. 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 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).

Algorithm 1 Sentiment Classification

```
1: procedure SENTIMENT ANALYSIS
2: lexicons ← selected sentiment lexicon
3: R_i ← review set
4: W_j ← word set of r_j(r_j ∈ R_i)
5: For each r_i in R_i
6:     For each w_z in W_j
7:         s_z ← polarity and valence analysis(w_z, lexicons)
8:         list ← list.append(s_z)
9:     S_j ← grammatical and syntactical heuristics
       (r_j, list, heuristics)
10:    if S_j > 0 then
11:        R_i^+.append(r_j).
12:    if S_j = 0 then
13:        R_i^0.append(r_j).
14:    if S_j < 0 then
15:        R_i^-.append(r_j).
16: trainMatrix ← train(R_i^+[: 8000], R_i^-[: 8000])
17: R_i^{0+}, R_i^{0-} ← naive bayes classifier(R_i^0, trainMatrix)
18: R_i^+.extend(R_i^{0+})
19: R_i^-.extend(R)i^{0-})
20: return R_i^+, R_i^-
```

Figure 2. Algorithm for Sentiment Classification

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 following:

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

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. To further investigate the information in R_i^0, after experimentally observing that typically includes a big number of reviews, we classify it into positive and negative using the Naive Bayes Classifier with the training data from R_i^+ and R_i^-. This way, 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. The process is described in Figure 2.

C. Topic Analysis

After dividing the review sets R_i and R_{i-1}, i.e., the review sets related to update u_i, into R_i^+, R_i^-, R_{i-1}^+ and R_{i-1}^- based on the sentiment classification method, we elicit the main topics from each of the classified review sets by exploiting the Latent Dirichlet Allocation (LDA) method [11]. 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 in 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 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 are returned as 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.

D. Calculating Topics Similarities

Based on the topic sets T_i^+, T_i^-, T_{i-1}^+ and T_{i-1}^- elicited from the review sets R_i^+, R_i^-, R_{i-1}^+ and R_{i-1}^-, 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:

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

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 t 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: $J(t_i, t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}$.

However, by using the Jaccard Similarity, we consider two given topics are similar only when they contain a particular number of common keywords, regardless of the probability of them. The meaning of each topic $t_i \in T$, denoted as $[kw_{i,1}, kw_{i,2}, ...kw_{i,k}]$, shall be more likely reflected by the high-probability keywords of t_i. Furthermore, the subset of only low-probability keywords may reflect different meanings. For example, a topic is denoted as {'update': 0.143, 'problem': 0.096, 'fix': 0.064, 'install': 0.03, 'uninstall': 0.029, 'open': 0.027, 'stop': 0.025, 'plea': 0.025, 'get': 0.022, 'reinstall': 0.019, 'bug': 0.019, 'start': 0.019, 'need': 0.014, 'application': 0.014, 'issue': 0.011, 'battery': 0.011, 'help': 0.01, 'face': 0.009, 'frustrate': 0.009, 'day': 0.008}. From the high-probability keywords of this topic, we can summarize that the topic is regarding the problems of updating, which requires being fixed. However, the low-probability keywords hardly reflect the topic, e.g., a keyword subset, {'reinstall', 'bug', 'start', 'need', 'application', 'issue', 'battery'}, reflects a very different issue regarding bugs and batteries.

Hence, when comparing the similarity of two given topics, the probability of the common keywords shall be taken into account. Considering that Jaccard coefficient is the normalized inner product [12], we herein adopt the similarity measure method incorporating also the inner product, the Kumar-Hassebrook (KH) similarity [13]. Provided between topic t_i and t_j, the c common keywords are denoted as $[kw_{ij,1}, kw_{ij,2}, ...kw_{ij,c}]$, with 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 similarity of the two given topics are calculated as follows.

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

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 monotonically, and the denominator decreases monotonically. Therefore, $KH(t_i, t_j)$ increases when t_i and t_j have more keywords in

common. In addition, when the probability of the common keywords increases, $\sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}$ increases. Because the denominator is greater than the numerator, and both are greater than 0, $KH(t_i, t_j)$ increases when the probabilities of the common keywords of t_i and t_j increase.

In this way, for two given topics t_i and t_j, when each keyword of these two topics is assigned the average value of the probability value set, then $KH(t_i, t_j) = J(t_i, t_j)$. Considering the monotonical increasing of the KH Similarity formula, it means that for t_i and t_j, when $KH(t_i, t_j) < J(t_i, t_j)$, the common keywords of these two topics hardly reflect the meaning of them. For example, two topics, denoted as the following set of keywords with the according probability of each keywords, are listed in Table I.

TABLE I. EXAMPLE TOPICS WITH KEYWORDS AND PROBABILITY

Topic 1	{'problem': 0.145, 'fix': 0.081, 'download': 0.051, 'please': 0.032, 'use': 0.025, 'reason': 0.021, 'service': 0.02, 'user': 0.015, 'issue': 0.015, 'data': 0.014}
Topic 2	{'version': 0.197, 'please': 0.121, 'go': 0.069, 'use': 0.043, 'option': 0.036, 'one': 0.028, 'lot': 0.027, 'revert': 0.021, 'way': 0.018, 'download': 0.018}

The Jaccard Similarity of these two topic is 0.176 when the KH Similarity is 0.064. We can observe that the common keywords, {'download', 'please', 'use'}, are of low probability in Topic 1 and keywords {'please', 'use'} in Topic 2, while neither topic is reflected by the common keywords. Topic 1 can be seen regarding the requests of fixing problems/bugs when Topic 2 is more related to keyword 'version' instead of 'download'. Thus, these two given topics cannot be considered as similar despite the high Jaccard Similarity.

E. Identifying Matching Topics

After computing similarities between pairs of review topics with KH similarity, we shall 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 high similarity. Starting from the pair of topics with highest similarity values,

We firstly use the Jaccard Similarity value of two particular topics as the threshold for their KH similarity. According to the formula of KH similarity given previous, we set the probability of each keyword in each topic equal to the average. Then

$$KH(t_{ai}, t_{bj}) = \frac{\sum_{x=1}^{c} p^2}{\sum_{x=1}^{k} p^2 + \sum_{x=1}^{k} p^2 - \sum_{x=1}^{c} p^2}$$

$$= \frac{c}{k + k - c} = J(t_{ai}, t_{bj})$$

Therefore, we select the topic pairs, whose KH similarity value greater than their Jaccard similarity value, as similar topics. Furthermore, from the topic pairs with the highest KH similarity value, we select the top n pairs of topics to investigate the changes and similarities of users opinion regarding the app.

Algorithm 2 Matching Topics Identification

```
procedure MATCHING TOPICS IDENTIFICATION
k ← number of topics
n ← number of selected similar topics
Ta ← {t_a1, t_a2, ..., t_ai, ..., t_ak}
Tb ← {t_b1, t_b2, ..., t_bj, ..., t_bk}
KHs_ij ← KHSimilarity(t_ai, t_bj)
Js_ij ← JaccardSimilarity(t_ai, t_bj)
For i, j ∈ {1, 2, ..., k}
    if KHs_ij ≥ Js_ij then
        S ← s_ij
If len(S) > n
    For each (t_ax, t_by) in S[:n]:
    Result.append(Summarize(t_ax, t_by))
Else:
    For each (t_ax, t_by) in S:
    Result.append(Summarize(t_ax, t_by))
Return Result
```

Figure 3. Algorithm for Matching Topic Identification

The aim of the algorithm (shown in Figure 3) is to select 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 and have also high similarity, will be considered as references to interpret the users opinions. Each particular topic generated by the LDA model contains a number of perspectives that can be interpreted by the keywords. Thus, it is possible that one particular topic have the similar similarity value to multiple topics, when they are similar regarding different perspectives which represented by their different common keywords.

TABLE II. EXAMPLE TOPICS WITH KEYWORDS AND PROBABILITY

Topic 1	{'time': 0.12, 'friend': 0.091, 'talk': 0.081, 'way': 0.069, 'see': 0.048, 'people': 0.045, 'communication': 0.038, 'want': 0.03, 'face': 0.023, 'world': 0.02 }
Topic 2	{'friend': 0.111, 'bring': 0.07, 'connect': 0.068, 'talk': 0.06, 'keep': 0.059, 'family': 0.054, 'application': 0.037, 'way': 0.021, 'touch': 0.017, 'contact': 0.016 }
Topic 3	{'see': 0.077, 'use': 0.056, 'people': 0.046, 'want': 0.044, 'contact': 0.034, 'thing': 0.031, 'year': 0.027, 'find': 0.023, 'know': 0.023, 'number': 0.019 }

For example, in Table II Topic 1 has the same Jaccard similarity value to both Topic 2 and 3. The two pairs of common keywords are {'friend', 'talk', 'way'} and {'people', 'see', 'want'}. Their KH similarity values are different but both high (0.276 and 0.137). From Topic 1, we could summarize that it is regarding using the app enabling people to communicate with friends any time they want and can see their faces as well. Despite it is considered similar to both Topic 2 and 3, Topic 2 focuses on the perspective of enabling communication between families and friends, when Topic 3 focus more on the perspective of contacting people with phone numbers and seeing them. Thus, by identifying both similar topic pairs, we shall have more thorough understanding of the users' opinions regarding the app.

III. CASE STUDY

A. 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 three 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 [14], a convenient language detecting package for Python language. Compared with PyEnchant [15], 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. Due to the fact that 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 [16] python package, with a further checking on the legitimacy of the sentences.

Filtering stop-words and lemmatization. In addition, the collected English review sentences are also transformed into lower cases, screened with stop-words, and lemmatized before topic modeling. In order to obtain more meaningful topic modeling results, we add the words that connect to only general information but have 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. In addition, considering the fact that the sentiment of each review item is identified, we eliminate all adjectives from the obtained tokens and select only nouns and verbs as tokens via the pos_tag function of NLTK.

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 [17], SentiWordNet [18], Affective Norms for English Words [19], and Word-Sense Disambiguation [20], and run closely with the accuracy of individual human. On the other hand, VADER approach is integrated in the NLTK package, which can be easily imported and performed using Python.

B. Dataset

Our study relies on real data. In particular, we focus on reviews submitted for 1-year period of Skype on the Android platform. We collected 153,128 user reviews submitted from 1.9.2016 to 31.8.2017. The reviews are tokenized into 234,064 individual sentences. After filtering the non-English review sentences, the number is reduced to 174,559.

We investigate the merits and issues concerning the major update of Skype released on Android platform on 1.6.2017 (u_i = version-1.6.2017). Within this period from 1.9.2016 to 31.8.2017, 76 updates were released. On average, the app has been updated nearly every five days. By observing the content of each update from the given information of Google Play, we find that some consecutive updates contain exact 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, when the rest of the update set as minor update. Amongst the major updates of Skype during this period, the update u_i provide significant 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 III. Accordingly, the review sets R^+_{i-1}, R^-_{i-1}, R^+_i, and R^-_i, for this study, consists of 65580, 29970, 36703, and 42306 reviews.

TABLE III. POSITIVE AND NEGATIVE REVIEWS AROUND A PARTICULAR APP UPDATE

	Positive reviews	Negative reviews	Total
Before 1.6.2017	65,580	29,970	95,550
After 1.6.2017	36,703	42,306	79,009
Total	102,283	72,276	174,559

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. Opposite to the situation before the update, we observe that after the update more negative reviews are given by the users than the positive ones, meaning that many users are not satisfied with this particular update or the app overall.

TABLE IV. NUMBER OF REVIEWS PER TOPIC

$t^+_{(i-1)1}$	$t^+_{(i-1)2}$	$t^+_{(i-1)3}$	$t^+_{(i-1)4}$	$t^+_{(i-1)5}$
13627	3104	3168	6283	4725
$t^-_{(i-1)1}$	$t^-_{(i-1)2}$	$t^-_{(i-1)3}$	$t^-_{(i-1)4}$	$t^-_{(i-1)5}$
8692	6016	4122	6105	9738
t^+_{i1}	t^+_{i2}	t^+_{i3}	t^+_{i4}	t^+_{i5}
3519	2892	4204	1895	3433
t^-_{i1}	t^-_{i2}	t^-_{i3}	t^-_{i4}	t^-_{i5}
2409	2700	2794	2408	3716
$t^+_{(i-1)6}$	$t^+_{(i-1)7}$	$t^+_{(i-1)8}$	$t^+_{(i-1)9}$	$t^+_{(i-1)10}$
7768	2796	3134	3186	3391
$t^-_{(i-1)6}$	$t^-_{(i-1)7}$	$t^-_{(i-1)8}$	$t^-_{(i-1)9}$	$t^-_{(i-1)10}$
4810	3399	3244	2798	2177
t^+_{i6}	t^+_{i7}	t^+_{i8}	t^+_{i9}	t^+_{i10}
4818	4132	4463	3237	4635
t^-_{i6}	t^-_{i7}	t^-_{i8}	t^-_{i9}	t^-_{i10}
3139	3893	3229	6252	4508

After sentiment analysis and classification, we perform the LDA topic analysis to identify the topics of the review set R^+_{i-1}, R^-_{i-1}, R^+_i, and R^-_i, 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 174,559 review sentences on Skype, we perform an LDA topic analysis on the review set R^+_{i-1}, R^-_{i-1}, R^+_i, and R^-_i. For each of the 4 sets of review data,

we use the Gensim topic modeling toolbox to train the 10-topic LDA models. For the 4 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. The numbers of reviews for each topic of the 4 data sets appear in Table IV (t^+_{in} represents the n^{th} topic of R^+_i). In general, reviews are divided into topics smoothly and, in most cases, each topic consists of around 4000 review sentences.

C. Topics Similarity Analysis

With the 40 identified topics, each of which is represented by 20 keywords, we compare the similarity between each topic from the 20 topics before the update and each one from the 20 topics after the update, which provides 400 topics pairs. The Jaccard similarity values for those 400 pairs range from 0 to 0.429 (i.e., 0 - 12 common keywords between two topics). Meanwhile, the KH similarity values range from 0 to 0.676. Therein, we identify the potential similar topics from the highest KH similarity value and eliminate the results where the KH similarity value is lower than the according Jaccard similarity value. In this way, we guarantee the common keywords of each identified topic pairs contain the overall probability value greater than average. We select the seven topic pairs with the highest KH similarity values for each comparison set. For each topic pair, we analyze the similarity of the two topics by observing their common keywords. Furthermore, from the unique keywords of each topic from one particular topic pair, we could also see the topic changes. The seven topic pairs with the highest KH similarity value are depicted in Table V.

TABLE V. PAIRS OF SIMILAR TOPICS

T^+_{i-1} - T^+_i	$(t^+_{(i-1)8}, t^+_{i5})$ $(t^+_{(i-1)6}, t^+_{i6})$ $(t^+_{(i-1)7}, t^+_{i9})$ $(t^+_{(i-1)6}, t^+_{i1})$ $(t^+_{(i-1)6}, t^+_{i5})$ $(t^+_{(i-1)1}, t^+_{i7})$ $(t^+_{(i-1)8}, t^+_{i3})$
T^+_{i-1} - T^-_i	$(t^+_{(i-1)6}, t^-_{i10})$ $(t^+_{(i-1)6}, t^-_{i5})$ $(t^+_{(i-1)8}, t^-_{i7})$ $(t^+_{(i-1)5}, t^-_{i6})$ $(t^+_{(i-1)7}, t^-_{i8})$ $(t^+_{(i-1)8}, t^+_{i10})$ $(t^+_{(i-1)2}, t^+_{i3})$
T^-_{i-1} - T^+_i	$(t^-_{(i-1)3}, t^+_{i6})$ $(t^-_{(i-1)7}, t^+_{i3})$ $(t^-_{(i-1)5}, t^+_{i1})$ $(t^-_{(i-1)3}, t^+_{i5})$ $(t^-_{(i-1)10}, t^+_{i8})$ $(t^+_{(i-1)1}, t^+_{i4})$ $(t^+_{(i-1)6}, t^+_{i9})$
T^-_{i-1} - T^-_i	$(t^-_{(i-1)3}, t^-_{i10})$ $(t^-_{(i-1)10}, t^-_{i3})$ $(t^-_{(i-1)6}, t^-_{i8})$ $(t^-_{(i-1)7}, t^-_{i9})$ $(t^+_{(i-1)7}, t^+_{i6})$ $(t^+_{(i-1)8}, t^+_{i6})$ $(t^-_{(i-1)8}, t^-_{i9})$

By further analyzing the common keywords in the obtained similar topic pairs, we verify the hypothesis H1 - H4 as follows.

H1. The topic similarities between T^+_{i-1} and T^+_i reflect the merits regarding the app A in general. The common keywords of the seven topics pairs with the highest KH similarity value appear in Table VI. For example, the common keywords of topics ($t^+_{(i-1)8}$, t^+_{i5}) reflects the acknowledgement from the users before and after the update. Because, the common keywords 'work' and 'call' indicate that the core function of the app works properly. The common keywords of topics ($t^+_{(i-1)6}$, t^+_{i6}) reflect the users acknowledge also the benefits brought by the app, e.g., enabling people to chat in groups, with video or by calling, so that people can see and hear from their friends. On the other hand, topic pair ($t^+_{(i-1)7}$, t^+_{i9}) contains the common keywords that reflect the users needs in adding features and fixing bugs. Considering the overall positive sentiment of the review sets, it is also possible the users reflect their satisfaction towards the work done by the developers regarding such matters. Topic pair ($t^+_{(i-1)6}$, t^+_{i5}) reflects the users satisfied with the video and audio quality of the app. As topic t^+_{i5} is also similar to $t^+_{(i-1)8}$, both topic pairs reflect similar information. Topic pair ($t^+_{(i-1)1}$, t^+_{i7}) reflects the contribution of the app to the society in a bigger picture, regarding the communication between friends and family and helping people keep in touch. Topic pair ($t^+_{(i-1)6}$, t^+_{i1}) contain few common keywords; however, as $t^+_{(i-1)6}$ is also similar to topic t^+_{i6} and t^+_{i5}, all these topic pairs reflect similar information.

TABLE VI. COMMON KEYOWRDS IN POSITIVE-POSITIVE REVIEWS

Topic Pairs	Common Keywords
($t^+_{(i-1)8}$, t^+_{i5})	['call', 'phone', 'sound', 'work']
($t^+_{(i-1)6}$, t^+_{i6})	['call', 'chat', 'friend', 'group', 'hear', 'make', 'people', 'person', 'phone', 'see', 'video']
($t^+_{(i-1)7}$, t^+_{i9})	['add', 'bug', 'everything', 'fix', 'hope', 'issue', 'make', 'need', 'please']
($t^+_{(i-1)6}$, t^+_{i1})	['people', 'use', 'year']
($t^+_{(i-1)6}$, t^+_{i5})	['call', 'make', 'phone', 'quality', 'video', 'voice']
($t^+_{(i-1)1}$, t^+_{i7})	['application', 'communicate', 'connect', 'family', 'friend', 'get', 'help', 'touch', 'way']
($t^+_{(i-1)8}$, t^+_{i3})	['connection', 'internet', 'keep', 'nothing', 'work']

H2. The topic similarities between T^+_{i-1} and T^-_i reflect the uncomfortable changes in the update u_i. The common keywords of the selected similar topic pairs are shown in Table VII. The common keywords of topics ($t^+_{(i-1)6}$, t^-_{i10}) reflects negative user reviews regarding the apps core feature exist, despite the positive feedback before this update. According to t^-_{i10}, the aspects which users complain about include calling in general, the user interface, video and sound quality, and connections. Meanwhile, t^-_{i10} is also considered similar to $t^+_{(i-1)8}$. Topic $t^+_{(i-1)8}$ indicates that before the update many users like the internet connection of this app with wifi on computer and tablet. Topic pair ($t^+_{(i-1)5}$, t^-_{i6}) reflects that issues concerning the user accounts, including logging in, signing up, passwords emerge after the update, where the users complain quite often. Furthermore, the topic pair ($t^+_{(i-1)7}$, t^-_{i8}) indicates that many users complain about the developers fixing problems negatively, despite many others reflect the issue with positive sentiment (see topic pair ($t^+_{(i-1)7}$, t^+_{i9})).

TABLE VII. COMMON KEYOWRDS IN POSITIVE-NEGATIVE REVIEWS

Topic Pairs	Common Keywords
($t^+_{(i-1)6}$, t^-_{i10})	['call', 'connect', 'hear', 'make', 'person', 'phone', 'quality', 'video', 'voice']
($t^+_{(i-1)6}$, t^-_{i5})	['make', 'use', 'year']
($t^+_{(i-1)8}$, t^-_{i7})	['need', 'work']
($t^+_{(i-1)5}$, t^-_{i6})	['account', 'go', 'keep', 'let', 'log', 'password', 'sign', 'try', 'win']
($t^+_{(i-1)7}$, t^-_{i8})	['fix', 'please', 'problem', 'thing']
($t^+_{(i-1)8}$, t^-_{i10})	['call', 'connection', 'drop', 'phone', 'sound', 'work']
($t^+_{(i-1)2}$, t^-_{i3})	['conversation', 'get', 'take', 'time', 'type']

H3. The topic similarities between T^-_{i-1} and T^+_i reflect the improvement in the update u_i. The common keywords of the selected similar topic pairs are shown in Table VIII. The common keywords of topics ($t^-_{(i-1)3}$, t^+_{i6}) reflect that before the update, many users have complaint regarding using the app for phone calls in general. After the update, a number of positive

reviews towards this matter appear. Considering the topic pair ($t^-_{(i-1)3}$, t^+_{i5}), users also switch the attitude regarding the video and sound quality to positive after the update. Topic pair ($t^-_{(i-1)7}$, t^+_{i3}) reflects the general positive acknowledgement of the update. Topic pair ($t^-_{(i-1)10}$, t^+_{i8}) reflects the users attitude towards the message notification feature has changed into positive. ($t^-_{(i-1)1}$, t^+_{i4}) reflects the general positive feedback regarding the new version while ($t^-_{(i-1)6}$, t^+_{i9}) reflects the positive feedback on having some bugs fixed in this version.

TABLE VIII. COMMON KEYOWRDS IN NEGATIVE-POSITIVE REVIEWS

Topic Pairs	Common Keywords
($t^-_{(i-1)3}$, t^+_{i6})	['call', 'get', 'hear', 'make', 'people', 'person', 'phone', 'see', 'talk', 'time', 'video']
($t^-_{(i-1)7}$, t^+_{i3})	['get', 'update', 'win', 'work']
($t^-_{(i-1)5}$, t^+_{i1})	['get', 'try', 'use', 'year']
($t^-_{(i-1)3}$, t^+_{i5})	['call', 'make', 'phone', 'quality', 'sound', 'time', 'video', 'voice']
($t^-_{(i-1)10}$, t^+_{i8})	['message', 'notification', 'open', 'see', 'send', 'show', 'take']
($t^-_{(i-1)1}$, t^+_{i4})	['version']
($t^-_{(i-1)6}$, t^+_{i9})	['bug', 'fix', 'get', 'lot', 'please']

H4. The topic similarities between T^-_{i-1} and T^-_i reflect the remaining issues regarding the app A. The common keywords of the selected similar topic pairs are shown in Table IX. For example, the common keywords of topics ($t^-_{(i-1)3}$, t^-_{i10}) reflects the users complaint regarding the general quality of the app remains after the update, specifically concerning connections, calling, audio and video quality, etc. On the other hand, topic pair ($t^-_{(i-1)10}$, t^-_{i3}) reflects the problem with sending and receiving messages with notifications still exist. Furthermore, crashing is also a persisting issue that many users complained about based on topic pair ($t^-_{(i-1)6}$, t^-_{i8}). Topic pair ($t^-_{(i-1)7}$, t^-_{i9}), despite having only one common keyword, reflects the users general negativity towards the update. Topic pair ($t^-_{(i-1)7}$, t^-_{i6}) and ($t^-_{(i-1)8}$, t^-_{i6}) reflect the issues regarding logging in and signing up with Microsoft account. Topic pair ($t^-_{(i-1)8}$, t^-_{i9}) reflects the issues regarding user contact list when specially t^-_{i9} reflects the users' complaints regarding contact list syncing and status.

TABLE IX. COMMON KEYOWRDS IN NEGATIVE-NEGATIVE REVIEWS.

Topic Pairs	Common Keywords
($t^-_{(i-1)3}$, t^-_{i10})	['call', 'connect', 'drop', 'hear', 'make', 'person', 'phone', 'quality', 'sound', 'time', 'video', 'voice']
($t^-_{(i-1)10}$, t^-_{i3})	['get', 'message', 'notification', 'open', 'see', 'send', 'show', 'take', 'time']
($t^-_{(i-1)6}$, t^-_{i8})	['crash', 'fix', 'give', 'keep', 'please', 'problem', 'star', 'think', 'time']
($t^-_{(i-1)7}$, t^-_{i9})	['update']
($t^-_{(i-1)7}$, t^-_{i6})	['let', 'login', 'microsoft', 'time', 'try', 'turn', 'update', 'win']
($t^-_{(i-1)8}$, t^-_{i6})	['account', 'keep', 'make', 'sign']
($t^-_{(i-1)8}$, t^-_{i9})	['add', 'contact', 'list', 'sync']

Conclusively, we could summarize the users' opinion before and after the particular update (version 1.6.2017) as follows:

The merits in general:

1) ($t^+_{(i-1)8}$, t^+_{i5}): calling feature works.
2) ($t^+_{(i-1)6}$, t^+_{i6}), ($t^+_{(i-1)1}$, t^+_{i7}): people can chat in group with video and calls, connecting with family and friends.
3) ($t^+_{(i-1)7}$, t^+_{i9}): added features and bugs fixed.
4) ($t^+_{(i-1)6}$, t^+_{i5}): the video and sound quality.

The uncomfortable changes:

1) ($t^+_{(i-1)6}$, t^-_{i10}): user interface, connection, and calling quality in general.
2) ($t^+_{(i-1)5}$, t^-_{i6}): the user accounts, including logging in, signing up, passwords.
3) ($t^+_{(i-1)7}$, t^-_{i8}): bugs fixes.

The improvement:

1) ($t^-_{(i-1)7}$, t^+_{i3}): update in general.
2) ($t^-_{(i-1)10}$, t^+_{i8}): message notification.
3) ($t^-_{(i-1)3}$, t^+_{i6}), (($t^-_{(i-1)3}$, t^+_{i5}): calling in general, video and sound quality.

The remaining issues:

1) ($t^-_{(i-1)3}$, t^-_{i10}): update in general.
2) ($t^-_{(i-1)10}$, t^-_{i3}): sending and receiving messages with notifications.
3) ($t^-_{(i-1)6}$, t^-_{i8}): crashing.
4) ($t^-_{(i-1)7}$, t^-_{i9}): the new version
5) ($t^-_{(i-1)7}$, t^-_{i6}), ($t^-_{(i-1)8}$, t^-_{i6}): login and signup with Microsoft accounts.
6) ($t^-_{(i-1)8}$, t^-_{i9}): contact list syncing and status update.

Interestingly, these points are verified by the short notes of Skype developers regarding their updates [21]. Specifically, the above topics can be associated with the following original developers claims: (a) General performance and reliability improvements (Version 2017.08.15, Version 2017.08.29: phone calls, video calls and messaging quality), (b) Improved sign in - sign back into your account more easily (Version 2017.08.15: "log in" features, user account related functions), (c) New controls added to help users manage vibration and LED notification alerts. (Version 2017.07.05: notification), (d) Improvements to PSTN call stability (Version 2017.07.05: connection), (e) Messaging improvements Add content to chats via the + button and enjoy more room for your messages. (Version 2017.08.02: messaging), (f) The ability to add or remove contacts from your profile (Version 2017.08.01), (g) Activity indicators - see who's currently active in your Chats list (Version 2017.08.02: contacts and statuses). Hence, due to the correlation between the previously mentioned issues detected using our method and the content of the following up updates, we can verify the existence of those issues. However, whether the reason of the according update is the user reviews is unknown. On the other hand, we can also detect the disagreement amongst users' opinions. For example, a number of users think the calling quality deteriorated after the update while many others think it was improved (($t^+_{(i-1)6}$, t^-_{i10}) and ($t^-_{(i-1)3}$, t^+_{i6})). A number of users also think the update improves the app when other users think it is just as bad as the previous (($t^-_{(i-1)7}$, t^+_{i3}) and ($t^-_{(i-1)3}$, t^-_{i10})). We can obtain more details regarding users' different opinions by further investigating the keywords-related review texts.

IV. 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 [22]. For the contemporary mobile apps, the app distribution platforms, e.g., Apple App-Store, and Google Play, enable such spontaneous reviews of the users and facilitate the developers in terms of maintaining and evolving mobile apps [23]. Such feedback and reviews contain helpful information for developers in terms of identifying missing features and improving software quality [24]. From those review information, user requirements can be elicited continuously and, in a crowd-based fashion [25] [26].

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 [2]–[4].

For such purpose, NLP, SA, and supervised learning are the common techniques used to classify user reviews [3] [4]. According to recent studies, the combination of NLP and SA techniques has high accuracy in detecting useful sentences [4]. These techniques are also used in many other studies in terms of review opinion mining [7] [9] [24].

Many studies have contributed to the user opinion mining of mobile apps. Fu et al. [5] 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 [6], ratings and API use [27], review classification and useful sentences detection [4]. 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 [8] proposes an approach focusing on feature extraction and sentiment analysis, which facilitates the evaluation of individual app features. Similarly, Iacob and Harrison [9] focuses also on the feature requests extraction but via means of linguistic rules and LDA topic modeling. Many studies also provide methods of using automatic classification method to study mobile app user reviews [28] [29] [30]

Compared to the previous mentioned approaches in user review opinion mining, our method aims towards the similar topic detection and analysis concerning not only the app in general but also the particular major 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 information regarding each individual update and will provide insights on the future updates.

V. Conclusion

In this study, we propose a method for analyzing the correlation of mobile apps user reviews before and after a particular app updates in order to detect how users' opinions change with the update released. After classifying the reviews before and after a particular update by positive and negative sentiment, we extract the topics of each segment. By comparing the similarities of these extracted topics, we identify both the positive and negative issues reflected by these reviews regarding the particular update and the app in general. Overall, this study is an exploratory investigation on using user review opinion mining techniques in detecting update-specific issues. The future studies shall extend the use of this method to the whole maintenance lifecycle of mobile apps to investigate the broader correlation between users feedback and the apps' update trends. Releasing strategies improvement based on this method will also be studied. Other factors, e.g., the different app categories, different platforms, and different mining and analysis techniques will also be taken into account.

References

[1] X. Li, Z. Zhang, and J. Nummenmaa, "Models for mobile application maintenance based on update history," in Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on. IEEE, 2014, pp. 1–6.

[2] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" IEEE Software, vol. 32, no. 3, 2015, pp. 70–77.

[3] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in 2015 IEEE 23rd international requirements engineering conference (RE). IEEE, 2015, pp. 116–125.

[4] 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 Software maintenance and evolution (ICSME), 2015 IEEE international conference on. IEEE, 2015, pp. 281–290.

[5] 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 Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 1276–1284.

[6] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in Proceedings of the 9th IEEE Working Conference on Mining Software Repositories. IEEE Press, 2012, pp. 108–111.

[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. ACM, 2014, pp. 767–778.

[8] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in Requirements Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, 2014, pp. 153–162.

[9] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013, pp. 41–44.

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

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

[12] T. T. Tanimoto, "Ibm internal report," Nov, vol. 17, 1957, p. 1957.

[13] B. V. Kumar and L. Hassebrook, "Performance measures for correlation filters," Applied optics, vol. 29, no. 20, 1990, pp. 2997–3006.

[14] Langdetect, https://pypi.python.org/pypi/langdetect, last accessed on 06/20/18.

[15] Pyenchant, https://pypi.python.org/pypi/pyenchant, last accessed on 06/20/18.

[16] NLTK, http://www.nltk.org, last accessed on 06/20/18.

[17] E. Cambria, R. Speer, C. Havasi, and A. Hussain, "Senticnet: A publicly available semantic resource for opinion mining." in AAAI fall symposium: commonsense knowledge, vol. 10, no. 0, 2010.

[18] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining." in Lrec, vol. 10, no. 2010, 2010, pp. 2200–2204.

[19] M. M. Bradley and P. J. Lang, "Affective norms for english words (anew): Instruction manual and affective ratings," Citeseer, Tech. Rep., 1999.

[20] M. Stevenson and Y. Wilks, "Word sense disambiguation," The Oxford Handbook of Comp. Linguistics, 2003, pp. 249–265.

[21] "Skype on Google Play," URL: https://play.google.com/store/apps/details?id=com.skype.raider, last accessed on 06/20/18.

[22] K. Schneider, "Focusing spontaneous feedback to support system evolution," in RE, pp. 165–174.

[23] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in Requirements Engineering Conference (RE), 2013 21st IEEE International. IEEE, 2013, pp. 125–134.

[24] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 582–591.

[25] N. Seyff, F. Graf, and N. Maiden, "Using mobile re tools to give end-users their own voice," in Requirements Engineering Conference (RE), 2010 18th IEEE International. IEEE, 2010, pp. 37–46.

[26] 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. Springer, 2015, pp. 247–253.

[27] 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., vol. 41, no. 4, 2015, pp. 384–407.

[28] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," Empirical Software Engineering, vol. 21, no. 3, 2016, pp. 1067–1106.

[29] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," Requirements Engineering, vol. 21, no. 3, 2016, pp. 311–331.

[30] Z. Sun, Z. Ji, P. Zhang, C. Chen, X. Qian, X. Du, and Q. Wan, "Automatic labeling of mobile apps by the type of psychological needs they satisfy," Telematics and Informatics, vol. 34, no. 5, 2017, pp. 767–778.