SemCluster: A Semi-supervised Clustering Tool for Crowdsourced Test Reports with Deep Image Understanding

Mingzhe Du
Shengcheng Yu
Chunrong Fang*
State Key Laboratory for Novel Software Technology,
Nanjing University, China
fangchunrong@nju.edu.cn

Tongyu Li
Heyuan Zhang
Zhenyu Chen
State Key Laboratory for Novel Software Technology,
Nanjing University, China

ABSTRACT

Due to the openness of crowdsourced testing, mobile app crowdsourced testing has been subject to duplicate reports. The previous research methods extract the textual features of the crowdsourced test reports, combine with shallow image analysis, and perform unsupervised clustering on the crowdsourced test reports to clarify the duplication of crowdsourced test reports and solve the problem. However, these methods ignore the semantic connection between textual descriptions and screenshots, making the clustering results unsatisfactory and the deduplication effect less accurate.

This paper proposes a semi-supervised clustering tool for crowdsourced test reports with deep image understanding, namely SemCluster, which makes the most of the semantic connection between textual descriptions and screenshots by constructing semantic binding rules and performing semi-supervised clustering. SemCluster improves six metrics of clustering results in the experiment compared to the state-of-the-art method, which verifies that SemCluster has achieved a good deduplication effect. The demo can be found at: https://sites.google.com/view/semcluster-demo.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**.

KEYWORDS

Semi-Supervised Clustering, Image Understanding, Textual Analysis, Semantic Binding Rules

ACM Reference Format:
Mingzhe Du, Shengcheng Yu, Chunrong Fang, Tongyu Li, Heyuan Zhang, and Zhenyu Chen. 2022. SemCluster: A Semi-supervised Clustering Tool for Crowdsourced Test Reports with Deep Image Understanding. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3540250.3558933

*Corresponding author

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3558933

1 INTRODUCTION

In crowdsourced testing, the mobile application testing tasks are distributed to a large number of the crowdsourcing workers, and these people utilize a variety of test environments to solve problems caused by fragmented mobile devices and complex network environments. However, crowdsourced testing also brings many problems, especially the problem of duplication of crowdsourced test reports. According to the data surveyed in [10], an average of 82% of the crowdsourced test reports in the actual industrial data are duplicate. This problem will greatly reduce the review efficiency of software developers, which will bring trouble to the timely resolution of software bugs. The solution to this problem in most of the existing researches is to extract textual features from crowdsourced test reports [1] and adopt the unsupervised clustering method to cluster the reports [5] [6]. These methods ignore the deep-level information contained in screenshots and the semantic connection between textual descriptions and screenshots.

In this paper, we design and implement a semi-supervised clustering tool for crowdsourced test reports with deep image understanding, namely SemCluster, to better solve the problem of duplicate reports. SemCluster extracts four features from the crowdsourced test reports and further uses various distance calculation methods to calculate the distance of each pair of feature vectors. According to the data distribution characteristics of the distance calculation results, SemCluster constructs the semantic binding rules, and under the guidance of the rules, semi-supervised clustering of the reports is carried out. SemCluster makes full use of the semantic connections between screenshots and textual descriptions, returns accurate and credible clustering results to users, and achieves the purpose of deduplication of crowdsourced test reports.

2 METHODOLOGY

SemCluster is divided into four modules, feature extraction, distance calculation, rule construction and semi-supervised clustering. The developer first uploads the crowdsourced test reports to SemCluster. In the feature extraction module, SemCluster extracts Structural Feature and Content Feature from screenshots, and Bug Behavior and Reproduction Step from textual description. In the distance calculation module, SemCluster uses a variety of algorithms to calculate the distance of each feature. In the rule construction module, this paper innovatively proposes semantic binding rules. In the semi-supervised clustering module, SemCluster uses K-Medoids

to cluster reports and iteratively modifies the clustering results according to the binding rules. SemCluster finally returns the clustering results to the developer, and the clustering results reveal the specific situation of duplicate reports so as to help solve the problem of the duplicate reports.

2.1 Feature Extraction

The crowdsourced test report consists of screenshots and textual descriptions. The screenshot is abstracted into Structural Feature and Content Feature. For textual descriptions, Bug Behavior and Reproduction Step can be extracted based on the contents.

Structural Feature Extraction GUI structure can be derived from the apk files of the apps with some automated technologies, e.g., UIAutomator. However, such technologies cannot work on GUI images. In addition, app widgets are extracted in certain areas. However, such areas may be blank. This paper proposes a new method that combines traditional computer vision algorithms (CV) and deep learning techniques (DL) to process screenshots. SemCluster first uses CV to identify the widgets from the screenshots and then uses a CNN model [9] and VGG-16 to classify the widgets. Finally, SemCluster organizes these widgets into a four-level tree structure. The root of the tree (level zero) is the screenshot. The first level is named "Group", which represents a group of widgets that can be roughly considered on the same horizontal line. The second level is named "Line", which represents a line of widgets in the same group whose coordinates are closer in height horizontally. The third level is named "Column", which represents a column of widgets with close vertical coordinates.

Content Feature Extraction SemCluster adopts a CNN model to classify the widgets. First, this paper builds a dataset containing over 36,000 screenshots of widgets [11], evenly distributed among 14 types of widgets and the dataset is divided into the training dataset, the validation dataset and the test dataset according to 7:2:1. The CNN model uses the AdaDelta algorithm as the optimizer, the categorical_crossentropy function as the loss function, and the batch size is set as 32. The model behaves well in identifying the widget category. The overall Precision and Recall values reach 90%, which means that the trained model can effectively extract the widget screenshot features, especially in identifying the features of widget contents. In this paper, the last SoftMax layer of the model is discarded and the result of the penultimate fully connection layer is directly output as a 4096-dimensional vector. The screenshot is represented by a variable-length set of 4096-dimensional vectors representing the valid widgets in the screenshot.

Bug Behavior Extraction This paper adopts a pre-trained TextCNN model to separate Bug Behavior from Reproduction Step and adopts the Word2Vec model to encode the Bug Behavior and convert it into a 100-dimensional vector. In addition, this paper uses a domain-specific keyword list to identify synonyms, antonyms and polysemies in mobile app testing.

Reproduction Step Extraction Reproduction Step describes the sequence of user actions from application launch to bug occurrence. Every operation S_i contains two parts of information, the specific operation op_i and the operation object obj_i. Reproduction Step RS can be abstracted into the following standard sequence: $RS = \langle S_1(op_1, obj_1), S_2(op_2, obj_2), ..., S_n(op_n, obj_n)\rangle$. In this paper, the component parser algorithm is used to analyze the part-of-speech of the text segment, so as to extract the specific operation and operation object from the textual description.

2.2 Distance Calculation

Structural Feature Distance Calculation SemCluster uses the APTED (All Path Tree Edit Distance) algorithm to calculate the Structural Feature distance. Tree edit distance is the minimum number of operations required to transform one tree structure into another. APTED considers three types of operations: (a) delete a node, and connect the child node of the deleted note to its parent node; (b) insert a node between an existing node and its child node; (c) rename a node. Since all nodes are assigned the same label in the distance calculation of Structural Feature, renaming a node is not required. The APTED algorithm uses the single-path function to calculate the edit distance between two trees first, and then to recursively calculate the edit distance between a subtree of one tree and all subtrees of the other tree. The calculated distance is sorted and normalized finally. The max-min normalization is as follows: $\frac{x-min}{max-min}$, where max represents the maximum value of all results, and min represents the minimum value of all results.

Content Feature Distance Calculation For the screenshots in the two crowdsourced test reports, SemCluster has obtained the number of widgets and the coordinate information of each widget. SemCluster first expresses the report whose screenshot contains less widgets as W_{source}, and the other report as W_{target}. Then, for each widget w_s in W_{source}, SemCluster finds a matching widget w_t^* from W_{target}. The selection criteria for w_t^* are as follows: (a) the widget in W_{target} that is closest to w_s according to the distance calculation of the coordinates; (b) the value of IoU (Intersection over Union) is greater than the preset threshold λ. It has been verified by experiments that the best effect is when the value of λ is 0.75. After finding w_t^*, calculate the Euclidean distance between two widget vectors. The average distance of all matched widget pairs is the Content Feature distance, and each result requires to be normalized using the max-min normalization formula.

Bug Behavior Distance Calculation Bug Behavior is in the form of text vectors. By convention, SemCluster uses the Euclidean distance as the Bug Behavior distance. The results also require to be normalized using the max-min normalization formula.

Reproduction Step Distance Calculation SemCluster treats the Reproduction Step as a sequence. For two sequences RS^a and RS^b, each node S_i^t in the sequence consists of an operation and an object. For each S_i^a in RS^a and each S_i^b in RS^b, the textual similarity between these nodes is calculated. Then SemCluster uses the DTW algorithm to find the most similar node pairs and calculate the distance between the two sequences. The results require to be normalized using the max-min normalization formula.

Feature Aggregation SemCluster calculates the semi-supervised distance SD between the two reports using a set of parameters θ_i representing the weight of each feature. The formula is as follows: $SD = \sum_{i=1}^{4}(\theta_i \times dis(X))$, where X represents four features, which are Structural Feature, Content Feature, Bug Behavior and Reproduction Step. The sum of θ_i is 1 and this paper sets θ_i to 0.25 in the real implementation.

2.3 Rule Construction

This paper constructs two kinds of binding rules to guide the clustering process, which comprehensively consider screenshots, textual descriptions and semantic connections between them. The two kinds of binding rules are as follows: (a) Must-Link: two data instances (crowdsourced test reports) must be assigned to the same cluster; (b) Cannot-Link: two data instances (crowdsourced test reports) cannot be assigned to the same cluster

Applying the binding rules to the feature distance range constraint, this paper obtains the following specific binding rules: $(a)\ Must-Link: Dis(SF) < \alpha\ \&\ Dis(BB) < \theta$; $(b)\ Cannot-Link: Dis(CF) > \beta\ \&\ Dis(RS) > \omega$. In the actual implementation process of this paper, by default α is set to 0.1, θ is set to 0.3, β is set to 0.7, and ω is set to 0.8. Parameters are determined based on preliminary attempts and small-scale experiments.

2.4 Semi-supervised Clustering Need to verify

SemCluster uses the K-Medoids algorithm in this module. Compared with the more commonly used K-Means algorithm, K-Medoids always selects the real crowdsourced test report as the center point in the entire clustering iteration process while K-Means continuously generates new data instances as the center points. Why?

Before the clustering process starts, it is first necessary to build a binding rule dictionary. According to the binding rules constructed in Section 2.4, it is determined that each crowdsourced test report pair belongs to Must-Link or Cannot-Link, or that there is no binding rule relationship between them. Finally, SemCluster obtains a Must-Link dictionary and a Cannot-Link dictionary.

The next step is to confirm the value of the target cluster number K. Following convention, this paper uses the elbow method to confirm the K value. After the K value is determined, the initial center point needs to be selected for the clustering process. First, SemCluster randomly selects K instances from the crowdsourced test reports as the "pre-selected" initial center points, and then checks whether there is a Must-Link relationship between the "pre-selected" initial center points. The crowdsourced test report pairs with Must-Link relationship are eliminated and SemCluster randomly selects again from the "unpreselected" crowdsourced test reports until there is no Must-Link relationship between the K initial center points.

SemCluster starts the iterative clustering process after the initial points are selected. In this paper, the number of iterations is set to 30 in the actual experimental process. In each round of iteration, the unclustered crowdsourced test reports should be calculated the semi-supervised distance with each center point, and clustered to the target cluster under the guidance of the binding rules.

For each target instance to be clustered, if there is no instance in the existing clusters that has a Must-Link relationship or a Cannot-Link relationship with the target instance, it is regarded a general situation, and the original clustering results can be maintained. If the following special cases are encountered, reclustering should be performed: (a) There are one or more clusters that satisfy: 1) there is an instance with a Must-Link relationship with the target instance; 2) there is no instance with a Cannot-Link relationship with the target instance. Then the target instance will be selected to cluster into such clusters, and will finally be clustered into the cluster with the smallest semi-supervised distance from the center point; (b) There are one or more clusters that satisfy: 1) there is an instance with a Cannot-Link relationship with the target instance; 2) there is no instance with a Must-Link relationship with the target instance. Then the target instance will be selected to cluster into other clusters, and will finally be clustered into the cluster with the smallest semi-supervised distance from the center point. For more special cases other than the above two cases, reclustering is not required, but will be regarded as "Conflict". After all the crowdsourced test reports are clustered, recalculate the center point of each cluster as the initial center point of a new round of iteration, and enter the next round of iteration. After all iterations are over, SemCluster gets the final clustering results so as to provide the developer with the specific duplication situation and realize the purpose of deduplication.

2.5 System Implementation

SemCluster applies Flask framework for the server end and Vue framework for the client end. SemCluster is connected to MySQL and stores account information and clustering results. Data files generated during processing are saved in local storage. The specific workflow is shown in the Figure 1.

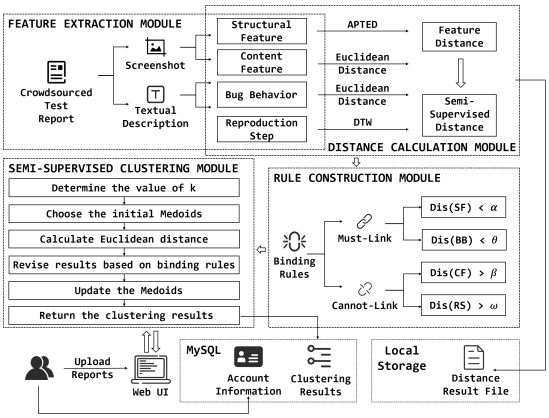

Figure 1: SemCluster Workflow

3 EXPERIMENT

This paper designed an empirical experiment to verify the effect of SemCluster. We collect 847 crowdsourced test reports from 18 mobile apps covering different categories (details in online package). We label these apps from A1 to A18, and the number of crowdsourced test reports for each app ranges from 4 to 152. Three experts in mobile application development and testing were invited to label these crowdsourced test reports according to the bugs found. After labeling and cross-validation, the bug categories per application range from 3 to 17 for a total of 109.

This paper compares SemCluster with the SETU algorithm proposed by Wang et al [10]. The experimental results show that the effect of SemCluster is significantly better than that of SETU. In 18 application tests, SemCluster increases Precision by 57.13%, Recall by 132.26%, F-measure by 76.81%, Purity by 16.82%, ARI by 2372.83%, and NMI by 51.70% on average.

SemCluster is compared with two configurations, Image and Text, which represent using only image features and only text features respectively. Compared with Image and Text, SemCluster has an improvement of 51.29% and 43.69% in Precision, 76.68% and 52.48%

in Recall, 56.91% and 45.11% in F-measure, 19.66% and 22.20% in Purity, 2372.83% and 787.76% in ARI, and 51.70% and 54.56% in NMI.

This paper designs Unspvsd using an unsupervised clustering method, and the only difference between SemCluster and Unspvsd is that the clustering process is not guided by binding rules in Unspvsd. According to the experimental results, Unspvsd is better than SETU, but not as good as SemCluster.

4 USER GUIDANCE

After the developer registers an account and logs in to SemCluster, the crowdsourced test report clustering can be started. The main page of SemCluster contains a step bar, through which the developer can clearly know which step SemCluster has reached.

File Upload Page. This page includes a window where the developer can upload a file in csv format. Before uploading any file, the developer can download the demo file provided by the website to view the specific format of the required file. After the developer uploads the file, the backend of SemCluster automatically starts feature extraction and distance calculation.

Distance Results Page. This page includes the distance results of four features and the final result of the semi-supervised distance. The developer observes the data distribution characteristics of these distance results to formulate appropriate binding rules.

Rule Construction Page. The developer can define the binding rules on this page. SemCluster supports the developer to add and delete the binding rules. The developer adds the features to be constrained according to the distance calculation results and sets a certain value. If the developer does not define the binding rules on this page, SemCluster will perform semi-supervised clustering under the guidance of the default binding rules.

Clustering Results Page. This page includes the final clustering results which are also stored in the database. The data displayed on this page includes the file name, the number of clusters, the details of the cluster, and the number of conflicts.

5 RELATED WORK

Crowdsourced test report deduplication technology is the key to solve the problem of duplication of crowdsourced test reports. Prifti et al. [8] designed and implemented a large-scale experimental study based on an open-source project and proposed a technique that focuses on searching for duplicates at specific locations in a repository. Alipour et al. [1] and Hindle et al. [4] conduct a more comprehensive analysis of the context of test reports to improve the accuracy of duplicate report detection, and further improve detection capabilities. Nguyen et al. [7] designed a tool called DBTM to detect duplicate reports using information retrieval-based and topic-based features. These works have a great inspiration for the feature extraction part and clustering part of this paper.

GUI image understanding technology for mobile app testing also plays an important role in solving the problem of duplication of crowdsourced test reports. Cooper et al. [3] propose TANGO, a method for detecting video-based duplicate reports by combining visual and textual information, and experimental evaluations confirm that TANGO can effectively identify duplicate reports and save developers' workload. Chen et al. [2] propose a method to automatically predict natural language labels for mobile application GUI components. Chen et al. found that more than 77% of applications have at least one image-based button lacking natural language labels. To solve this problem, a deep learning model was proposed to learn and predict based on The label of the button for the image. These works have great inspirations for the extraction of image features and the design of the experiment in this paper.

6 CONCLUSION

The duplication of crowdsourced test reports has increasingly become a key issue affecting the efficiency of crowdsourced testing. This paper proposes SemCluster, which performs feature extraction and distance calculation on crowdsourced test reports, and introduces semantic binding rules to guide unsupervised clustering process. Developers can get the clustering results of the crowdsourced test reports by uploading the original file of the crowdsourced test reports in SemCluster and constructing semantic binding rules, which reveals duplication of crowdsourced test reports and improves the deduplication efficiency. The experimental verification results in this paper show that the proposed SemCluster succeeds in clustering of crowdsourced test reports and solving the problem of the duplication of crowdsourced test reports.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for insightful comments. This research is partially supported by the National Natural Science Foundation of China (61932012, 62141215), Science, Technology and Innovation Commission of Shenzhen Municipality (CJGJZD20200617103001003) and Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX22_0174).

REFERENCES

[1] Anahita Alipour, Abram Hindle, and Eleni Stroulia. 2013. A contextual approach towards more accurate duplicate bug report detection. In *2013 10th Working Conference on Mining Software Repositories*. 183–192.
[2] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhut, Guoqiang Li, and Jinshui Wang. 2020. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In *2020 IEEE/ACM 42nd International Conference on Software Engineering*. 322–334.
[3] Nathan Cooper, Carlos Bernal-Cárdenas, Oscar Chaparro, Kevin Moran, and Denys Poshyvanyk. 2021. It takes two to tango: Combining visual and textual information for detecting duplicate video-based bug reports. In *2021 IEEE/ACM 43rd International Conference on Software Engineering*. 957–969.
[4] Abram Hindle, Anahita Alipour, and Eleni Stroulia. 2016. A contextual approach towards more accurate duplicate bug report detection and ranking. *Empirical Software Engineering* 21, 2 (2016), 368–410.
[5] He Jiang, Xin Chen, Tieke He, Zhenyu Chen, and Xiaochen Li. 2018. Fuzzy clustering of crowdsourced test reports for apps. *ACM Transactions on Internet Technology* 18, 2 (2018), 1–28.
[6] Di Liu, Yang Feng, Xiaofang Zhang, James Jones, and Zhenyu Chen. 2020. Clustering crowdsourced test reports of mobile applications using image understanding. *IEEE Transactions on Software Engineering* (2020).
[7] Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N Nguyen, David Lo, and Chengnian Sun. 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *2012 Proceedings of the 27th IEEE/ACM international conference on automated software engineering*. 70–79.
[8] Tomi Prifti, Sean Banerjee, and Bojan Cukic. 2011. Detecting bug duplicate reports through local references. In *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*. 1–9.
[9] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[10] Junjie Wang, Mingyang Li, Song Wang, Tim Menzies, and Qing Wang. 2019. Images don't lie: Duplicate crowdtesting reports detection with screenshot information. *Information and Software Technology* 110 (2019), 139–155.
[11] Shengcheng Yu, Chunrong Fang, Zhenfei Cao, Xu Wang, Tongyu Li, and Zhenyu Chen. 2021. Prioritize crowdsourced test reports via deep screenshot understanding. In *2021 IEEE/ACM 43rd International Conference on Software Engineering*. 946–956.