Zhang L, Huang XY, Jiang J *et al.* CSLabel: An approach for labelling mobile app reviews. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(6): 1076–1089 Nov. 2017. DOI 10.1007/s11390-017-1784-1

CSLabel: An Approach for Labelling Mobile App Reviews

Li Zhang, *Senior Member*, *CCF*, *Member*, *IEEE*, Xin-Yue Huang, Jing Jiang*, *Member*, *CCF*, and Ya-Kun Hu

State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China

E-mail: lily@buaa.edu.cn; huangxinyue_buaa@163.com; jiangjing@buaa.edu.cn; 2900228779@qq.com

Received April 20, 2017; revised September 15, 2017.

Abstract Mobile apps (applications) have become a popular form of software, and the app reviews by users have become an important feedback resource. Users may raise some issues in their reviews when they use apps, such as a functional bug, a network lag, or a request for a feature. Understanding these issues can help developers to focus on users' concerns, and help users to evaluate similar apps for download or purchase. However, we do not know which types of issues are raised in a review. Moreover, the amount of user reviews is huge and the nature of the reviews' text is unstructured and informal. In this paper, we analyze 3 902 user reviews from 11 mobile apps in a Chinese app store — 360 Mobile Assistant, and uncover 17 issue types. Then, we propose an approach CSLabel that can label user reviews based on the raised issue types. CSLabel uses a cost-sensitive learning method to mitigate the effects of the imbalanced data, and optimizes the setting of the support vector machine (SVM) classifier's kernel function. Results show that CSLabel can correctly label reviews with the precision of 66.5%, the recall of 69.8%, and the F_1 measure of 69.8%. In comparison with the state-of-the-art approach, CSLabel improves the precision by 14%, the recall by 30%, the F_1 measure by 22%. Finally, we apply our approach to two real scenarios: 1) we provide an overview of 1 076 786 user reviews from 1 100 apps in the 360 Mobile Assistant and 2) we find that some issue types have a negative correlation with users' evaluation of apps.

Keywords mobile app, user review, classification

1 Introduction

As a new form of software, mobile application (commonly known as app) has become a fashion all over the world. The mobile application distribution platforms (Apple App Store, Google Play Store, etc.) allow users to easily search, purchase, and install apps. The download numbers of these platforms are astronomic, with around 1 billion downloads per month in Apple App Store[1]. Also, these platforms allow users to submit feedback for downloaded applications, and ratings and reviews are publicly visible. The valuable information in user reviews can help developers to discover the issues related to their apps, such as bug reports, feature requests[2], and help users to evaluate whether an app is worth downloading or purchasing.

Considering the huge amount of user reviews and the unstructured nature of reviews, manual inspection is time-consuming and inefficient. Therefore, it is necessary to first investigate which types of issues are raised in a review, and then, an automated approach is needed to label user reviews with issue types they contain.

There are already literatures focusing on this problem, i.e., how to automatically classify or label the user reviews based on the raised issue types[3-5]. However, there are limitations in current approaches. First of all, the previous studies are aimed at English reviews and the U.S. app stores, while there is no empirical study based on Chinese reviews and Chinese app stores. Whether the issue types are suitable for Chinese apps or not is unknown. Second, in McIlroy *et al.*'s study[5], the percentages of some issue types are low, e.g., the percentage of the uninteresting content is 0.7 in the reviews from Apple App Store. This resulted in the imbalanced data during the classification process. The imbalanced data reduces the performance of the classi-

Regular Paper

Special Section on Software Systems 2017

This work is supported by the National Natural Science Foundation of China under Grant No. 61672078, and the State Key Laboratory of Software Development Environment of China under Grant No. SKLSDE-2017ZX-06.

*Corresponding Author

fication of user reviews.

To solve problems mentioned above, we first manually label 3 902 user reviews in a Chinese app store, i.e., the 360 Mobile Assistant, and uncover 17 types of issues in the app store. Then, we present an approach called CSLabel that utilizes the cost-sensitive learning and support vector machine (SVM) to label user reviews. The approach helps the developers to gain an overview of the users' feedback, provides an overview of an app store in China, and identifies the issue types that have negative correlation with users' evaluation of apps. Also, CSLabel uses a cost-sensitive learning method to mitigate the effects of the imbalanced data, optimizes the kernel function of the SVM classifier, and achieves better results than McIlroy *et al.*'s multi-label approach[5]. We perform a study to answer the following three research questions.

*RQ*1: *What are the issue types presented in Chinese app stores?* We identify 17 types of issues, such as feature request, functional complaints, and so on.

*RQ*2: *How well can we automatically label user reviews?* Our approach CSLabel can correctly label user reviews based on the 17 issue types with the precision of 66.5%, the recall of 69.8%, and the F_1 measure of 68.1%. Compared with the results using the multi-label approach[5], our approach increases the F_1 measure by 22%.

*RQ*3: *Is CSLabel useful for developers?* We apply CSLabel on an app store in China, get an overview of the app store, and find that some issue types have a negative correlation with users' evaluation of apps.

This paper is organized as follows. Section 2 presents the process of uncovering issue types. Section 3 presents our approach. Section 4 presents the application of our approach on two real scenarios. Section 5 discusses the practical implications, the data source differences, and the threats to the validity of our study. Section 6 presents the related work. Section 7 concludes our work.

2 Preliminary Study

In this section, we try to answer what issue types are presented in Chinese app stores (RQ1).

User reviews contain the issues that users are experiencing, such as a crashing, a feature request, and/or a network problem. Understanding these issues can help developers to better understand users' requirements. In this section we explore the types of issues presented in a Chinese app store. McIlroy *et al.* already uncovered 14 types of issues in user reviews[5]. Based on their work, we analyze the data in a Chinese app store, and conclude that there are 17 types of issues in the app store (listed in Subsection 2.3).

We introduce how our datasets are collected, how we identify issue types from user reviews, and how we manually label the reviews below.

Our process to answer RQ1 is divided into several subsections: background, data collection, manual inspection, and results.

2.1 Background

In this subsection we provide a brief background about the app store we study and introduce the reasons why we choose to focus on average and bad reviews.

We choose the 360 Mobile Assistant①, one of the most popular app stores in China. The 360 Mobile Assistant is an Android application distribution platform run by Qihoo 360. Until February 2015, the total number of users in this app store exceeded 800 million. The cumulative downloads of applications reached 64 billion times, and the average daily distribution reached 180 million times②.

Apps distributed in this store are all free. Users who download an app can review it. A review in this app store contains a date, a rating (which can be good, average, or bad), and a comment text.

Previous studies make the assumption that in the Apple App Store, one-star and two-star reviews are indicative of negative issues[5-7]. Following their work, we inspect only the reviews which were rated as average or bad in the 360 Mobile Assistant.

2.2 Data Collection

Due to the huge amount of apps in the app store, we need to select a portion of apps for manual study.

In the 360 Mobile Assistant, the apps belong to different categories, e.g., shopping, news and reading, according to the services they provide. Based on the overall ranking offered by the platform, we select the top-ranked app of each category until July 3, 2016. Then, we remove the apps of which the total number of average and bad reviews is less than 100. Finally, we get 11 apps: TaoBao (category: shopping),

①http://zhushou.360.cn/, Sept. 2017.

②http://www.cctime.com/html/2015-8-14/2015814115221654.htm, Oct. 2017.

Meitu Xiuxiu (category: photography and video), JinRiTouTiao (category: news and reading), Ali Pay (category: finance), Wechat (category: communication and social), Youku (category: music and video), 360 Defender (category: system and security), 360 ZhuoMian (category: theme and wallpaper), 360 YunPan (category: office and business), DiDiChuXin (category: maps and travel), and ZuoYeBang (category: education).

After determining the apps to be analyzed, we build a simple web crawler to automatically download the user reviews of these apps from the website of the 360 Mobile Assistant. In total, we download 128 697 average or bad user reviews for the 11 apps from the 360 Mobile Assistant.

Due to the high cost of manually checking all these reviews, we study a statistical representative sample of these reviews③. This sample of reviews is randomly chosen to achieve a 95% confidence level and a 5% confidence interval. For example, "Taobao" has a total of 12 594 average and bad reviews. The statistically representative sample for 12 594 reviews is 373 reviews, which is calculated to achieve a 95% confidence level and a 5% confidence interval.

Finally, we randomly select 3 902 reviews from the 128 697 average and bad reviews.

2.3 Issue Types

After getting the statistically representative sample set (3 902 reviews), we follow a process called coding as suggested by Seaman *et al.* to identify the issue types in user reviews[8-9]. This process is used to extract values for quantitative variables from qualitative data in an iterative way.

At the beginning, we choose the issue type set defined by McIlroy *et al.* as the starting set[5]. For each review, the first author manually inspects it and labels it with the issue types that the review points out. If an issue in the review is not included in the issue type set, the first author will define a new issue type and add it to the issue type set. After that, the first author restarts the labelling process based on the new issue type set. The second author and the third author also take this labelling process independently. When the three authors finish the labelling process, we discuss together to determine the final issue type set.

We end up with 17 issue types and they are shown in Table 1.

This issue type set differs from the issue type set presented in McIlroy *et al.*'s paper[5] in the following ways.

1) The uninteresting content is changed into the content complaint. The uninteresting content includes the reviews of which the specific content is unappealing. However, there are some reviews complaining the lack of content. For example, a user wrote, "I can't find the TV series and films I want to see." Therefore we merge these two kinds of complaint on the content of app into the content complaint.

Table 1. Issue Types and Descriptions

Issue Type	Description
Additional cost	The user complains about the hidden cost to enjoy the full experience of the app
Compatibility issue	App has problems on a specific device or an OS version
Content complaint	Specific content is unappealing or complains about the lack of content
Crashing	App is often crashing
Feature removal	One or more specific features are ruining the app
Feature request	App needs additional feature(s)
Functional complaint	An unexpected behavior or failure occurs
Installation issue	A failure occurs in the installation process
Network connection issue	App has network connection problem, such as network lag
Other	The review is not useful or does not point out the problem
Privacy and ethical issue	App invades privacy or is unethical
Property safety	App has threatened the user's property safety
Resource heavy	App consumes too much battery or memory
Response time	App is slow to respond to input, or is laggy overall
Traffic wasting	App takes more network traffic than the user expects
Update issue	User blames an update for introducing new problems
User interface	User complains about the design, controls or visuals

③http://www.raosoft.com/samplesize.html, Sept. 2017.

2) The installation issue is added as a new type. Some users cannot successfully install the app. For example, a user wrote, "Why cannot I update to the new version? Every time it failed to install." Thus we add this type to receive the specific complaint about the problems occurred in the process of installing apps.

3) The property safety is added as a new type. In China, credit card payment is not so popular as that in the United States. The electronic payment has developed rapidly and becomes a very popular payment method. Users who use electronic payments may encounter property safety issues. For example, a user wrote, "No gesture password, I should not be updated, the password of the phone can be changed casually and my funds are not guaranteed." Therefore we add this type to receive the specific complaint about the property safety.

4) The network problem is changed into the network connection issue, and the traffic wasting is added as a new type. The public free WiFi is not so popular in China and the charge of cellular network is relatively expensive. Therefore, the Chinese users care about the smartphone traffic that the app costs. For example, a user wrote, "Compared to other apps, this app costs more traffic." Accordingly we extract the traffic wasting issue out from the network problem issue, as a new type. According to the data we analyze, except the traffic wasting issue, the network problems are mainly the complaints about the network connection issue. Therefore, we change the network problem into the network connection issue, and if there are other network problems, they will be labelled as Other.

So far, we get the issue types presented in the Chinese app store.

Moreover, we sample 1 000 reviews of other 20 apps to check whether there are other issue types.

We crawl the ratings of 31 849 apps in the 360 Mobile Assistant, and calculate the average rating. The average rating is 7.05. We randomly select 10 apps with a rating higher than the average rating, and 10 apps with a rating lower than the average rating. These 20 apps include both popular apps and unpopular apps. These apps are: Wanneng Wifi Password, DingDing, HongYanChuanShu, AcFun, Faceu, HuDongZuoYe, JuHuaSuan, 8684GongJiao, LOFTER, JiaoTong Bank, LuDaShi, MeiZhuang Camera, BiLin, MX Player, DuoKanYueDu, AnZhuoBiZhi, YouDaoYunBiJi, DangDang, GaoDe Map, MeiRiYingYuTingLi. We download the average or bad user reviews for these 20 apps, and then randomly select 50 reviews from each app.

The first, second and third author independently inspect each review and label it with the issue types that the review points out. If an issue in the review is not included in the current issue type set, any of the authors will define a new issue type and add it to the issue type set.

After labeling the 1 000 reviews, we find that there is no new issue type. It indicates that the 17 issue types are sufficient for labelling mobile app reviews.

2.4 Reviews Labelling

After getting the issue types, the first and second author independently label the 3 902 reviews based on the 17 issue types. The differences in the labelling results are merged after we have reached a consensus. This process can make sure that each review is inspected by two authors so that the occasional mistake could be reduced.

Also, we take a measurement on how reliable the labels are. Intra class correlation coefficient (ICC)[10] is a piece of statistics which describes how strongly units in the same group resemble one another. The common guidelines for ICC are as follows: if ICC is less than 0.4, it means the correlation is poor; if ICC is between 0.40 and 0.59, it means the correlation is fair; if ICC is between 0.60 and 0.74, it means the correlation is good; if ICC is between 0.75 and 1.00, it means the correlation is excellent.

We calculate the ICC between the first author and the second author to measure the reliability of manual labelling. The results are shown in Table 2. The initial ICCs are calculated after the authors independently label the reviews, and for all the issue types, the ICCs are good or excellent. The final ICCs are calculated after the authors discuss the differences, and the ICCs in all issue types are 1. The two authors reach 100% final agreement for all the reviews.

The result of manual labelling is shown in Table 3. We notice that the percentages of some issues are fairly low, e.g., the percentage of the additional cost is 0.82, and the percentage of the traffic wasting is 0.41. When we train the multi-label classifier, we use the labelled samples. The positive samples belong to the category, while the negative samples do not belong to the category. For the issue types with low percentages, the number of positive samples differs greatly from the number of negative samples, e.g., for the additional cost, 0.82% of the samples are positive, and 99.18% of samples are negative. In the subsequent classifica-

tion process, measures need to be taken to mitigate the impact of this imbalanced distribution of data.

Table 2. Intra Class Correlation Coefficient

Issue Type	Initial ICC	Final ICC
Additional cost	1.00	1
Compatibility issue	0.72	1
Content complaint	0.80	1
Crashing	0.91	1
Feature removal	0.89	1
Feature request	0.75	1
Functional complaint	0.89	1
Installation issue	0.95	1
Network connection issue	0.79	1
Other	0.93	1
Privacy and ethical issue	1.00	1
Property safety	1.00	1
Resource heavy	1.00	1
Response time	1.00	1
Traffic wasting	0.80	1
Update issue	0.91	1
User interface	0.86	1

Table 3. Percentages of Issue Types

Issue Type	Percentage
Additional cost	0.82
Compatibility issue	2.07
Content complaint	2.84
Crashing	14.96
Feature removal	6.58
Feature request	8.32
Functional complaint	15.45
Installation issue	1.15
Network connection issue	4.07
Other	34.21
Privacy and ethical issue	0.53
Property safety	0.94
Resource heavy	3.17
Response time	4.86
Traffic wasting	0.41
Update issue	17.73
User interface	1.28

3 CSLabel: An Automatic Labelling Approach

In this section, we try to answer how well we can automatically label the user reviews (RQ2). We propose CSLabel for multi-label classification of user reviews. Fig.1 presents the overall framework of our work.

Our entire process contains several parts: feature extraction, model building, evaluation, and results.

In the feature extraction phase, our goal is to extract the features of reviews' texts (Subsection 3.1). In the model building phase, our goal is to build a multi-label classification model from review features (Subsection 3.2). In order to mitigate the effects of the imbalanced data, we use CostSensitive Classifier[11]. And we optimize the setting of the SVM classifier's kernel function. We introduce the evaluation measures in Subsection 3.3 and present our results in Subsection 3.4.

3.1 Feature Extraction

Since the text is unstructured data, it must be converted into a construable form for the computer first.

Vector Space Model (VSM) is a text representation model suitable for large-scale corpus[12]. In this model, the text space is regarded as a vector space composed of a set of orthogonal eigenvectors. Each dimension of the vector corresponds to a feature in the text, and each dimension itself represents the weight of the corresponding feature in the text. The model can be described by the following formula:

$$\begin{aligned} &\boldsymbol{vector}(d_i) \\ &= (weight_1(d_i), weight_2(d_i), ..., weight_n(d_i)), \end{aligned}$$

where n is the number of feature items selected for feature extraction and $weight_j(d_i)$ is the weight (relevance and importance) of the j-th text feature in the document d_i.

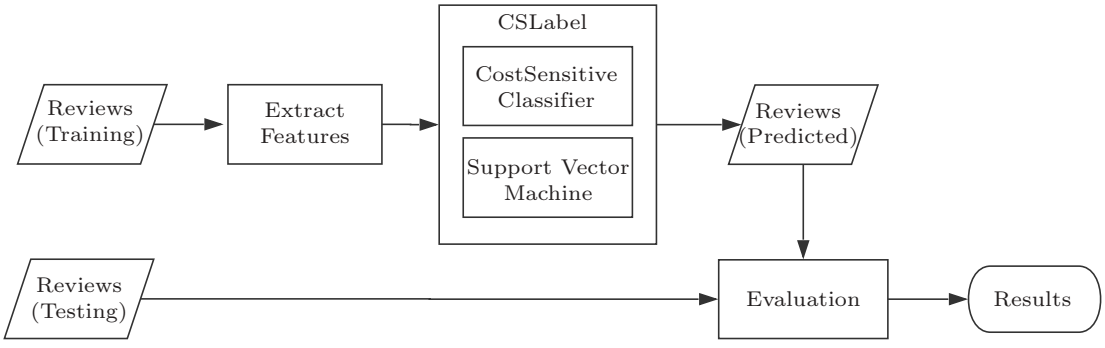

Fig.1. Overall framework of CSLabel.

For English text, a word is a feature. However, Chinese sentences need to be segmented to words first, using the Chinese text segmentation technology. Jieba④ Chinese text segmentation is a Python text segmentation component. We use this component to segment the review text, and we remove single numbers and non-Chinese characters.

Following previous work[5], we retain the stop words because some of them are meaningful to determine the issue type, such as "buyao"(means "do not" or "do not need/want"), and filter words that appear less than three times in our dataset. Such filtering eliminates rare words (spelling mistakes or unimportant words), and reduces the complexity of classification.

As for the weights of features, tf-idf (term frequency-inverse document frequency) algorithm is a commonly used method to calculate the weights[13]. The main idea of the tf-idf is that if a word or phrase appears in a document with a high term frequency and is rare in the other documents, the word or phrase is considered to have a good ability to classify.

Above all, in order to build feature vectors, we use the String To Word Vector filter, which implements the tf-idf algorithm. It is available in WEKA[14].

3.2 Model Building

In this subsection, an introduction of our classification approach is provided.

3.2.1 Multi-Label Classification

In our case, a review may contain multiple issue types, thereby the problem we need to solve is actually a multi-label classification. Binary Relevance (BR) is one of the representative algorithms for tackling the multi-label classification problems, which transforms the multi-label classification problem into the multiple binary classification problem. We choose BR because it is simple with linear complexity[15]. This means we need to build multiple classifiers, and the overall evaluation is also needed.

3.2.2 Cost-Sensitive Learning

As shown in Table 3 in Subsection 2.3, we notice that the percentages of issue types are different. The percentages of some issues are fairly low, e.g., the percentage of additional cost is 0.82, and the percentage of traffic wasting is 0.41. For these issues, the number of negative samples is much larger than that of positive samples. These imbalanced data may lead to a tendency for the classifier to predict new samples as negative samples.

In this paper, we use cost-sensitive learning[16] to deal with this problem.

The core of the cost-sensitive learning method is the cost matrix. In practical applications, the costs of different types of misclassification are not the same.

The cost matrix is defined as shown in Table 4.

Table 4. Cost Matrix

	True = 0	True = 1
Predict = 0	C_{00}	C_{01}
Predict = 1	C_{10}	C_{11}

C_{ij} is the cost of misclassifying class j into category i. Obviously, $C_{00} = C_{11} = 0$, and C_{01}, C_{10} are two different misclassification costs.

Since our data is imbalanced, we can reweight it based on different misclassification costs. When the cost of predicting a positive sample into a negative sample is high, we increase the weight of positive samples.

We use a meta-classifier to make the base classifier cost-sensitive. This meta-classifier supports increasing the weight of examples, which is fully equivalent to oversampling the minority class. It differs from resampling the minority class. For example, if resampling is done by randomly duplicating instances in the minority class, we may have several occurrences of an instance and just one occurrence of a different instance. Using costs enforces each sample to be replicated the same times. Also, it can avoid downsampling the majority class, and thus take full advantage of the available data. The meta-classifier is implemented in WEKA as the CostSensitive Classifier[11].

In addition, we need to specify the cost matrix for each issue type. In order to determine the specific value of the cost matrix, for each issue type, we traverse to get the cost matrix value which gives the best classification result. Then, we apply the best cost matrix value of each issue type to the training process.

3.2.3 Support Vector Machine

A support vector machine is a discriminative classifier formally defined by a separating hyperplane. When given labelled training data, the support vector machine outputs an optimal hyperplane which can classify new examples.

④https://github.com/fxsjy/jieba, Sept. 2017.

In previous studies, we find that SVMs perform well in text classification[17-18].

For SVMs, a data point is viewed as a p-dimensional vector. If an SVM separates such points with a $(p-1)$-dimensional hyperplane, it is called a linear classifier. In order to deal with the situation that the original problem is not linearly separable in a finite-dimensional space, SVMs use kernel functions to map the original finite-dimensional space into high-dimensional spaces, e.g., radial basis function kernel. In case of the number of samples is small and the number of feature dimensions is very big, the map of non-linear classification is usually inaccurate and it is possible to erroneously divide the feature space, which may result in worse results than the linear model.

Therefore, we choose to use a linear SVM in our approach, that is, the WEKA's SVM implementation, SMO Classifier[11,19]. We use the PolyKernel with exponent set to 1, in order to make the model a linear support vector machine. Besides, we keep other parameters by default. The former method multi-label approach[5] uses the default setting of the LibSVM classifier in MEKA⑤, that is, it uses the radial basis function kernel (non-linear classification). The result compared with that of the multi-label approach is presented in Subsection 3.4.

3.3 Evaluation

3.3.1 Cross-Validation

Cross-validation is a commonly used technique to derive a more accurate estimate of prediction performance[20]. We perform a 10-fold cross-validation for our approach CSLabel and the multi-label approach on our dataset. The dataset is divided into 10 groups (i.e., folds), nine of which will be taken as training data and one as testing data. Each run of experiment produces a corresponding result. The average of the results of 10 times is used as an estimate of the result for our approach CSLabel and the multi-label approach.

3.3.2 Evaluation Measures

For the binary classification, the commonly used evaluation measures are precision, recall, and F_1 measure. Usually the class of interest is positive, and the other is negative. There are four cases of the classifier's prediction, and the total numbers of four cases are recorded as: TP: the number of positive samples predicted to be positive; FN: the number of positive samples predicted to be negative; FP: the number of negative samples predicted to be positive; TN: the number of negative samples predicted to be negative.

Precision is defined as

$$P = \frac{TP}{TP + FP}.$$

Recall is defined as

$$R = \frac{TP}{TP + FN}.$$

F_1 measure is the harmonic mean of precision and recall, namely:

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R},$$
$$F_1 = \frac{2TP}{2TP + FP + FN}.$$

In this paper, we build a classifier for each issue type, that is, we build 17 classifiers in total. Therefore it is necessary to conduct an overall evaluation of the classifiers. There are different evaluation measures used for multi-labelled data[5,15]. Micro-average is a commonly used measure, and it is the arithmetic average of the performance metrics for each sample.

If the number of classes is denoted by k, then:

micro-precision is defined as

$$\frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} TP_k + \sum_{k=1}^{K} FP_k};$$

micro-recall is defined as

$$\frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} TP_k + \sum_{k=1}^{K} FN_k};$$

micro-F_1 is defined as

$$\frac{2 \times \text{micro-precision} \times \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}}.$$

3.4 Results

We apply our approach CSLabel and the multi-label approach to our dataset. The multi-label approach uses the Proportional Cut Threshold algorithm and SVM classifier, which are implemented in MEKA[5]. The results are presented in Table 5 (best results in bold).

Table 5. Evaluations of Multi-Label Approach and CSLabel

Evaluation Measure	Multi-Label Approach	CSLabel
Micro-precision	0.583	**0.665**
Micro-recall	0.535	**0.698**
Micro-F_1	0.558	**0.681**

⑤http://meka.sourceforge.net/, Sept. 2017.

It can be found that CSLabel achieves 66.5% micro-precision, 69.8% micro-recall, 68.1% micro-F_1. In comparison with the multi-label approach[5], CSLabel has a 14% increase in micro-precision, a 30% increase in micro-recall, and a 22% increase in micro-F_1. The result proves that our work takes an effective way to improve the results.

The results of each issue type compared with those of the multi-label approach[5] are shown in Table 6. For most of the issue types, CSLabel performs better than the multi-label approach. For example, for installation issue, CSLabel improves the F_1 measure from 11.6% to 69.6%, and for update issue, CSLabel improves the F_1 measure from 31.3% to 82.3%.

Table 6. F_1 Measure of Multi-Label Approach and CSLabel for 17 Issue Types

Issue Type	Multi-Label Approach	CSLabel
Additional cost	0.327	0.571
Compatibility issue	0.116	0.427
Content complaint	0.528	0.479
Crashing	0.739	0.807
Feature removal	0.528	0.627
Feature request	0.313	0.478
Functional complaint	0.443	0.552
Installation issue	0.116	0.696
Network connection issue	0.498	0.634
Other	0.674	0.724
Privacy and ethical issue	0.359	0.526
Property safety	0.443	0.507
Resource heavy	0.700	0.717
Response time	0.518	0.657
Traffic wasting	0.739	0.533
Update issue	0.313	0.823
User interface	0.327	0.458

4 Applications of Proposed Approach

In this section, we try to answer whether CSLabel is useful for developers (RQ3). We apply our approach to reviews from a wider range of apps, and study the implications of the application to stakeholders. More specifically, we conduct an empirical study on the reviews of each category in the 360 Mobile Assistant.

Based on the distribution of issue types in the app store, stakeholders would be able to observe the issue types with the most user complaints. Also, stakeholders could investigate which issue types have negative correlation with users' evaluation of the apps.

4.1 Data Collection

We select the 100 most downloaded apps per category from 11 categories in the 360 Mobile Assistant to study. The 11 categories are: shopping, photography and video, news and reading, finance, communication and social, music and video, system and security, theme and wallpaper, office and business, maps and travel, and education. These categories are what the 11 apps mentioned in Subsection 2.2 belong to.

In order to assess the impact of the issues on the apps, we first establish a score to evaluate the application. It is related to the number of good reviews, average reviews and bad reviews of the app. Each good review has a weight α, each average review has a weight β, and each bad review has a weight γ. Then, the score of an app is:

$$\begin{aligned}&(\text{the number of the good reviews} \times \alpha + \\ &\text{the number of the average reviews} \times \beta + \\ &\text{the number of the bad reviews} \times \gamma) / \\ &\text{the number of all the reviews.}\end{aligned}$$

We try several combinations of the values of α, β, and γ, and the results are similar. The results presented in Subsection 4.2 are calculated based on values of $\alpha = 10$, $\beta = 5$, and $\gamma = 1$.

After calculating the scores of 100 apps of each category, we crawl the reviews of each app. However, due to the restrictions on the crawler in 360 Mobile Assistant, we can only crawl some of the latest reviews. At last, we retain only 1 076 786 average and bad user reviews. Then we follow the same feature extraction steps presented in Subsection 3.1.

4.2 Results

We run CSLabel on the 1 076 786 user reviews to gain the distribution of issue types of each category in the app store. For each category, we calculate the percentage of each issue type for the 100 apps respectively, and then take the average percentage as the result.

Table 7 shows the issue type distributions for the 360 Mobile Assistant. It can be found that the issue type distribution of each category is similar.

The percentages of some issues are high and the percentages of others are low. For example, the update issue has a higher percentage than the others. It can be concluded that the users are used to blaming the update for introducing new problems, thereby developers

Table 7. Issue Type Distributions for 11 Categories

App Type	C&S (%)	N&R (%)	M&V (%)	S&S (%)	Shopping (%)	M&T (%)	O&B (%)	Finance (%)	T&W (%)	P&V (%)	Edu. (%)
Additional cost	0.53	0.42	0.48	0.50	0.32	0.55	0.70	0.46	0.31	0.91	0.54
Compatibility issue	0.48	0.38	0.67	0.51	0.42	0.89	0.42	0.52	0.28	0.25	0.27
Content issue	1.32	1.53	1.36	1.37	1.17	1.37	2.72	1.44	1.37	1.62	0.93
Crashing	7.33	9.29	7.72	6.87	7.79	7.54	8.96	9.48	5.55	7.21	8.91
Feature removal	7.92	6.65	8.47	7.29	7.01	6.47	8.70	6.37	9.98	7.19	6.53
Feature request	12.38	10.51	11.90	13.19	12.08	14.71	13.73	11.82	10.96	10.60	10.76
Functional complaint	11.18	11.43	9.83	11.60	10.35	11.19	11.52	13.64	12.62	10.34	9.54
Installation issue	0.52	0.51	0.40	0.62	0.70	0.86	0.58	0.70	0.46	0.43	0.36
Network connection issue	7.20	7.12	7.91	8.13	8.40	8.49	8.38	7.49	7.96	7.43	7.47
Other	14.15	15.18	14.87	12.68	17.83	15.17	14.55	20.09	11.04	12.61	13.46
Privacy and ethical issue	1.64	1.93	1.39	1.46	2.69	3.25	2.71	3.71	1.79	1.34	2.45
Property safety	0.19	0.05	0.05	0.12	0.06	0.07	0.09	0.07	0.04	0.06	0.04
Resource heavy	4.08	3.90	3.66	4.89	4.05	4.68	5.04	4.92	3.32	3.90	4.09
Response time	2.39	3.36	2.46	2.58	3.59	3.76	4.97	5.44	2.29	2.12	3.60
Traffic wasting	0.32	0.29	0.35	0.45	0.31	0.32	0.40	0.42	0.46	0.43	0.32
Update issue	71.97	70.98	72.49	73.35	69.34	69.33	68.04	63.56	76.71	75.43	74.96
User interface	0.39	2.05	0.34	0.64	0.27	0.42	0.39	0.37	0.31	0.32	0.97

Note: C&S: communication and social; N&R: news and reading; M&V: music and video; S&S: system and security; M&T: maps and travel; O&B: office and business; T&W: theme and wallpaper; P&V: photography and video; Edu.: education.

should be careful when they are ready to submit an update. In addition, providing more tips in the app about the new features may help users easier to use. The functional complaint, the feature request, the feature removal, and the crashing also have higher frequencies.

However, what is the correlation between the percentage of the issue type and the score of the app?

We calculate the Pearson correlation coefficient between the percentages of the issue types and apps' scores for each category, and the results are shown in Table 8.

The Pearson correlation coefficient is between -1 and 1. When the value is close to 1, there is a positive correlation between the percentage of each issue type and the app's score; when the value is close to -1, there is a negative correlation between the percentage of each issue type and the app's score. We can draw some conclusions from Table 8 as follows.

1) The additional cost, the feature removal, and the feature request have a negative correlation with the app's score. If developers want to improve the app's score, solving these three issues may help.

2) Though the percentage of update issue is high, it does not show a strong correlation with the score. In contrast, although the percentages of the additional cost, the feature removal, and the feature request are not so high as that of the update issue, they have a negative correlation with the score.

3) For some categories, the privacy and ethical issue has a positive correlation with the app's score, like communication and social, finance, and office and business. Furthermore, the percentage of the privacy and ethical issue is fairly low. We manually inspect some reviews that have privacy and ethical issue. Some users complain about that the app takes their privacy information to recommend them contents, but some other users may consider this feature is useful. Also, some users complain about that the app provides some unethical contents such as pornographic videos, but some other users may be curious about or want these contents. Thus, the features which a fraction of users complain about may be accepted or even enjoyed by some other users.

Through the above analysis, we can find that our approach can help stakeholders to find the most critical issues.

Table 8. Pearson Correlation Coefficient Between the Percentages of the Issue Types and Apps' Scores

App Type	C&S (%)	N&R (%)	M&V (%)	S&S (%)	Shopping (%)	M&T (%)	O&B (%)	Finance (%)	T&W (%)	P&V (%)	Edu. (%)
Additional cost	−0.41	−0.57	−0.26	−0.34	−0.53	−0.22	−0.20	−0.70	−0.29	−0.19	−0.13
Compatibility issue	−0.15	−0.23	−0.14	−0.11	−0.12	0.01	0.01	−0.27	−0.30	−0.07	−0.07
Content issue	−0.11	−0.14	−0.30	0.00	−0.28	−0.32	0.01	0.12	−0.12	−0.04	−0.20
Crashing	0.05	−0.02	0.09	−0.19	0.05	−0.03	−0.01	0.04	−0.06	−0.03	0.17
Feature removal	−0.28	−0.34	−0.45	−0.29	−0.31	−0.45	−0.20	−0.49	−0.25	−0.50	−0.37
Feature request	−0.24	−0.42	−0.35	−0.32	−0.27	−0.41	−0.33	−0.40	−0.32	−0.53	−0.26
Functional complaint	−0.02	−0.06	0.00	−0.03	0.10	0.09	−0.01	0.24	−0.06	−0.25	0.06
Installation Issue	−0.14	−0.03	−0.21	−0.01	0.13	0.10	0.16	0.04	0.08	−0.16	0.00
Network connection Issue	−0.15	−0.40	−0.24	−0.35	−0.18	−0.25	−0.01	−0.39	−0.46	−0.45	−0.22
Other	−0.12	0.09	0.05	−0.07	0.01	0.13	0.35	0.24	−0.27	−0.01	0.17
Privacy and ethical issue	0.40	0.27	0.20	0.18	−0.03	0.35	0.44	0.42	0.14	0.06	0.23
Property safety	−0.07	0.15	−0.15	−0.07	0.16	−0.25	0.06	0.00	0.08	−0.03	0.08
Resource heavy	−0.30	−0.17	0.07	−0.03	−0.24	−0.13	−0.11	−0.29	0.07	−0.33	−0.13
Response time	0.07	0.08	−0.11	−0.06	0.07	0.04	0.15	0.11	−0.05	−0.08	0.15
Traffic wasting	−0.23	−0.07	−0.32	−0.20	−0.29	0.03	−0.03	−0.44	−0.03	−0.37	−0.07
Update issue	0.17	0.08	0.20	0.06	0.11	0.11	−0.22	−0.08	0.29	0.33	−0.04
User interface	−0.22	0.10	−0.16	−0.22	−0.41	−0.16	−0.20	−0.55	−0.02	−0.28	0.09

Note: C&S: communication and social; N&R: news and reading; M&V: music and video; S&S: system and security; M&T: maps and travel; O&B: office and business; T&W: theme and wallpaper; P&V: photography and video; Edu.: education.

5 Discussion

5.1 Practical Implications

CSLabel can provide stakeholders an overview of the issues in the user reviews, and can help stakeholders to find the correlation between the issue types and the users' evaluation of the app.

When applying CSLabel to a real scene, some manual settings and choices may affect the effect of the approach. In our experiments, the issue types we set out are discovered from the reviews of the entire app store. If a stakeholder does not concern about certain issues, he or she can simply remove them from the issue type set. Or, a stakeholder has a unique problem, for example, users of shopping platform apps like "TaoBao" may have complaints on third party business, and then the stakeholder can add these issues to the issue type set.

5.2 Data Source Differences

The reviews used in Table 3 were crawled in July, 2016. At that time, the 360 Mobile Assistant did not place restrictions on crawlers, thereby we have crawled all the reviews of the 11 apps since they were first released. However, when we collected the reviews in Table 7 in March, 2017, the restrictions of the 360 Mobile Assistant have been placed on crawlers, and we can only crawl less than 2 000 latest reviews for each app.

This difference of data source affects the distribution of issue types.

For example, from Table 3, we can find that the percentage of the update issue is 17.73, but in Table 7, the percentages of 11 categories are between 63 and 77. We collect the ChangeLogs of the 11 apps of which the reviews are used in Table 3, and we count the number of updates in 2014∼2016. The results are shown in Fig.2 (360 ZhuoMian stopped updating from 2014; thus it is not shown in Fig.2). It can be found that the frequency of updates increases over time. For example, DiDiChuXing updated 9 times in 2014, 18 times in 2015, and 25 times in 2016. More updates bring more issues about update. In Table 7, we only get the latest reviews of apps. As a result, the percentage of the Update Issue is higher than that of reviews in Table 3.

5.3 Threats to Validity

As for any empirical study, there are some threats of validity to our work. In this subsection, we introduce these threats in detail, and how we mitigate their impact.

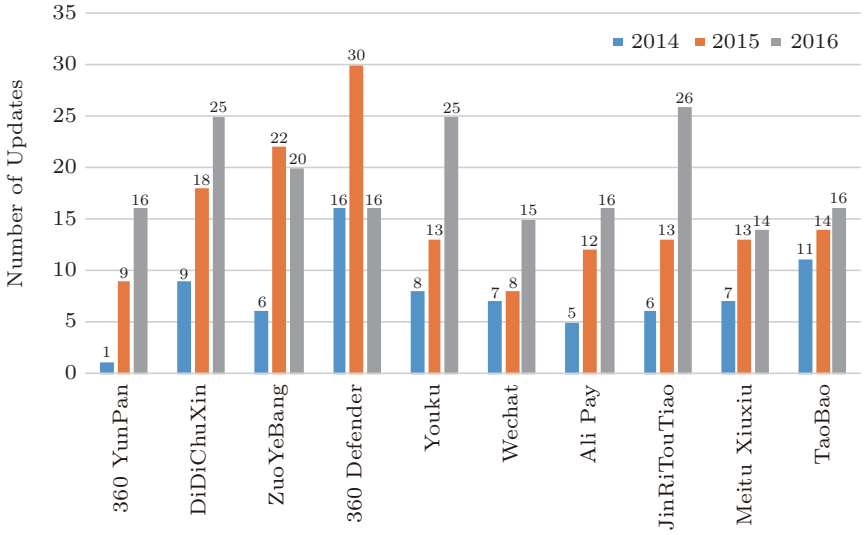

Fig.2. Number of updates during 2014∼2016.

5.3.1 *Threats to Construct Validity*

Since the way we label the reviews is manual, some reviews may not be properly labelled. To mitigate the effects of human error, the first author and the second author independently label the reviews, and they merge the differences in their labelling results. We calculate the intra class correlation coefficient to measure the reliability in Subsection 2.4. In the future, we plan to analyze more user reviews, and let at least three people to label each review.

The apps we selected cannot represent all apps, which may make our issue type set not adapt to each case, but our approach is independent of the selected issues.

5.3.2 *Threats to Internal Validity*

For the selection of SVM classifier, we refer to the previous studies of text categorization. SVM is a text classification tool with good performance. Also, SVM is known to be sensitive to hyper-parameters. We use the default values for hyper-parameters except for the PolyKernel with exponent set to 1. In the future, we plan to tune SVM to find the best performing parameters, and try more machine learning classification techniques. Ignoring good reviews may lead to the missing of some issues because there may also be complaints in good reviews. However, adding more issue types could not affect the availability of our classification approach. In the future, it is necessary to take more experiments to generalize our findings on all the reviews.

5.3.3 *Threats to External Validity*

Our dataset and empirical study are based on the 360 Mobile Assistant, and whether our results can be generalized to other app stores is unknown. In the future, we plan to study similar research questions based on other app stores, and compare the results with the findings from the 360 Mobile Assistant.

The training set of our reviews dataset includes 3 902 reviews from 11 applications. Though we select 11 applications from different categories, these 11 applications may still be different from other applications. It is unknown whether the 3 902 reviews are the representative of other mobile applications. In future work, we plan to manually label more reviews from more applications, and compare them with the 3 902 reviews. We will explore how the choice of applications in training set affects the performance of the classifier.

6 Related Work

We survey the related work in two major areas.

6.1 Mining Mobile User Feedback

Previous work shows that the user feedback in an app store has useful information for software improvements and evolution. Pagano and Brügge[2] manually surveyed user feedback in form of ratings and reviews in the Apple App Store in 2013. They found that the reviews typically contain multiple topics, such as user experience, bug reports, and feature requests, and negative feedback such as shortcomings is typically destruc-

tive. Harman *et al.*[21] studied the apps in the Blackberry app store in September 2011, and showed that there is a strong correlation between customer ratings and the rank of app downloads.

Due to the large number of user feedback, some researchers tried to use automatic methods to extract valuable information from user reviews. Di Sorbo *et al.*[22] introduced SURF (Summarizer of User Reviews Feedback), which uses sophisticated summarization techniques for summarizing thousands of reviews and generating an interactive, structured and condensed agenda of recommended software changes. Iacob and Harrison[23] designed MARA (Mobile App Review Analyzer), which is a prototype for automatic retrieval of mobile app feature requests from online reviews. Fu *et al.*[24] proposed WisCom, which is a system that can analyze user ratings and comments in mobile app stores. They used topic model to analyze apps' topic distribution, and found out the strongest complaints that users have. Carreño and Winbladh[25] used the opinion mining method ASUM[26] to extract the features of app store user reviews and compared it with the k-median method. Guzman and Maalej[27] used the NLTK toolkit[28] to extract features from the user reviews, used the SentiStrength tool[29] to detect the sentiment polarity of the users, and used the topic model (LDA)[30] to group the features and produce a high-level summary. Chen *et al.*[31] presented an approach AR-Miner to help developers find the most informative user reviews using: 1) text analysis and machine learning to filter out non-informative reviews and 2) topic analysis to recognize topics treated in the reviews classified as informative ones.

The above studies focused on the topics in the user reviews. Our work addresses a different problem. We find the types of issues raised in a review, which can help stakeholders to quickly understand what the users complain about.

6.2 Classification and Labelling of User Reviews

Maalej and Nabil[3] classified user reviews into four basic types: 1) bug reports; 2) feature requests; 3) user experiences; 4) ratings. These types include errors encountered by users in the use, the lack of functional requirements, and the experience of a particular scene, as well as praise or criticism. Panichella *et al.*[4] pointed out that the topic analysis techniques are useful for discovering topics in review texts, but they cannot reveal the users' intentions for reviews containing specific topics. Therefore, they considered the following four categories as the base categories according to users' intentions: information giving, information seeking, feature request, and problem discovery.

In the above two studies, the user reviews were classified into four categories. In our work, 17 types of issues are proposed. It is more precise, and we could obtain the user requirements more accurately.

McIlroy *et al.*[5] presented 14 types of issues for apps in Apple App Store and Google Play Store. The authors compared several machine learning classifiers. We apply our approach CSLabel and their approach on a same dataset. Compared with their approach, CSLabel increases the precision by 14%, the recall by 30%, and the F_1 measure by 22%.

7 Conclusions

In this paper, we manually analyzed 3 902 user reviews in a Chinese app store, 360 Mobile Assistant, and uncovered 17 issue types. Due to the huge amount and the unstructured nature of the data, we proposed CSLabel to label the user reviews based on the 17 issue types. CSLabel uses a cost-sensitive learning method and a linear SVM to predict the issue types of reviews. We evaluated CSLabel on 3 902 user reviews, and our approach can correctly label reviews with a precision of 66.5%, a recall of 69.8%, and a F_1 measure of 69.8%. In comparison with the multi-label approach, CSLabel increases the precision by 14%, the recall by 30%, and F_1 measure by 22%. Moreover, we applied CSLabel on 1 076 786 user reviews in the Chinese app store. Results helped stakeholders to get an overview of the app store and identify the issue types that have a negative correlation with users' evaluation of apps.

References

[1] Pagano D, Maalej W. User feedback in the Appstore: An empirical study. In *Proc. the 21st IEEE International Requirements Engineering Conference* (*RE*), July 2013, pp.125-134.

[2] Pagano D, Brügge B. User involvement in software evolution practice: A case study. In *Proc. the 35th International Conference on Software Engineering* (*ICSE*), May 2013, pp.953-962.

[3] Maalej W, Nabil H. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proc. the 23rd IEEE International Requirements Engineering Conference* (*RE*), Aug. 2015, pp.116-125.

[4] Panichella S, Di Sorbo A, Guzman E, Visaggio C A, Canfora G, Gall H C. How can I improve my app? Classifying user

reviews for software maintenance and evolution. In *Proc. IEEE International Conference on Software Maintenance and Evolution* (*ICSME*), Sept. 29-Oct. 1, 2015, pp.281-290.

[5] McIlroy S, Ali N, Khalid H, Hassan A E. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 2016, 21(3): 1067-1106.

[6] Maas A L, Daly R E, Pham P T, Huang D, Ng A Y, Potts C. Learning word vectors for sentiment analysis. In *Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Volume 1, June 2011, pp.142-150.

[7] Pang B, Lee L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. the 42nd Annual Meeting on Association for Computational Linguistics*, July 2004, pp.271-278.

[8] Seaman C B, Shull F, Regardie M, Elbert D, Feldmann R L, Guo Y, Godfrey S. Defect categorization: Making use of a decade of widely varying historical data. In *Proc. the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Oct. 2008, pp.149-157.

[9] Seaman C B. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 1999, 25(4): 557-572.

[10] Shrout P E, Fleiss J L. Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin*, 1979, 86(2): 420-428.

[11] Witten I H, Frank E, Hall M A, Pal C J. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2016.

[12] Salton G, Yang C S. On the specification of term values in automatic indexing. *Journal of Documentation*, 1973, 29(4): 351-372.

[13] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 1988, 24(5): 513-523.

[14] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten I H. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 2009, 11(1): 10-18.

[15] Tsoumakas G, Katakis I, Vlahavas I. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, Maimon R L (ed.), Springer, 2009, pp.667-685.

[16] Elkan C. The foundations of cost-sensitive learning. In *Proc. the 17th International Joint Conference on Artificial Intelligence*, Volume 17, Aug. 2001, pp.973-978.

[17] Dumais S, Platt J, Heckerman D, Sahami M. Inductive learning algorithms and representations for text categorization. In *Proc. the 7th International Conference on Information and Knowledge Management*, Nov. 1998, pp.148-155.

[18] Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys* (*CSUR*), 2002, 34(1): 1-47.

[19] Platt J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods — Support Vector Learning*, Schoelkopf B, Burges C, Smola A (eds.), MIT Press, 1998.

[20] Seni G, Elder J F. Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. Morgan & Claypool, 2010.

[21] Harman M, Jia Y, Zhang Y. App store mining and analysis: MSR for app stores. In *Proc. the 9th IEEE Working Conference on Mining Software Repositories* (*MSR*), June 2012, pp.108-111.

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

[23] Iacob C, Harrison R. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proc. the 10th IEEE Working Conference on Mining Software Repositories* (*MSR*), May 2013, pp.41-44.

[24] Fu B, Lin J, Li L, Faloutsos C, Hong J, Sadeh N. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proc. the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2013, pp.1276-1284.

[25] Carreño L V G, Winbladh K. Analysis of user comments: An approach for software requirements evolution. In *Proc. the 35th International Conference on Software Engineering* (*ICSE*), May 2013, pp.582-591.

[26] Jo Y, Oh A H. Aspect and sentiment unification model for online review analysis. In *Proc. the 4th ACM International Conference on Web Search and Data Mining*, Feb. 2011, pp.815-824.

[27] Guzman E, Maalej W. How do users like this feature? A fine grained sentiment analysis of app reviews. In *Proc. the 22nd IEEE International Requirements Engineering Conference* (*RE*), Aug. 2014, pp.153-162.

[28] Manning C D, Schütze H. Foundations of Statistical Natural Language Processing (1st edition). MIT Press, 1999.

[29] Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 2010, 61(12): 2544-2558.

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

[31] Chen N, Lin J, Hoi S C, Xiao X, Zhang B. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In *Proc. the 36th International Conference on Software Engineering*, May 31-June 7, 2014, pp.767-778.

Li Zhang received her Bachelor's, Master's and Ph.D. degrees in computer science and technology from Beihang University, Beijing, in 1989, 1992 and 1996, respectively. She is now a professor in the State Key Laboratory of Software Development Environment of Beihang University, Beijing, where she is leading the expertise area of system and software modeling. Her main research area is software engineering, with specific interest in requirements engineering, software/system architecture, model-based engineering, model-based product line engineering, and empirical software engineering.

Xin-Yue Huang received her B.E. degree in computer science and technology from Beihang University, Beijing, in 2015. She is now a M.S. candidate in the State Key Laboratory of Software Development Environment of Beihang University, Beijing. Her research interests include empirical software engineering, data mining, and machine learning.

Jing Jiang is an assistant professor in the State Key Laboratory of Software Development Environment of Beihang University, Beijing. Her research interests include software engineering, empirical software engineering, data mining, and recommendation. She received her B.S. and Ph.D. degrees in computer science from Peking University, Beijing, in 2007 and 2012, respectively.

Ya-Kun Hu received his B.E. degree in computer science and technology from Beihang University in 2017. He works in the State Key Laboratory of Software Development Environment of Beihang University, Beijing, during 2016∼2017. His research interests include natural language processing and deep learning.