Augmenting App Reviews with App Changelogs: An Approach for App Reviews Classification

Chong Wang*, Tao Wang*, 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*—Recent research on the automatic classification of app reviews either focused on grouping app reviews into categories relevant to software evolution, or employed app reviews as the only research data to improve app reviews classification. Although it was reported that app review classification can benefit from supplementing user reviews with the data from other sources, only a few studies employed app changelogs for this purpose. This paper explores how to augment app reviews with changelogs to improve the accuracy and performance of classifying functional and non-functional requirements in app reviews. Specifically, we propose AUG-AC as an approach to extract feature words from app changelogs and construct the augments for app reviews. Next, we designed a series of experiments to evaluate our approach, varying in the length of AC-based augments for app reviews. The results show that AUG-AC outperforms the existing method by using app changelogs as a source of data next to app reviews.**

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

I. INTRODUCTION

With the rapid growth of mobile applications, massive sets of data are provided by the crowds. Particularly, app reviews, a type of explicit feedback from the users, have been recognized as an important source of user requirements for app updating and maintenance [1-3]. However, current research [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, 14] also explored the use of user feedback from other sources in requirements elicitation. For example, Vu et al. [4] employed user reviews of packaged software in Amazon to pre-extract phrases for mining user opinions from app reviews, while Jiang et al. [5] combined product reviews from Amazon with app reviews as the research data. These authors [4-5] 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 many 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 30% of app reviews are informative for app updates). To reduce the manual effort in filtering out non-informative samples and identify valuable information for developers, this paper explores if other information of apps, especially the pieces with less noise (i.e. 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. We were motivated by these findings, and set out to explore how to use official app changelogs to improve the accuracy and performance of classifying requirements in app reviews. Especially, this paper intends to explore how to make use of app changelogs in the automatic classification of app reviews from the perspective of requirements types, and finally aid developers in the maintenance and updating of apps.

The paper is structured as follows. Sect. II is on related works. Sect. III presents our approach. Sect. IV reports on the experimental results evaluating and comparing the accuracy and performance of our approach with others. Sect. V discusses our findings. Sect. VI is on validity threats. Sect. VII concludes.

II. RELATED WORK

Considering the automatic classification of app reviews, some researchers proposed categories relevant to software maintenance and evolution. Maalej et al. [1] introduced several probabilistic techniques to classify app reviews into four categories, i.e. bug reports, feature requests, user experience, and text rating. Guzman et al. [2] proposed seven categories relevant to software evolution, viz. bug report, feature strength, feature shortcoming, user request, praise, complaint, and usage scenario. The categories proposed in these two studies partially overlap, since the authors intended to help app vendors and developers filter critical reviews relevant to different aspects of software maintenance. Other researchers considered the categories of app reviews from the perspective of requirements types. In particular, our previous works in [8-9] employed classic machine learning algorithms to identify and classify functional and non-functional requirements (FRs and NFRs) from app reviews. Another similar study [10] performed automatic analysis on app reviews for NFRs elicitation and prioritization. In all these studies, however, app reviews were the only type of research data to apply and compare specified classifiers.

To the best of our knowledge, only very few studies employed research data from other sources to analyze app reviews. Those other sources include user reviews of software [4], of products [5], and app descriptions [14], etc. However, both [4, 5] aimed to extract and cluster user opinions, rather than

*: The authors contributed equally to this work.

This work is supported by the National Key Research and Development Plan under grant No. 2018YFB1003800, and the National Natural Science Foundation of China under grant Nos. 61702378 and 61672387.

classifying requirements into different categories. In [14], Liu et al. used app descriptions, another typical data in app stores, to guide the analysis of user reviews. Gao et al. [7] used app changelogs to identify emerging issues from app reviews, instead of identifying and classifying requirements.

In contrast to these previous studies, the focus of our work is mainly on how to augment app reviews with app changelogs in order to improve the accuracy and performance of classifying FRs and NFRs from app reviews.

III. OUR APPROACH

To employ official app changelogs for the automatic identification and classification of requirements from app reviews, we propose an approach, called AUG-AC, to AUGment app reviews with the text feature words extracted from App Changelogs (AC). In this section, we give an overview of AUG-AC to explore how to improve the automatic classification of app reviews by employing official app changelogs. Each step of our approach will be detailed in a subsection.

The experimental data collected and manually labeled in our previous work [13] will be reused to evaluate the performance and AUC-AC. The dataset includes 6000 app review sentences of three apps (one from Apple App Store and two from Google Play) and 2024 app changes filtered from 2005 official changelogs of 30 apps (3 $categories \times 10$ $apps$ in Apple App Store). As described in [13], these app review sentences and changes were labeled with six types of requirements, including four types of NFRs defined in ISO 25010 [11] (i.e. *Usability, Reliability, Portability* and *Performance*), FR, and 'Others'- the type referring to those review sentences and app changes that fit neither FRs nor the four NFRs listed above.

A. Overview

Our approach consists of four main steps, as Figure 1 shows.

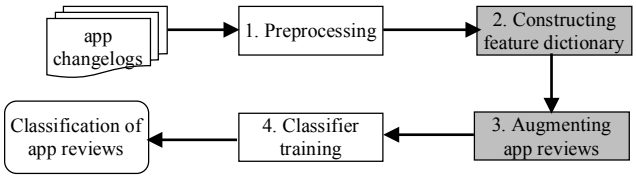


Figure 1. Overview of AUG-AC approach, focusing on the steps in grey.

The first step is to preprocess the sampled app reviews and changelogs to get respective text feature words of each requirements type (Sect. III.B). The second step (which is the one foci of our work) creates a AC-based feature words dictionary for augmenting app reviews (Sect. III.C). The third step (the other focus of our work) augments app reviews with the text features extracted in the second step to train the classifier and group the app reviews into six pre-specified types of requirements (Sect. III.D). In the last step, those augmented app reviews generated in Sect. III.D construct the training set of the specified classifier. The accuracy of applying this classifier to categorizing app reviews in the test set will be evaluated by the standard metrics Precision, Recall and F-measure (Sect. III.E).

B. Text Preprocessing

In this step, multiple Natural Language Processing (NLP) techniques were applied to the text of app review sentences and changes. Specifically, Natural Language Toolkit (NLTK) was adopted to perform stopword removal, punctuation removal, and lemmatization.

Next, considering each type of requirements, we intended to extract text feature words from the app changelogs and treat them as the candidate augmented words for the app reviews labeled as this type. In general, the concerns of app changelogs grouped in each type of requirements may not always be well represented by the frequency and importance of a word. Thus, for each type of requirements, Latent Dirichlet Allocation (LDA) was employed to extract text feature words in app changelogs. By applying LDA, app changelogs labeled as a certain type of requirements into can be clustered into one topic and produce topic words for this cluster (i.e. each type of requirements). In this work, topic words of each cluster form the initial set of text feature words to be augmented to those app reviews that are labeled as the corresponding type of requirements.

C. Constructing AC-based Feature Dictionary

This step aims to construct an AC-based feature dictionary, consisting of the text feature words that was extracted from app changelogs and to be augmented to app reviews. As already said in the beginning of Sect. III, six types of requirements have been specified as the category labels for both app reviews and changelogs. Accordingly, for each requirements type i, a AC-based feature dictionary D_i is needed (1) to store the text feature words extracted from app changelogs typed as i, and (2) to provide candidate AC-based augmented words for app reviews typed as i. In this paper, D_i is initialized as a set containing the top 20 features words t_{ij} extracted from app changelogs labeled with requirements type i.

Algorithm 1: Constructing AC-based Feature Dictionary

Input: D_i – initial AC-based feature dictionary.

Output: D_i' – extended AC-based feature dictionary.

```
for each requirements type i
    import D_i;
    Insert D_i to D_i';
    for each t_ij ∈ D_i
        insert Synonym(t_ij) into D_i';
        insert Antonym(t_ij) into D_i';
    end for
    return D_i';
end for
```

Furthermore, we conducted a pilot study to compare the text feature words extracted from app reviews and changelogs. The preliminary results indicate that the reflection of app reviews on app changelogs is often expressed as the synonyms and antonyms of a certain text feature word, rather than using the same terms. Since more text feature words benefit pre-training of the classifier, the size of D_i is recommended to be extended to cover more candidate AC-based augmented words. For this purpose, WordNet in NLTK was applied to the initial dictionary D_i to generate the extended dictionary D_i' for the requirements type i. More specifically, for each text feature word t_{ij} in D_i, all its synonyms and antonyms identified in WordNet were added

to generate D_i'. Algorithm 1 provides the details of constructing an AC-based feature dictionary for each type of requirements. This results in D_i', which will be used in the next step of AUG-AC to augment the app reviews labeled as requirements type i. In our work, D_i' consists of two parts, i.e. (1) the top 20 text feature words extracted from the type i-labeled app changelogs and (2) all the synonyms and antonyms of these 20 words.

D. Augmenting App Reviews

In this step, we select text feature words in the AC-based feature dictionary created in Sect. III.C, in order to augment app reviews. These augmented app reviews construct the training set of the classifier for app reviews classification.

To achieve a higher accuracy in requirements classification from app reviews, we proposed to augment app reviews with the text feature words derived from those app changelogs whose requirements type is identic with that of the app reviews to be augmented. Specifically, for each type of requirements, we first used Word2Vec in NLTK to calculate the similarity between the AC-based text feature words and the app reviews labeled as the same type. Below, formula (1) was defined to perform the similarity calculation task, where r_{ik} denotes the type i-labeled app review sentence k expressed by a vector $r_{ik} = (t_{ik,1}, t_{ik,2}, \cdots t_{ik,m} \cdots t_{ik,n})$, $t_{ik,m}$ denotes the m-th feature word in r_{ik}, t_{ij} denotes the AC-based text feature word labeled as requirements type i, $w_{ik,m}$ denotes the weight of the feature word $t_{ik,m}$ (produced by BoW) in the app review sentence r_{ik}, $sim(t_{ik,m}, t_{ij})$ denotes the similarity between the AC-based feature word t_{ij} and the app review-based feature word $t_{ik,m}$ (calculated by Word2Vec), and n denotes the number of AC-based feature words to be augmented to the app review sentences.

$$\mathrm{Sim}(r_{ik}, t_{ij}) = \frac{\sum_{m=1}^{n} \left(w_{ik,m} * sim(t_{ik,m}, t_{ij})\right)}{\sum_{m=1}^{n} w_{ik,m}} \quad (1)$$

Considering each app review sentence r_{ik}, the similarity between r_{ik} and t_{ij}, i.e. the value of $\mathrm{Sim}(r_{ik}, t_{ij})$, will be ranked. As a result, the top n AC-based feature words will be added to the end of this review sentence as the semantic augment. This means that n can be treated as the length of AC-based augment for app reviews. Algorithm 2 describes how to augment app review sentences with AC-based text feature words extracted in Sect. III.B and generated in Sect. III.C. Note that in our work, the value of n is pre-specified and fixed for augmenting app reviews. How much length of AC-based augments, i.e. the number of feature words augmented to app reviews, could bring more accurate prediction of app review classification will be discussed in Sect. IV.

Algorithm 2: Generating AC-Augmented App Reviews

Input: AR_i – app reviews labeled as requirements type i
n - the length of AC-based augment

Output: Aug_AR_i – augmented AR_i by adding n words

```
for each requirements type i
    for each r_ik ∈ AR_i
        for each word t_ij ∈ D_i'
            calculate Sim(r_ik, t_ij);
        end for
        sort t_ij ∈ D_i' by Sim(r_ik, t_ij) in descending order;
        add the first n t_ij to r_ik to produce new_r_ik;
        insert new_r_ik into Aug_AR_i;
        Return Aug_AR_i;
    end for
end for
```

E. Classifier Training and Evaluation

According to the experimental results in [1,9,13], Naïve Bayes has been reported to outperform other machine learning algorithms in the automatic classification of app reviews. Therefore, this work adopted Naïve Bayes as the classification technique to categorize FR and NFRs from app reviews. To evaluate the performance of Naïve Bayes, 10-fold cross validation was applied to reduce its overfitting in identification and classification of app reviews. In addition, we adopted the standard metrics *Precision, Recall* and *F-measure* to evaluate the accuracy of Naïve Bayes on the automatic classification of app reviews.

$$\text{Weighted average } (Precision_i \backslash Recall_i \backslash F-measure_i) = \frac{\sum_{i \in type} (Precision_i \backslash Recall_i \backslash F-measure_i) * Number_i}{\sum_{i \in type} Number_i} \quad (2)$$

More specifically, for each requirements type i, $\mathrm{Precision_i}$ is the fraction of the app reviews that are correctly classified as requirements type i, $\mathrm{Recall_i}$ is the fraction of the app reviews of requirements type i that are correctly classified as type i, and $\mathrm{F-measure_i}$ is the harmonic average of the precision and recall. Furthermore, we introduced weighted average precision, recall and F-measure (see formula (2) below) to evaluate the accuracy of classifying app reviews into categories of each requirements type. In formula (2), $Number_i$ denotes the number of app review sentences labeled as requirements type i in the test set of Naïve Bayes.

IV. RESULTS

This section reports the results of our experimental study and compares the accuracy of Naïve Bayes in the automatic classification of requirements from app reviews. The Naïve Bayes algorithm was programmed with Python. All the experiments were conducted on a 2.50GHz Core i5 CPU with 8GB RAM under Windows 10.

A. Impact of Length of Augments on App Reviews Classficiation

As mentioned in Sect. III.C, the length of AC-based augments depends on the number of AC-based text feature words that added to the specified app reviews. Figure 2 shows the precision, recall and F-measure of the automatic classification of augmented app reviews with increasing number of AC-based augmented words (from 5 to 70 words) with an interval of 5 words, by applying Naïve Bayes. Note that in these experiments, the maximum length of AC-based augments is set as 70 words. The reasons are: (1) in [9], Lu and Peng have reported that augments with 1.9 times of the length of an app review sentence leaded to the best results in app reviews classification; and (2) in our dataset, the longest app review sentence has 37 words, and 70 words are around 1.9 times of the maximum length of included app review sentences.

As observed in Figure 2, F-measure is growing rapidly when less than 35 AC-based feature words are augmented to app reviews. There are two peaks when the length of augments is 35

and 45 words respectively. When the number of added AC-based feature words is greater than 50, the value of F-measure decreased or fluctuated within a narrow range.

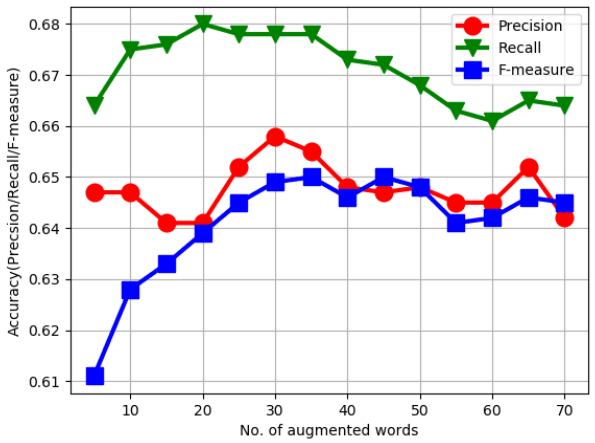
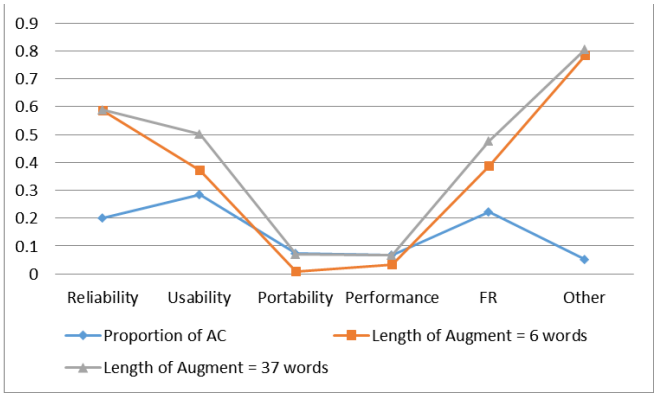


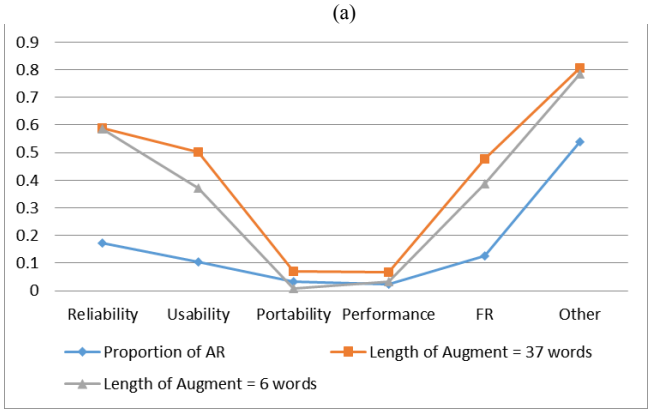
Figure 2. Accuracy of classifying augmented app reviews with varying number of AC-based text feature words.

Furthermore, Table I zooms in the Precision, Recall and F-measure for classifying each type of requirements in app reviews, by employing AC-based augments with different lengths. Typically, the experiments run on the app reviews augmented by 6 (i.e. the average length of app reviews in our dataset) and 37 (i.e. the maximum length of app reviews in our dataset) AC-based text feature words. The results were listed in the columns for precision, recall and F-measure in Table I. We found that for two types or requirements – *Reliability* and *Other*, the accuracy of app reviews classification seldom differs in the lengths of augments for app reviews. Whereas, for *'Usability'*, *'Portability'*, *'Performance'* and *'FR'* typed app reviews, the longer AC-based augment leads to much higher accuracy of classifying app reviews. Specifically, we analyzed the influence of the proportion of each type of requirements identified in app reviews or changelogs on the accuracy of classifying app reviews with different length of AC-based augments.

As shown in Figure 3(a), for two types of requirements – '*Usability'* and *'FR'*, the higher proportion of app changelogs labeled as these two requirements types leads to more accurate identification and classification of these two types of requirements in app reviews. Similarly, for the other two requirements types – '*Portability'* and *'Performance'*, the lower proportion of app changelogs resulted in a lower accuracy of classifying app reviews labeled as these two types. Whereas, it is surprising to find that lower proportion of app changelogs typed as '*Reliability*' and '*Other*' produced the two highest accuracy of classifying these two types of requirements in app reviews. Regarding the proportion of six specified types of requirements in app reviews, Figure 3(b) indicates that the higher (lower) accuracy of classifying app reviews labeled as a certain requirements type usually responds to the higher (lower) proportion of app reviews labeled as this type of requirements.



(a)



(b)

Figure 3. Influence of the proportion of each type in (a) app changelogs and (b) app reviews on the F-measure of classifying augmented app reviews.

TABLE I. PROPORTION OF APP REVIEWS/CHANGELOGS AND ACCURACY OF REQUIREMENTS CLASSIFICATION IN AUGMENTED APP REVIEWS (AC = APP CHANGELOGS, AR = APP REVIEWS)

Type	Proportion of AR	Proportion of AC	Length of Augment = 6 words			Length of Augment = 37 words		
			Precision	***Recall***	***F-measure***	***Precision***	***Recall***	***F-measure***
Reliability	0.172	0.199	0.641	0.544	0.586	0.586	0.588	0.587
Usability	0.104	0.285	0.845	0.239	0.370	0.656	0.408	0.502
Portability	0.034	0.073	0.100	0.004	0.007	0.410	0.040	0.071
Performance	0.025	0.068	0.200	0.018	0.034	0.400	0.037	0.068
FR	0.126	0.222	0.582	0.289	0.386	0.468	0.486	0.476
Other	0.538	0.053	0.672	0.951	0.785	0.750	0.873	0.807
Weighted average			0.641	0.665	0.611	0.656	0.678	0.652

B. Comparision with AUG-BoW

Similar to AUG-BoW in [9], our proposed AUG-AC also aims at augmenting app reviews for more accurate classification of app reviews. As we mentioned in Sect. II, AUG-AC differs in employing official app changelogs to augment app reviews for the identification and classification of requirements in app reviews. To compare and evaluate the performance of these two methods, we conducted a series of experiments varying in the methods for generating augments and the length of augments. As listed in the first row of Table II, we repeated AUG-BOW in sampled app reviews in our dataset; while in the second to the fourth row of Table II, the experiments evaluated AUG-AC in the cases that the lengths of augments were the average length of app reviews (i.e. 6.15 words), 1.9 times of this average length, and the maximum length of included app reviews (i.e. 37 words) respectively.

TABLE II. RESULTS ON CLASSIFYING APP REVIEWS AUGMENTED WITH DIFFERENT TECHNIQUES.

Techniques	Length of augment	Precision	Recall	F-measure
AUG-BoW [9]	1.9 × length of an app review [9]	0.651	0.642	0.569
AUG-AC	6 words	0.646	0.666	0.610
AUG-AC	15 words	0.650	0.674	0.631
AUG-AC	37 words	0.656	0.678	0.652

Our experimental results are in Table II. Therein, we observe that once our proposed AUG-AC was applied to generate AC-based augments for app reviews, the accuracy of app reviews classification increases regardless of the length of augments. That is, our proposed AUG-AC outperformed AUG-BoW [9] by employing app changelogs for classifying requirements in app reviews. Furthermore, we compared the time spent on the automatic classification of augmented app reviews when applying AUG-BoW and AUG-AC in our dataset respectively. For this purpose, the time spent in two typical experiments – in the 1st and 4th row of Table II, was calculated. Note that in this work, we only concentrated on the time spent on augmenting app reviews with app changelogs by these two methods. The reason is that both AUG-BoW and AUG-AC adopted Naïve Bayes as the classifier, and these two techniques may take the same time period to classify augmented app reviews. The results are: (1) AUG-BoW took 1315.77 seconds to construct 'customized' augments for each app review in the training set of Naïve Bayes, and (2) AUG-AC took 106.64 seconds to complete the 37-word augments for any included app reviews. It was obvious that app reviews classification based on AUG-AC completed much faster than that based on AUG-BoW.

V. DISCUSSION

A. Analysis on the length of AC-based augments

Regarding the length of AC-based augments, we observed that the longer AC-based augments leaded to a higher accuracy in classifying app reviews. However, the accuracy did not continuously increase by augmenting app reviews with more than around 40 text feature words. The reason could be that in our AC-based feature dictionary, the AC-based feature words to be added to app reviews were ranked according to their topic relevance with the specified type of requirements. In turn, the top 40 feature words selected for the construction of AC-based augments were the most 'type-sensitive' ones, and also enough, to provide much more accurate results in the app reviews classification.

Next, the proportion of each type of requirements in app reviews and changelogs was not always similar, indicating that these two data sources concentrated on different types of requirements. Taking '*Usability*' NFR as an example, its proportion in app changelogs is around 2.7 times of that in app reviews. This finding implies that although apps may be upgraded to fit different types of user requirements, the official changelogs always pay more attention to this type of requirement. The reason could be that for user, changes typed as '*Usability*' were critical for making decision on whether this release is appropriate for their demands. Regarding to the *'Other'* type, the much lower proportion of app changelogs indicates that they have less noise than app reviews.

Furthermore, we found that the rank of the proportion of different types of requirements in app reviews nearly follow the rank of accuracy in classifying these requirements types in app reviews. The reason could be that both the training and test set of Naïve Bayes consisted of app reviews. In contrast, the rank of the proportion of different types of requirements in app changelogs did not always correspond to the rank of accuracy in classifying app reviews. For example, lower proportion of *'Reliability'* and *'Other'* types in app changelogs contributed to more accurate results in app review classification. One reason could be that for these two types of requirements, the AC-based text feature words that were used to augment app reviews are quite similar to the feature vectors of app reviews. The other reason could be that compared with the other four types of requirements, those that employed AC-based text feature words are much more 'topic sensitive' to identify app reviews typed as *'Reliability'* and *'Other'*. All the findings are positive to our exploration on using app changelogs to classify requirements in app reviews, and encourage further research on this topic.

B. Comparison between AUG-AC and AUG-BoW

Considering the accuracy of app reviews classification, we found that our proposed AUG-AC method outperformed AUG-BoW provided in [9] – the work that inspired our ideas to improve the accuracy of app reviews classification with app changelogs. Compared with AUG-BoW, AUG-AC spent much less time in augmenting app reviews. The main reason is that in AUG-AC, the augments of app reviews were constructed by calculating the similarity between the AC-based feature words of a certain type of requirements and the app reviews labeled as this type. That is, for each type of requirements, 20 feature words extracted from app reviews and their synonyms and antonyms (around 300 words) selected from WordNet are candidates to be calculated and ranked. Whereas, AUG-BoW augmented app reviews by calculating the similarity between extracted text feature words of a certain requirements type and all the app reviews labeled as any type of requirements. These findings can be deemed as the main advantage of our AUG-AC and the main difference between AUG-BoW and AUG-AC.

VI. THREATS TO VADILITY

We followed the guidelines in [15-16] to evaluate the possible threats to validity of our experimental results.

Construct validity: Our work reused the dataset in [13]. All the app reviews and changelogs in this dataset 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. As indicated in [13], we believe this threat to construct validity is partially mitigated by following the aforementioned labelling process.

Internal validity: There is an internal threat to validity concerning how the proposed AUG-AC and the Naïve Bayes classifier were programmed. We implemented them by Python. The results of this exploratory study may vary if AUG-AC and Naïve Bayes are implemented in other ways, e.g. in Weka. Thus, how to improve the implementation of AUG-AC and Naïve Bayes remains to be studied. Another internal validity threat is that, not all the app changelogs and reviews were collected from the same platform. Especially, the changelogs of WhatsApp were collected from Apple App Store and the app reviews were from Google Play. Different concerns of users in different platforms may lead to the fact that the AC-based text feature words provide insufficient semantics to app reviews, which may further result in inaccurate classification of app reviews. Therefore, more research is needed on app reviews and changelogs from the same platform.

External validity: We investigated the user reviews of three apps and changelogs of 30 apps which span over three categories and two major mobile operating systems. We believe that the threats to external validity are partially alleviated. Due to the time and resource limitation, we did not cover many apps, and we plan cover more categories of apps to increase the external validity of the study results.

VII. CONCLUSIONS AND FUTURE WORK

This work explored to augment app reviews with official app changelogs, in order to improve the accuracy of classifying requirements, including FR and four types of NFRs, in app reviews. For this purpose, AUG-AC was proposed to augment app reviews with not only the text feature extracted from app changelogs but also their synonyms and antonyms generated by Word2Vec, in the case that both the employed app changelogs and the app reviews to be augmented are labeled as the same type of requirements. Next, a series of experiments was designed to evaluate the performance of AUG-AC by varying the length of AC-based augments. The experimental results indicate that the AC-based augment of app reviews implemented by AUG-AC can improve the accuracy of classifying requirements in app reviews.

To further evaluate the performance of AUG-AC, our next steps are: (1) re-evaluation of AUG-AC on a balanced dataset by leveraging the proportions of different types of requirements in current dataset; (2) evaluation of AUG-AC with app reviews and changelogs of other apps in other categories of Apple App Store or other app repositories (e.g. Google Play, other Android app stores, etc.); and (3) validation of AUG-AC by using the dataset labeled by more types of NFRs (e.g. Security).

REFERENCES

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

[2] E. Guzman, M. El-Haliby, B. ,"Ensemble Methods for App Re-view Classification: An Approach for Software Evolution", in Proc. of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15), Lincoln, USA, 2015, pp.771-776.

[3] S. Panichella, A. Di Sorbo, et al., "How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution", in Proc. of IEEE International Conference on Software Maintenance and Evolution (ICSME'15), Bremen, Germany, 2015, pp.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", in Proc. of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16), Singapore, 2016, pp.726-731.

[5] W. Jiang, H. Ruan, et al, "For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews", In Proceeding of the 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'14), Taiwan, China, 2014, pp.584-595.

[6] N. Chen, J. Lin, S.C.H. et al, "AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace", in Proc. of the 36th International Conference on Software Engineering (ICSE '14), Hyderabad, India, ACM, 2014, pp.767-778.

[7] C. Gao, J. Zeng, M. R. Lyu and I. King, "Online App Review Analysis for Identifying Emerging Issues", in Proc. of the 40th International Conference on Software Engineering (ICSE'18), Gothenburg, Sweden, ACM, 2018, pp.48-58.

[8] H. Yang and P. Liang, "Identification and Classification of Requirements from App User Reviews", in Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE'15), Pittsburgh, USA, 2015, pp.7-12.

[9] M. Lu and P. Liang, "Automatic Classification of Non-Functional Requirements from Augmented App User Reviews", in Proc. of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17), Karlskrona, Sweden, 2017, pp.344-353.

[10] E.C. Groen, S. Kopczynska, et al., "Users-The Hidden Software Product Quality Experts?", in Proc. of the 25th International Requirements Engineering Conference (RE'17), Lisbon, Portugal, 2017, pp.80-89.

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

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

[13] C. Wang, F. Zhang, P. Liang, et al. "Can App Changelogs Improve Requirements Classification from App Reviews? An Exploratory Study", In Proc. of the 12th International Symposium on Empirical Software Engineering and Measurement (ESEM'18). ACM, Oulu, Finland, 2018, pp.43:1-43:4.

[14] YZ. Liu, L Liu, HX Liu, and WY Wang, "Analyzing reviews guided by App description for the software development and evolution", Journal of Software: Evolution and Process, vol. 30, no. 12, 2018, pp. ,

[15] Forrest Shull, Janice Singer, and Sjøberg Dag I. K. Guide to Advanced Empirical Software Engineering. Springer, 2008.

[16] Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Regnell Bjorn, and Anders Wesslen. Experimentation in Software Engineering. Springer, 2012.