# Research Progress in the Processing of Crowdsourced Test Reports

Naiqi Wang[1,2], Lizhi Cai[1,2], Mingang Chen[2(✉)], and Chuwei Zhang[3]

[1] School of Information Science and Engineering,
East China University of Science and Technology, Shanghai, China
slytherinwnq@163.com
[2] Shanghai Key Laboratory of Computer Software Testing and Evaluating,
Shanghai Development Center of Computer Software Technology,
Shanghai, China
{clz, cmg}@ssc.stn.sh.cn
[3] Shanghai Foreign Affairs Service Center, Shanghai, China
zhangchuwei@sfasc.com.cn

**Abstract.** In recent years, crowdsourced testing, which uses collective intelligence to solve complex software testing tasks has gained widespread attention in academia and industry. However, due to a large number of workers participating in crowdsourced testing tasks, the submitted test reports set is too large, making it difficult for developers to review test reports. Therefore, how to effectively process and integrate crowdsourced test reports is always a significant challenge in the crowdsourced testing process. This paper deals with the crowdsourced test reports processing, sorts out some achievements in this field in recent years, and classifies, summarizes, and compares existing research results from four directions: duplicated reports detection, test reports aggregation and classification, priority ranking, and reports summarization. Finally explored the possible research directions, opportunities and challenges of the crowdsourced test reports.

**Keywords:** Software testing · Crowdsourced testing · Reports processing

## 1 Introduction

In 2006, Howe first proposed the concept of crowdsourcing, a distributed problem solving and production organization model in the Internet environment [1]. The company or organization outsources the tasks that were carried by the full-time staff to an unidentified, large-scale group to complete the task through an open Web platform.

Subsequently, in response to this definition, various circles have conducted a variety of researches on the working forms of crowdsourcing. The application of crowdsourcing to software testing has also become a new trend. A number of crowdsourced testing platforms have emerged on the Internet, such as Amazon Mechanical Turk, Baidu MTC, UTest, MoocTest and so on. Through these crowdsourced testing platforms, software testing can shorten the test cycle, increase the diversity of the test environment, and improve the quality of software products.


H. Gao et al. (Eds.): TridentCom 2019, LNICST 309, pp. 150–160, 2020.
https://doi.org/10.1007/978-3-030-43215-7_11

However, due to crowdsourced testing characteristics, crowdsourced testing still faces many challenges, such as the division of crowdsourced testing tasks, the incentives for crowdsourced workers, and crowdsourced test reports processing. Due to the large number of crowdsourced workers testing one software at the same time, and the uneven level of crowdsourced workers. The number of test reports received by developers is often large, the duplicate ratio is high, and the quality gap is large. This also makes it difficult for developers to quickly find effective information from these test reports and fix bugs based on these test reports.

Therefore, a lot of work is used to study crowdsourced test reports processing, such as test reports aggregation, classification, prioritization to select higher quality test reports, and multi-document reports summarization to enhance the efficiency of developer review reports. This paper retrieves the relevant papers of crowdsourced test reports processing from the three major databases: IEEE, ACM and SpringerLink, strives to carry out comprehensive summaries, and discusses the future work of crowdsourced test reports processing.

## 2 Background

### 2.1 Crowdsourced Software Testing

Mao et al. gave a complete interpretation of crowdsourcing work in the field of software engineering. They proposed that crowdsourcing software engineering is the behavior of external software engineering tasks in the form of public convening a large number of potential, undefined online workers [2].

In software testing, the developer as a task requester uploads the software to be tested and the test tasks list to the crowdsourced testing platform. After a large number of crowdsourced workers test it, the feedback is sent to the software developer. The procedure of crowdsourced software testing is shown in Fig. 1. A large number of online workers participate in the completion of test tasks, which can provide a good simulation of real application scenarios and real user performance, with short test cycles and relatively low costs [3]. These are some of the advantages of crowdsourced testing over traditional testing. However, due to the large number of crowdsourced workers required for the test project, the levels of crowdsourced workers involved in the test task are mixed, so the quality gap between the test reports submitted is also large. What's more, the measure of rewarding crowdsourced workers in most crowdsourcing platforms is the number of valid reports submitted by workers, which also leads to crowdsourced workers submitting as many reports as possible. All of the above reasons have led to the disadvantages of large gap in the quality of crowdsourced test reports, high duplicate ratio, and a low number of effective reports. Developers often need a lot of energy and time to review these crowdsourced test reports, and the efficiency of fixing bugs based on reports is very low.

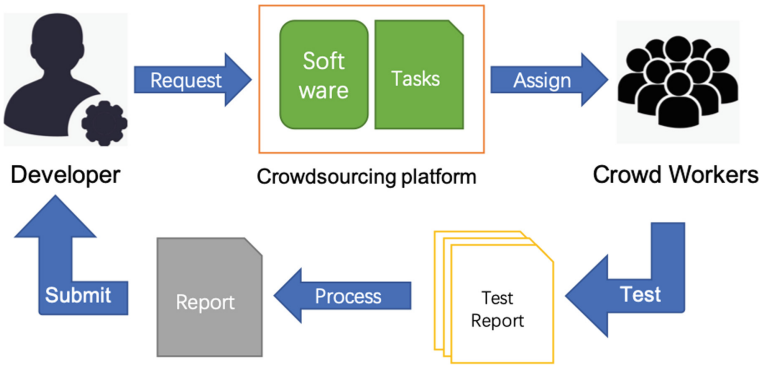


**Fig. 1.** The procedure of crowdsourced testing

### 2.2 Crowdsourced Test Reports

Unlike traditional test reports, Crowdsourced workers who receive test tasks in crowdsourced testing are often not professional software testers. Therefore, the length of the crowdsourced test reports is relatively short, including only the tester's test environment, input conditions, results, and whether or not the test passed.

In addition, as during the test, it is much more convenient to collect the results of the screenshot than to summarize the results into text, crowdsourced workers often use a large number of screenshots to help indicate the test results. Therefore, the crowdsourced test reports feature short text and rich screenshots [4, 5].

In the crowdsourced testing process, crowdsourced workers submit a large number of quality test reports, and crowdsourced testing platform auditors or task requesters will face difficulties in how to effectively integrate and process crowdsourced test reports [6]. Researchers have put a lot of effort in researching how to detect duplicate test reports and select higher-quality reports from the reports set.

## 3 Crowdsourced Test Reports Processing Research

This paper combines the research results of various scholars and summarizes the researches on crowdsourced test reports into four directions:

**Duplicate reports detection.** For a large number of test reports submitted by crowd- sourced workers, duplicate reports in the reports set are detected and further screened (See Sect. 3.1 for details).

**Clustering and classification of crowdsourced test reports.** Based on the duplicate reports detection of the test reports, similar test reports are clustered, or a classifier is constructed to classify the report (See Sect. 3.2 for details).

**Quality assessment and prioritization of crowdsourcing test reports.** Automate testing the content of a test report, conducting a quality assessment, and ranking all reports according to the quality of the reports within the report set (See Sect. 3.3 for details).

**Crowdsourced test reports summarization.** Multi-text summarization of crowd sourced test reports to simplify review of test reports (See Sect. 3.4 for details).

The crowdsourced testing reports processing flow is shown in Fig. 2.

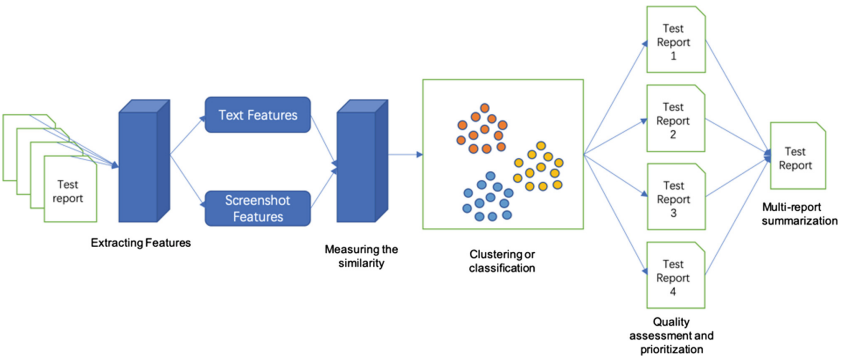


**Fig. 2.** The procedure of crowdsourced test reports processing

### 3.1 Duplicate Reports Detection

For the duplicate reports detection of test reports, researchers have proposed many methods to detect, such as using NLP [7–11], constructing discriminant [12], using information retrieval technology [13, 14] and using Machine learning techniques [12, 15–17].

**Using NLP.** In 2007, Runeson et al. first proposed the use of Natural Language Processing to achieve detection of similar defect reports [7]. They first segmented the reports text, removed all affixes and the stop words, replaced the synonyms. Each defect report is then characterized, converted into a bag-of-words and modeled as a feature vector, and each feature is a word of the report. The similarity between defect reports is calculated by the cosine distance between the feature vectors. Yang et al. used a combination of information retrieval and word embedding techniques to calculate the text similarity of the bug title and description part [8]; Rocha et al. used the bug attribute field calculation [9]; Hindle et al. used software quality context information, software structure items and system development to calculate the similarity between reports [10].

In addition to using the textual information of the test report itself, Wang et al. proposed the use of execution information to aid in the repeated detection of defect reports, Wang et al. set a similarity discrimination mechanism to determine the similarity between reports [11]. However, some studies have shown that only a small number of reports contain execution information, so relying on execution information to assist in detecting duplicate reports is not very effective.

**Using Information Retrieval.** Information Retrieval (IR) refers to the process and technology of obtaining information resources from the relevant information source set according to the needs of information users [13]. Sun et al. proposed a retrieval

function (REP) to achieve detection of repeated defect reports. REP can not only use the summary of defect reports and textual similarity of description fields, but also use non-text fields for similarity analysis [14]; Nguyen et al. combines IR technology and topic models to achieve repeated reports detection [15]. They first used the BM25 algorithm to calculate the similarity between reports, and also used the topic model for calculations.

**Using Machine Learning.** The development of machine learning and artificial intelligence has brought about tremendous progress in the automatic classification of computers. Sun et al. constructed a discriminant and trained a discriminant model using a Support Vector Machine (SVM) to retrieve Top-K relevant similar reports from a set. Liu et al. used Learning to Rank (L2R) technique to build sorting models to achieve repetitive reports searches, while they also used knowledge in Wikipedia to discover semantic relationships between words and documents [16]. Banerjee et al. used multi-label classification to assign multiple duplicate predictions to each report and use fusion roles to combine the results [17].

### 3.2 Clustering and Classification of Crowdsourced Test Reports

Due to the working mode of crowdsourced testing, each test project will receive a large number of test reports. Therefore, how to effectively cluster and classify a large number of test reports is the focus of various scholars.

**Clustering of Crowdsourced Test Reports.** Clustering of crowdsourced test reports is the aggregation of similar reports into a cluster according to a similarity algorithm to condense a large number of scattered test reports into one cluster. Similar reports generally contain similar, identical bugs, so clustering similar reports can effectively classify test reports of the same type into the same reports cluster.

Wang et al. used a method of clustering and reclassification to process the crowdsourced test reports [18]. They first performed feature extraction on the test reports, including textual information, the sentiment orientation score of the reports and the experience of the crowdsourced workers as characteristics, and clustered using the K-means algorithm. Subsequently, the training data is constructed from the most similar clusters to construct the classifier, which can mitigate the impact of local bias on the classification effect.

The test reports submitted by the crowdsourced workers often have the following problems, the invalid problem, that is, the reports includes many non-error test reports and empty test reports that do not contain error information; the imbalance problem, that is, due to the difference between the worker levels, there is a big gap between language description and quality of the reports; multi-bug problem, that is, some workers submit reports containing multiple bugs, which need to be grouped into different clusters. In order to solve these problems, Jiang et al. proposed a new framework called the Test Report Fuzzy Clustering Framework (TERFUR) [19]. They first construct a filter to remove invalid reports, and then build a preprocessor to solve the test reports description too short. Finally, propose a two-stage merging algorithm to cluster redundant reports and multiple bug reports.

In response to the short text and rich screenshots of the crowdsourced test reports, Feng et al. divided reports into two parts, a text description part and a screenshot part [20]. The text description part is processed using NLP technology to obtain the distance vector between the text histogram and the text; the screenshot part distance is obtained by using the Spatial Pyramid Matching (SPM) [21] technique for the screenshot part. The two-part matrix is combined to obtain a mixed distance matrix. Hao et al. used this method to obtain the distance between test reports and cluster the reports using the Hierarchical Agglomerative Clustering (HAC) method [4]. Also, because in the case where such a cluster is not known in advance, the HAC can be adaptively clustered by setting a threshold value without first defining the number of clusters. The method proposed by Yang et al. is also using SPM to obtain the text and screenshot distance matrix, but in the clustering method selection, they chose a different way [22]. After obtaining the similarity value between all reports, they first choose two reports' similarity values. Then, they compare the first report in the remaining test reports with the existing two cluster representatives. This report is inserted if the similarity value is greater than or equal to the threshold they defined. In a cluster with a higher similarity value, if the similarity value is smaller than the threshold, a new cluster will be added. The next one is compared to similar values for the three existing clusters. In the comparison of similarity values and thresholds, if the similarity value is higher, it is added to the cluster of higher similarity values, otherwise, it is represented as a new cluster, so reciprocating until all the reports are traversed again.

In the selection of text clustering methods, Wang et al. used K-means clustering, but this method needs to pre-define the number of clusters that need to be clustered. Wang et al. use element silhouette to determine the number of clusters. The number of candidate K is set to 1–1000 to determine the best solution [18]. While Hao et al. used HAC to determine the number of clusters based directly on the threshold distance value, simplifies the process of clustering parameter setting [4]. However, in the method of image matching, the SPM method used in the papers above treats the features of different regions equally, and does not take into account the difference in the characteristics of different regions. What's more, this method does not fully play the role of the text information in the screenshot, and should be considered in future research.

**Classification of Crowdsourced Test Reports.** The classification of crowdsourced test reports is that the researchers extract features from the test reports set according to the features that need to be extracted, and construct a classifier from the extracted features. Reports repeatability testing, severity prediction, and bug triaging are available according to the capabilities of the classifier.

Wang et al. constructed a classifier for text features, sentiment features, and experience of workers in the results after clustering [18]. Later, they focused on the cross-domain issues faced by the crowdsourced test reports classifier [23]. Test reports in different fields often have their own special professional terms, and directly use different areas of test reports to build classifiers. These words will make the accuracy of the classifier reduced, so Wang et al. first trained a Stacked Denoising Autoencoders (SDA) model and generated advanced features of the reports based on the textual information of the reports, and finally trained the classifier on the advanced feature

vector. In this way, the cross-domain problem in the crowdsourced test reports classifier training can be solved. They also proposed the Local Information Based Active Classification (LOAF) to classify true faults from crowdsourced test reports [24]. LOAF recommended a small number of instances that provide the most information in the local community, and continually asks users about their tags and then classifies them. LOAF utilizes an active learning approach that greatly reduces the amount of data that needs to be labeled and simplifies the annotation work required to build a classifier.

Wang et al. used text information and image information for classification too, but unlike [4, 22], they did not use SPM to calculate screenshot information. Instead, a SETU framework was proposed that combines information from screenshots and textual descriptions to detect duplicate population test reports. SETU extracts four types of features to characterize screenshots (image structure features and image color features) and text descriptions (TF-IDF features and word embedding features) and designs a hierarchical algorithm to detect four-based repeat similarities the scores come from four characteristics.

### 3.3 Quality Assessment and Prioritization of Crowdsourced Test Reports

In software engineering, it is very important to obtain software defect information from the defect reports for repair. The quality of the defect reports is also very important for the inspection reports [26, 27]. In the crowdsourced testing, although the crowdsourced test reports contain less information than traditional defect reports, due to its large number, how to conduct automated quality assessment of crowdsourced test reports to select higher quality reports and give priority to crowdsourced test reports is very important for the processing of the test reports.

**Quality Assessment for Crowdsourced Test Reports.** In the quality assessment for reports, some classification-based work has achieved the ability to classify reports according to reports quality. The classifier constructed by CURES [18], it's criteria for judging test reports are text features, emotional parameters, and worker experience. The text feature is the description of the bug by the crowdsourced workers. The sentiment parameter is a score of −5 to 5, because studies have shown that the reported sentimental tendency may reflect the subjective feeling of the bug; the workers' experience is divided into the number of reports submitted, the number of true faults reported, and the proportion of true faults in the submitted reports. LOAF is the use of active learning methods to continuously learn the reports containing the true faults, so as to classify the reports [24].

Chen et al. proposed a framework for mobile applications called TERQAF [28]. They defined a series of indicators to measure the ideal properties of the test reports, and summarized the values of all the indicators to determine the test by using the step conversion function to judge the quality of the reports. To the best of the author's knowledge, this is also the first work to investigate the quality of test reports and to address the quality assessment of test reports.

**Prioritization of Crowdsourcing Test Reports.** For how to sort crowdsourced test reports, Feng et al. proposed a combination of risk strategy and diversity strategy

considerations [29]. After modeling the test reports to vector, they first considered the risk strategy, that is, selecting the report with the highest risk value; Then considered the diversity strategy, that is, selecting the report with the largest distance from the selected report matrix in the candidate reports. Finally, they combined the two strategies and continually select reports to enter the reports set to complete the sorting task.

Feng et al. used text information and screenshot information, through the NLP and SPM to obtain the distance matrix [20]. They use a diversity strategy, when a developer reviews a report, the next one pushed is the one that is the most distant from the current review report. This strategy allows developers to find more errors in a short period of time.

For traditional test reports, Yu et al. trained neural network-based test reports priority prediction model [30], they added an additional layer in the neural network to learn the relationship between the severity and tester, used evolutionary learning to adapt to the addition of new features, and reused data between similar projects as an initial training set for new projects to speed up the training process.

### 3.4 Crowdsourced Test Reports Summarization

Text summarization has always been a hot issue in the field of NLP research. Text summarization is technically divided into extractive summarization and abstractive summarization, which is divided into single-document summarization and multi-document summarization from the abstracted document format. In the traditional test reports processing, because of its long text, it contains more information, single-document and multi-document summarization are both needed. However, the characteristics of the crowdsourced test reports are that the amount is large and the text is short. Therefore, the summarization work for the crowdsourced test reports is mainly a multi-document summarization.

In traditional test reports processing, Mani et al. applied four well-known unsupervised summarization algorithms for the error reports summarization in order to solve the problem that the supervised method requires manual annotated corpora and the generated digest may be biased toward training data [31]. To build a more efficient automatic summarizer to summarize bug reports, [32, 33] investigated whether the current conversation-based automatic summarizer is suitable for bug reports, and they found that the quality of the generated summary is similar to the summary generated by email or other conversations. They also trained an automatic summarizer for the bug reports, and achieved good results.

For the crowdsourced test reports summarization, Hao et al. clustered the reports firstly [4]. In each cluster, the PageRank algorithm is used to find a report with the largest amount of information as the main report, and the rest is used as the candidate reports. Separated the sentence with the screenshot in the candidate reports and cluster again. In each cluster, PageRank is used to sort the sentences and screenshots, and then the selected sentences are added to the master report to assist the review according to the set summary compression ratio. Jiang et al. first investigated the existing methods of attribute construction in Mining Software Repositories (MSR). Then, a new method called Crowd-Attribute is proposed to infer the valid attributes of the bug reports

summary from the crowdsourced workers to generate reports data. Jiang et al. used Crowd-Attribute to construct 11 new attributes and proposed a new supervised algorithm, Logistic Regression with Crowdsourcing Attributes (LRCA) [34].

Currently, methods for crowdsourced test reports summarization are mostly extractive, multi-document summarization. Researchers use certain rules to extract the information they need from a large number of test reports and combine them into a single test report that helps developers to review submitted test reports more quickly.

## 4 Summary

This paper reviews the processing methods of crowdsourcing test reports in recent years. From the crowdsourcing test reports, duplicate reports detection, reports clustering and classification, report quality assessment and prioritization, and crowdsourced test reports summarization, we summarize the work in the four directions above. We can find that the quality of the report itself is uneven due to the large difference in the level of crowdsourced workers, which will affect the results of the crowdsourced test reports. And in the crowdsourced test reports processing, the methods currently used by researchers are more traditional, and the processing of text and screenshots is relatively simple. Therefore, future research on the processing of crowdsourced test reports can focus on the following points:

*(1) Using more advanced NLP models and image processing methods to process the reports:*

In recent years, NLP technology has developed rapidly. Based on the pre-trained model, BERT, XLNet have achieved excellent results in tasks such as text classification, reading comprehension, and short text matching, but the current text features extraction in crowdsourced test reports processing still uses a more traditional approach. Similarly, the current analysis of screenshots for crowdsourced test reports, the only documents available for review are the SPM method and the extraction of image structure and color. Therefore, how to use the more advanced NLP model and image processing method to process the crowdsourced test reports is the research directions that researchers can choose.

*(2) Constructing a recommendation system that combines the processed, classified test reports with the developer information. And assign bugs based on developer professionalism:*

Using clustering and classification methods can effectively bring together reports with similar characteristics, and similar reports often reveal the same bugs. In software engineering, different bugs are generally fixed by different developers. Therefore, pushing a test report that is highly relevant to a professional based on the developer's professional information can make it more efficient for developers to review the reports.

*(3) Generating test case based on the processed test reports:*

Mattia et al. proposed a method for automatically generating test cases based on bug reports of mobile applications. They used a combination of program analysis and natural language processing to generate executable test cases from bug reports, which can be correctly applied to most reports [35]. How to use the processed large-scale crowdsourced test reports to generate the corresponding test cases is also worth studying.

**Acknowledgment.** This work is funded by National Key R&D Program of China (No. 2018YFB 1403400).

## References

1. Howe, J.: The rise of crowdsourcing. Wired Mag. **14**(6), 1–4 (2016)
2. Mao, K., Capra, L., Harman, M., et al.: A survey of the use of crowdsourcing in software engineering. J. Syst. Softw. **126**, 57–84 (2017)
3. Latoza, T., Hoek, A.: Crowdsourcing in software engineering: models, motivations, and challenges. IEEE Softw. **33**(1), 74–80 (2016)
4. Hao, R., Feng, Y., Jones, J., Li, Y., Chen, Z.: CTRAS: crowdsourced test report aggregation and summarization. In: ICSE 2019 (2019)
5. Zhang, T., Chen, J., Luo, X., Li, T.: Bug reports for desktop software and mobile apps in GitHub: what is the difference? IEEE Softw. **36**, 63–71 (2017)
6. Zhang, X.F., Feng, Y., Liu, D., Chen, Z.Y., Xu, B.W.: Research progress of crowdsourced software testing. Ruan Jian Xue Bao/J. Softw. **29**(1), 69–88 (2018)
7. Runeson, P., Alexandersson, M., Nyholm, O.: Detection of duplicate defect reports using natural language processing. In: Proceedings of the 29th International Conference on Software Engineering, pp. 499–510. IEEE Computer Society (2007)
8. Yang, X., Lo, D., Xia, X., Bao, L., Sun, J.: Combining word embedding with information retrieval to recommend similar bug reports. In: ISSRE 2016, pp. 127–137 (2016)
9. Rocha, H., Valente, M.T., Marques-Neto, H., Murphy, G.C.: An empirical study on recommendations of similar bugs. In: SANER 2016, pp. 46–56 (2016)
10. Hindle, A., Alipour, A., Stroulia, E.: A contextual approach towards more accurate duplicate bug report detection and ranking. Empir. Softw. Eng. **21**, 368–410 (2016)
11. Wang, X., Zhang, L., Xie, T., et al.: An approach to detecting duplicate bug reports using natural language and execution information. In: ACM/IEEE 30th International Conference on Software Engineering, ICSE 2008, pp. 461–470. IEEE (2008)
12. Sun, C., Lo, D., Wang, X., Jiang, J., Khoo, S.-C.: A discriminative model approach for accurate duplicate bug report retrieval. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1. ACM (2010)
13. Information Retrieval. https://en.wikiipedia.org/wiki/information_retrieval. Accessed 10 Sept 2019
14. Sun, C., Lo, D., Khoo, S.C., et al.: Towards more accurate retrieval of duplicate bug reports. In: 26th IEEE/ACM International Conference on Automated Software Engineering, pp. 253–262. IEEE Computer Society (2011)
15. Nguyen, A.T., Nguyen, T.T., Nguyen, T.N., et al.: Duplicate bug report detection with a combination of information retrieval and topic modeling. In: 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 70–79. ACM (2012)
16. Liu, K., Tan, H.B.K., Zhang, H.: Has this bug been reported? In: 20th Working Conference on Reverse Engineering (WCRE), pp. 82–91. IEEE (2013)

17. Banerjee, S., Syed, Z., Helmick, J., Cukic, B.: A fusion approach for classifying duplicate problem reports. In: ISSRE 2013, pp. 208–217 (2013)
18. Wang, J., Cui, Q., Wang, Q., et al.: Towards effectively test report classification to assist crowdsourced testing. In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM (2016)
19. Jiang, H., Chen, X., He, T., et al.: Fuzzy clustering of crowdsourced test reports for apps. ACM Trans. Internet Technol. **18**(2), 1–28 (2018)
20. Feng, Y., Jones, J.A., Chen, Z., et al.: Multi-objective test report prioritization using image understanding. In: 31st IEEE/ACM International Conference on Automated Software Engineering, pp. 202–213 (2016)
21. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Computer Vision and Pattern Recognition, pp. 2169–2178. IEEE (2016)
22. Yang, Y., Yao, X., Gong, D.: Clustering study of crowdsourced test report with multi-source heterogeneous information. In: Tan, Y., Shi, Y. (eds.) DMBD 2019. CCIS, vol. 1071, pp. 135–145. Springer, Singapore (2019). https://doi.org/10.1007/978-981-32-9563-6_14
23. Wang, J., Cui, Q., Wang, S., et al.: Domain adaptation for test report classification in crowdsourced testing. In: International Conference on Software Engineering: Software Engineering in Practice Track. IEEE Press (2017)
24. Wang, J., Wang, S., Cui, Q., et al.: Local-based active classification of test report to assist crowdsourced testing. In: 31st IEEE/ACM International Conference. ACM (2016)
25. Wang, J., Li, M., Wang, S., Menzies, T., Wang, Q.: Images don't lie: duplicate crowdtesting reports detection with screenshot information. Inf. Softw. Technol. **110**, 139–155 (2019)
26. Nazar, N., Jiang, H., Gao, G., et al.: Source code fragment summarization with small scale crowdsourcing based features. Front. Comput. Sci. **10**(3), 504–517 (2016)
27. Jiang, H., Zhang, J., Ma, H., et al.: Mining authorship characteristics in bug repositories. Sci. China Inf. Sci. **60**(1), 012107 (2017)
28. Chen, X., Jiang, H., Li, X., et al.: Automated quality assessment for crowdsourced test reports of mobile applications. In: 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE Computer Society (2018)
29. Feng, Y., Chen, Z., Jones, J.A., Fang, C., Xu, B.: Test report prioritization to assist crowdsourced testing. In: 10th ACM Joint Meeting on Foundations of Software Engineering, pp. 225–236 (2015)
30. Yu, L., Tsai, W.-T., Zhao, W., Wu, F.: Predicting defect priority based on neural networks. In: Cao, L., Zhong, J., Feng, Y. (eds.) ADMA 2010. LNCS (LNAI), vol. 6441, pp. 356–367. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17313-4_35
31. Mani, S., Catherine, R., Sinha, V.S., Dubey, A.: AUSUM: approach for unsupervised bug report summarization. In: ACM SIGSOFT International Symposium on the Foundations of Software Engineering, pp. 1–11 (2012)
32. Rastkar, S., Murphy, G.C., Murray, G.: Automatic summarization of bug reports. IEEE Trans. Softw. Eng. **40**(4), 366–380 (2014)
33. Kokate, P., Wankhade, N.R.: Automatic summarization of bug reports and bug triage classification. Int. J. Sci. Technol. Manag. Res. **2**(6) (2017)
34. Jiang, H., Li, X., Ren, Z., et al.: Toward better summarizing bug reports with crowdsourcing elicited attributes. IEEE Trans. Reliab. **68**, 1–21 (2018)
35. Fazzini, M., Prammer, M., d'Amorim, M., Orso, A.: Automatically translating bug reports into test cases for mobile apps. In: 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018), pp. 141–152. ACM (2018)