Too Many User-Reviews! What Should App Developers Look at First?

Ehsan Noei, Feng Zhang, and Ying Zou

Abstract—Due to the rapid growth in the number of mobile applications (apps) in the past few years, succeeding in mobile app markets has become ruthless. Online app markets, such as Google Play Store, let users rate apps on a five-star scale and leave feedback. Given the importance of high star-ratings to the success of an app, it is crucial to help developers find the key topics of user-reviews that are significantly related to star-ratings of a given category. Having considered the key topics of user-reviews, app developers can narrow down their effort to the user-reviews that matter to be addressed for receiving higher star-ratings. We study 4,193,549 user-reviews of 623 Android apps that were collected from Google Play Store in ten different categories. The results show that few key topics commonly exist across categories, and each category has a specific set of key topics. We also evaluated the identified key topics with respect to the changes that are made to each version of the apps for 19 months. We observed, for 77% of the apps, considering the key topics in the next versions shares a significant relationship with increases in star-ratings.

Index Terms—Mobile application, Empirical study, Software release, User-review

1 Introduction

MOBILE app markets, such as Google Play Store [1], are immensely competitive for app developers [2]. Google Play Store uses the star-rating mechanism to gather users' feedback and ratings. By the star-rating mechanism, users can rate each app from one star (the lowest) to five stars (the highest). Star-ratings can influence the income of app development companies [3], [4] as users rely on star-ratings for choosing new apps to download [3]. With over three million Android apps [5], [6], low rated apps usually cannot survive in the competitive market of mobile apps.

A star-rating can be associated with a user-review. A user-review is an informal piece of text without a predefined structure [7] and could contain useful information, such as bug reports, feature requests, and user experiences [7], [8]. Recent studies (e.g., [7], [8], [9], [10], [11]) have shown the unavoidable importance of star-ratings and user-reviews in success and profitability of apps.

Many studies have been conducted to summarize user-reviews and classify them into meaningful groups [12]. For example, Villarroel *et al.* [13] help app developers to improve the release planning by classifying user-reviews into clusters of bug reports and feature requests. Villarroel *et al.* [13] rank clusters of user-reviews based on different metrics, such as the number of user-reviews. However, we observe that the number of user-reviews does not always share a significant relationship with star-ratings. Iacob *et al.* [14] summarize the topics of user-reviews which cover users' complaints. Recent work is subject to the context of a single or a few app(s). However, none of the earlier work has specified a set of topics with a broader view, i.e., category, that are statistically significantly related to star-ratings; while, every category shares a specific set of characteristics and requirements [15].

We introduce the *key topics* which are the topics of user-reviews that share a significant relationship with star-ratings. For instance, we identify *messages* as a key topic of the category of *social*. Example user-reviews are *"Love this app. Great to be able to check up on the site for my messages when I'm on the go!"*, *"Wouldn't let me read my messages so I uninstalled it [...]"*, and *"It wont let me recieve messages or send them fix it and i'll rate 5 star again"*. The above user-reviews show users' concerns regarding *messages* in the category of *social*. Addressing all the issues that are reported in user-reviews is a time-consuming task. Therefore, it is beneficial to understand and consider the key topics for better managing time and efforts in order to receive higher star-ratings.

For each category, we identify a set of key topics by analyzing the contribution of each topic to the performance of the model that is built with the star-ratings as the dependent metric. For example, we find that *speed*, *battery consumption*, and *searching* are the key topics of the category of *travel and local*. Moreover, we observe that the most frequent topics of each category are not necessarily the key topics of the same category, i.e., all the most frequent topics do not share a significant relationship with star-ratings. Furthermore, our proposed approach needs minor human intervention to identify the key topics.

We investigate a large number of user-reviews, i.e., $4,193,549$ user-reviews of 623 apps in ten different categories. With such a number of user-reviews, we conduct a fine-grained analysis per topic on user-reviews and the associated star-ratings for 19 months.

We evaluate the identified key topics based on the reported changes of apps for 19 months. Release notes could be a great indicator of the changes that are made to each app. Johann *et al.* [16] reported a precision of up to 88% for app descriptions in identifying app features. For each app, we calculate the similarity score of the release notes with key

• *E. Noei and Y. Zou are with the Department of Electrical and Computer Engineering, Queen's University, Ontario, Canada.*
E-mail: {e.noei, ying.zou}@queensu.ca
• *F. Zhang is with the School of Computing, Queen's University, Ontario, Canada.*
E-mail: feng@cs.queensu.ca

Manuscript received DATE; revised DATE.

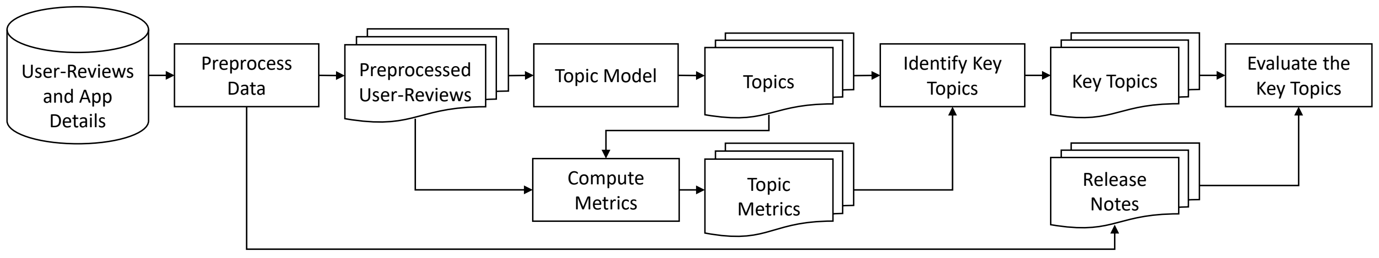


Fig. 1: Overview of the study setup.

TABLE 1: List of the categories, followed by the number of apps, initial number of user-reviews, and number of consistent and informative user-reviews.

Category	#Apps	#User-reviews	#Consistent User-reviews	#Informative User-reviews
Business	42	277,564	187,343	102,375
Communication	70	3,055,079	1,972,061	950,007
Health and Fitness	57	342,003	249,534	171,220
Media and Video	44	1,020,481	651,444	342,610
Photography	56	827,870	631,629	274,611
Productivity	67	1,271,049	925,077	438,505
Shopping	63	660,784	484,433	293,943
Social	99	2,739,505	1,712,657	953,311
Tools	76	1,476,290	1,019,811	477,804
Travel and Local	49	435,628	293,355	189,163

topics versus non-key topics. Higher star-ratings are received when release notes are more similar to the key topics for 77% of the subject apps.

Paper Organization. Section 2 describes the study setup. Section 3 discusses the key topics identification method and the identified key topics. In Section 4, we evaluate the relationship between the key topics and changes in star-ratings. Section 5 lists the potential threats to the validity of this work. Section 6 discusses the related work. Finally, Section 7 concludes the paper and provides the future direction.

2 Study Setup

Figure 1 shows an overview of the study setup. As shown in Figure 1, we preprocess the user-reviews and release notes. Then, we identify the topics of each user-review. After that, we compute the metrics of the topics and identify the key topics of each category. Finally, we evaluate the changes in star-ratings with respect to release notes.

2.1 Data Source

We retrieve the user-reviews, app details, and release notes from Google Play Store. Despite other existing platforms, such as iOS and Windows platforms, Android has by far the largest number of users [2]. Due to the processing time for handling all the user-reviews in over thirty categories of Google Play Store, we study ten categories. We randomly selected ten categories to avoid introducing a bias towards a specific set of categories, e.g., popular categories. The selected categories are listed in Table 1. The distribution of the number of user-reviews varies across the categories. Some categories, such as *communication* and *social*, have a variety of popular apps and users.

We implement a crawler to extract app details and the associated user-reviews on a daily basis. The crawler retrieves all the new user-reviews and merges them into our database. We ran the crawler from $April$ 2014 to $November$ 2015. Initially, we retrieved $14,241,915$ user-reviews for $2,723$ apps. To ensure that we have enough data to study the evolution of star-ratings over time, we exclude $2,100$ apps based on two criteria: (i) the apps that get updated less than ten times during our study as we study the relation of addressing the issues raised by the key topics with star-ratings over time, and (ii) the apps that receive less than ten user-reviews between each update to avoid the apps with few user-reviews skew the findings [17], [18]. The numbers of selected apps in each category are listed in Table 1. By removing $2,100$ apps, the number of user-reviews decreased from $14,241,915$ to $12,106,253$ (i.e., 15% of the initial user-reviews excluded from our study). The numbers of releases for all the apps are illustrated in Figure 2a.

2.2 Preprocessing Data

In this section, we explain the preprocessing steps, such as applying natural language processing techniques.

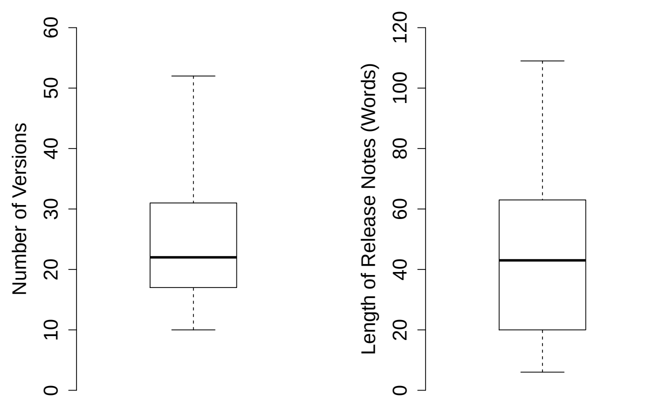
2.2.1 Removing Non-English User-reviews

We filter out non-English user-reviews using Language Detector [19]. We remove $468,473$ user-reviews (i.e., 4%) that were written in a non-English language.

2.2.2 Filtering Out Inconsistent User-reviews

Google Play Store faces inconsistencies among the user-reviews [7], [8]. For instance, two users who perceive the same quality from the same app may give different star-ratings based on their subjective experience. Some user-reviews have negative content, but they are associated with high star-rating, and vice versa. As an example, we found *"Absolutely terrible. Waste of time"* associated with 5 stars. The inconsistent user-reviews can affect the accuracy of the findings. To identify the inconsistent user-reviews, we compare star-ratings with sentiment scores [20] of user-reviews.

Different sentiment analysis approaches may produce different sentiment scores [21]. We evaluated two popular tools [21], including SentiStrength [22] and Natural Language Toolkit (NLTK) [23]. We manually investigated the output of both SentiStrength and NLTK on a sample of user-reviews with the size of 384, the confidence level of 95%, and the confidence interval of 5. SentiStrength achieved 74% correct sentiment scores, while NTLK achieved 62% correct



(a) Number of Versions (b) Length of Release Notes

Fig. 2: The number of versions and length of release notes (i.e., the number of words) for all the apps during the 19-month period of the study.

sentiment scores. Moreover, Thelwall *et al.* [22] show that SentiStrength has an acceptable accuracy for texts and comments in social media, such as 70.7% correct sentiment scores for the YouTube comments. Having such a character makes SentiStrength a good candidate to measure the sentiment scores of user-reviews [24].

SentiStrength scores user-reviews between -5 and $+5$. The most negative user-reviews receive -5, and the most positive user-reviews receive $+5$. User-reviews with the score of -1, 0, and $+1$ are considered as neutral user-reviews [22]. Any score above $+1$ is a positive sentiment score, and any score below -1 is a negative sentiment score [22]. User-reviews are associated with star-ratings between 1 and 5. The majority of users do not download an app with an average star-rating of less than three [6]. We count the star-ratings of 3 as a neutral rating, any star-rating below 3 as a negative rating, and any star-rating above 3 as a positive one.

We only consider the user-reviews with consistent sentiment scores and star-ratings to reduce the risk of having inconsistent user-reviews skew the findings. In total, we identify $8,127,344$ consistent user-reviews out of $11,137,780$ user-reviews (i.e., 73% of all the retrieved user-reviews). The total number of user-reviews and consistent user-reviews are listed in Table 1.

2.2.3 Filtering Out Uninformative User-reviews

An uninformative user-review, such as *"This app is OK"*, provides minor information for app developers [13], [25]. To filter out uninformative user-reviews, we rely on the AR-MINER approach proposed by Chen *et al.* [25]. The AR-MINER applies Expectation Maximization for Naïve Bayes method [26] to build a classifier that distinguishes between informative and uninformative user-reviews. In total, we get $4,193,549$ informative user-reviews (i.e., 52% of the consistent user-reviews). The last column in Table 1 shows the number of informative user-reviews.

2.2.4 Correcting Typos

Typos in user-reviews would disturb the analysis techniques [27]. To reduce the risk of missing valuable information in the user-reviews, we correct the typos using Jazzy Spell Checker [28]. We apply Jazzy with a dictionary of $645,289$ English words.

We also replace the commonly used abbreviations and informal messaging vocabularies with meaningful words. We get the abbreviations and informal vocabularies from the available online sources [29], [30]. For example, we replace *"gr8"* with *"great"*.

2.2.5 Processing Natural Language

We apply natural language processing techniques [31] to better identify the topics of user-reviews. We also apply natural language processing on the release notes to better evaluate the identified key topics. The lengths of the release notes, in terms of the number of words, are illustrated in Figure 2b. Release notes have a median of 43 words.

Coreference Resolution. Coreference happens when two or more expressions refer to the same referent [32]. For instance, suppose an example user-review: *"I really like the pictures in this app. They give me a more comprehensive view"*. The second sentence uses a pronoun (i.e., they). However, the machine does not understand what does "they" refer to. We use the Stanford deterministic coreference resolution [33] to resolve the coreferences in the user-reviews. The above example will be converted to *"I really like the pictures in this app. The pictures give me a more comprehensive view"*.

Sentence Annotation. Some users write long user-reviews with several concerns, such as listing the pros and cons in details, in one single review. We use Stanford CoreNLP [31] to fragment the user-reviews with several concerns into pieces. Hence, each piece could share an independent concept. Stanford CoreNLP annotates the words in user-reviews. It produces the base forms of words and parts of speech. It also identifies the structure of sentences in terms of words and phrases dependencies. The two sentences in the example above refer to the same concept that is *"pictures"*. If the sentences did not share the same concern, we would have fragmented them into two pieces. For example, *"I can easily search the product names. However, the app is a battery killer."* is fragmented into two pieces of *"I can easily search the product names."* and *"However, the app is a battery killer"*. Therefore, we can have each fragment with an independent subject.

2.2.6 Building Corpus

A text corpus is a large set of texts for statistical analyses, such as topic modeling [34]. To build the corpus of fragments of user-reviews, we normalize the user-reviews before applying topic modeling, including removing the stop words [35], removing the punctuations, and stemming the words [36]. We also normalize the release notes to better compare them with the identified topics and evaluate the key topics.

2.3 Topic Modeling

Topic modeling techniques let us assign topics to each user-review, so we can understand the topics that each user-review is about. We use the Latent Dirichlet Allocation (LDA) technique [37] to capture the topics of user-reviews. LDA allows sets of observations to be described by unobserved groups and determines the similar parts of data. Moreover, Henriksson *et al.* [38] showed that LDA-based topic modeling generates the best synonyms of words which is useful when

TABLE 2: The identified topics of user-reviews.

#	Topic Name	Top Words	Brief Description
T_1	**Advertisements**	ad, screen, remove, annoy, button	Advertisement-related issues are along with this topic.
T_2	**Authentication Issues**	account, log, wrong, check, password	More related to authentication process of an app.
T_3	**Battery Consumption**	battery, save, life, power, charge	More about the issues of energy and battery consumption.
T_4	**Bug Reports**	star, problem, only, point, issue	Users explain their problem, specially the bugs.
T_5	**Comparing Versions**	update, version, latest, free, upgrade	Discussing different versions, such as free vs. full version.
T_6	**Connection**	call, voice, connect, contact, number	The words are about calls and internet connections. quality.
T_7	**Device Compatibility**	phone, android, mobile, device, tablet	Comparisons based on different devices are along with this topic.
T_8	**Feature Requests**	better, improve, perform, add, feature	The verbs, such as "add", are in this topic to request a feature.
T_9	**Language Support**	language, translate, English, speak	Users discuss language issues, such as adding a specific language.
T_{10}	**Messages**	post, message, view, text, receive	The words are more related to sending and receiving messages.
T_{11}	**Pictures**	photo, picture, edit, share, view	Related to viewing, sharing, and processing pictures.
T_{12}	**Playing Audio & Video**	video, play, watch, player, quality	The words are related to watching and playing audio & video.
T_{13}	**Praising Features**	well, perfect, proper, interesting, pretty	User praise various features.
T_{14}	**Purchases**	deal, money, price, store, shop	The words of this topic deal with purchasing and monetary issues.
T_{15}	**Repeating Issues**	time, every, day, try, long	Adverbs of time can be seen in this topic.
T_{16}	**Searching**	google, search, map, find, locate	Searching is the main concern in this topic.
T_{17}	**Social Networking**	people, meet, talk, chat, friend	About connecting users to other people, such as friends.
T_{18}	**Speed**	slow, load, quick, fast, speed	More related to speed of an app, a process, or a functionality.
T_{19}	**Storage**	file, data, manage, space, card	The words are more about storage management.
T_{20}	**Task Tracking & Notifications**	help, track, activity, list, sync	The words are related to tracking reminders, and notifications.
T_{21}	**Technical Support**	guy, support, hope, team, job	User discuss and share issues regarding technical supports.
T_{22}	**User Interface**	screen, button, bar, interface, theme	Issues and experiences about user interface.
T_{23}	**Web Browsing**	browser, open, speed, load, slow	The words in this topic are more related to web browsing issues.

comparing key topics with release notes (See Section 4). Latent parameters, such as the number of topics, need to be set up first. The most important parameters are:

- α which is set up according to the distribution of topics.
- β is the distribution of word for each topic.
- θ_d is the distribution of topics for a document d.
- ϕ_t is the distribution of words that describe topics t.
- t_{wd} is the topic for w_{th} word in document d.
- $|N|$ is the number of topics.
- $|W|$ is the number of words in the vocabulary.
- $|w|$ is the number of words in each document.
- $|W_d|$ is the total number of words in all documents.

More details about the LDA parameters can be found in the work of Blei *et al* [37]. Panichella *et al.* [39] proposed an approach to determine the optimal configuration of LDA for software engineering tasks. Most of the parameters are calculated automatically with respect to the distribution of data (i.e., words, documents, and topics) and input parameters using JGIBBLDA [40] (i.e., the tool that we use to identify the topics). We employ two approaches (i.e., Griffiths *et al.* [41] and Cao *et al.* [42]) to identify the optimum number of topics. The method of Griffiths *et al.* [41] compares the likelihood of the topic models for each number of topics. The best number of topics has the highest likelihood value. The method of Cao *et al.* [42] investigates the optimum number of topics in LDA based on the topic density. The approach of Griffiths *et al.* suggests 15 as the optimum number of topics. However, the approach of Cao *et al.* suggests 35 as the optimum number of topics. Hence, the best guess would be between 15 and 35. To avoid losing any potential topic, we pick a safe approach by choosing the maximum of the suggested numbers of topics (i.e., 35 topics).

We apply LDA with $2,000$ Gibbs sampling iterations [43]. Although more iteration is better, with $2,000$ iterations, we can achieve a high accuracy with topic modeling [41], [43]. The Gibbs sampling is a Markov chain Monte Carlo algorithm [44] to acquire a sequence of observations. The observations are approximated by a specified multivariate probability distribution [45].

We manually analyze the identified topics with a help of an external evaluator who is a graduate student in software engineering. The external evaluator has over six years experience in software and app development. We discussed each topic with the external evaluator until we reach an agreement. We merge the topics that have the same or very close semantic meaning based on the words given by the LDA and the associated user-reviews. We merge the topics having considered the following criteria:

- Each topic is defined by a set of words. We compare the words of each topic (especially the words appearing more frequently for each topic). If the words have the same semantic meaning, we merge them into one group.
- We also look into the user-reviews that are associated with each topic, especially in the cases that the words of each topic were not clear enough to understand the meaning of the topics. If the user-reviews of each topic are explaining the same concept, we merge them together.

We list the final 23 topics in Table 2 followed by the five most frequently appearing words from the output of LDA and a brief description of each topic. Table 3 shows the frequency of topics in each category.

2.4 Computing Metrics

We follow the *Goal / Question / Metric* (GQM) paradigm [46], [47] to capture the metrics. We bring up nine questions to quantify the user-reviews and corresponding topics. Then, we identify the metrics that can be measured to address the questions. We measure 27 metrics of user-reviews. Table 4 shows the GQM model and list of metrics.

2.4.1 Number of Releases

Khalid *et al.* [48] reported that many users complain about app updates. Hence, the number of releases might affect

TABLE 3: Distribution of the topics in different categories. Each number demonstrates the percentage of the corresponding topic in each category.

#	Topic Name	Business	Commu-nication	Health and Fitness	Media Video	Photo-hraphy	Produ-ctivity	Shopping	Social	Tools	Travel and Local
T_1	**Advertisements**	4%	4%	2%	4%	3%	3%	2%	4%	4%	2%
T_2	**Authentication Issues**	2%	4%	1%	1%	1%	2%	2%	3%	1%	2%
T_3	**Battery Consumption**	2%	1%	1%	1%	1%	18%	3%	1%	9%	2%
T_4	**Bug Reports**	2%	2%	1%	2%	3%	2%	3%	4%	3%	4%
T_5	**Comparing Versions**	6%	4%	2%	4%	2%	3%	3%	5%	6%	4%
T_6	**Connection**	5%	14%	2%	6%	5%	4%	3%	7%	7%	3%
T_7	**Device Compatibility**	8%	3%	2%	4%	3%	8%	2%	3%	7%	2%
T_8	**Feature Requests**	10%	4%	2%	3%	5%	5%	2%	2%	4%	3%
T_9	**Language Support**	1%	2%	1%	2%	3%	2%	1%	2%	3%	1%
T_{10}	**Messages**	4%	12%	1%	2%	2%	2%	2%	10%	2%	2%
T_{11}	**Pictures**	2%	1%	1%	1%	26%	1%	1%	4%	1%	1%
T_{12}	**Playing Audio & Video**	1%	2%	1%	31%	4%	1%	1%	4%	2%	1%
T_{13}	**Praising Features**	7%	7%	5%	6%	12%	8%	9%	11%	9%	5%
T_{14}	**Purchases**	1%	1%	1%	1%	1%	1%	40%	1%	1%	6%
T_{15}	**Repeating Issues**	1%	1%	3%	1%	2%	2%	3%	2%	2%	1%
T_{16}	**Searching**	1%	1%	2%	1%	1%	1%	3%	1%	2%	33%
T_{17}	**Social Networking**	2%	8%	1%	2%	3%	2%	3%	14%	2%	2%
T_{18}	**Speed**	6%	5%	5%	6%	5%	5%	8%	8%	6%	9%
T_{19}	**Storage**	8%	1%	1%	2%	2%	7%	1%	1%	5%	1%
T_{20}	**Task Tracking and Notifications**	12%	4%	60%	3%	6%	12%	6%	4%	7%	9%
T_{21}	**Technical Support**	5%	3%	2%	3%	3%	4%	2%	2%	5%	3%
T_{22}	**User Interface**	6%	6%	4%	5%	5%	5%	2%	6%	7%	4%
T_{23}	**Web Browsing**	3%	11%	1%	13%	3%	3%	2%	2%	5%	2%

TABLE 4: The GQM model to quantify the metrics of user-reviews.

Goal: Quantifying the user-reviews and the associated topics.		
Questions	**Metric**	**#**
How often each app gets released?	Number of releases	1
What is the estimated amount of information in each user-review based on the identified topics?	Entropy of topics	1
How well is the users' experience with each topic?	Sentiment scores	6
How many users discuss the same topic?	Number of user-reviews	2
What is the proportion of positive and negative user-reviews?	Proportion of user-reviews	2
How much information can be extracted from the user-reviews?	Number of words and sentences	12
What is the proportion of each topic in the user-reviews?	Proportion of topics	1
How often does each topic appear in the user-reviews?	Topic recurrence length	1
For how long does each topic appear in the user-reviews?	Duration of topics	1
	Total:	27

the user-reviews. For each app, we count the number of releases during the period of our study (i.e., $April$ 2014 to $November$ 2015).

2.4.2 Entropy of Topics

According to the Shannon entropy [49], entropy measures the amount of information contained in user-reviews. We measure the entropy of topics using Equation (1).

$$H(a) = -\sum_{i=1}^{n} Pr_a(T_i) \cdot log(Pr_a(T_i)) \tag{1}$$

In Equation (1), $H(a)$ represents the entropy of an app a. n shows the number of topics. T_i demonstrates the i^{th} topic. $Pr_a(T_i)$ shows the probability of the occurring topic T_i in user-reviews of the app a. We compute $Pr_a(T_i)$ using Equation (2).

$$Pr_a(T_i) = \frac{\eta_a(T_i)}{\eta_a} \tag{2}$$

In Equation (2), η_a shows the number of user-reviews for an app a, and $\eta_a(T_i)$ represents the number of user-reviews with the i^{th} topic for the same app.

2.4.3 Sentiment Score of Topics

We use SentiStrength [20], [22] to measure the sentiment scores of the topics for each fragment of user-reviews. SentiStrength scores user-reviews by a quantitative value between -5 and $+5$ from the most negative to the most positive. For each app, we measure the statistical characteristics of the sentiments scores, including the mean, median, minimum, maximum, 1^{st} quartile, and 3^{rd} quartile of the sentiment scores. Moreover, for each topic, we calculate the number of negative and positive user-reviews. We also count the proportion of negative and positive user-reviews.

2.4.4 Number of Words and Sentences

Sizes of user-reviews can implicitly show the helpfulness of user-reviews [50]. As the number of words and sentences increases, users might provide more information about a specific topic. To calculate the sizes of user-reviews, we count the number of words and sentences of each fragment of user-reviews. We use Stanford Parser [51] and Stanford Tokenizer [31] to count the number of words and sentences. For each topic, we measure the mean, median, minimum, maximum, 1^{st} quartile, and 3^{rd} quartile of the number of words and sentences.

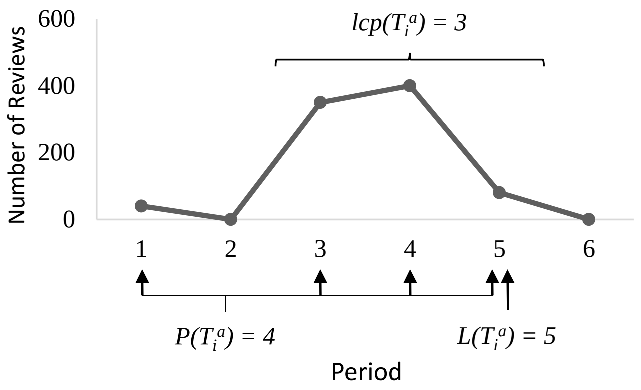


Fig. 3: An example for computing the *topic recurrence length*.

2.4.5 Proportion of Topics

Some topics frequently appear in a specific category, while some topics appear infrequently. For example, in the category of *shopping*, users talk more about *buying and purchasing* than *watching videos*. To capture the varying popularity of a topic, we measure the proportion of each topic for each app.

2.4.6 Topic Recurrence Length

A topic can be hot in the past but off subject now. Inspired by prior studies [52], [53], we introduce the *topic recurrence length* metric to capture the consecutive occurrences of each topic. We use the following equation to measure the topic recurrence length of each topic for each app:

$$TRL(T_i^a) = lcp(T_i^a) \cdot e^{\frac{1}{n}((P(T_i^a)+L(T_i^a))} \tag{3}$$

For each app, we break the 19 months duration our study into the periods between each release. In Equation (3), T_i^a shows the i^{th} topic of the user-reviews for an app a. n is the total number of periods between the releases of an app a during the period of our study. lcp is the longest consecutive period of T_i^a with at least one user-review between each release. $P(T_i^a)$ is the number of periods with at least one user-review for a topic T_i^a. $L(T_i^a)$ is the last index of the longest period.

We give an exponential effect to $P(T_i^a)$ as the emphasize of TRL is on the recurrence of topics. As the recurrence of a topic increase, TRL grows exponentially. Similarly, we gave $L(T_i^a)$ the same effect. Figure 3 shows an example for computing the topic recurrence length. Suppose that we want to calculate the topic recurrence length of the i^{th} topic for an app a. Presume this app has updated and released 5 times during our study. Then, the number of periods is 6. Let's assume the distribution of the number of user-reviews for each period is $\{40, 0, 350, 400, 80, 0\}$. For this distribution, the longest consecutive period $lcp(T_i^a) = 3$, the number of periods with at least one user-review $P(T_i^a) = 4$, and the last index of the longest period is 5. Consequently, $TRL(T_i^a)$ (i.e., the topic recurrence length) equals to 13.45 using equation (3). A greater value of TRL shows that the topic had appeared more often. For example, a topic with a $TRL = 13.45$ (as in the example above) recurs more often than a topic having a $TRL = 10$.

2.4.7 Duration of Topics

During the lifetime of an app, some topics might appear for a specific time but disappear from the user-reviews later. For example, if an app suffers from battery drains, the users might comment intensively on battery consumption problems. After the problem being resolved, users may not complain about battery issues anymore. Therefore, this topic would disappear from the user-reviews. To measure the duration of each topic, we calculate the difference between the earliest and latest time of a continuous period with at least one user-review regarding the same topic.

3 Key Topics Identification

In this section, we describe the key topics identification method and the identified key topics.

3.1 Method

Google Play Store reports the average of star-ratings as an indicator of the overall star-rating of an app [1]. Similarly, for each app, we investigate the relationship between the average of star-ratings and the identified topics, such as *user interface* and *battery consumption*. The independent metrics are listed in Table 2.

3.1.1 Correlation Analysis

Having correlated metrics affects the stability of our models negatively and makes it difficult to distinguish the impact of the metrics [54]. We use the variable clustering analysis technique according to Spearman's $|\rho| > 0.7$ [55] to investigate the correlation between the explanatory metrics [56]. The Spearman correlation evaluates the monotonic relationship between continuous or ordinal metrics.

3.1.2 Computing Contribution of the Metrics

Proportional Marginal Variance Decomposition (PMVD) is an approach that is based on sequential R^2s. It averages over orderings by using weighted averages with data-dependent weights. Therefore, PMVD mitigates the risk of dependence on the orderings of the metrics of topics. We apply PMVD [57], [58] to detect the topics that give the highest contributions to star-ratings.

For each topic, we measure a set of metrics as explained in Section 2.4). Using PMVD, we group the metrics of each topic together. Therefore, PMVD would be able to calculate the contribution of each topic (i.e., each group of metrics) on star-ratings. For simplicity, we explain the calculation of PMVD for $m_i^{t_j}$ (i.e., i^{th} metric of j^{th} topic) in Equation (4). In this Equation, s is the set of understudy metrics and n denotes the number of metrics in set s. r is a permutation of metrics of set s. $\omega(m_i^{t_j}|s)$ is the additional R^2 when adding the $m_i^{t_j}$ to the model as shown in Equation (5). $\sigma(s)$ is the set of all $n!$ permutations of metrics of set s. $\rho(s)$ denotes the data-dependent PMVD weights that is calculated using Equation(6) [57].

$$PMVD(m_i^{t_j}) = \frac{1}{n!}\sum_{\sigma(s)} \rho(r)\omega(m_i^{t_j}|s) \tag{4}$$

$$\omega(m_i^{t_j}s) = R^2(m_i^{t_j}s) - R^2(s) \tag{5}$$

TABLE 5: List of the key topics of each category. The key topics are sorted according to the PMVD scores (in parentheses).

Category	Key Topics	# Key Topics
Business	Battery Consumption (0.33), Speed (0.31), Device Compatibility (0.25)	3
Communication	Searching (0.58), Battery (0.11), Speed (0.08)	3
Health and Fitness	Task Tracking and Notifications (0.43), Messages (0.21), Speed (0.08), Battery Consumption (0.05)	4
Media and Video	Playing Audio & Video (0.71), Speed (0.19), Repeating Issues (0.06)	3
Photography	Pictures (0.43), Advertisement (0.24), Searching (0.06), User Interface (0.06), Speed (0.05), Battery Consumption (0.05)	6
Productivity	Pictures (0.20), Advertisements (0.12), Authentication Issues (0.12), Repeating Issues (0.07), Task Tracking and Notifications (0.07)	5
Shopping	Searching (0.59), Bug Reports (0.10), Purchases (0.07)	3
Social	Messages (0.15), User Interface (0.08), Comparing Versions (0.08), Social Networking (0.06), Pictures (0.06), Searching (0.05)	6
Tools	Battery Consumption (0.24), Speed (0.23), Bug Reports (0.17), Language Support (0.12)	4
Travel and Local	Searching (0.21), Battery Consumption (0.19), Advertisements (0.13), Speed (0.08)	4

Equation (6) shows an overview of computing the PMVD weights. The weights have been derived from a set of axioms (see Feldman [58]), such as the axiom of proper exclusion. The axiom of proper exclusion makes an independent metric with a coefficient 0 to have an R^2 share of 0 [57]. In Equation (6), $\sigma(s)$ is the set of all $n!$ permutations of metrics of set s. r is a permutation of metrics in set s. $\rho(r)$ shows the weight for a list of metrics r. $\Gamma(s)$ can be computed using Equation (7) for a permutation r of the metrics of set s. In Equation (7), m_{ri} is the i^{th} metric of a permutation r of set s.

$$\rho(r) = \frac{\Gamma(r)}{\sum_{\sigma(s)} \Gamma(s)} \tag{6}$$

$$\Gamma(r) = \prod_{i=1}^{n-1} \omega(m_{ri+1}, ..., m_{rn} | m_{r1}, ..., m_{ri})^{-1} \tag{7}$$

Using PMVD allows us to identify the likelihood of independent metrics that are correctly ordered with respect to their relative importance. We compute the expected contribution of the metrics of each topic to the model performance. We use the R package RELAIMPO [57] that provides methods and metrics, such as the PVMD, for examining relative importance in linear models. We group the metrics of each topic together. PMVD gives the expected contribution of each topic based on its group of metrics. We mark a topic as a key topic when the expected contribution of the metrics that describe the topic (i.e., the PMVD score) is more than 0.05 [57].

3.2 Findings

The identified key topics are listed in Table 5. As shown in Table 5, each category has its own set of key topics that shares a significant relationship with star-ratings. For instance, the key topics of the category of *business* are *battery consumption*, *speed*, and *device compatibility*. Some categories, such as *business*, have fewer key topics in comparison with other categories. The categories of *business*, *communication*, and *shopping* have only three key topics each, while the categories of *photography* and *social* have the most number of key topics with six key topics.

App developers should take care of the issues that are related to the key topics of the category in which they publish their apps to achieve higher star-ratings. App developers can also refer to related work, such as Chen *et al.* [25], Villarroel *et al.* [13], and Di Sorbo *et al.* [59], to identify the bug reports and feature request that are reported in the user-reviews of their own apps. Following our findings, they can focus on the user-reviews that are related to the key topics to better manage time, budget, and efforts.

We made some additional interesting observations, as listed below, that can further help app developers:

1) For each category, some key topics have a higher relationship with star-ratings in comparison with the other key topics. PMVD scores are listed in parentheses in Table 5. PMVD scores are between 0 and 1 [57]. A higher value of a PMVD score shows a higher contribution to the performance of the model. For instance, in the category of *business*, *battery consumption* and *speed* are relatively more important than the issues related to *device compatibility*. Hence, developers would be able to prioritize the key topics.
2) By comparing the most frequent topics of each category and the key topics of the same category, we find that the key topics are not *necessarily* the most *frequent* topics of the user-reviews (see Table 3). For example, in the category of *business*, *task tracking and notifications* is more frequent than other topics while it is not a key topic. This is an interesting observation as developers are likely to be distracted by the frequent topics that do not actually share a significant relationship with star-ratings. The list of our extracted key topics helps developers prioritize the development efforts to address the issues that are related to the key topics.
3) Although we study the relationship between a group of metrics and star-ratings, highlighting the most significant metrics can provide developers with more insights into the important aspects. Table 6 shows the significant metrics of each model. For example, the *sentiment scores* of *battery consumption (T_3)* and *messages (T_{10})* share significant relationships with star-ratings in the category of *health and fitness*.
4) Some key topics are shared by the majority of the categories, while other key topics only appear in one category, such as *play audio & video* in the category of *media and video*. The most popular topic is *speed* which appears in seven categories (i.e., *business*, *communication*, *health and fitness*, *media and video*, *photography*, *tools*, and *travel and local*). The common key topics should be taken care of by the majority of app developers

TABLE 6: The significant metrics of each category and the adjusted R^2 of the model that is built for each category. The topics are indexed in Table 2.

Category	Significant Metrics	Adjusted R^2
Business	T_2(#Words), T_3(Sentiment), T_9(#Words)	0.93
Communication	T_3(Sentiment, #Sentences), T_8(Sentiment), T_9(Sentiment), T_{12}(Sentiment), T_{14}(#Sentences), T_{16}(Sentiment), T_{12}(Sentiment), T_{20}(#Sentences)	0.87
Health and Fitness	T_3(Sentiment), T_{10}(Sentiment), T_{20}(#Sentences),	0.73
Media and Video	T_1(Sentiment), T_7(#Sentences), T_8(#Sentences), T_{11}(#Words), T_{12}(Sentiment), T_{18}(Sentiment)	0.94
Photography	T_1(Sentiment), T_3(Negative Reviews%), T_{11}(Sentiment), T_{18}(Sentiment, #Words)	0.86
Productivity	T_2(#Negative Reviews), T_{11}(Sentiment)	0.75
Shopping	Entropy, T_4(Sentiment), T_{19}(#Sentences), T_{10}(Sentiment), T_{16}(Sentiment), T_{21}(Sentiment), T_{23}(#Sentences)	0.90
Social	T_4(Sentiment), T_9(#Sentences), T_{10}(Sentiment), T_{16}(Sentiment)	0.74
Tools	T_4(Sentiment) , T_9(Sentiment, TRL), T_{16}(Proportion), T_{18}(Sentiment)	0.73
Travel and Local	T_1(Sentiment), T_{16}(#Words)	0.82

and app development companies. For example, testing companies should pay more attention to the issues that are related to the common key topics.

The key topics of each category are the topics of user-reviews that share a significant relationship with star-ratings. The key topics are not the most frequent topics of user-reviews.

4 Evaluation

In this section, we evaluate the relationship between the identified key topics and star-ratings. We check if the changes in star-ratings after each release are related to the key topics.

4.1 Method

We follow the steps below to evaluate the identified key topics for each category. For simplicity, let's describe the approach for the release$_s$ of an app, where release$_s$ is a sample release of the app during the period of our study.

1) We calculate (i) the average of star-ratings that are received after release^{s-1} and before release^s, and (ii) the average of star-ratings that are received after release^s and before release^{s+1}. Then, we measure the difference between the above averages (Δ).
2) We identify the positive changes in star-ratings where $\Delta > 0$. Then, we calculate *weighted* cosine similarity [60] between each release note and key topics versus non-key topics. Release notes highlight the major updates in an app [61]. We calculate the similarity using the words in release notes and vectors of words that are extracted by applying the topic modeling (see Section 2.3). We get the similarity scores for the key topics (λ_s^k) and non-key topics ($\lambda_s^{\neg k}$). If $\lambda_s^k > \lambda_s^{\neg k}$, then positive changes are associated with a higher similarity score of the key topics.
3) For each app, we build two vectors: (i) Λ^k for the key topics (Equation (8)) and (ii) $\Lambda^{\neg k}$ for the non-key topics (Equation (9)).

$$\Lambda^k = \{\lambda_m^k, \lambda_{m+1}^k, ..., \lambda_n^k\} \tag{8}$$

$$\Lambda^{\neg k} = \{\lambda_m^{\neg k}, \lambda_{m+1}^{\neg k}, ..., \lambda_n^{\neg k}\} \tag{9}$$

λ_i^k and $\lambda_i^{\neg k}$ are the similarity scores for the i^{th} version of the app. $m \leq i \leq n$ where m is the oldest release and n is the latest release of the app during our study.
4) Mann-Whitney U test [62] is a non-parametric test to determine two datasets have the same distribution without an assumption of the normality of datasets. We compare Λ^k and $\Lambda^{\neg k}$ using paired Mann-Whitney U test [63]. As a null hypothesis, we assume that two λs are similar. The Mann-Whitney U test rejects the hypothesis with a $p - value < 0.05$ [62].
5) We group the apps into two groups: (i) one group where there is a significant difference in the similarity scores between the key topics and non-key topics (p-value$<$ 0.05), and (ii) one group where there is no significant difference in the similarity scores between the key topics and non-key topics. If we observe a greater proportion of apps in the first group, we can conclude that the release notes that are related to the key topics are more associated with positive changes in star-ratings.

In addition, we repeat the above steps considering the negative changes in star-ratings (i.e., $\Delta < 0$). The goal is to find out if the negative changes in star-ratings are also related to the key topics.

4.2 Findings

First, we describe our findings regarding the positive changes in star-ratings. Then, we explain our findings regarding the negative changes.

4.2.1 *Positive Changes*

For 77% of the apps on average, having a similar release note to the key topics shares a significant relationship with increases in star-ratings. For each category, the second column of Table 7 shows the proportion of apps with p-values less than 0.05 (see step 5 of the evaluation method). The last column of Table 7 shows the differences between the average of the similarity scores for the key topics and non-key topics.

TABLE 7: Proportion of apps with significant differences.

Category	p-value<0.05	p-value ≥ 0.05	Diff
Business	76%	24%	+0.08
Communication	77%	23%	+0.08
Health and Fitness	82%	18%	+0.06
Media and Video	77%	23%	+0.05
Photography	80%	20%	+0.04
Productivity	67%	33%	+0.02
Shopping	81%	19%	+0.07
Social	67%	33%	+0.04
Tools	87%	13%	+0.05
Travel and Local	71%	29%	+0.02

4.2.2 Negative Changes

By repeating the experiment considering the negative changes in star-ratings, we observed that, for 26% of the apps on average, there is a significant difference between the key topics and non-key topics. To investigate the reasons for the above observation, we randomly selected 384 user-reviews of the cases where there is a decline in star-ratings. We find two main reasons:

1) The changes related to the issue of the key topics (reported in the release notes) could not satisfy users. Consequently, the same issues are repeated in the user-reviews of the next versions.
2) Developers make a change related to the key topics that makes users unhappy. For example, the *user interface* is a key topics for the category of *social*. We observed that after a release note concerning the *user interface*: *"We've added a brand new notification center, so now you can choose which items you get."*, star-ratings decreased. The users were complaining about the new changes.

Developers should consider the key topics carefully for the next releases to reduce the risk of receiving low star-ratings.

For 77% of the apps on average, having a similar release note to the key topics shares a statistically significant relationship with positive changes in star-ratings.

5 Threats to Validity

In this section, we discuss the potential threats to the validity of our experiments and findings.

5.1 Internal

Regarding removing inconsistent user-reviews, we manually analyzed the inconsistent user-reviews on a statistically representative sample with the confidence level of 95% and the confidence interval of 5 (i.e., 384 user-reviews). We found that 78% are fake or inconsistent user-reviews. In the evaluation section, we consider all the release notes of the subject apps regardless of the user-reviews. Additionally, we repeat the evaluation approach with an alternation where we only consider the release notes that come after the user-reviews that discuss a key topic. Even with the new alternation in the experiment, we still observe increases in star-ratings. In some cases that developers may conveniently write phrases like "bug fixes" in the release notes, the key topics of *bug reports* and *feature requests* would cover them. However, such short release notes are not recommended [64]. Google Play Store does not provide access to all the user-reviews. Hence, any analysis on the user-reviews might encounter dealing with an incomplete set of data. Martin *et al.* [65] observed that using an incomplete set of user-reviews in Blackberry World app store introduces bias to the findings. They conclude that using incomplete data in other app stores may also bias the findings. To reduce such a bias on the findings, we collect all the user-reviews (i.e., $14,241,915$ user-reviews) gradually during 19 months from $April\ 1,\ 2014$ to $October\ 31,\ 2015$.

5.2 External

Threats to external validity concern the possibility to generalize the findings [66]. Due to the processing time for handling all the user-reviews, we gather the user-reviews from ten randomly selected categories. Therefore, our extracted key topics are limited to the ten categories. However, future research can follow our approach to mine the key topics for other categories.

6 Related Work

Considering the importance of crowdsourcing [67], the number of studies that focus on the user-reviews is on the rise [12]. Several papers have investigated the user-reviews that are posted on mobile app markets. The goal of such papers is to extract knowledge from a relatively huge body of unstructured text to alleviate the process of app maintenance and release [13], [25], [64], [68]. In this section, we summarize the related work along two research directions: (i) summarizing user-reviews and (ii) relating user-reviews to source code.

6.1 Summarizing User-Reviews

Chen *et al.* [25] proposed a framework, called AR-Miner, to identify and filter out uninformative user-reviews. They employed textual analysis techniques to detect and rank informative user-reviews. Using AR-Miner, we filter out uninformative user-reviews from our study.

Panichella *et al* [69] used natural language processing, text analysis, and sentiment analysis to classify the user-reviews into five main intentions of users, including information giving, information seeking, feature requests, problem discovery, and others. Di Sorbo *et al.* [59] presented an approach, called SURF, to summarize user-reviews of mobile apps. Di Sorbo *et al.* [59] employ two levels of classification: (i) intention classification [69], and (ii) topic classification. In our work, we use automated techniques that can process millions of user-reviews to summarize the topics.

Table 8 provides a comparison between the topics that are extracted by Di Sorbo *et al.* [59] with our topics. We identified 23 topics of user-reviews. Our topics not only covers all the topics that are discovered by Di Sorbo *et al.* [59] but also considers a wider range of topics. As shown in Table 8, Di Sorbo *et al.* [59] did not cover nine topics. According to our results, not all the nine topics that are covered by Di Sorbo *et al.* [59] appear as key topics.

Gu and Kim [70] classified user-reviews in five groups of evaluation, praises, feature requests, bug reports, and others.

TABLE 8: Comparison between the topics that are discovered by Sorbo *et al.* [59] and our topics.

Topics by Sorbo *et al.* [59]	**Our Topics**
The Topics in Common	
App	Removed as uninformative user-reviews
GUI	User interface
Contents	We cover more specific topics, such as pictures and social networking
Pricing	Comparing versions
Features or functionality	Feature requests
Improvement	Feature requests and bug reports
Updates/versions	Comparing versions
Resources	We have topics with finer grains, such as battery consumption and storage
Security	Authentication issues
Download	Removed as uninformative user-reviews
Model	Device compatibility and comparing versions
Company	Technical support
The Topics not in Common	
Not Covered	Advertisements
Not Covered	Connection
Not Covered	Language Support
Not Covered	Messages
Not Covered	Purchases
Not Covered	Search
Not Covered	Speed
Not Covered	Task tracking and notifications
Not Covered	Web browsing

They applied aspect opinion mining [71] and sentiment analysis to find the most popular features of an app. However, their work did not reveal the relation of such features with star-ratings.

Topic modeling is widely used in different domains, and interesting results have been inferred [72]. Consequently, some researchers rely on topic modeling technique [37], [73] to summarize the user-reviews. For example, Iacob and Harrison [74] and Guzman and Maalej [24] applied LDA on user-reviews to extract feature requests. Fu *et al.* [8] proposed an approach to analyze star-ratings and user-reviews. Similar to our work, Fu *et al.* [8] crawled and retrieved a large number of user-reviews too, i.e., $13,286,706$ user-reviews. None of the papers mentioned above have studied the topics that share a significant relationship with star-ratings. Although providing a summary of user-reviews is beneficial, developers need to have an understanding of the key topics to achieve higher star-ratings.

6.2 Relating User-Reviews to Source Code

Source code of the majority of mobile apps is not publicly available which is a challenge for researchers. Ciurumelea *et al.* [68] proposed an approach, called UUR, to organize user-reviews with respect to users' requests. Having the user-reviews organized, they could recommend some source code using code localization. Ciurumelea *et al.* [68] defined five high-level topics: (i) *compatibility* which is covered by *device compatibility* in our study, (ii) *usage* which is covered by *user interface* and other topics, such as *social networking*, (iii) *resources* which is covered by *storage* and *battery consumption* topics, (iv) *pricing* which is covered by *comparing versions*, and (v) *protection* which is covered by *authentication issues* in our study. Palomba *et al.* [75] followed the approach proposed by Panichella *et al.* [11] to classify user-reviews and map them to source code. Palomba *et al.* [75] recommended the source code changes required to address the user-reviews by measuring the asymmetric Dice similarity coefficient [76] between the words in user-reviews and the words in each class of source code. Palomba *et al.* [7] studied 100 Android apps to show that implementing users' requests increases star-ratings.

Discussion. Related work (e.g., [24], [25], [70], [75]) can integrate the key topics into their approach in order to receive higher star-ratings. When summarizing user-reviews [13], [25], [59], paying attention to the key topics prevents distractions introduced by frequent topics that share a minor relationship with star-ratings. For example, CLAP [13] clusters user-reviews, then prioritizes the clustered user-reviews. CLAP should give higher priority to the clusters of user-reviews that are related to the key topics. Second, our work can make a great contribution to the papers that relate user-reviews to source code. For example, by having the key topics in mind, developers can focus on the issues that are needed to be addressed first for the next releases. For example, CHANGEADVISOR [75] follows four major steps to recommend changes in an app: (i) user-reviews identification, (ii) processing source code, (iii) user-reviews classification, and (iv) identifying changes. In the first step (i.e., user-reviews identification), CHANGEADVISOR can give the user-reviews that discuss the key-topics a higher priority; thereby, higher priority of changes in the app. Future research should thoroughly investigate all aspects of integrating the key topics into related work, and changes in star-ratings before and after considering the key topics.

7 CONCLUSION

In this paper, we identify the key topics on which the developers should focus for the next releases. The identified key topics provide app developers with a smaller subset of user-reviews for investigation, rather than all the user-reviews. We study the topics of $4,193,549$ user-reviews from Google Play Store that are collected in a 19 month period. Our analysis is based on 623 Android apps in ten randomly selected categories. We employ PMVD to find the key topics of each category. We find that each category has a specific set of key topics that are not necessarily the most frequent topics of user-reviews. Considering the key topics is recommended as they share a statistically significant relationship with star-ratings. Finally, we evaluated our findings using release notes. For 77% of the apps on average, having a similar release note to the key topics shares a statistically significant relationship with positive changes in star-ratings.

In the future, we will integrate the earlier work that summarizes user-reviews with our approach. Thus, developers would be able to prioritize and summarize future changes by focusing on the identified key topics.

ACKNOWLEDGMENTS

We thank the anonymous reviewers who previously reviewed our paper for providing constructive feedback. We also thank Dr. Foutse Khomh from Ecole Polytechnique de Montréal for his valuable suggestions that helped us to improve our work. Also, we thank Guoliang Zhao for helping us in analyzing the topics.

References

[1] Google, "Google play store," [Online]. Available: http://play.google.com/, 2017.

[2] A. Stats, "Number of android applications," [Online]. Available: http://www.appbrain.com/stats/number-of-android-apps, 2018.

[3] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk, "The impact of api change- and fault-proneness on the user ratings of android apps," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, 2015.

[4] H.-W. Kim, H. Lee, and J. Son, "An exploratory study on the determinants of smartphone app purchase," in *Proceedings of the 11th International DSI and the 16th APDSI Joint Meeting*, 2011.

[5] Statista, "Number of apps available in leading app stores," [Online]. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores, 2016.

[6] E. Noei, M. D. Syer, Y. Zou, A. E. Hassan, and I. Keivanloo, "A study of the relation of mobile device attributes with the user-perceived quality of android apps," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3088–3116, 2017.

[7] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "User reviews matter! tracking crowdsourced reviews to support evolution of successful apps," in *Proceedings of the 31st International Conference on Software Maintenance and Evolution*. IEEE, 2015, pp. 291–300.

[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 *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 1276–1284.

[9] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Proceedings of the 21st International Conference on Requirements Engineering*. IEEE, 2013, pp. 125–134.

[10] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in *Proceedings of the 35th International Conference on Software Engineering*. IEEE, 2013, pp. 582–591.

[11] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *Proceedings of the 31st International Conference on Software Maintenance and Evolution*, 2015.

[12] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Transactions on Software Engineering*, vol. PP, no. 99, 2016.

[13] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *38th International Conference on Software Engineering*. ACM, 2016, pp. 14–24.

[14] C. Iacob, V. Veerappa, and R. Harrison, "What are you complaining about?: a study of online reviews of mobile applications," in *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 2013, p. 29.

[15] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[16] T. Johann, C. Stanik, W. Maalej *et al.*, "Safe: A simple approach for feature extraction from app descriptions and app reviews," in *25th International Conference on Requirements Engineering*. IEEE, 2017, pp. 21–30.

[17] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.

[18] H. Khalid, M. Nagappan, and A. E. Hassan, "Examining the relationship between findbugs warnings and app ratings," *IEEE Software*, vol. 33, no. 4, pp. 34–39, 2016.

[19] Optimaize, "Language detection library for java," [Online]. Available: https://github.com/optimaize/language-detector/, 2017.

[20] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[21] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *Proceedings of the 31st Conference on Software maintenance and evolution*. IEEE, 2015, pp. 531–535.

[22] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 1, pp. 163–173, 2012.

[23] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

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

[25] 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.

[26] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2, pp. 103–134, 2000.

[27] C. Nord, *Text analysis in translation: Theory, methodology, and didactic application of a model for translation-oriented text analysis*. Rodopi, 2005, no. 94.

[28] Jazzy, "Jazzy spell checker," [Online]. Available: http://jazzy.sourceforge.net/, 2017.

[29] Allacronyms, "Acronyms and abbreviations," [Online]. Available: https://www.allacronyms.com/, 2017.

[30] Netlingo, "Top 50 most popular text terms," [Online]. Available: http://www.netlingo.com/top50/popular-text-terms.php, 2017.

[31] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.

[32] D. Crystal, *Dictionary of linguistics and phonetics*. John Wiley & Sons, 2011, vol. 30.

[33] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky, "Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task," in *Proceedings of the 50th Internrational Conference on Computational Natural Language Learning: Shared Task*, 2011, pp. 28–34.

[34] W. Martin, B. Al, and P. van Sterkenburg, "On the processing of a text corpus: From textual data to lexicographical information," in *Lexicography: Principles and Practice*, ser. Applied Language Studies Series. London: Academic Press, 1983.

[35] A. Rajaraman, J. D. Ullman, J. D. Ullman, and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press Cambridge, 2012, vol. 77.

[36] J. B. Lovins, *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory, 1968.

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

[38] A. Henriksson, H. Moen, M. Skeppstedt, V. Daudaravičius, and M. Duneld, "Synonym extraction and abbreviation expansion with ensembles of semantic spaces," *Journal of biomedical semantics*, vol. 5, no. 1, p. 6, 2014.

[39] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013, pp. 522–531.

[40] X.-H. Phan and C.-T. Nguyen, "Jgibblda," [Online]. Available: http://jgibblda.sourceforge.net/, 2008.

[41] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[42] J. Cao, T. Xia, J. Li, Y. Zhang, and S. Tang, "A density-based method for adaptive lda model selection," *Neurocomputing*, vol. 72, no. 7, pp. 1775–1781, 2009.

[43] A. E. Raftery, S. Lewis *et al.*, "How many iterations in the gibbs sampler," *Bayesian statistics*, vol. 4, no. 2, pp. 763–773, 1992.

[44] W. R. Gilks, *Markov chain monte carlo*. Wiley Online Library, 2005.

[45] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," *Standford University*, vol. 518, no. 11, pp. 1–3, 2002.

[46] V. R. Basili, "Software modeling and measurement: the goal/question/metric paradigm," University of Maryland at College Park, Tech. Rep., 1992.

[47] R. Van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, "Goal question metric (gqm) approach," *Encyclopedia of software engineering*, 2002.

[48] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile app users complain about? a study on free ios apps," *IEEE Software*, vol. 10, 2015.

[49] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[50] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti, "Automatically assessing review helpfulness," in *Proceedings of the 2006 Conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 423–430.

[51] M.-C. De Marneffe, B. MacCartney, C. D. Manning *et al.*, "Generating typed dependency parses from phrase structure parses," in *Proceedings of the 5th International Conference on Language Resources and Evaluation*, vol. 6, no. 2006, 2006, pp. 449–454.

[52] S. E. S. Taba, F. Khomh, Y. Zou, A. E. Hassan, and M. Nagappan, "Predicting bugs using antipatterns," in *Proceedings of the 29th International Conference on Software Maintenance*. IEEE, 2013, pp. 270–279.

[53] G. Held and T. Marshall, *Data compression; techniques and applications: Hardware and software considerations*. John Wiley & Sons, Inc., 1991.

[54] F. E. Harrell, *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer, 2001.

[55] T. H. Nguyen, B. Adams, and A. E. Hassan, "Studying the impact of dependency network measures on software quality," in *Proceedings of the 26th International Conference on Software Maintenance*. IEEE, 2010, pp. 1–10.

[56] "Harrell miscellaneous," [Online]. Available: http://cran.r-project.org/web/packages/Hmisc/index.html.

[57] U. Grömping *et al.*, "Relative importance for linear regression in r: the package relaimpo," *Journal of statistical software*, vol. 17, no. 1, pp. 1–27, 2006.

[58] B. E. Feldman, "Relative importance and value," *Available at SSRN 2255827*, 2005.

[59] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 499–510.

[60] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[61] J. Foerderer, T. Kude, S. Mithas, and A. Heinzl, "Does platform owner's entry crowd out innovation? evidence from google photos," *Information Systems Research*, 2018.

[62] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[63] S. L. Prescott, C. Macaubas, T. Smallacombe, B. J. Holt, P. D. Sly, and P. G. Holt, "Development of allergen-specific t-cell memory in atopic and normal children," *The Lancet*, vol. 353, no. 9148, pp. 196–200, 1999.

[64] E. Noei, D. A. Da Costa, and Y. Zou, "Winning the app production rally," in *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2018, pp. 283–294.

[65] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 123–133.

[66] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to Advanced Empirical Software Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[67] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.

[68] A. Ciurumelea, A. Schaufelbhl, S. Panichella, and H. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *Proceedings of the 24th International Conference on Software Analysis Evolution and Reengineering*. IEEE, 2017.

[69] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "Ardoc: app reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 1023–1027.

[70] X. Gu and S. Kim, """ what parts of your apps are loved by users?"(t)," in *30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2015, pp. 760–770.

[71] N. Kobayashi, K. Inui, and Y. Matsumoto, "Extracting aspect-evaluation and aspect-of relations in opinion mining." in *EMNLP-CoNLL*, vol. 7. Citeseer, 2007, pp. 1065–1074.

[72] M. Hoffman, F. R. Bach, and D. M. Blei, "Online learning for latent dirichlet allocation," in *advances in neural information processing systems*, 2010, pp. 856–864.

[73] E. Noei and A. Heydarnoori, "Exaf: A search engine for sample applications of object-oriented framework-provided concepts," *Information and Software Technology*, vol. 75, pp. 135–147, 2016.

[74] 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*, ser. MSR '13. IEEE, 2013, pp. 41–44.

[75] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proceedings of the 39th International Conference on Software Engineering*, 2017, pp. 106–117.

[76] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.