Can App Changelogs Improve Requirements Classification from App Reviews? An Exploratory Study *

Chong Wang, Fan Zhang, Peng Liang
School of Computer Science
Wuhan University, China
{cwang, liangp}@whu.edu.cn

Maya Daneva, Marten van Sinderen
School of Computer Science
University of Twente, The Netherlands
{m.daneva, m.j.vansinderen}@utwente.nl

ABSTRACT

[Background] Recent research on mining app reviews for software evolution indicated that the elicitation and analysis of user requirements can benefit from supplementing user reviews by data from other sources. However, only a few studies reported results of leveraging app changelogs together with app reviews. **[Aims]** Motivated by those findings, this exploratory experimental study looks into the role of app changelogs in the classification of requirements derived from app reviews. We aim at understanding if the use of app changelogs can lead to more accurate identification and classification of functional and non-functional requirements from app reviews. We also want to know which classification technique works better in this context. **[Method]** We did a case study on the effect of app changelogs on automatic classification of app reviews. Specifically, manual labeling, text preprocessing, and four supervised machine learning algorithms were applied to a series of experiments, varying in the number of app changelogs in the experimental data. **[Results]** We compared the accuracy of requirements classification from app reviews, by training the four classifiers with varying combinations of app reviews and changelogs. Among the four algorithms, Naïve Bayes was found to be more accurate for categorizing app reviews. **[Conclusions]** The results show that official app changelogs did not contribute to more accurate identification and classification of requirements from app reviews. In addition, Naïve Bayes seems to be more suitable for our further research on this topic.

CCS CONCEPTS

• Software and its engineering → Requirements analysis

KEYWORDS

App reviews, app changelogs, requirements analysis, machine learning, data-driven requirements engineering

ACM Reference format:

Chong Wang, Fan Zhang, Peng Liang, Maya, Daneva, Marten van Sinderen. 2018. Can App Changelogs Improve Requirements Classification from App Reviews? An Exploratory Study. In *Proceedings of 12th International Symposium on Empirical Software Engineering and Measurement (ESEM'18). ACM, Oulu, Finland, 4 pages.*

* This work is supported by the National Basic Research Program of China under grant No. 2014CB340404, and the National Natural Science Foundation of China under grant Nos. 61702378 and 61672387.

ESEM'18, October 11-12, 2018, Oulu, Finland

ACM ISBN 978-1-4503-5823-1/18/10…$15.00
https://doi.org/10.1145/3239235.3267428.

1 INTRODUCTION

Rapid growth of Mobile Internet has resulted in massive sets of data provided by the crowds. Especially, app reviews, a type of implicit feedback from the users, have been recognized as an important source of user requirements for app updating and maintenance [1-3]. Until now, researchers [1-2] mainly concentrated on how to extract features or topics from a large number of app reviews and then classify these topics into categories relevant to software evolution. Several studies [4-7] also explored the use of user feedback from other sources in requirements elicitation. For example, [4] employed user reviews of packaged software in Amazon to pre-extract phrases for mining user opinions from app reviews, while [5] combined product reviews from Amazon with app reviews as the research data. These authors observed that user reviews of software have similar characteristics to app reviews: (1) the number of reviews is increasing rapidly every day; (2) review texts contain lots of noise words, including emoji, non-English words, misspelled words, user-defined abbreviations; and (3) most reviews are non-informative (as reported in [6], only around one-third of app reviews are informative for app updates). Previous research had to make great manual efforts to preprocess app reviews before applying supervised machine learning algorithms for the automatic extraction and classification of software aspects or user requirements from numerous app reviews. To reduce manual effort in filtering out non-informative samples and identify valuable information for developers, this emerging results paper intends to explore if other information of apps, especially the pieces with higher value density, namely app changelogs, could be a significant help.

App changelogs are posted by software vendors regularly in weeks or months. These official texts are written in a standardized way and comprise primary changes of the releases. A 2018 ICSE study [7] has successfully employed app changelogs to identify emerging issues in app reviews. Moreover, our preliminary findings revealed that the changelogs of a certain released app partially reflected user demands addressed in the app reviews of its succeeding version. We were motivated by these findings, and set out to explore whether the use of app changelogs can improve the accuracy of requirements classification from app reviews.

In the next sections, we first present related works, and describe our research questions, method, and data. We then report and discuss the results, and treat validity threats. Finally, we conclude with future directions.

2 RELATED WORK

Regarding the automatic classification of app reviews, Maalej et al. [1] introduced several probabilistic techniques to classify app reviews into four categories, namely bug reports, feature requests, user experience, and text rating, while [2] proposed seven other categories relevant to software evolution. However, these authors did not consider the categories from the perspective of requirements types. In our previous works [8-9], we employed classical machine learning algorithms to extract and classify functional and non-functional requirements (NFRs) from app reviews in which app reviews were the only type of research data to apply and compare specified classifiers. Another similar study in [10], concentrating on elicitation and prioritization of quality requirements, also performed automatic analysis on app reviews.

To the best of our knowledge, only a few studies analyzed app reviews with the supplement of research data from other sources, such as user reviews of software [4] or products [5]. However, both [4, 5] aimed to extract and cluster user opinions, rather than classifying requirements into different categories. Gao et al. [7] took app changeslogs as the ground truth to identify emerging issues from app reviews, instead of identifying and classifying requirements. In contrast to these sources [4-5, 7-10], the focus of our work is mainly on the effect of app changelogs on classifying functional and non-functional requirements from app reviews.

3 RESEARCH DESIGN

3.1 Research Questions

The goal of this work is to explore whether app changelogs can improve the automatic classification of requirements from app reviews. To this end, we formulate two research questions (RQs):

RQ1: *Will requirements classification of app reviews be improved if we train the classifier with app changelogs?*

RQ2: *Which among four classical classification algorithms (Naïve Bayes vs. Bagging vs. J48 vs. KNN) leads to better results?*

Considering the characteristics of app reviews and changelogs, the answer to RQ1 is needed to understand the impact of the official dataset with no noise words (i.e. app changelogs) on the automatic classification of requirements from app reviews. Next, since several classical classification algorithms have been applied in the automatic classification of app reviews, RQ2 is expected to investigate which classification techniques performs better when using app changelogs to train the classifiers. The answer of RQ2 would help evaluate the accuracy of different classifiers and identify the best one for the further research on this topic.

3.2 Research Method and Data

To answer the two RQs, we followed [12] and designed a series of experiments, which were conducted in four phases:

3.2.1 Data Collection. The experiment data is composed of two datasets. More specifically, app reviews were collected from iBooks in the category 'Books & Reference' in Apple App Store, WhatsApp in the category 'Communication' in Google Play, and TripAdvisor in the category 'Travel & Local' in Google Play. 6000 sentences in these user reviews were included as the dataset of app reviews. The dataset of app changelogs was crawled from Apple App Store, containing 2005 changelogs of 30 apps ($3\ \text{categories} \times 10\ \text{apps}$) released till May 2018, as shown in Table 1. Note that these three categories are the same as those which the aforementioned three apps belong to, since we assumed that the changelogs of the same app or the apps from the same category may improve the accuracy of app reviews classification.

Next, we observed that the changes in the 'recent updates' of an app changelog are (partially) duplicated with the ones in the 'latest updates' of its preceding changelogs. By excluding 4233 duplicates from the crawled changes in Table 1, a final set of 2024 changes forms the dataset of app changelogs for our experiments.

Table 1: Overview of Crawled App Changelogs

Category	No. of Apps	No. of Changelogs	No. of Changes
Communications	10	749	2352
Books & References	10	568	1930
Travel & Local	10	688	1975
Total:	30	2005	6257

3.2.2 Sample Selection and Labeling. To explore the impact of app changelogs on the requirements classification, all the 2024 changes in the dataset of app changelogs were sampled.

All four classification techniques used in our study are supervised machine learning approaches, which need to be trained using a labeled truth set. The training set contained already classified instances to classify the instances in the test set. For the truth set creation, we conducted a manual content analysis and labeling for all the sampled 6000 app review sentences and 2024 changes. First, three coders (two were computer science master students, and one - the second author, was a computer science bachelor student) were asked to conduct two groups (two coders in each group) of pilot labeling on 300 app review sentences ($100\ \text{sentences} \times 3\ \text{apps}$) and 150 changes ($5\ \text{changes} \times 30\ \text{apps}$) respectively. One of the two master students has richer experience on labeling requirements in app reviews, and worked in the two groups. Before that, we briefed these three coders in a meeting to introduce the task and explain the NFR standard (ISO 25010 [11]) used for labeling NFRs with some examples. Two pilot labeling tasks resulted in 88% agreements on app review sentences and 87% on app changes respectively. After discussing and resolving all the disagreements, we developed a coding guide to precisely define each type and increase the quality of manual labeling. Finally, these two master students completed labeling on 6000 app review sentences, and the second author and one master student came to an agreement on the labels for 2024 app changes.

Table 2: Impact of App Changelogs on App Reviews Classification using Naïve Bayes (R = app reviews, C = app changelogs)

Training set 1	Test set 1: 500R			Training set 2	Test set 2: 500R (no 'Others'-labeled R)		
	Precision	Recall	F1	(no 'Others'- labeled R)	Precision	Recall	F1
2000C	0.550	0.529	0.539	2000C	0.618	0.599	0.608
1500C+500R	0.597	0.588	0.592	1500C+500R	0.638	0.620	0.629
1000C+1000R	0.601	0.592	0.596	1000C+1000R	0.628	0.620	0.624
500C+1500R	0.605	0.598	0.601	500C+1500R	0.629	0.628	0.628
2000R	0.612	0.588	0.600	2000R	0.629	0.627	0.628

Note that ISO 25010 treats eight types of NFRs: *Functional suitability*, *Performance efficiency*, *Compatibility*, *Usability*, *Reliability*, *Security*, *Maintainability*, and *Portability*. During the pilot labeling, however, we found that functional suitability, compatibility, maintainability, and security were seldom observed in either app reviews or changelogs. Therefore, we included only four types of NFRs (*Usability*, *Reliability*, *Portability*, *and Performance*) in this research, plus the type of functional requirements (FR) and a requirements type that we labeled 'Others' to refer to those requirements that are neither FR nor fit the four NFRs indicated above.

3.2.3 Text Preprocessing. To improve classifier training and the accuracy of requirements classification from app reviews, Natural Language Processing techniques, including stopword removal, stemming, and lemmatization, were applied to the text of app review sentences and changes. Specifically, we employed: (1) Stanford CoreNLP to preprocess the texts of sampled app review sentences and changes; (2) TF-IDF (Term Frequency - Inverse Document Frequency) to count the frequency and evaluate the importance of a word extracted from the text of the sample; and (3) StringToWordVector in Weka to implement TF-IDF technique and extract keywords from app reviews and changelogs.

3.2.4 Classifier Training and Evaluation. Based on the size of the truth set, especially the size of app changes, we set the ratio of the data for the training set to that for the test set as 8:2, i.e., the four classifiers evaluated in Section 4 will by trained by 80% of the random app review sentences and/or app changes. In addition, we adopted the standard metrics *Precision*, *Recall*, and *F-measure (F1)* to evaluate and compare the accuracy of the four classification algorithms (Naïve Bayes, Bagging, J48, and KNN), on the automatic classification of app reviews.

4 Results

4.1 Impact of App Changelogs on App Reviews Classification

To answer RQ1, we designed a series of experiments to calculate the accuracy of automatic requirements classification from app reviews, varying in the proportion of app changelogs in the training sets. These experiments run on a training set of 2000 instances (app review sentences, app changes, or both) and a test set of 500 app review sentences. First, 'Others'-labeled instances were included when we randomly selected specified numbers of app review sentences and changes to form training set 1 and test set 1. Next, we observed that (1) app changes were seldom labeled as 'Others' and (2) almost all the 'Others'-labeled app sentences were non-informative for app updating. To reduce the noise and enhance the accuracy of the app reviews classification, training set 2 and test set 2 were constructed to exclude app review sentences labeled as 'Others'. Table 2 summarizes the results of app reviews classification only using Naïve Bayes, since this classifier has been reported to outperform other classification algorithms [1, 9].

As shown in Table 2, when the classifier was only trained by app changes, the results of requirements classification from app reviews were the worst regardless whether we included 'Others'-labeled instances or not. Once app reviews were added into the training set, the accuracy of Naïve Bayes was obviously increased (at least by 9.8% in test set 1 and 2.6% in test set 2) and then remained stable with the varying proportion of app changes in the training sets. We also observed that F1 score was increased by at least 4.7% after excluding 'Others'-labeled app review sentences in both the training and test sets. In the case that training set 2 is composed of only app changes, the classifier is much more accurate (i.e., F1 score was increased by 12.8%) for predicting requirements types from app reviews. The reason could be that experimental data with less noise normally provide higher quality samples for more accurate prediction.

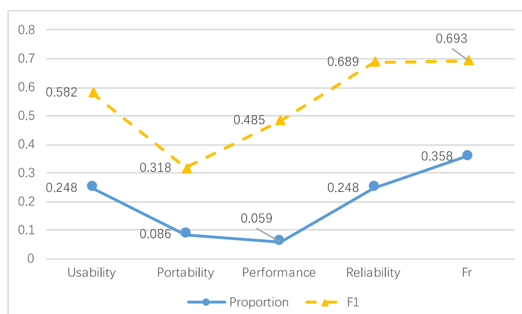

Figure 1: Influence of the proportion of each type of requirements on the accuracy of Naïve Bayes.

Furthermore, we analyzed the relationship between the proportion of each type of requirements in the experimental data and the accuracy of applying Naïve Bayes on the data. Due to space limitation, we only show the results of running Naïve Bayes on the data in the third row of Table 2, where training set 2 contains 1000 app changes and 1000 review sentences. As shown in Fig.1, for the four types of requirements - Usability, Portability, Reliability, and FR, the higher proportion of instances labeled as a certain requirements type leads to more accurate prediction for the corresponding type of requirements from app reviews. Regarding the requirements typed as ‘performance’, however, less samples resulted in higher accuracy. The reason could be that the extracted words characterizing ‘performance’ are not only easier to be identified for grouping this type of requirements, but also harder to be confused with other types of requirements from app reviews.

4.2 Classification Techniques

Table 3 shows the results of comparing Naïve Bayes, Bagging, J48, and KNN. Note that for RQ2, all the experiments were conducted on the instances without ‘Others’-labeled app reviews, according to the preliminary results obtained in Sect. 4.1. In Table 3, the numbers in bold represent the highest F1 scores of the evaluated classification technique when running on the specified dataset. Finally, the results reveal that Naïve Bayes and Bagging performed better than either J48 or KNN. While in most cases, Naïve Bayes is more accurate for predicting different types of requirements than Bagging.

Table 3: F1 score of the Evaluated Classification Techniques

Training set 2	Test set 2	Classification technique			
		Naïve Bayes	Bagging	J48	KNN
2000C	500R	0.608	**0.627**	0.575	0.534
1500C+ 500R	500R	**0.629**	0.628	0.529	0.500
1000C + 1000R	500R	**0.624**	0.580	0.522	0.496
500C +1500R	500R	**0.628**	0.599	0.577	0.597
2000R	500R	**0.628**	0.610	0.577	0.539

4.3 Discussion

For RQ1, we found that although app changelogs contained less noise words and provided high-quality training texts for the classifiers, they did not add up to more accurate classification of different types of requirements from app reviews. The findings were not encouraging, leading to the conclusion that employing higher-quality data from the different source (e.g., app changelogs) cannot improve the automatic classification of lower-quality instances from another source (e.g., app reviews). One reason could be that the proportion of various requirements types in app changelogs is quite different from that in app reviews. It is necessary to explore the impact of unbalanced datasets on the accuracy of requirements classification. Another reason could be that the words used to identify a certain type of requirements in official app changelogs might not be the same as that used in app reviews. It is meaningful to improve feature identification and selection for requirements classification.

For RQ2, we observed that Naïve Bayes worked best for the automatic classification of requirements from app reviews. This finding agrees with the results in [1, 9]. Thus, Naïve Bayes can be an appropriate classification technique to achieve high accuracy in multiclass case in our further research.

5 LIMITATIONS

We evaluated the potential threats to the validity [12] of our results as follows. First, the collected app changelogs and reviews were analyzed by three coders independently, on the premise that they had a consistent understanding on different types of requirements, especially on NFR types defined in ISO 25010. To reduce the risk of inconsistent understanding and mislabeling, a brief meeting and two pilot studies were conducted to exchange their understandings, resolve conflicts, and finally get a consensus on the categorization of specified types of requirements before labeling the remaining samples. Second, all the experiments were implemented in Weka, a suite of machine learning tool written in Java. The results of this exploratory study may vary if the evaluated classification algorithms are programmed in a different language, such as Python. Therefore, how to improve the implementation of those classifiers remains to be studied.

6 CONCLUSIONS AND FUTURE WORK

This emerging results paper explored the effect of app changelogs on the automatic classification of requirements from app reviews. For this purpose, multiple training sets containing different numbers of app changes were constructed as the inputs of evaluated classifiers. The results indicate that the employment of app changelogs did not contribute much to more accurate classification of FRs and four types of NFRs from app reviews. Naïve Bayes is suggested as an appropriate supervised machine learning algorithm for the automatic classification of app reviews.

Since the current findings are not encouraging, our next steps are: (1) to re-evaluate the influence of using app changelogs on requirements classification from app reviews by leveraging the proportions of different types of requirements in the samples, and (2) to investigate how to improve the accuracy of classifying requirements from app reviews by introducing other methods for feature extraction.

REFERENCES

[1] W. Maalej, Z. Kurtanovic, H. Nabil, C. Stanik. 2016. On the automatic classification of app reviews. Requirements Engineering, 21, 3, 311-331.

[2] E. Guzman, M. El-Haliby, B. Ensemble Methods for App Re-view Classification: An Approach for Software Evolution. *ASE’15*, 771-776.

[3] S. Panichella, A. Di Sorbo, et al. How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution. *ICSME’15*, 281-290.

[4] P.M. Vu, H.V. Pham, T.T. Nguyen, T. T. Nguyen. Phrase-based extraction of user opinions in mobile app reviews. *ASE’1*6, Singapore, 726-731.

[5] W. Jiang, H. Ruan, et al, For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews. *PAKDD’14*, 584-595.

[6] N. Chen, J. Lin, S.C.H. et al, AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace. *ICSE’14*, 767-778.

[7] C. Gao, J. Zeng, M. R. Lyu and I. King. Online App Review Analysis for Identifying Emerging Issues. *ICSE’18*, 48-58.

[8] H. Yang and P. Liang. Identification and Classification of Requirements from App User Reviews. *SEKE’15*, 7-12.

[9] M. Lu and P. Liang. Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. *EASE’17*, 344-353.

[10] E.C. Groen, S. Kopczynska, et al, Users-The Hidden Software Product Quality Experts?, *RE’17*, 80-89.

[11] ISO. 2011. ISO/IEC 25010, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. FDIS.

[12] R. Wieringa. 2014. Design Science Methodology for Information Systems and Software Engineering. Springer, ISBN 978-3-662-43838-1.