CTRAS: A Tool for Aggregating and Summarizing Crowdsourced Test Reports

Yuying Li
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
yuyingli@smail.nju.edu.cn

Rui Hao
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
ruihao@smail.nju.edu.cn

Yang Feng
Department of Informatics
University of California, Irvine
Irvine, CA, USA
yang.feng@uci.edu

James A. Jones
Department of Informatics
University of California, Irvine
Irvine, CA, USA
jajones@uci.edu

Xiaofang Zhang
School of Computer Science and Technology
Soochow University
Suzhou, China
xfzhang@suda.edu.cn

Zhenyu Chen
Mooctest Inc.
Nanjing, China
chenzhenyu@mooctest.com

ABSTRACT

In this paper, we present CTRAS, a tool for automatically aggregating and summarizing duplicate crowdsourced test reports on the fly. CTRAS can automatically detect duplicates based on both textual information and the screenshots, and further aggregates and summarizes the duplicate test reports. CTRAS provides end users with a comprehensive and comprehensible understanding of all duplicates by identifying the main topics across the group of aggregated test reports and highlighting supplementary topics that are mentioned in subgroups of test reports. Also, it provides the classic tool of issue tracking systems, such as the project-report dashboard and keyword searching, and automates their classic functionalities, such as bug triaging and best fixer recommendation, to assist end users in managing and diagnosing test reports. Video: https://youtu.be/PNP10gKIPFs

CCS CONCEPTS

• **Software and its engineering → Maintaining software**.

KEYWORDS

Crowdsourced Testing; Duplicate Report; Document Summarization; Image Understanding

ACM Reference Format:
Yuying Li, Rui Hao, Yang Feng, James A. Jones, Xiaofang Zhang, and Zhenyu Chen. 2019. CTRAS: A Tool for Aggregating and Summarizing Crowdsourced Test Reports. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '19), July 15–19, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3293882.3339004

ISSTA '19, July 15–19, 2019, Beijing, China

ACM ISBN 978-1-4503-6224-5/19/07…$15.00
https://doi.org/10.1145/3293882.3339004

1 INTRODUCTION

Crowdsourced testing has become a popular mobile application testing method because it can provide testing with numerous, diverse, and real-world conditions, such as device models, location information, network environments, user settings, and operating systems. Unlike traditional testing approaches, because crowdsourced testing employs a large number of crowd workers, who may not have strong knowledge on software testing, to complete the testing tasks, and the number of completed tasks influences the rewards for the crowd workers, the requester often receive an overwhelming number of reports containing many duplicates and low-quality ones. Therefore, the process of triaging, understanding, and diagnosing these reports naturally becomes a time-consuming yet inevitable task for the requesters.

To assist developers in dealing with duplicate reports, the conventional bug-management systems, such as the widely used Bugzilla and Mantis, have provided keyword-search-based features for users to identify similar reports to reduce duplicated submissions. However, the settings and inherent features of *mobile* crowdsourced testing bring challenges into applying these techniques. Zhang *et al.* found that testers of mobile applications prefer to submit screenshots rather than detailed paragraphs of text to describe bug [8] that leads to mobile test reports often contain insufficient text descriptions and rich screenshots in comparison with desktop software. Under this situation, while text-analysis-based methods become less effective because of short and inaccurate text descriptions, automatically identifying information from screenshots becomes critical for developers to understand reports. On the other hand, prior research on assisting with duplicate detection for crowdsourced testing focuses on three main approaches: (1) automatically clustering test reports, (2) filtering out duplicates to reduce the number of test reports, and (3) prioritizing test reports for diagnosis [2, 6, 7]. These methods do not leverage the information among the test reports within duplicate clusters, which Bettenburg *et al.* found is

useful for providing extra information for report comprehension and debugging [1].

Thus, in this paper, we describe the architecture, implementation, and usage of CTRAS, a novel tool that is capable of leveraging the information of duplicate test reports to assist test-report management by automatically clustering, aggregating, summarizing, and assigning test reports. Different from the conventional bug/test-report-processing techniques, instead of discouraging developers from submitting duplicates and filtering them out, our technique aims at leveraging the additional information provided by them and summarizing both the *textual and image information* from the grouped duplicates to a comprehensive and comprehensible report. CTRAS measures the similarity regarding both the *text description and screenshots*. Then, it automatically conducts clustering on test reports to aggregate duplicates. Based on the clustering result, CTRAS *summarizes* and *visualizes* the crucial information of each report cluster, including main topics, supplementary topics, weighted keywords, duplicate relationships, and then automatically *assigns* the aggregated test reports to developers to assist the test report management and diagnosis.

CTRAS has the following features:

- CTRAS is a web-based test report management tool that is capable of aggregating and summarizing the crowdsourced test reports. Based on the aggregation report, CTRAS provides an efficient way for developers to process the overwhelming number of test reports.
- CTRAS identifies useful information from the duplicates, to supplement the main topics and further assist developers to understand the reports.
- CTRAS visualizes the duplicate connection map for each aggregated test reports, which is helpful for developers to diagnose bugs.

2 TOOL DESIGN

2.1 Preliminary Definitions

To ease the explanation, in this paper, we use the term *test report* to refer to the original test report submitted by the crowd workers and define the following concepts:

- *Master report*: an individual test report that is identified as providing the most information for end users.
- *Supplementary*: the representative information item, *i.e.,* either text or screenshot, that describes the shared topics or features of several reports within the duplicate report group.
- *Summary*: A summarized report for a group of duplicates which consists of one master report and several supplementaries.

2.2 Architecture

We present the architecture of CTRAS in Figure 1. CTRAS is composed of four main components: aggregator, summarizer, analyzer, and visualizer. CTRAS aims at assisting software developers in four software-maintenance tasks: bug triaging, test-report comprehension, duplicate bug-report aggregation, and expert recommendation. First, the aggregator computes the distance between each pair of test reports based on both the textual description as well as the screenshots to form a distance matrix. Based on the distance matrix, the aggregator is capable of detecting duplicates and grouping them

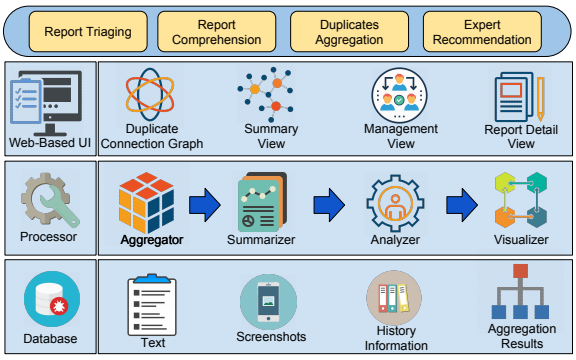

Figure 1: Components and features of CTRAS

into clusters. And then, the summarizer identifies the most informative test report, *i.e., master report*, and distinct topics for each duplicate group. Summarizer refines these distinct topics and identifies the most representative items to form supplementaries. These supplementaries, along with the master report, comprise the *summary* that summarizes the cluster of duplicates to help developers comprehend the failures and localize bugs. Based on the *summary*, the analyzer mines employ the historical information to identify the domain expert and recommend the expert for fixing the bug. Further, to show the aggregate result and help the developers track the summarization procedure, the visualizer also shows *aggregation graphs* and *duplicate-relationship graphs*. The core code of the CTRAS is available via github: https://github.com/iris-42/CTRAS

2.3 Aggregator

Aggregator first computes the distance between each pair of test reports and output the distance matrix. We adopt the hybrid strategy, proposed by Feng *et al.* [2], to measure the distance regarding both of the textual descriptions and screenshots of test reports. CTRAS employs NLP techniques to process the textual descriptions, including the process of tokenization, stop-word removal, and keyword vector building. To measure the distance between screenshots, CTRAS employs SPM (Spatial Pyramid Matching) to extract the SIFT (Scale-Invariant Feature Transform) features and compute the Chi-Square distance. The hybrid distance matrix among all reports is built upon the weighted harmonic mean of these two similarities. Based on the distance matrix, the aggregator adopts Hierarchical Agglomerative Clustering (HAC), to group the duplicates.

2.4 Summarizer

Summarizer component has two fundamental functions: (1) identifying the master report, (2) extracting supplementary topics from the duplicates. The summarizer abstracts test reports within a duplicate group into a weighted graph, of which the vertex is the report and the edge connecting two vertexes is weighted based on the distance between the two reports. It further employs the PageRank algorithm on the graph to measure the importance of reports and identifies the master report with the highest weight. After identifying the master report, CTRAS compares each sentence and screenshot of other reports with the master report and marks different ones. CTRAS applies HAC algorithm on the set of the marked sentences to form some candidate sentence clusters. Similarly, CTRAS applies the same strategy on the marked screenshot

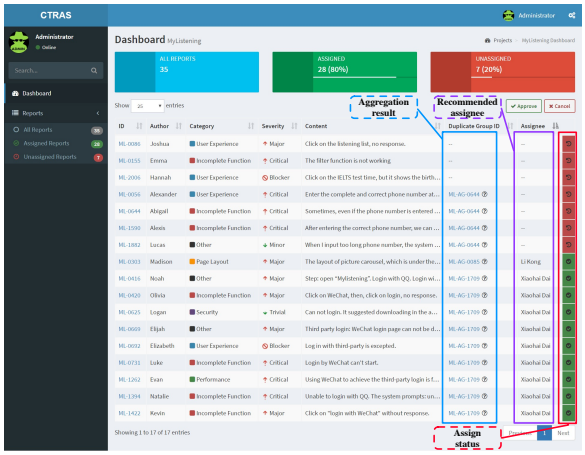

Figure 2: The test-report-list view of CTRAS

set and get the candidate screenshot clusters. Within each candidate clusters, CTRAS further applies the PageRank algorithm to highlight the most representative item, *i.e.,* a sentence or screenshot as the supplementary.

Note that, for some test reports that are not aggregated into any duplicate report groups (*i.e.,* their descriptions and screenshots are not similar to any other test reports), CTRAS allows users to manually process and aggregate them.

2.5 Analyzer

CTRAS employs the weighted keywords of each of the test-report clusters to mine the historical assignment information in the database. Based on this historical information, CTRAS matches the best fixer for each group of test reports. Compared with similar conventional tools, with which users need to assign reports manually, users only need to check the correctness of the recommendation and choose to confirm or modify it. Note that for some new reports, there may be no domain expert; under this situation, CTRAS cannot automatically identify the proper fixer and as such leaves the recommendation as "None".

2.6 Visualization

Besides the basic operation views, CTRAS presents the *summary view* to show the detailed information. In the summary view, CTRAS shows the basic information, presents all content of the master report, and highlights representative sentence and screenshots of the supplementaries. In addition, a tag cloud is presented to depict the keywords within the duplicate group. Further, CTRAS presents an aggregation graph to show the distance and relationships among test reports and supplementaries of the summary.

3 INTERFACE AND USAGE

Figures 2 and 3 show two main views of CTRAS. After a user, such as a project manager, logs into CTRAS, she can choose the application to work on. Once an application has been chosen, Figure 2 is presented. And at the top of Figure 2, CTRAS shows the total number of submitted test reports, along with the number of assigned and unassigned test reports. Below that is the list of individual test reports. Each test report is presented in a row, along with user-submitted information, such as the test-report identifier, authors, category, severity, and a snippet of the textual description. Additionally, automatically assigned information is further presented in each row, such as its membership to an aggregate report and the developer to which it was assigned.

Note that CTRAS can aggregate and summarize test reports. The red button (*i.e.,* the *Auto Assign*) control the automatic test report aggregation and assignment process. Upon completion of such processing, the aggregation result and assignee are displayed for each test report, and the user can either approve or change the aggregation and assignment. Also, the user can then view individual test reports by clicking a test-report ID, or view aggregated reports by clicking the Aggregator ID.

We present the view of an exemplary *summary* of the duplicate group "ML-AG-1709" in Figure 3. In the top of the center pane, we present the basic information about the group, including names of all crowd testers who submitted test reports, the assignee of this report group, tags and so on. Additionally, the view shows the detailed information, including both the textual description as well as the screenshots, of the *master report* which is identified as the most informative one of this group.

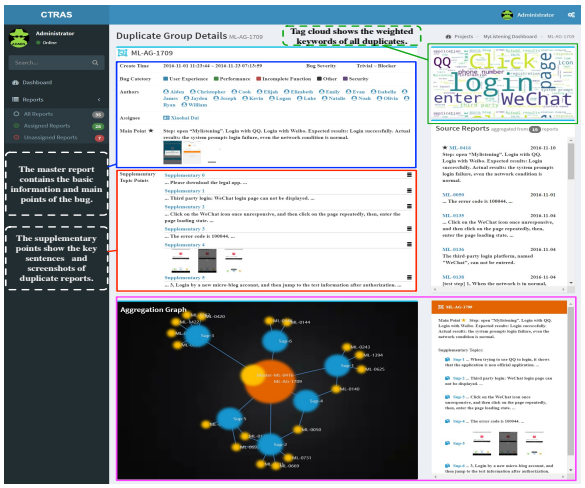

Figure 3: The summary view of CTRAS

Below this pane, supplementaries are shown. Supplementaries are extracted from multiple test reports within the aggregate report. These supplementaries can enrich context and descriptions of the *master report*, which may help users to diagnose and fix bugs. Also, to help users catch the key points of these supplementaries, CTRAS highlights the most representative sentence or screenshot within each of them. Below the tag cloud, individual test reports can be found and inspected.

Finally, at the bottom of the view is a graph that depicts the relationships between test reports and their supplementaries. The orange circle denotes the summary, and yellow circles represent individual reports, among which the largest one overlaps the orange circle is the master report. The blue circles surrounding the orange circle represent the supplementaries. The yellow circles radiating from the blue one represent the individual test report that shares the information of the supplementary. The length of the edge denotes the distance between nodes. Note that users can click on any of these nodes and view their details in the right pane. Given this

visualization, users can reach a high-level understanding of the aggregation as well as the information of this group.

4 EVALUATION

We conduct a task-based experiment which involved 8 industrial crowdsourced projects and 16 participants to evaluate the performance of CTRAS. The dataset used in this experiment were collected through the national software-testing contest[1]. In this contest, contestants are required to test applications on mobile devices and submit test reports. More than 10 organization committee members and professional testers manually reproduce the reported bug and label these reports. The description of the dataset is shown in Table 1, in which, $|R|$ is the number of reports, $|S|$ is the number of screenshots, $|R_s|$ is the number of reports that contains at least one screenshot, $|D|$ is the number of duplicates and $|F|$ is the number of revealed faults in the test reports.

CTRAS is designed for presenting a comprehensive and comprehensible summary for users to improve the efficiency of processing test reports without loss of accuracy. To evaluate the effectiveness of the summary, we disabled the summarizer component to get the modified version of CTRAS. Based on the modified version of CTRAS, which can provide all functionalities, such as automatic aggregation and batch processing, except the summarization and *summary view*, we can evaluate to what extent the summarization and *summary view* can improve the efficiency and accuracy in processing test reports.

We recruited 16 participants and all of them were the graduate student of computer science who had at least five-years experience in programming but not any experience in using these subject applications. We first provided the participants with the ground-truth information of all reported faults, which is written by the organization committee, to help them to form a high-level understanding of faults. Given this information, each participant was required to label the test report with the fault it revealed based on the version of CTRAS with or without the *summary view*. Each participants is required to complete the labeling tasks of four different applications: two tasks are completed on the version with a summary view and two tasks are completed on the version without the summary view. Totally, we collect 32 results under each operating settings.

Table 1: Statistical Information of Testing Applications

	Name	Version	Category	$\|R\|$	$\|S\|$	$\|R_s\|$	$\|D\|$	$\|F\|$
$p1$	Game-2048	3.14	Games	210	219	164	96	18
$p2$	HW Health	2.0.1	Health	262	327	201	109	28
$p3$	HJ Normandy	2.12.0	Education	269	436	241	123	28
$p4$	MyListening	1.6.2	Education	473	418	306	128	31
$p5$	JayMe	2.1.2	Social	1400	1997	1168	678	112
$p6$	Kimiss	2.7.1	Beauty	79	58	48	31	6
$p7$	Slife	2.0.3	Health	1346	2238	1124	885	55
$p8$	Tuniu	9.0.0	Travel	531	640	418	236	54
$total$				4570	6333	3670	2286	332

We evaluated the CTRAS from two aspects: the time cost of labeling one test report and the accuracy, which is calculated as the ratio of the number of correctly labeled reports to the total number of reports. The evaluation results reveal that CTRAS with the *summary view* reach an accuracy of 0.88 and the time cost of labeling one test report is 5.27 seconds. The accuracy on the version without a summary view is 0.88 and the time cost is 11.86 seconds. On average, the summary can help participants to save 51.7% time cost with the fault while there is no significant loss of accuracy. Moreover, we also evaluated the performance of the aggregator and summarizer, detailed experimental results are available in [3].

[1] http://www.mooctest.org

5 RELATED WORK

There are several works discussing the problem of bug report summarization. Jiang *et al.* [5]leveraged the authorship characteristics to assist bug report summarization. In this paper, given a new bug report created by a contributor, the classifier trained over annotated bug reports by the contributor is highly likely to generate better summaries than classifiers trained over annotated bug reports by others. Jiang *et al.* [4] proposed a PageRank-Based Summarization Technique (PRST), which utilized the information in a master report and associated duplicates to summarize the master report. However, the goal of these works is to filter out the duplicates, none of these approaches leverages the duplicate reports to help people understand bugs more comprehensively. Besides, none of these researches leverages the image information on bug report summarization problems, whereas our technique aims at leveraging the additional information provided by them, and summarizing both the textual and image information from the grouped duplicates to a comprehensive and comprehensible report.

6 CONCLUSION

In this paper, we present CTRAS, a web-based tool for aggregating and summarizing crowdsourced test reports to assist professional testers to manage and understand crowdsourced test reports. CTRAS provides the end users a comprehensive and comprehensible understanding of all duplicates by identifying the main topics across the group of aggregated test reports and highlighting supplementary topics that are mentioned in subgroups of test reports.

ACKNOWLEDGEMENT

The work is partly supported by the National Key Research and Development Program of China (2018YFB1403400) and the National Natural Science Foundation of China (61690201, 61772014, 61802171).

REFERENCES

[1] N. Bettenburg, R. Premraj, and T. Zimmermann. 2008. Duplicate bug reports considered harmful ... really?. In *IEEE International Conference on Software Maintenance*. 337–345.
[2] Yang Feng, James A. Jones, Zhenyu Chen, and Chunrong Fang. 2016. Multi-objective test report prioritization using image understanding. In *Ieee/acm International Conference on Automated Software Engineering*. 202–213.
[3] Rui Hao, Yang Feng, James A. Jones, Yuying Li, and Zhenyu Chen. 2019. CTRAS: Crowdsourced Test Report Aggregation and Summarization. In *Proceedings of the 41th International Conference on Software Engineering*. ACM.
[4] He Jiang, Najam Nazar, Jingxuan Zhang, Tao Zhang, and Zhilei Ren. 2017. PRST: a pagerank-based summarization technique for summarizing bug reports with duplicates. *International Journal of Software Engineering and Knowledge Engineering* 27, 06 (2017), 869–896.
[5] He Jiang, Jingxuan Zhang, Hongjing Ma, Najam Nazar, and Zhilei Ren. 2017. Mining authorship characteristics in bug repositories. *Science China Information Sciences* 60, 1 (2017), 012107.
[6] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau Cheng Khoo. 2012. A discriminative model approach for accurate duplicate bug report retrieval. In *ACM/IEEE International Conference on Software Engineering*. 45–54.
[7] Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun. 2009. An approach to detecting duplicate bug reports using natural language and execution information. In *ACM/IEEE International Conference on Software Engineering*. 461–470.
[8] Tao Zhang, Jiachi Chen, Xiapu Luo, and Tao Li. 2017. Bug Reports for Desktop Software and Mobile Apps in GitHub: What is the Difference? *IEEE Software* (2017).