Crowdsourced Report Generation via Bug Screenshot Understanding

Shengcheng Yu
State Key Laboratory for Novel Software Technology, Nanjing University, China
Email: ysc_nju@outlook.com ACM Member Number: 4662439

***Abstract*—Quality control is a challenge of crowdsourcing, especially in software testing. As some unprofessional workers involved, low-quality yieldings may hinder crowdsourced testing from satisfying requesters' requirements. Therefore, it is in demand to assist crowdworkers to raise bug report quality. In this paper, we propose a novel auxiliary method, namely CroReG, to generate crowdsourcing bug reports by analyzing bug screenshots uploaded by crowdworkers with image understanding techniques.**

The preliminary experiment results show that CroReG can effectively generate bug reports containing accurate screenshot captions and providing positive guidance for crowdworkers.

***Index Terms*—Crowdsourced Testing, Mobile App Testing, Bug Report Generation**

I. Problem & Background

Crowdsourced testing has been proved to be much more effective than traditional in-house testing in some areas [1]. In mobile application testing, crowdsourced testing has become a mainstream method, because of the fragmentation of device models, multitude of mobile devices, variety of OS versions, and diversity of testing scenarios [2]. However, the openness of crowdsourcing also makes it easy to contribute to poor quality results. Quality control is a challenge of crowdsourcing, especially in professional areas, such as software testing [3].

Quality control strategies in crowdsourcing systems are mainly classified into crowdworker profiles, task design, control execution, etc.[4] This paper proposes a novel report generation method in control execution to improve the quality of bug reports[1]. It is easy for most of the crowdworkers (even common mobile application users) to capture screenshots in mobile devices [2]. However, it is difficult for untrained testers to write professional bug reports [5].

There are many successful stories of image understanding in many areas. This inspires us to introduce image understanding techniques to improve the quality of crowdsourced reports. A novel method, namely CroReG, is proposed to understand the crowdsourced screenshots and automatically generate bug reports using the combination of Deep Learning and Optical Character Recognition (OCR) techniques. We use *im2txt*[2] model to transfer screenshots into text caption, and OCR to extract texts existing in the screenshots. The text similarity from the two modules will be calculated to further generate bug reports. It is expected that generated bug reports can accurately reflect the bug shown in the screenshots.

[1]"Bug Report" and "Report" are used interchangeably in this paper.
[2]https://github.com/tensorflow/models/tree/master/research/im2txt

Convolutional Neural Network (CNN) has a strong feature self-learning and classification recognition ability due to its unique structure [6][7]. CroReG also borrows the ideas of machine translation, using a variant of Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) [8], to translate image feature vectors into captions. LSTM memorizes the previous information and applies it to the current calculation. OCR is introduced into CroReG to recognize characters existing in the screenshots accurately.

In order to evaluate the effectiveness of CroReG, we conduct a preliminary experiment on some bug screenshots from different categories of mobile applications. 4 captions are generated using CroReG for each screenshot. According to manual evaluation, the correctness of the generated reports reached 90%, and the description in the reports are in highly simulated natural language tune. The results confirm that most generated reports are tightly related to the bugs shown in the screenshots. We believe that CroReG can not only improve the quality of bug reports from untrained crowdworkers but also reduce the work expense of skillful crowdworkers.

II. Methodology

In this paper, we firstly propose a report generation method CroReG based on crowdsourced bug screenshots. The generation procedure is divided into two parts: image translation and text extraction. These two parts work respectively, and the results from these two modules will be further processed and generate final bug reports for the screenshots. The framework of CroReG is shown in Fig. 1.

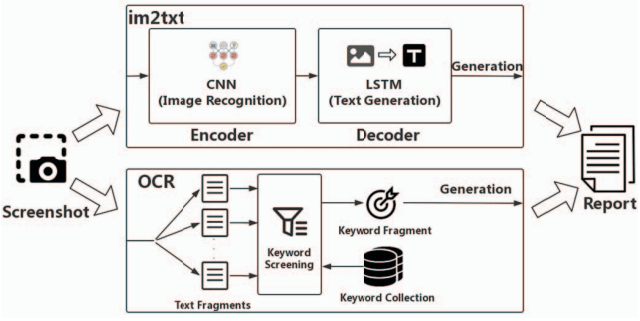

Fig. 1. The Framework of CroReG

A. Image Translation

In this module, CroReG utilizes a deep learning model named *im2txt* [9], which is composed of an encoder and a

978-1-7281-2508-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ASE.2019.00161

decoder. The encoder is a CNN, and a vector with a fixed length will be obtained as a feature extracted from the screenshot. The decoder is an LSTM network, and the feature vector is processed into a natural-language bug caption. The specific CNN model in CroReG is Inception-v3 [10]. Moreover, the dataset is acquired from Baidu MTC[3] and MOOCTEST[4]. It contains 26387 screenshots with manual captions. The dataset is divided into three disjoint sets: training set, verification set, and testing set. A keyword list file related to bugs is also generated. The captions generated by *im2txt* model will be ranked according to the possibility and analyzed according to the results from the OCR process. Top 3 matching captions will be presented to users, and other captions will be discarded.

B. Text Extraction

Mobile application pages always contain rich bug-related text information when bugs appear, such as pop-ups. Therefore, the OCR technique is applied to extract text information in bug screenshots. CroReG uses a third-party remote interface to identify uploaded screenshots. The system will generate several candidate text fragments, and such text fragments will be screened according to preset keywords in the keyword list file generated in the image translation module. Each candidate fragment will be compared with the keyword list to judge whether it contains one or more keywords. The keyword fragments will be processed to exclude redundant information and to generate highly readable bug captions corresponding to the uploaded screenshots.

C. Report Generation

The results from the image translation module and the text extraction module of the same screenshot is processed together to calculate the caption similarity using Levenshtein distance [11], which refers to the minimum number of operations required to convert one string to another during the editing operation, including replacing, deleting, or inserting a character. Then, the results will be merged, and final reports will be presented.

D. An Example of CorReG

Here is an example to illustrate our research. Our tool support only Chinese language currently, so for easy reading, we translate the results of this example into English language. The screenshot of the example says that "Sorry, 'Dingding Guide' (the application name) has stopped running". The image translation module generated 3 captions with corresponding possibility, which are "Software stop running", "Stop running", "Software stop running and exit", and the text extraction module generated 1 caption, which is "Stop Runni", and the mistake is due to the second character in the word "running" is in the next line, so it is wrongly ignored by CroReG. Finally, the generated report is "Software stop running". Though both modules have some subtle errors, the final report is still correct due to the joint process of CroReG, which proves the high accuracy and usability of CroReG.

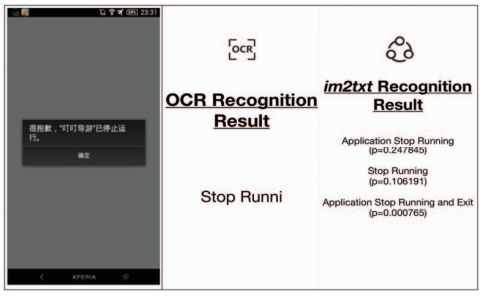

Fig. 2. An Example of Result from CroReG

III. Preliminary Experiment

We have conducted a preliminary experiment to evaluate the effectiveness of CroReG using bug screenshots from mobile applications of different categories, including video game, tool, news, etc. The image translation module generates 3 most possible captions for each screenshot, and the text extraction module generates 1 caption for each screenshot. After the merging process, One caption is finally generated as the bug report for the bug screenshot. We evaluate the experimental reports manually, and the accuracy reached 90%. Our research has reached a high accuracy to generate bug reports.

As to verify the universality of CroReG, we select screenshots from different categories of mobile applications. Results show that CroReG has better effect when the screenshots have strong features such as with text prompts, flashback, etc. Some bugs related to the business logic of the application are harder for CroReG to generate bug reports, such as defects of specific rules in video game applications.

Overall, the preliminary experiment shows that CroReG has good support for most bug screenshots, and can provide much assistance for crowdworkers.

IV. Conclusion & Future Work

Although the study is preliminary, we believe that CroReG can open a new direction on quality control for crowdsourced testing. Many techniques in CroReG can be improved in the future. Crowdworkers tend to upload screenshots of a series of operation sequence. We are working further to merge reports of a series of screenshots to generate a more detailed and complete report, providing better guidance for developers to fix corresponding bugs. A quality assessment of report quality is necessary for crowdworkers when they upload screenshots. It can help crowdworkers to upload high-quality screenshots and then CroReG can generate high-quality reports.

Acknowledgment

The work is partly supported by the National Key Research and Development Program of China (2018YFB1003901), the National Natural Science Foundation of China (61802171), the Fundamental Research Funds for the Central Universities (021714380017), and the Open Foundation of State Key Laboratory for Novel Software Technology in Nanjing University (ZZKT2017B09).

[3]http://mtc.baidu.com
[4]http://mooctest.net

REFERENCES

[1] R. Gao, Y. Wang, Y. Feng, Z. Chen, and W. Eric Wong, "Successes, challenges, and rethinking – an industrial investigation on crowdsourced mobile application testing," *Empirical Software Engineering*, 2019.

[2] Y. Feng, J. A. Jones, Z. Chen, and C. Fang, "Multi-objective test report prioritization using image understanding," in *31st IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2016.

[3] Z. Chen and B. Luo, "Quasi-crowdsourcing testing for educational projects," in *International Conference on Software Engineering*, 2014.

[4] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh, "Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions," *ACM Computing Survey*, 2018.

[5] R. Hao, Y. Feng, J. A. Jones, Y. Li, and Z. Chen, "Ctras: Crowdsourced test report aggregation and summarization," in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[9] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[11] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, 1966.