Identifying Key Features from App User Reviews

Huayao Wu, Wenjun Deng, Xintao Niu, and Changhai Nie
State Key Laboratory for Novel Software Technology and
Department of Computer Science and Technology
Nanjing University, China
{hywu, niuxintao, changhainie}@nju.edu.cn, wenjundeng@smail.nju.edu.cn

***Abstract*—Due to the rapid growth and strong competition of mobile application (app) market, app developers should not only offer users with attractive new features, but also carefully maintain and improve existing features based on users' feedbacks. User reviews indicate a rich source of information to plan such feature maintenance activities, and it could be of great benefit for developers to evaluate and magnify the contribution of specific features to the overall success of their apps. In this study, we refer to the features that are highly correlated to app ratings as *key features*, and we present KEFE, a novel approach that leverages app description and user reviews to identify key features of a given app. The application of KEFE especially relies on natural language processing, deep machine learning classifier, and regression analysis technique, which involves three main steps: 1) extracting feature-describing phrases from app description; 2) matching each app feature with its relevant user reviews; and 3) building a regression model to identify features that have significant relationships with app ratings. To train and evaluate KEFE, we collect 200 app descriptions and 1,108,148 user reviews from Chinese Apple App Store. Experimental results demonstrate the effectiveness of KEFE in feature extraction, where an average F-measure of 78.13% is achieved. The key features identified are also likely to provide hints for successful app releases, as for the releases that receive higher app ratings, 70% of features improvements are related to key features.**

***Index Terms*—key features, feature extraction, user reviews, app store analysis**

I. Introduction

With the increasing popularity and revenue of mobile application (app) market, app stores, such as Apple App Store and Google Play, have become the primary source for the distribution and acquisition of mobile applications [1]. A prominent nature of app stores is the exploitation of knowledge and creativity of distributed crowds to establish an open ecosystem. This also indicates a paradigm shift in software engineering practices, and has greatly influenced app development, maintenance, and evolution [2], [3].

One powerful mechanism of app stores is that they allow users to rate, and also post reviews to express their opinions and experiences on particular apps [4], [5]. Such a communication channel not only facilitates users for selecting their favourable apps, but also provides app developers a rich source of information for assisting release plannings [6]. However, the volume of available user reviews is simply too large to be manually analyzed [7]–[9]. Researchers have therefore developed a variety of approaches to automatically digest the content in user reviews to extract the most interesting information they carry [10]–[19]. For example, some approaches aim to categorize user reviews into different groups (e.g., bug report, feature request, aspect evaluation, etc.) [20]–[24], while some others attempt to cluster and prioritize user reviews to identify the most crucial topics [7], [25].

Apart from the summarization of user reviews into informative topics, it is also important for app developers to understand the impact of particular features on the success of their apps [26]. A *feature* of an app typically indicates an essential functionality or a service that the app can provide for users [27]. To make the app survive in the highly competing app market, developers might want to find the features that users complain a lot, so that they can try to modify these features in the next releases to retain users [28], [29]; developers might also be interested in the features that are popular in competing apps, because such features can be directly borrowed for requirement elicitation [27], [30].

In order to achieve an effective analysis of the many features of a given app, a primary challenge (and also, an elementary step) is the automatic and accurate extraction of app features. App descriptions and user reviews are good candidate sources for this purpose, but it is not easy to accomplish, because texts in such sources are usually written in unstructured, noisy, and ambiguous natural languages [27], [31]. Currently, despite that there are approaches, such as SAFE [31], for handling English text, the support for Chinese is yet to be investigated. Addressing Chinese app descriptions and reviews could be potentially beneficial for the ever growing and profitable Chinese app market. But this asks for novel approaches, because different linguistic rules and workflows are generally required to process Chinese text [32], and simply reusing existing English-oriented approaches with automatic translation tools is unlikely to be an effective choice [25].

Once features of a given app are extracted, the second challenge is the identification of features that are important for developers' consideration. Empirical studies have demonstrated the influence of app ratings on the rank of app downloads [33], [34], so it is rational to evaluate and magnify the "contribution" of specific features on the app's overall rating score. Currently, Keertipati et al. [35] and Licorish et al. [36] have applied regression-based approaches for prioritizing features, but their approaches rely on a human rater for assigning severity scores for each review sentence. Noei et al. [37] identified key topics that share significant relationships with app ratings, but their approach works at the level of topics of a complete app category, instead of concrete features of each specific app.

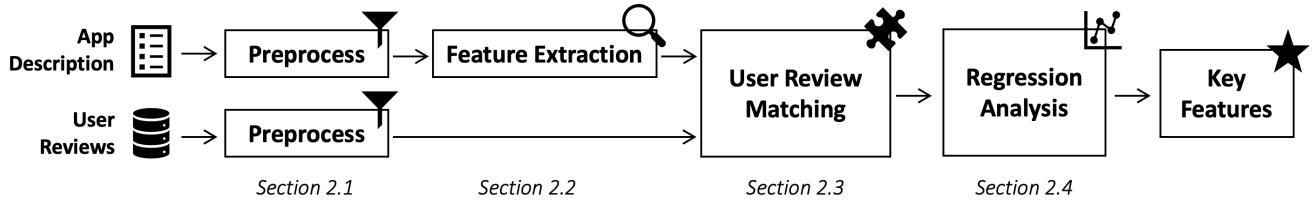

Fig. 1. The overall workflow of the KEFE approach.

In this paper, we propose the KEFE approach to address the above two challenges, i.e., feature extraction and evaluation. In particular, we introduce the concept of *key feature* to refer to the features that are highly correlated to app ratings. Such key features indicate features that should be carefully addressed by app developers in next releases, as proper maintenance of these features is likely to result in more positive reviews and/or fewer negative reviews, and correspondingly, a higher rating score.

In order to identify key features of a given app, the KEFE approach leverages the information in both app description and user reviews. Specifically, 1) KEFE first applies a textual pattern-based approach and a deep machine learning classifier to extract feature-describing phrases from app description; 2) KEFE then matches each app feature with its relevant user reviews by using another classifier; and 3) finally, KEFE applies a multi-linear regression model to yield a set of key features, each of which has a significant relationship with positive and/or negative user reviews received for the given app. We note that the first two steps of KEFE are especially developed to processing Chinese text. Nevertheless, as long as the features can be extracted and associated with user reviews (e.g., by using an approach like SAFE [31]), KEFE remains capable for key features identification.

The KEFE approach presented in this paper differs from previous studies that rely on part of speech and traditional classifier to analyze app descriptions [27], [31], as it further exploits grammatical relations of sentences and utilizes a more powerful deep machine learning classifier to address the feature extraction challenge. Moreover, KEFE relies on the contribution of features to app ratings to identify key features of a specific app, instead of user sentiment or review frequency to a feature [26], [28], [29], or key topics that are common to an entire app category [37]. This indicates a different strategy to address the feature evaluation challenge.

To train and evaluate the KEFE approach, we created a dataset of Chinese Apple App Store, which includes 200 app descriptions and 1,108,148 user reviews. The experimental results demonstrate the effectiveness of KEFE in extracting features from app descriptions, where a *Precision* of 82.49% and a *Recall* of 74.83% are observed. KEFE also performs well in matching app features with their relevant user reviews, achieving an average *F-measure* of 62.02%. Moreover, the identified key feature indeed indicate the features that are important for app developers' consideration. Especially, for the app releases that receive higher app ratings, about 70% of features improvements are related to key features.

Summing up, we make the following contributions in this paper:

- We propose a novel approach, KEFE, to identify key features that have significant relationships with app ratings, i.e., the features that should be carefully addressed by app developers.
- We create a dataset from Chinese Apple App Store, which includes 200 app descriptions and 1,108,148 user reviews.
- We evaluate the performance of KEFE on our dataset, in which experimental results demonstrate its effectiveness and usefulness.

This rest of this paper is organised as follows: Section II introduces the detailed process of the KEFE approach. Section III explains the dataset and experiment design for evaluating the performance of KEFE. Section IV presents the experimental results. Section V discusses potential threats to validity. Section VI introduces related works, and Section VII concludes this paper.

II. Approach

Given an app, the primary goal of the KEFE approach is to automatically identify key features that have significant relationships with its rating scores, based on the information in app description and user reviews. Figure 1 gives the overall workflow of KEFE, which includes the following four main components:

(1) **Preprocess**: removing irrelevant and noisy information from app description and user reviews;
(2) **Feature Extraction**: applying a textual pattern-based filter and a deep machine learning classifier to extract feature-describing phrases from app description;
(3) **User Review Matching**: applying a deep machine learning classifier to determine user reviews that are relevant to each feature;
(4) **Regression Analysis**: applying a multi-linear regression model to identify key features.

In the next sections, we will explain the detailed process for each of the above components.

TABLE I
THE TEXTUAL PATTERNS FOR PHRASE EXTRACTION

#	Pattern	Sentence	Phrase Extracted
1	VOB (verb-object)	user can *send messages*	send messages
2	ATT (attribute)	provide a *music player*	music player
3	VOB (verb-object) + [ATT (attribute)]	user can *listen to classical music*	listen to classical music
4	FOB (fronting-object)	any type of *music* can be *searched*	search music
5	COO (coordinate) + VOB (verb-object)	user can *send pictures* and *videos*	send pictures & send videos
6	COO (coordinate) + VOB (verb-object) + [ATT (attribute)]	users can *listen to classical* and *pop music*	listen to classic music & listen to pop music
7	DBL (double) + VOB (verb-object)	there are plenty of *novels* to *read*	read novels
8	COO (coordinate) + RAD (right adjunct) + VOB (verb-object)	user can *download information* about *tickets*, *hotels*, etc	download tickets information & download hotels information
9	COO (coordinate) + RAD (right adjunct) + VOB (verb-object) + [ATT (attribute)]	user can *download information* about *train tickets*, *air tickets*, etc.	download train tickets information & download air tickets information
10	Individual word	all resources are available upon *registration*	registration

A. Preprocess

The preprocess step removes noisy sentences and characters from app description and user reviews. This is a common step in studies of user review analysis [28], [37].

1) Preprocessing App Description: App description represents a relatively formal text that describes what the app is about and what it can do for users. Given an app description, KEFE removes sentences that describe contact and subscription information (e.g., email address, telephone number, and membership fees) and the sentences after them. Such sentences typically appear at the end of the description, and do not describe app features. KEFE also removes redundant punctuation marks, special symbols (e.g., %, #, @, etc.), emojis, and characters except for Chinese and English from the description (we note that some features might be described in English phrases in Chinese app stores).

2) Preprocessing User Reviews: In contrast to app descriptions, user reviews are usually in short length, and contain informal and mixed vocabularies. For each user review, KEFE first removes all characters that are not written in Chinese and English, in order to reduce potential disturbance to future analysis. Next, since there is no explicit boundary between words in Chinese, KEFE applies the *pyltp* toolkit [38] to split the review text into sentences, and segment each sentence into words. At last, KEFE removes stop words that are common in Chinese app reviews based on a stop-words list. This stop-words list is created by extending a standard list of Chinese language[1], aiming to include words that are common in user reviews but are not relevant to app features. At the end of this step, if a sentence in user review contains no word after stop words removal, this sentence will be removed.

B. Feature Extraction

Different studies usually have different ways to define a feature [28], [31], [33]. In this study, we refer to a *feature* to an essential functionality or a service that the app can provide for users. This can indicate a description of functional capabilities

[1]https://github.com/goto456/stopwords

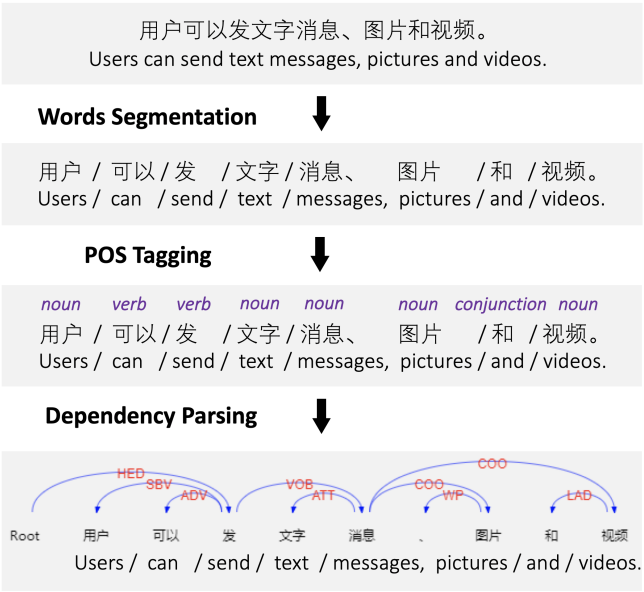

Fig. 2. An example of the phrase extraction process.

of the app (e.g., "upload images" and "online streaming"), or a resource that users can access (e.g., "music library").

In order to extract feature-describing phrases from app description, KEFE applies a process that is similar to that of Jiang et al. [27] (but with different techniques). This process first uses textual pattern-based approach to extract candidate phrases, and then uses a classifier to determine whether each of these candidate phrases is feature-describing or not.

1) Phrase Extraction: Given an app description with clean sentences, KEFE first extracts candidate phrases (i.e., a small group of words) that could represent app features. Figure 2 uses an example to illustrate this process.

The first step of phrase extraction is words segmentation. KEFE applies the *pyltp* toolkit [38] to segment the app description into words. Here, to improve the performance of the default words segmentation model of *pyltp*, an additional

words segmentation directory is created and used. This directory contains common collocations of Chinese characters that frequently appear in app descriptions.

After words segmentation, KEFE applies Part-Of-Speech (POS) tagging to label each segmented word in the sentence with its appropriate part of speech. With the above results, KEFE further conducts dependency parsing to reveal the syntactic structure of the sentence. This step will produce a syntax parsing tree that indicates the grammatical relations between words in the given sentence[2]. Take the sentence in Figure 2 as an example. According to the syntax parsing tree, we can see that the head predict of the sentence is *send*, and *send* and *messages* follow a Verb-Object relation (VOB). KEFE applies the *pyltp* toolkit [38] to implement POS tagging and dependency parsing.

Once the syntax parsing tree is constructed, KEFE applies a textual pattern-based approach to extract candidate phrases. The use of textual patterns is a common choice for feature extraction, and there are several patterns developed for English app descriptions (e.g., verbs, adjectives, and nouns are usually considered as important parts of speech in defining features) [27], [31]. To derive the textual patterns for KEFE, in this study, we first examined the grammatical relations behind existing POS patterns [27], [31] to investigate whether they can be reused for processing Chinese text. Next, we collected a large number of Chinese app descriptions, and performed the above phrase extraction process to build their respective syntax parsing trees. We then manually analyzed these trees to identify grammatical relations that are related to app features. After these steps, we have developed ten patterns for KEFE, as shown in Table I.

Specifically, each pattern in Table I gives a combination of grammatical relations, indicating a common linguistic rule for describing features (the square bracket indicates that the relation can appear more than once in the sentence). For example, according to pattern VOB (verb-object relation), the phrase "*send messages*" will be extracted as a candidate phrase from the sentence in Figure 2.

2) Feature Classification: Once a set of candidate phrases is extracted, KEFE applies a deep machine learning classifier to determine whether each of these phrases is a feature-describing phrase or a non-feature-describing phrase. This represents a common binary classification problem, where each data instance consists of a candidate phrase f, and a boolean label L indicating whether this candidate phrase describes a feature or not.

In this study, we choose to use BERT (Bidirectional Encoder Representations from Transformers) [39] to implement the classifier. BERT is a recent method of pre-training language representations, which allows users to fine-tune a specific classifier based on a general-purpose pre-trained model (specifically, the BERT-Base model for Chinese language[3] is used in this study). The use of BERT has achieved very promising results in a wide range of natural language processing tasks [39]. We also compared BERT with several traditional classifiers in our preliminary experiments (including Naïve Bayes, C4.5, and Random Forest), and BERT tends to exhibit the best performance.

[2]More details about the grammatical relations can be found at https://www.ltp-cloud.com/intro_en

[3]https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

C. User Review Matching

After extracting feature-describing phrases from app description, the next step of KEFE is to find user reviews that are relevant to each feature. Specifically, for a given set of feature-describing phrases, $F = \{f_1, f_2, \ldots, f_n\}$, and a set of user reviews $R = \{r_1, r_2, \ldots, r_m\}$, the user review matching process will produce the set $R_F = \{R_1, R_2, \ldots, R_n\}$, where $R_i \subseteq R$ is the set of reviews that comment on feature f_i $(1 \leq i \leq n)$, that is, feature f_i is mentioned in every review $r \in R_i$. Here, sets R_i and R_j may have overlaps, because a user review may talk about multiple features. For example, the review "I cannot send pictures and videos" is relevant to two features: *send pictures* and *send videos*.

To implement user review matching, we also choose to use BERT [39] to train a deep machine learning classifier (with the same pre-trained model[3]). Similarly, this also represents a binary classification problem, where each data instance consists of a feature-describing phrase f, a user review r, and a boolean label L indicating whether f is mentioned in r. The trained BERT model will then assess the semantic similarity between the given feature f' and user review r', and produce two class probabilities (matching and non-matching). We take the class with higher probability, i.e., $p > 0.5$, as the result of classification.

Note that the previous SAFE [31] approach also addresses the user review matching problem. But it uses the same feature extractor to extract feature-describing phrases from both app description and user reviews, and applies a binary semantic similarity function to determine matching. However, app descriptions and user reviews usually have different characteristics (e.g., user reviews are shorter, and more informal [27]). In our preliminary experiments, we found that it is ineffective to use the same textual pattern-based approach to extract features from Chinese user reviews. We thus choose to use a classifier to implement the user review matching process.

D. Regression Analysis

The last step of KEFE is to apply regression analysis technique to identify key features that have significant relationships with app ratings. The input of this step includes the set of feature-describing phrases $F = \{f_1, f_2, \ldots, f_n\}$, and their corresponding user reviews $R_F = \{R_1, R_2, \ldots, R_n\}$; the output is a set of key features. In this study, since Apple App Store uses a five-stars scale to represent user rating scores, we consider ratings of one or two stars as *negative reviews*, and ratings of four or five stars as *positive reviews*.

In order for an app to receive a higher rating score in future releases, it is rational to increase the number of positive reviews, and at the same time, decrease the number of negative

reviews. KEFE thus uses a regression model to investigate the relationships between the number of all positive (or negative) reviews received (i.e., the dependent variable), and the number of positive (or negative) reviews that are relevant to each feature (i.e, the independent variables).

Specifically, KEFE first calculates the number of all negative (or positive) reviews received in each of the past m days, $W = [w_1, w_2, \ldots, w_m]$, as well as the number of negative (or positive) reviews that are relevant to each feature f_i over the same time period, $W_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}]$, for $1 \leq i \leq n$. Here, because user reviews that are posted a long time ago might contain no useful information for the current release, we choose to use the reviews that are posted in the last $m = 180$ days only (about half a year). In addition, we remove features that are not mentioned in any review of the past m days, because such features tend to have no impact on the app ratings. Then, KEFE builds a multi-linear regression model that uses variables W_i to predict the outcome of variable W. The regression equation used is in the form of:

$$W = \beta_0 + \beta_1 \times W_1 + \beta_2 \times W_2 + \cdots + \beta_n \times W_n$$

where β_0 is a constant, and β_i is the correlation coefficient associated with variable W_i $(1 \leq i \leq n)$.

KEFE uses the *statsmodels* package of Python to build the regression model. This process will produce a p-value for each independent variable to indicate whether the relationship between this variable and the outcome is statistically significant. Finally, we consider the variables with p-value < 0.05 as significant variables, and the features associated with these variables are identified as key features.

KEFE will apply the above process based on positive and negative reviews, respectively. The key features identified are then merged together to produce the final set of key features of the given app. We note that the concrete improvement strategies to address key features are left to app developers' decision. For example, developers can choose to improve key features of positive reviews (e.g., by adding functionalities) to amplify their attractions, and/or to improve key features of negative reviews (e.g., by fixing bugs) to reduce users' complaints.

III. Experiment Design

In this section, we first describe the research questions we seek to investigate, and then the detailed approaches we used for evaluation. In order to assist others to replicate our findings and reuse the KEFE approach in further work, we provide our dataset, natural language processing resources, classification models, and executable scripts of KEFE at this paper's companion website: https://github.com/GIST-NJU/KEFE.

The experiments of this study were conducted on several machines, each of which is equipped with Intel Xeon@2.5GHz CPU, 32GB memory, and Windows 10 operating system.

A. Research Questions

In this study, we set out the following three research questions to investigate the performance of the three major steps of the KEFE approach. The first two research questions concern the effectiveness of feature extraction and user review matching. While the last research question concerns the usefulness of the identified key features for app development.

- **RQ$_1$** *How effective is KEFE in extracting feature-describing phrases from app descriptions?*
- **RQ$_2$** *How effective is KEFE in matching app features with their relevant user reviews?*
- **RQ$_3$** *How useful are the key features identified by KEFE for release plannings?*

B. Dataset

The dataset used in this study was retrieved from QiMai[4], which is a commercial platform for collecting and analyzing app-related data of popular Chinese app stores (including Apple App Store and nine app stores of Android platform). Here, we chose Apple App Store as the source of this study, because QiMai provides the most complete information for this app store.

To create the dataset, we randomly selected ten categories from Chinese Apple App Store. This helps to avoid introducing bias towards specific categories [3]. Then, for each category, we randomly selected ten free apps and ten paid apps from their respective top-20 ranked lists. We chose to focus on those popular apps, because they tend to receive more attentions from users, and thereby have enough information for carrying out this study. In addition, to ensure that the features of the app are well documented, the description of the app selected should contain more than 100 Chinese characters. At the end of this step, we had a collection of 200 apps (10 categories $\times$ 20 apps per category).

We extracted app descriptions of these 200 apps for answering RQ_1. We then selected five apps (we will explain the selection of these subject apps in the next section), and extracted their user reviews of a period of roughly five years for answering RQ_2 and RQ_3 (including user name, title, content, date, and rating score of each review). This results in a total number of 1,108,148 raw user reviews in our dataset.

C. Evaluation Approaches

In this section, we present the approaches for answering each of the three research questions posed in Section III-A.

1) Approach for answering RQ_1: The first research question concerns whether the feature-describing phrases can be precisely extracted from app description. To this end, we randomly selected 150 apps from our dataset for training the feature classifier of KEFE (15 apps per category)[5], and used the remaining 50 apps for evaluation.

Specifically, for each of the 150 apps used for training, we applied the Preprocess and Phrase Extraction processes (see Sections II-A and II-B) to extract candidate phrases from app description. We then manually annotated each of these phrases

[4]https://www.qimai.cn

[5]The stop-words list and words segmentation directory (see Sections II-A and II-B) used in KEFE were also created based on a deep observation and analysis of these 150 apps.

TABLE II
THE CONFUSION MATRIX

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP	FP
	Negative	FN	TN

as either feature-describing phrase or non-feature-describing phrase. This process was performed independently by two authors of this paper, and we have set a guideline to ensure that they follow the same criteria to do the annotation. During this process, if the two authors assign different labels to a data instance (i.e., one author think that it is feature describing, but the other author disagrees), a third author will be involved and the final label will be decided based on the vote of all three authors. This annotation process ends with 3,121 feature-describing phrases and 6,135 non-feature-describing phrases, and these data were used as the training set to fine-tune the BERT classifier for feature extraction.

Next, for each of the remaining 50 apps, we manually extracted a set of *golden* features from its app description as the ground truth for evaluation. This process was also performed independently by two authors with a pre-defined guideline. We note that the extraction of golden features is different from the annotation of phrases, as the former requires to pick out feature-describing phrases directly from app description, while the latter only requires to assign a boolean label to each candidate phrase. We also note that both of the two authors are major in computer science and have enough experience with software development, so it is not difficult for them to perform the extraction task. Similarly, if the two authors cannot reach consensus (i.e., there exists any difference in the phrases extracted), a third author will be involved in the discussion to resolve disagreements.

Finally, we applied KEFE to automatically extract feature-describing phrase from app descriptions of the above 50 apps, and compared these phrases against their corresponding golden features. We then calculated the *Precision*, *Recall*, *F-measure*, and *Accuracy* that can be achieved:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, FP, FN, and TN indicate the true positives, false positives, false negatives, and true negatives of the confusion matrix (as shown in Table II), respectively.

TABLE III
THE SUBJECT APPS AND RELEASES SELECTED FOR ANSWERING RQ_2 AND RQ_3

Name	**# Reviews**	**# Releases**	
		Positive	**Negative**
Alipay	99,837	9	9
DingTalk	34,637	7	6
NetEase Music	330,381	9	9
Tencent Video	187,553	7	6
WeChat	455,740	8	8
Sum	1,108,148	40	38

We note that KEFE is the first approach for extracting features from Chinese app descriptions. The previous state-of-the-art feature extractors, SAFE [31] and SAFER [27], cannot be directly applied in our dataset, because they rely on several English-oriented textual patterns, and we find that most of them are not suitable for processing Chinese text. Meanwhile, some additional treatments (e.g., word segmentation) are also required in Chinese-oriented natural language processing tasks. Hence, we just chose to report the results of these two previous extractors for handling English app descriptions as references, in order to demonstrate that the performance of KEFE is at a competitive level.

2) Approach for answering RQ_2: The second research question concerns whether user reviews that comment on each feature can be precisely extracted. In this study, we randomly selected five apps from the 50 apps used in RQ_1 (i.e., the evaluation set) as the subjects for answering RQ_2 and RQ_3. Table III gives the number of user reviews collected for these apps.

Here, since it is infeasible to manually annotate all the above user reviews to evaluate the BERT classifier, we randomly selected 2,500 user reviews for each app to create the train and test data sets. Specifically, we first applied the Preprocess step (see Section II-A) to remove irrelevant reviews from the above review set. Then, for each combination of app feature and user review, a boolean label was manually assigned to indicate whether the content of the review is relevant to the feature. Similar to the annotation process of candidate feature-describing phrases (Section III-C1), we have set a guideline to perform the annotation task of user review matching. Each data instance here was annotated by one author only, because this step involves a collection of roughly 300,000 data instances (i.e., combinations of features and reviews).

Finally, for each app, we shuffled its data instances and divided them into train-set (60% of instances), dev-set (20% of instances) and test-set (20% of instances). All train-sets and dev-sets were merged together and used to fine-tune the BERT classifier, and the test-set of each app was used for evaluation. Similarly, we calculated the *Precision*, *Recall*, *F-measure*, and *Accuracy* that can be achieved.

3) Approach for answering RQ_3: The last research question concerns whether the key features identified by KEFE are useful for app development. To establish the connection

TABLE IV
THE RESULTS OF FEATURE EXTRACTION

App Category	# Characters	# Golden Features	Precision	Recall	F-measure	Accuracy
Business	755	43.8	89.83%	72.60%	80.30%	83.19%
Tool	589.6	23.6	84.35%	82.20%	83.26%	89.49%
Travel	986.0	39.0	85.05%	84.62%	84.83%	89.86%
Social	345.8	15.0	86.67%	69.33%	77.04%	87.55%
News	665.0	18.8	81.00%	86.17%	83.51%	89.97%
Navigation	565.4	20.8	79.17%	54.81%	64.77%	74.38%
Music	720.0	19.6	79.57%	75.51%	77.49%	86.31%
Life	466.6	28.2	86.21%	70.92%	77.82%	82.24%
Education	697.4	28.6	82.44%	75.52%	78.83%	87.76%
Entertainment	1250.8	9.4	70.59%	76.60%	73.47%	96.37%
Average	704.2	24.7	82.49%	74.83%	78.13%	86.71%
SAFE [31]			55.90%	43.40%	45.80%	NA
SAFER [27]			61.79%	80.27%	69.11%	69.57%

between key features and decisions made for the next releases, we consider the positive and negative changes in app ratings, and investigate whether these changes are related to the improvements of key features. Here, we refer to activities that seek to optimize a feature to better satisfy users' needs as the *improvement* of the feature (including adding functionalities, enhancing efficiency, fixing bugs, etc.). As such, a key feature is considered useful, if improving this feature can lead to a higher rating score.

To this end, we used release notes as the practical ground truth [37], as they usually contain the major updates of app features [11], [40]. We first selected a series of releases with positive, or negative, changes in rating scores for evaluation (we refer to such releases as positive, or negative, releases). Given an app with t releases, $\{r_1, r_2, \ldots, r_t\}$, a release r_i $(1 < i < t)$ will be selected if it satisfies the following criteria:

- the release note of r_i explicitly mentions the improvement of at least one feature;
- there is a positive, or negative, change between $A_1 =$ the average of ratings received during $[r_i, r_{i+1}]$, and $A_2 =$ the average of ratings received during $[r_{i-1}, r_i]$ (i.e., $\Delta = A_1 - A_2 > 0$ for positive releases, and $\Delta < 0$ for negative releases);
- the number of reviews received during both $[r_i, r_{i+1}]$ and $[r_{i-1}, r_i]$ are greater than 500. This avoids to introduce bias from unreliable rating changes.

By filtering release notes of the five apps used in RQ_2, we retrieved a total number of 78 releases as the subjects for answering RQ_3. The numbers of positive and negative releases selected for each app are shown in the last column of Table III.

For each subject release r_i, we followed the same process of golden feature extraction (as explain in Section III-C1) to manually extract a set of features N that are improved in this release. Here, a feature is extracted if and only if some optimizations of this feature are explicitly mentioned in the release note of r_i (e.g., "now you can receive money in group chats" indicates an improvement of the feature "group chats"); Vague descriptions (e.g., "several bug fixes") will not be considered as feature improvement. In addition, We ignored features that are newly introduced, because the main focus of this study is on the improvements of existing features.

TABLE V
THE RESULTS OF USER REVIEW MATCHING

Name	Precision	Recall	F-measure	Accuracy
Alipay	92.96%	43.42%	59.19%	98.58%
DingTalk	95.20%	88.26%	91.60%	99.82%
NetEase Music	94.74%	42.35%	58.54%	99.28%
Tecent Video	95.00%	25.00%	39.58%	98.45%
WeChat	94.92%	45.16%	61.20%	99.30%
Average	94.56%	48.84%	62.02%	99.09%

We then applied KEFE based on the set of existing features and user reviews received before r_i to identify key features. Finally, by comparing the set of key features obtained against N, we calculated the Hit Ratio (Hit) of key features:

$$Hit = \frac{\text{\# key features in } N}{\text{\# features in } N}.$$

Here, if we can observe a high value of Hit for positive releases and a low value for negative releases, we can conclude that the improvement of key features is more likely to receive higher rating scores.

IV. RESULT

This section presents results for answering our research questions posed in Section III-A.

A. RQ_1: Feature Extraction

The first research question investigates the effectiveness of the feature extraction approach used in KEFE. The first three columns of Table IV give the average number of Chinese characters in app descriptions, and the average number of golden features extracted for the 50 apps used for evaluation (10 categories $\times$ 5 apps per category). From Table IV, we can see that the number of features mentioned in app descriptions varies across different categories, where the apps of *Business* and *Travel* categories tend to describe the most features. Overall, an average number of 24.7 golden features are extracted for each app.

TABLE VI
THE RESULTS OF KEY FEATURES IDENTIFICATION

Name	# Features	# Key Features	Hit (KEFE)		Hit (Random)	
			Positive Release	Negative Release	Positive Release	Negative Release
Alipay	15	5.9	71.30%	20.37%	42.69%	42.72%
DingTalk	56	8.5	65.24%	26.19%	16.73%	10.24%
NetEase Music	17	7.9	77.78%	46.30%	44.88%	42.84%
Tecent Video	16	4.2	85.71%	58.33%	28.02%	28.89%
WeChat	22	7.9	49.79%	12.50%	35.59%	35.42%
Average	25.2	6.9	69.96%	32.74%	33.58%	32.02%

The last four columns of Table IV give the *Precision*, *Recall*, *F-measure*, and *Accuracy* obtained by applying the feature extraction approach of KEFE. From these results, we can see that KEFE generally performs well for all kinds of apps. On average, it can achieve a *Precision* value of 82.49%, and a *Recall* value of 74.83%; while in terms of *F-measure* and *Accuracy*, these values are 78.13% and 86.71%, respectively. In particular, KEFE tends to exhibit the best performance for the *Business*, *Tool*, *Travel*, and *News* categories, where the *F-measure*s observed are all higher than 80%.

In addition, the last two rows of Table IV give the overall average results of two state-of-the-art feature extractors for English app descriptions as references [27], [31]. Note that it is difficult to directly compare with these data here, because the previous extractors cannot be directly applied in our dataset. But according to the data reported in Table IV, we are comfortable to claim that the performance of KEFE is indeed at a competitive level for feature extraction.

Answer to $\mathbf{RQ_1}$: KEFE can effectively extracts feature-describing phrases from app descriptions, where an average *F-measure* of 78.13% can be achieved.

B. RQ_2: User Review Matching

The second research question investigates the effectiveness of the user review matching approach used in KEFE. Table V gives the *Precision*, *Recall*, *F-measure*, and *Accuracy* that can be achieved.

From Table V, we can see that KEFE can precisely extract user reviews that are relevant to each app feature. Specifically, the *Precision* observed is at least 92% for all five apps, and an average value of 94.56% can be achieved. However, we can see that the approach tends to have a relatively poor performance in terms of *Recall*, where only an average value of 48.84% is achieved. Nevertheless, the approach can still achieve a *Recall* value of 88.26% for *DingTalk*, and the average *F-measure* is 62.02%.

The relatively low *Recall* obtained indicates that the user review matching approach of KEFE might miss some user reviews. In practice, since there are usually a large volume of user reviews, developers might want to extract substantially more relevant user reviews for further analysis rather than irrelevant ones (in this study, building the model with the most accurate data). In this case, the low *Recall* might not greatly influence the performance of the following key features identification.

Answer to $\mathbf{RQ_2}$: KEFE is highly precise in matching app features with their relevant user reviews, but the *Recall* observed is relatively low. The average *F-measure* of user review matching is 62.02%.

C. RQ_3: Key Feature Identification

The last research question concerns the usefulness of the key feature identification approach. Because there is no available approach that can analyze Chinese app description and user reviews to derive important features, we choose to implement a random approach as the baseline for comparison. Specifically, for each of the 78 subject releases (see Table III), we first apply KEFE to perform user review matching and identify the set of key features. Then, the random approach will randomly pick up the same number of 'key' features from the set of all features as the results (their executions are repeated 30 times in each release to account the randomness).

Table VI gives the number of features for each app, the average number of key features that KEFE identifies, and the average value of Hit that can be achieved by KEFE and random approaches. From Table VI, we can see that key features account for a relatively small proportion of features that are described in app descriptions. This is important for the practical application of a key feature identification approach, because app developers might want to take only a few yet important features into consideration. In particular, *DingTalk* has the largest number of features among all five apps studied, while only 15% (8.5/56) of these features are identified as key features. On average, there are 25.2 features in the subject apps, and KEFE suggests only 27% (6.9/25.2) of them for developers' consideration.

We then consider the usefulness of key features identified by KEFE. From Table VI, generally, we can observe a high value of Hit in positive releases, and a low value in negative releases. For example, for *Alipay*, the average Hit observed for positive releases is 71.3%. This indicates that 71.3% of features improvements in these release are related to key features (i.e., key features play a more important role in these releases than non-key features). By contrast, for the negative releases of *Alipay*, this proportion is only 20.37% (i.e., most improvements in these releases are related to non-key features). This finding reveals that improving key features in the next release is more beneficial than improving non-key features. On average, the identified key features account

for 69.96% of features improvements in releases that receive higher app ratings.

With respect to the random approach, we can see that it tends to achieve similar Hit values for both positive and negative releases, and the Hit observed is substantially lower than that of KEFE in all positive releases. On average, the randomly selected features can only account for 33.58% of features improvements in positive releases; and the lowest proportion, 16.73%, is observed for the app that has the largest number of features, i.e., *DingTalk*. This finding indicates that it is unlikely to receive higher app ratings by improving features that are randomly determined. KEFE can thus play a better role in the identification of important features.

Answer to $\mathbf{RQ_3}$: the identified key features indeed indicate features that should be carefully addressed by app developers. For the releases that receive higher app ratings, about 70% of features improvements are related to key features.

V. Threats to Validity

As far as internal threats to validity are concerned, the results reported in this study might be influenced by manual analysis involved in the annotation of training data (for two machine learning classifiers) and the extraction of golden features (from both app descriptions and release notes). To mitigate this risk as much as possible, we have set guidelines to specify the procedure, criteria and examples, to avoid potential subjective bias or mistakes, and have carefully examined the annotation results. We note that the authors responsible for performing these tasks are all major in computer science and have a good understanding of software development.

Another potential threats to validity derives from the evaluation of usefulness of the identified key features, because planning next app releases can be a very complex problem and involve many trade-offs in the real world. It is ideal to invite actual app developers to determine whether the identified key features are useful to them, but it is labor intensive and can also introduce subjective bias. Instead, we choose to use release notes as a practical ground truth, which can reflect real-world decisions made by developers to some extent. In addition, the performance of feature extraction and user review matching processes might be influenced by the stop-words list and words segmentation directory used. We choose to make these data publicly available in our dataset, so that others can replicate and extend our approaches.

As far as external threats to validity are concerned, in this study, we used 50 app descriptions for evaluating the performance of KEFE in feature extraction (on average, each app has 24.7 features), and collected 1,108,148 user reviews and 78 releases for evaluating the performance of key features identification. Despite that these apps all come from Chinese Apple App Store, they are selected from ten different categories, and thereby cover a diverse set of feature descriptions and user groups. We acknowledge that KEFE might exhibit different performance on different apps. Especially, since KEFE usually requires a number of user reviews for building a statistically significant regression model, a poor performance might be observed for apps with few user reviews (it is nevertheless easy to conduct manual analysis in this case). In the future, we plan to evaluate the performance of KEFE on more apps across different app stores.

VI. Related Work

App store analysis is a popular field of software engineering researches, in which a wide and diverse range of approaches have been developed [6]. In this section, we discuss related works from two directions: 1) summarization of user reviews, and 2) feature extraction and evaluation.

A. Summarization of User Reviews

User reviews are usually in large volumes and contain noisy and irrelevant information. In order to extract useful information from user reviews, a variety of studies have been conducted [8], [9], [14], [16], [18], [20], [41]. For example, Guzman et al. [21], Panichella et al. [22], and Ciurumelea et al. [23] proposed approaches to classify user reviews into specific groups according to the information they carry. Mcilroy et al. [24] assigned multiple labels to each user review to indicate different types of issues. Di Sorbo et al. [12] took both review topics and user intentions into account, and developed the SURF approach to generate user review summaries.

In addition to the summarization of user reviews into informative topics, there are also studies that attempt to prioritize topics in order of importance. AR-Miner [7] is an exemplary approach for this purpose. It employs a pre-trained classifier to identify informative reviews, and then groups these reviews into topics and ranks these topics by priority. Later, Scalabrino et al. [25] developed the CLAP approach. CLAP especially categorizes user reviews into seven groups (e.g., functional bug reports, suggestion for new features, etc.), and uses a classifier to assign high or low priority to each review cluster; the cluster with high priority indicates topics that developers are advised to address in the next release.

Topics in user reviews can also be time-sensitive with fluctuations over releases. Gao et al. [11] developed the PAID approach to investigate the changes in topics of user reviews. Gao et al. [15] also developed IDEA, which uses an anomaly detection method to identify abnormal topics. Here, a topic is considered abnormal, if it frequently appears in current user reviews but not previously. Later, the DIVER [17] approach was developed to enhance the effectiveness and efficiency of IDEA in an industrial environment.

In order to further support app developers in accommodating change requests in user reviews, there are also studies that seek to link user reviews to other software artifacts. For example, Palomba et al. [13] devised an approach that can base structure, semantics, and sentiments of use reviews to recommend and localize source code changes. Zhang et al. [19] propose to use Word2Vec to map user feedbacks to issues reports, which can achieve a high accuracy in change localization.

More recently, Noei et al. [37] investigated the *key topics* of user reviews that are significantly correlated to app ratings.

They found that different app categories typically have different sets of key topics, and these topics are not always the topics with the highest frequency.

B. Feature Extraction and Evaluation

Unlike the above studies that mainly focus on the topics covered by user reviews, there are also studies that seek to analyze specific (fine-grained) features, i.e., essential functionalities or services, that the app provides [26]–[31], [33], [35], [36].

A common and elementary step of such feature-related studies is feature extraction. Harman et al. [33] proposed a data mining approach to extract features from app description. Johann et al. [31] developed the SAFE approach, which employs a set of pre-defined textual patterns to extract features from both app descriptions and user reviews. The extracted features were then compared to determine whether each feature is actually mentioned in user reviews. The feature extraction challenge was also highlighted by Jiang et al. [27], while they built a classifier to extract features from app description, and then recommended missing features for similar apps.

Once the features are extracted, developers might want to evaluate the relative importance of different features for release plannings. Guzman and Maalej [28] scored the features based on user sentiments, where app features were extracted from user reviews by a collocation finding algorithm. Gu and Kim [29] also concentrated on the sentiment of users about app features. Likewise, they identified features from user reviews, but they distinguished between feature evaluation and feature request. Keertipati et al. [35] and Licorish et al. [36] proposed approaches for prioritizing features extracted from negative user reviews. They combined review frequency, rating, emotion, and deontic to build a regression model, and illustrated the use of their approaches by case studies.

Recently, Malik et al. [26] introduced the concept of *hot features*, i.e., the features that are most frequently mentioned in user reviews, for feature evaluation and recommendation. They used sentiment analysis to explore users' opinions on the hot features, where the features that users prefer could be recommended for similar apps.

Discussions. The KEFE approach presented in this paper aims to identify key features based on app description and user reviews. It differs from the above studies in two main aspects. First, the feature extraction and user review matching approaches used in KEFE are designed to process Chinese text, while previous studies [27], [31] mainly focus on English text and provide few multi-language supports. Moreover, previous studies [27], [31] typically rely on part of speech and traditional classifier to extract features, while KEFE further exploits grammatical relations of sentences and utilizes a more powerful deep machine learning classifier.

Second, KEFE studies the features that are most closely related to app ratings, instead of user sentiments [28], [29], or frequency of reviews [26]. There is one study [37] that seeks to investigate the correlation between topics and app ratings, but it focuses on topics of a complete app category, instead of concrete features of a specific app. The studies of Keertipati et al. [35] and Licorish et al. [36] are the closest existing studies to KEFE, where they build a regression model to prioritize features based on negative user reviews. KEFE differs from their studies, as it additionally uses the information in app description, and built the regression model based on different variables (both positive and negative reviews are accounted). KEFE is also automatic, and empirically evaluated, while the previous studies [35], [36] rely on a human rater to decide the severity of each sentence, and only report results of two case studies.

In addition, KEFE aims to extract existing features from app description, and analyzes user reviews that comment on these features. This differs from the analysis of feature request based on user reviews [20], [25], because the key features identified must be an existing feature of the given app. Accordingly, if a review talks about new feature requests only, this review will not be used to perform regression analysis in KEFE.

VII. Conclusion

The features greatly impact the success of apps. This study presents the KEFE approach to identify key features of a given app based on its app description and user reviews. Such key features indicate features that have significant relationships with user rating scores, and thereby should be carefully addressed by app developers. In order to effectively identify key features, KEFE firsts applies a textual pattern-based filter and a BERT classifier to extract feature-describing phrases from app description. KEFE then matches each app features with its relevant user reviews by another BERT classifier. Finally, KEFE builds a regression model to investigate the relationships between features and positive/negative user reviews, where the features associated with statistically significant explanatory variables are identified as key features. The KEFE approach is trained and evaluated on a dataset with 200 app descriptions and 1,108,148 user reviews. The experimental results reveal that KEFE is effective in both feature extraction and user review matching, achieving average *F-measure*s of 78.13% and 62.02%, respectively. The key features identified could also provide useful information for developers' consideration, because for the releases that receive higher app ratings, key features account for about 70% of all features improvements.

Acknowledgement

This work is supported in part by the National Key Research and Development Program of China (No. 2018YFB1003800), National Natural Science Foundation of China (No. 61902174, No. 62072226), and Natural Science Foundation of Jiangsu Province (No. BK20190291).

References

[1] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telematics and informatics*, vol. 28, no. 1, pp. 22–31, 2011.

[2] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora, "Mobile app development and management: results from a qualitative investigation," in *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, 2017, pp. 133–143.

[3] A. AlSubaihin, F. Sarro, S. Black, L. Capra, and M. Harman, "App store effects on software engineering practices," *IEEE Transactions on Software Engineering*, to be published, doi: 10.1109/TSE.2019.2891715.

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

[5] F. Palomba, M. Linares-Vasquez, 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 (ICSME)*, 2015, pp. 291–300.

[6] 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. 43, no. 9, pp. 817–847, 2016.

[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 (ICSE)*, 2014, pp. 767–778.

[8] C. Gao, H. Xu, J. Hu, and Y. Zhou, "AR-Tracker: Track the dynamics of mobile apps via user review mining," in *Proceedings of the 2015 Symposium on Service-Oriented System Engineering*, 2015, pp. 284–290.

[9] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *Journal of Systems and Software*, vol. 125, pp. 207–219, 2017.

[10] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach," in *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, 2015, pp. 749–759.

[11] C. Gao, B. Wang, P. He, J. Zhu, Y. Zhou, and M. R. Lyu, "PAID: Prioritizing app issues for developers by tracking user reviews over versions," in *Proceedings of the 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 35–45.

[12] 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 24th International Symposium on Foundations of Software Engineering (FSE)*, 2016, pp. 499–510.

[13] 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 (ICSE)*, 2017, pp. 106–117.

[14] W. Luiz, F. Viegas, R. Alencar, F. Mourão, T. Salles, D. Carvalho, M. A. Gonçalves, and L. Rocha, "A feature-oriented sentiment rating for mobile app reviews," in *Proceedings of the World Wide Web Conference (WWW)*, 2018, pp. 1909–1918.

[15] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proceedings of the 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 48–58.

[16] C. Gao, J. Zeng, D. Lo, C.-Y. Lin, M. R. Lyu, and I. King, "Infar: Insight extraction from app reviews," in *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2018, pp. 904–907.

[17] C. Gao, W. Zheng, Y. Deng, D. Lo, J. Zeng, M. R. Lyu, and I. King, "Emerging app issue identification from user feedback: Experience on wechat," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 279–288.

[18] S. Hassan, C.-P. Bezemer, and A. E. Hassan, "Studying bad updates of top free-to-download apps in the google play store," *IEEE Transactions on Software Engineering*, vol. 46, no. 7, pp. 773–793, 2018.

[19] T. Zhang, J. Chen, X. Zhan, X. Luo, D. Lo, and H. Jiang, "Where2change: Change request localization for app reviews," *IEEE Transactions on Software Engineering*, to be published, doi: 10.1109/TSE.2019.2956941.

[20] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proceedings of the 23rd International Requirements Engineering Conference (RE)*, 2015, pp. 116–125.

[21] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution," in *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, 2015, pp. 771–776.

[22] 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 24th International Symposium on Foundations of Software Engineering (FSE)*, 2016, pp. 1023–1027.

[23] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H. C. 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 (SANER)*, 2017, pp. 91–102.

[24] 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, pp. 1067–1106, 2016.

[25] S. Scalabrino, G. Bavota, B. Russo, M. Di Penta, and R. Oliveto, "Listening to the crowd for the release planning of mobile apps," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 68–86, 2019.

[26] H. Malik, E. M. Shakshuki, and W.-S. Yoo, "Comparing mobile apps by identifying 'hot' features," *Future Generation Computer Systems*, vol. 107, pp. 659–669, 2020.

[27] H. Jiang, J. Zhang, X. Li, Z. Ren, D. Lo, X. Wu, and Z. Luo, "Recommending new features from mobile app descriptions," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 4, pp. 1–29, 2019.

[28] 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 Requirements Engineering Conference (RE)*, 2014, pp. 153–162.

[29] X. Gu and S. Kim, "What parts of your apps are loved by users?" in *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, 2015, pp. 760–770.

[30] T. Iqbal, N. Seyff, and D. Mendez, "Generating requirements out of thin air: Towards automated feature identification for new apps," in *International Requirements Engineering Conference Workshops (REW)*, 2019, pp. 193–199.

[31] T. Johann, C. Stanik, W. Maalej *et al.*, "SAFE: A simple approach for feature extraction from app descriptions and app reviews," in *Proceedings of the 25th International Requirements Engineering Conference (RE)*, 2017, pp. 21–30.

[32] K.-F. Wong, W. Li, R. Xu, and Z.-S. Zhang, "Introduction to chinese natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 2, no. 1, pp. 1–148, 2009.

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

[34] 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 Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, 2014.

[35] S. Keertipati, B. T. R. Savarimuthu, and S. A. Licorish, "Approaches for prioritizing feature improvements extracted from app reviews," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 2016, pp. 1–6.

[36] S. A. Licorish, B. T. R. Savarimuthu, and S. Keertipati, "Attributes that predict which features to fix: Lessons for app store mining," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 108–117.

[37] E. Noei, F. Zhang, and Y. Zou, "Too many user-reviews, what should app developers look at first?" *IEEE Transactions on Software Engineering*, to be published, doi: 10.1109/TSE.2019.2893171.

[38] W. Che, Z. Li, and T. Liu, "Ltp: A chinese language technology platform," in *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, 2010, pp. 13–16.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[40] J. Foerderer, T. Kude, S. Mithas, and A. Heinzl, "Does platform owner's entry crowd out innovation? Evidence from google photos," *Information Systems Research*, vol. 29, no. 2, pp. 444–460, 2018.

[41] N. Jha and A. Mahmoud, "Mining non-functional requirements from app store reviews," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3659–3695, 2019.