IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0000000

LLM-Guided Crowdsourced Test Report Clustering

YING LI[1], YE ZHONG[2], LIJUAN YANG[3], YANBO WANG[4], PENGHUA ZHU[5]

[1]North China Institute of Aerospace Engineering, Beijing, China (e-mail: lan_yanjing@qq.com)
[2]Dalian University of Technology, Dalian, China (e-mail: zhongye_0712@163.com)
[3]North China Institute of Aerospace Engineering, Beijing, China (e-mail: ylj23@nciae.edu.cn)
[4]Beijing Aerospace Automatic Control Institute, Beijing, China (e-mail: wangyb1@126.com)
[5]North China Institute of Aerospace Engineering, Beijing, China (e-mail: zhupenghua_nciae@163.com)

Corresponding author: Penghua Zhu (e-mail: zhupenghua_nciae@163.com).

This work was supported by Hebei Province Higher Education Science and Technology Research Project (CXY2023013).

ABSTRACT

This paper proposes a clustering method for crowdsourced test reports based on a large language model to solve the limitations of existing methods in processing repeated reports and utilizing multi-modal information. Existing crowdsourced test report clustering methods have significant shortcomings in handling duplicate reports, ignoring the semantic information of screenshots, and underutilizing the relationship between text and images. The emergence of LLM provides a new way to solve these problems. By integrating the semantic understanding ability of LLM, key information can be extracted from the test report more accurately, and the semantic relationship between screenshots and text descriptions can be used to guide the clustering process, thus improving the accuracy and effectiveness of clustering. The method in this paper uses a pre-trained LLM (such as GPT-4) to encode the text in the test report, and uses a visual model such as CLIP to encode the application screenshots, converting the text descriptions and images into high-dimensional semantic vectors. The cosine similarity is then used to calculate the similarity between the vectors, and semantic binding rules are constructed to guide the clustering process, ensuring that semantically related reports are assigned to the same cluster and semantically different reports are assigned to different clusters. Through experimental verification, this method is significantly superior to traditional methods in several evaluation indicators, demonstrating its great potential in improving the efficiency and quality of crowdsourced test report processing. In the future, this method is expected to be widely used in the process of software testing and maintenance, and further promote technological progress.

INDEX TERMS Large Language Model, Crowdsourced Testing, Test Report Clustering

I. INTRODUCTION

THE existing clustering methods for crowdsourced testing reports exhibit notable limitations in addressing duplicate reports, particularly in harnessing the semantic information from screenshots and textual descriptions. The emergence of large language models (LLMs) offers fresh perspectives for tackling this issue. By integrating the semantic comprehension capabilities of LLMs, we can more precisely extract critical information from testing reports and leverage the semantic relationships between screenshots and textual descriptions to guide the clustering process, thereby enhancing the accuracy and effectiveness of clustering. Therefore, this paper introduces a crowdsourced testing report clustering approach based on large language models, which fully exploits the multimodal information in testing reports, such as the semantic associations between textual descriptions and application screenshots. This approach aims to overcome the limitations of existing methods and is experimentally validated for its effectiveness. Not only does this method improve the processing efficiency of testing reports, but it also provides developers with more precise error detection and remediation support, thereby advancing the application and development of crowdsourced testing in software development processes.

II. RESEARCH BACKGROUND & MOTIVATION

A. THE SIGNIFICANCE OF SOFTWARE TESTING

As the complexity of software systems continues to escalate, ensuring software quality has become paramount. Software testing serves not only as a means to guarantee the functional

integrity of software but also as a vital link in ensuring its reliability, security, and performance. High-quality software not only fulfills user needs and enhances user satisfaction but also mitigates maintenance costs and bolsters enterprises' competitive edge in the market.

Firstly, software testing forms the cornerstone for ensuring the proper functioning of software. Development processes inherently introduce a myriad of errors and defects, which, if undetected and unrepaired, can lead to software failures in real-world operations. Through systematic testing, these issues can be identified early in the development phase, preventing them from impacting user experience post-release. Early detection and resolution of defects minimize repair costs, prevent problem proliferation, and enhance development efficiency.

Secondly, software testing plays a pivotal role in safeguarding software security. With the proliferation of the internet and rapid advancements in information technology, software security challenges have become increasingly severe. Security incidents such as hacking attacks, data breaches, and system intrusions pose grave threats to individual privacy and corporate data. Security testing can uncover vulnerabilities and potential threats within software, enabling timely remediation measures to enhance software security and protect user data and privacy.

Furthermore, performance testing is another essential aspect of software testing. As user demands escalate and competition intensifies, software performance becomes a crucial factor influencing user choices. Performance testing assesses software's behavior under varying loads and stress conditions, ensuring stable operation even under high concurrency and heavy traffic. It identifies and optimizes performance bottlenecks, enhancing system response speed and processing capabilities, thereby raising user satisfaction and market competitiveness.

Software testing's significance extends beyond the development phase into software maintenance and updates. As software versions iterate and update, the introduction of new features and optimization of existing ones can introduce new defects or cause anomalies in existing functions. Consequently, regression testing and comprehensive testing before each version update are crucial to ensure the stability and reliability of new versions.

In conclusion, software testing holds a pivotal position throughout the software development lifecycle. It ensures software's functionality, reliability, security, and performance, elevating user experience and satisfaction, minimizing maintenance costs, and fortifying enterprises' market competitiveness. As software systems become more complex and user demands proliferate, the importance of software testing will become even more pronounced, serving as a cornerstone for safeguarding software quality and driving technological advancements.

B. ADVANTAGES AND CHALLENGES OF CROWDSOURCED TESTING

Crowdsourced testing, as an emerging paradigm in software testing, has gained widespread adoption in the software development domain due to its openness and flexibility. Unlike traditional in-house testing, crowdsourced testing distributes testing tasks to a vast pool of external testers, leveraging their diverse devices and environments, thereby achieving broader test coverage and heightened efficiency. However, alongside its numerous advantages, crowdsourced testing also confronts distinct challenges.

First of all, the foremost advantage of crowdsourced testing lies in its extensive coverage. Traditional in-house testing is often constrained by the limited number of testers and the diversity of testing environments, making it challenging to encompass all potential usage scenarios. Conversely, crowdsourced testing, by assigning tasks to testers worldwide, enables testing across various hardware devices, operating systems, and network environments, uncovering more latent issues and defects. This diversity significantly enhances the comprehensiveness and effectiveness of testing, ensuring software stability and compatibility in real-world usage.

In addition, crowdsourced testing can substantially reduce testing costs. Compared to assembling a dedicated in-house testing team, crowdsourced testing relies on external testers who are paid based on task completion rather than fixed salaries and benefits. This pay-per-task model significantly lowers the testing expenses for enterprises, making high-quality testing services accessible even to small and medium-sized enterprises.

Furthermore, crowdsourced testing boasts high flexibility and adaptability. Testers can select tasks based on their availability and interests, unrestrained by fixed working hours or locations. This flexibility allows crowdsourced testing to swiftly adapt to diverse project requirements, particularly for time-sensitive projects such as game testing and mobile app testing, where it can swiftly complete numerous tasks, ensuring timely product releases.

However, alongside its benefits, crowdsourced testing faces unique challenges. Firstly, its openness leads to varying quality in test reports. Given the disparate technical proficiencies and experiences of testers, submitted reports may suffer from unclear descriptions, incomplete information, or even duplicate reports. Notably, the duplication rate in crowdsourced testing reports can reach up to 82%. Duplicate reports waste developers' time and effort, potentially obscuring critical issues amidst the flood of redundant information, and hindering timely resolution. Thus, effectively filtering and processing crowdsourced test reports to ensure their quality is a pressing issue.

Secondly, security and privacy concerns are paramount in crowdsourced testing. Distributing software products to external testers can expose risks of intellectual property and data breaches. For applications involving sensitive data, such as financial and medical software, safeguarding user data and corporate secrets during testing is crucial.

Additionally, coordination and management pose a significant challenge in crowdsourced testing. With testers dispersed globally, effectively assigning tasks, tracking progress, evaluating results, and promptly addressing issues necessitates intricate management. Lack of effective coordination can lead to task delays and resource waste, impacting testing efficiency and outcomes.

To address these challenges, crowdsourced testing necessitates the integration of advanced technologies and management methodologies. For instance, leveraging automation tools and big data analytics can refine test report filtering and processing, enhancing quality and efficiency. Establishing robust security mechanisms and privacy safeguards can protect the testing process, preventing data and intellectual property breaches. Incorporating project management tools and collaboration platforms can elevate coordination and management efficiency in crowdsourced testing, ensuring timely and high-quality task completion.

C. RESEARCH MOTIVATION

1) Limitations of Current Crowdsourced Testing Report Clustering Methods

In the realm of software development, crowdsourced testing significantly enhances the breadth and depth of testing by harnessing the collective resources of numerous external testers. Nevertheless, the current clustering methods employed for crowdsourced testing reports exhibit numerous limitations that hinder their full potential.

Neglect of Semantic Information in Screenshots: Beyond textual descriptions, crowdsourced testing reports commonly include screenshots of applications. These screenshots contain abundant visual information and contextual cues that provide vital clues about the reported issues. Nonetheless, many existing clustering methods treat screenshots merely as pixel data, disregarding their semantic content. For instance, visually similar screenshots may represent distinct issues, while dissimilar ones could depict the same problem. Disregarding such semantic information leads to inaccurate clustering results and fails to effectively group similar reports.

Inadequate Exploitation of Text-Image Relationships: There exists a profound connection between application screenshots and their accompanying textual descriptions. This relationship offers a more comprehensive contextual perspective, which could enhance clustering outcomes. Yet, numerous existing methods treat text and images as independent entities, failing to leverage their semantic correlations. Neglecting these interdependencies undermines the utilization of all available information within the reports, compromising the reliability and precision of clustering results.

Insufficient Handling of Duplicate Reports: Duplicate reports are a pervasive issue in crowdsourced testing. Given the independence of testers, it is common for multiple testers to report the same issue. Current clustering methods often resort to simplistic categorization based on superficial features when dealing with duplicates, struggling to distinguish between reports that genuinely share the same underlying cause. This not only exacerbates the review burden for developers but also risks drowning critical issues amidst a deluge of redundant information, hindering timely resolution.

Limitations of Clustering Algorithms: Contemporary clustering algorithms, such as K-means and hierarchical clustering, excel in handling structured data but encounter limitations when dealing with high-dimensional and unstructured data (e.g., text and images). These algorithms are often sensitive to initial parameters and distance metrics, leading to unstable clustering results. Moreover, their high computational complexity for large-scale data renders them inadequate for meeting real-time requirements in practical applications.

2) Development and Application Potential of Large Language Models

With the rapid advancements in deep learning technologies, Large Language Models (LLMs) have achieved remarkable progress in the field of Natural Language Processing (NLP). By training on vast datasets, LLMs demonstrate a formidable capacity to comprehend and generate natural language, showcasing their robust linguistic processing abilities.

LLMs excel in natural language understanding tasks. BERT (Bidirectional Encoder Representations from Transformers), a pre-trained model introduced by Google, leverages a bidirectional Transformer architecture to capture contextual information within sentences. In tasks such as reading comprehension, question-answering systems, and text classification, BERT and its variants have achieved outstanding results. Furthermore, LLMs can be applied to tasks including sentiment analysis, named entity recognition, and relation extraction. By pre-training on large-scale corpora and fine-tuning for specific tasks, these models acquire extensive linguistic knowledge and task-related features, thereby enhancing task accuracy and robustness.

Large language models have also made significant progress in natural language generation tasks. GPT-4 (Generative Pre-trained Transformer 4), proposed by OpenAI, is a generative model that leverages a unidirectional Transformer architecture to generate coherent natural language paragraphs based on input text. As the latest iteration in the series, GPT-4 boasts an even larger parameter scale and more robust text generation capabilities. Large language models find applications in text generation, dialogue systems, machine translation, and more. For instance, GPT-4 is capable of producing high-quality news articles, technical documentation, and literary works, while also engaging in human-like conversations and providing answers to questions. Its formidable generative power presents novel opportunities for automated content creation, intelligent customer service, and other applications.

Beyond their excellence in NLP, LLMs exhibit immense potential in multimodal processing. By integrating textual and visual information, these models enable cross-modal understanding and generation. CLIP (Contrastive Language-Image Pretraining), a multimodal model proposed by OpenAI, accomplishes tasks such as image classification, image generation, and image-text retrieval through joint training of text and

images. The application prospects of LLMs in multimodal processing are vast. For instance, in autonomous driving, combining visual and linguistic information enables more precise environmental perception and decision-making. In healthcare, analyzing medical images and textual records aids in diagnosis and treatment. In smart homes, voice commands and image recognition facilitate control and management of intelligent devices.

Based on the aforementioned remarkable performances of LLMs in natural language understanding, generation, and multimodal processing, they hold immense application potential in crowdsourced test report clustering. Firstly, LLMs can extract rich semantic features from textual descriptions in test reports, capturing intricate details described by testers. Concurrently, when integrated with image recognition technologies, these models can extract valuable information from screenshots, such as interface elements and error messages. The fusion of these multimodal features significantly enhances the representational power of test reports, providing a more accurate foundation for clustering. Moreover, LLMs' ability to capture semantic associations between text and images facilitates the identification of reports describing the same issues but phrased differently. This capability is crucial in handling duplicate and similar reports, as it enables more precise categorization of similar reports, alleviating developers' workload and enhancing issue resolution efficiency.

D. RESEARCH QUESTIONS AND MAIN CONTRIBUTIONS

As software development complexity escalates, crowdsourced testing has emerged as a vital means of identifying and rectifying software defects. By harnessing the resources of a vast pool of external testers, crowdsourced testing offers a diverse array of testing environments and device configurations. Nevertheless, it also introduces a deluge of duplicative and redundant test reports, significantly burdening developers' review processes. To enhance the efficiency in processing crowdsourced test reports, clustering techniques have been widely applied to group similar reports together. However, existing clustering methods for crowdsourced test reports exhibit notable limitations, struggling to fully leverage the multimodal information within the reports, such as the semantic correlations between textual descriptions and application screenshots.

Large Language Models (LLMs), exemplified by GPT-4, have demonstrated formidable capabilities in the realm of Natural Language Processing (NLP). By training on massive datasets, LLMs can comprehend and generate natural language texts, excelling in various NLP tasks. Leveraging LLMs' semantic understanding and generation abilities, we can more precisely extract key information from test reports and integrate screenshot content for comprehensive analysis, thereby enhancing clustering accuracy and efficiency. The main contributions of this paper are as follows:

1) A crowdsourced test report clustering method based on Large Language Models is proposed, which extracts features from both application screenshots and textual descriptions.
2) Semantic binding rules are formulated utilizing the semantic relationships between screenshots and textual descriptions to guide the clustering process.
3) The effectiveness and superiority of the proposed method are experimentally validated across multiple evaluation metrics.

III. METHODOLOGY

A. FEATURE EXTRACTION

Feature Extraction is a pivotal step in achieving effective clustering, primarily involving the extraction of both image and textual features from application screenshots and textual descriptions within crowdsourced testing reports. In this methodology, we harness the prowess of advanced large language models (LLMs) for feature extraction, aiming to derive richer and more precise semantic information. Specifically, we employ a pre-trained LLM, GPT-4, for text encoding, and a pre-trained visual model, CLIP, for encoding application screenshots, thereby transforming both textual descriptions and images into high-dimensional semantic vectors.

Textual Feature Extraction: LLMs exhibit formidable capabilities in text comprehension and representation. These models are leveraged to process the textual descriptions within crowdsourced testing reports, whereby each segment of text is fed into a pre-trained LLM to extract its high-dimensional semantic vector representation. GPT-4, with its extensive contextual understanding and generation abilities, adeptly captures intricate semantic relationships. By utilizing these models, every sentence and paragraph within the textual descriptions can be transformed into a high-dimensional vector, accurately portraying their semantic content. These high-dimensional vectors then facilitate subsequent similarity computations and clustering tasks.

Image Feature Extraction: For image feature extraction, pre-trained visual models are employed. CLIP, introduced by OpenAI, is a model that jointly trains on image and text data, enabling the mapping of both modalities into a shared semantic space. When applying CLIP, application screenshots from crowdsourced testing reports are input into the model, yielding their corresponding vector representations in the semantic space. Compared to traditional convolutional neural networks (e.g., VGG-16), CLIP excels in capturing semantic information within images due to its consideration of the interplay between images and texts. This approach yields image vectors enriched with semantic information, facilitating subsequent similarity computations and clustering procedures.

B. SIMILARITY CALCULATION

Having obtained the semantic vector representations for both images and texts, we proceed to calculate their pairwise similarities. Cosine similarity is a commonly employed method for this purpose, owing to its suitability in high-dimensional vector spaces. In this approach, cosine similarity is utilized

to compute the degree of similarity between the vectors. Cosine similarity measures the angle between two vectors; the smaller the angle, the higher the similarity. The formula for cosine similarity is as follows:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{1}$$

where $\mathbf{A} \cdot \mathbf{B}$ represents the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$, $\|\mathbf{A}\|$ represents the norm (length) of vector $\mathbf{A}$, and $\|\mathbf{B}\|$ represents the norm of vector $\mathbf{B}$.

By adopting this method, we can derive the semantic similarity between application screenshots and textual descriptions, providing foundational data for subsequent clustering processes. This semantic similarity serves as a crucial input, enabling the grouping of related items based on their underlying meanings, rather than mere superficial similarities.

C. CONSTRUCTION OF SEMANTIC BINDING RULES

The profound semantic comprehension ability of large models enables them to more accurately capture the semantic relationships between application screenshots and textual descriptions. These models can be leveraged to devise more effective semantic binding rules. In this methodology, we employ two types of constraints to guide the clustering process, which take into account both application screenshots, textual descriptions, and their mutual semantic relationships. The two types of rules used to specify these relationships are:

- ***YES-LINK***: Indicates that two data instances (reports) should be assigned to the same cluster.
- ***NO-LINK***: Indicates that two data instances (reports) should be allocated to different clusters.

These rules possess transitivity properties. If Report A and Report B are YES-LINK, and Report B and Report C are also YES-LINK, then Report A and Report C must necessarily be YES-LINK; similarly, if Report A and Report B are YES-LINK, while Report B and Report C are NO-LINK, Report A and Report C should be NO-LINK, and so forth.

Large models can be utilized to perform semantic matching for every pair of test reports, and based on the aforementioned rules, the relationships between reports can be formulated. For instance, if the similarity of the combined feature vectors of two reports exceeds a certain threshold, they can be labeled as YES-LINK; conversely, if the similarity falls below a threshold, they are labeled as NO-LINK. The application of these rules in conjunction with the semantic matching capabilities of large models can significantly enhance the accuracy and consistency of the clustering process.

D. REPORT CLUSTERING

During the clustering process, an initial set of semantic binding rules is constructed. For all pairs of reports, their similarities are queried, and the semantic binding rules are applied, ultimately resulting in a collection of report pairs categorized as YES-LINK and NO-LINK. In each iteration of the clustering, the set of semantic binding rules is utilized to verify any instances that violate these rules, and subsequent re-clustering is performed accordingly. The K-Medoids algorithm, a variant of clustering algorithms, is employed in this methodology. The specific clustering procedure involves:

1) Determining the Number of Clusters (K).
2) Randomly selecting K initial medoids and verifying them against the semantic rules to ensure no instances violating the must-link and cannot-link rules coexist within the initial set of medoids.
3) In each iteration, calculating the distances between all non-medoid instances and the medoids, and applying the set of semantic binding rules for verification and adjustment.
4) Termination of iterations occurs when all instances satisfy the semantic binding rules, and the medoids remain unchanged, indicating a stable clustering result.

Compared to the K-Means algorithm, the K-Medoids clustering algorithm maintains actual test reports as the cluster centers (medoids) during the iterative clustering process, thereby avoiding the potential deviation of centroids from actual data points.

In summary, the workflow of the method can be summarized as Figure 1.

IV. EXPERIMENT

To evaluate the effectiveness of the Large Language Model (LLM)-guided crowdsourced test report clustering method, researchers formulated three research questions and conducted experiments on a dataset comprising 847 reports. The experimental results demonstrated the superiority of the proposed method over baseline approaches and different configurations across various evaluation metrics.

A. RESEARCH QUESTIONS

Three questions were posed to assess the effectiveness of the LLM-guided crowdsourced test report clustering method:

- **RQ1**: How does the proposed method perform compared to baseline methods?
- **RQ2**: How does different components contribute to the proposed method?
- **RQ3**: Are there significant differences in clustering results under different prompts?
- **RQ4**: What is the time overhead of the proposed method?

B. TEST SUBJECTS

The dataset used in the experiments encompassed 847 crowdsourced test reports from 18 mobile applications, spanning diverse application categories such as finance, communication, lifestyle, etc. The number of reports per application ranged from 4 to 152. Table 1 provides detailed information about the dataset.

The dataset was sourced from one of the most popular and representative crowdsourced testing platforms in China. This platform offers crowdsourced test reports for real-world

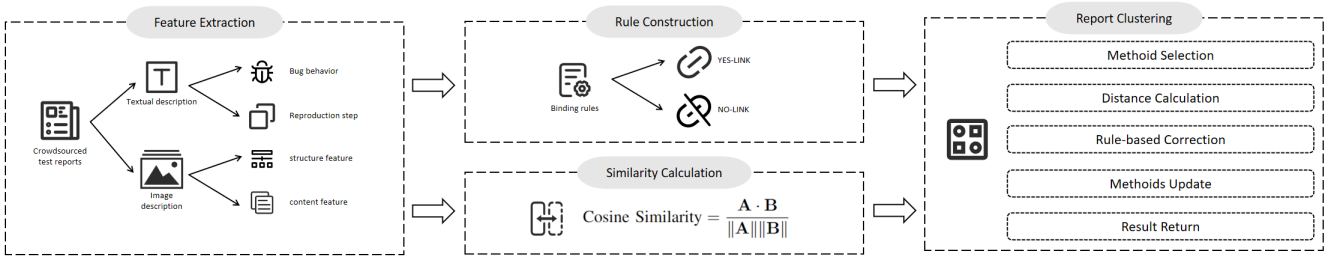

FIGURE 1: Method Workflow

TABLE 1: Apps and Test Reports under Investigation

App	Domain	# Report	Bug Category	Report / Category
A1	Finance	134	9	14.89
A2	System	29	6	4.83
A3	Business	13	3	4.33
A4	Reading	13	3	4.33
A5	Reading	26	4	6.50
A6	Reading	152	8	19.00
A7	Reading	41	4	10.50
A8	System	75	5	15.00
A9	Finance	26	5	5.20
A10	Life	5	3	1.67
A11	Finance	10	2	5.00
A12	Travel	17	17	1.00
A13	Communi.	131	8	16.38
A14	Travel	88	15	5.87
A15	Music	51	8	6.38
A16	Education	4	2	2.00
A17	Life	12	3	4.00
A18	System	24	5	4.80
Sum / Avg		847	109	6.93

applications and has supported numerous academic studies. To ensure the representativeness of the dataset, researchers invited three experts with extensive experience in mobile application development and testing to annotate the reports.

C. EVALUATION INDICATORS

To evaluate the effectiveness of the Large Language Model (LLM)-guided crowdsourced test report clustering method, researchers employed multiple evaluation metrics, including Precision, Recall, F1-measure, Purity, Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI).

Precision: The proportion of correctly identified reports among the clustering results.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall: The proportion of actual correct reports that are correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-measure: The harmonic mean of Precision and Recall, offering a balanced perspective.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Purity: The extent to which a cluster contains a single class. Let N be the total number of samples, C_k represent the kth cluster, L_j the jth true class, and $|C_k \cap L_j|$ the size of the intersection between cluster C_k and true class L_j.

$$\text{Purity} = \frac{1}{N} \sum_{k} \max_{j} |C_k \cap L_j| \tag{5}$$

Adjusted Rand Index (ARI): The Adjusted Rand Index (AR) is used to evaluate the quality of clustering, adjusting for chance.

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \tag{7}$$

Normalized Mutual Information (NMI): Normalized Mutual Information measures the consistency between the clustering results and the true classes. Specifically, $I(\Omega; C)$ represents the Mutual Information, calculated as: $I(\Omega; C) = H(\Omega) - H(\Omega \mid C)$, where $H(\Omega)$ and $H(C)$ are the entropies of the clustering results and true classes, respectively.

$$NMI = \frac{2 \cdot I(U; V)}{H(U) + H(V)} \tag{8}$$

D. EXPERIMENTAL RESULTS

In this section, we present the experimental results of an LLM-directed crowdsourcing test designed to evaluate the effectiveness of various clustering methods. The detailed experimental data can be found in Table 2 (where "Deep" refers to DeepPrior and "LLM" refers to LLM-guided), while a summary is provided in Table 3. Figure 2 shows a t-SNE visualization of the clustering outcomes, clearly illustrating the distinct separation of clusters, which highlights the effectiveness of our methods in clustering crowdsourced test reports.

For RQ1, the LLM-guided crowdsourced test report clustering method shows significant improvement over the benchmark methods SETU and DeepPrior. Specifically, the LLM-directed crowdsourced test report clustering method showed significant improvements in accuracy, recall, F1 values, purity, ARI, and NMI. For example, the accuracy and recall rates of the method are 0.86 and 0.92, respectively, while SETU is 0.23 and 0.26, and DeepPrior is only 0.31 and 0.42, respectively. In terms of purity, the method is 0.89, which is 71.2%

App	SETU	Deep	LLM
A1	0.21	0.30	0.90
A2	0.15	0.23	0.88
A3	0.30	0.38	0.82
A4	0.28	0.37	0.84
A5	0.18	0.38	0.93
A6	0.18	0.20	0.92
A7	0.41	0.40	0.81
A8	0.14	0.29	0.91
A9	0.32	0.41	0.76
A10	0.31	0.43	0.83
A11	0.18	0.20	0.91
A12	0.20	0.20	0.89
A13	0.25	0.21	0.81
A14	0.15	0.25	0.80
A15	0.21	0.38	0.91
A16	0.31	0.48	0.83
A17	0.12	0.40	0.89
A18	0.25	0.40	0.83

(a) Precision

App	SETU	Deep	LLM
A1	0.25	0.63	0.93
A2	0.28	0.58	0.89
A3	0.28	0.34	0.84
A4	0.33	0.35	0.85
A5	0.12	0.18	0.83
A6	0.39	0.59	0.98
A7	0.30	0.56	0.97
A8	0.10	0.20	0.80
A9	0.13	0.34	0.96
A10	0.28	0.57	0.99
A11	0.22	0.20	0.85
A12	0.25	0.28	0.92
A13	0.27	0.38	0.95
A14	0.31	0.37	0.80
A15	0.09	0.18	0.82
A16	0.30	0.58	0.99
A17	0.37	0.64	0.89
A18	0.40	0.59	0.82

(b) Recall

App	SETU	Deep	LLM
A1	0.23	0.40	0.91
A2	0.20	0.33	0.88
A3	0.29	0.36	0.83
A4	0.23	0.36	0.84
A5	0.15	0.25	0.88
A6	0.26	0.29	0.95
A7	0.34	0.47	0.88
A8	0.12	0.23	0.85
A9	0.17	0.37	0.85
A10	0.34	0.49	0.91
A11	0.20	0.20	0.88
A12	0.22	0.24	0.90
A13	0.26	0.27	0.88
A14	0.21	0.30	0.80
A15	0.13	0.25	0.86
A16	0.34	0.53	0.91
A17	0.18	0.49	0.89
A18	0.30	0.48	0.83

(c) F1-Measure

App	SETU	Deep	LLM
A1	0.42	0.55	0.92
A2	0.72	0.85	0.92
A3	0.46	0.46	0.87
A4	0.36	0.66	0.86
A5	0.48	0.57	0.84
A6	0.31	0.40	0.85
A7	0.65	0.83	0.91
A8	0.58	0.71	0.84
A9	0.60	0.63	0.87
A10	0.48	0.56	0.88
A11	0.33	0.50	0.88
A12	0.48	0.53	0.88
A13	0.57	0.42	0.93
A14	0.74	0.81	0.89
A15	0.72	0.73	0.87
A16	0.37	0.39	0.91
A17	0.56	0.70	0.95
A18	0.51	0.51	0.96

(d) Purity

App	SETU	Deep	LLM
A1	-0.02	0.08	0.91
A2	0.01	0.03	0.89
A3	0.01	0.16	0.99
A4	-0.02	0.07	0.94
A5	-0.02	0.12	0.98
A6	0.01	0.13	0.96
A7	0.01	0.14	0.98
A8	-0.03	-0.02	0.97
A9	-0.01	0.00	0.92
A10	0.01	0.02	0.98
A11	-0.03	0.13	0.89
A12	-0.01	0.18	0.90
A13	0.03	0.07	0.98
A14	-0.03	0.04	0.97
A15	-0.01	0.13	0.93
A16	-0.04	-0.01	0.95
A17	-0.01	0.18	0.94
A18	-0.02	0.19	0.94

(e) ARI

App	SETU	Deep	LLM
A1	0.32	0.42	0.97
A2	0.35	0.47	0.98
A3	0.20	0.41	0.92
A4	0.20	0.50	0.97
A5	0.22	0.47	0.90
A6	0.25	0.32	0.94
A7	0.30	0.39	0.96
A8	0.37	0.48	0.90
A9	0.20	0.27	0.95
A10	0.27	0.33	0.97
A11	0.31	0.38	0.96
A12	0.37	0.46	0.91
A13	0.22	0.36	0.98
A14	0.20	0.38	0.94
A15	0.26	0.30	1.00
A16	0.35	0.37	0.94
A17	0.23	0.28	0.92
A18	0.42	0.45	0.97

(f) NMI

App	SETU	Deep	LLM
A1	20.9s	19.4s	12.5s
A2	8.6s	8.1s	3.7s
A3	7.1s	6.3s	1.8s
A4	7.2s	6.4s	1.8s
A5	8.4s	7.7s	2.9s
A6	23.3s	19.1s	16.4s
A7	10.2s	9.5s	4.3s
A8	12.6s	13.0s	7.8s
A9	8.6s	7.6s	3.1s
A10	6.4s	5.6s	0.9s
A11	6.8s	6.2s	1.4s
A12	7.7s	6.6s	2.1s
A13	18.6s	19.4s	12.0s
A14	15.7s	14.1s	10.1s
A15	10.6s	10.0s	5.8s
A16	6.3s	5.5s	0.8s
A17	7.2s	6.3s	1.6s
A18	8.1s	7.4s	2.9s

(g) Time overhead

TABLE 2: Detailed Experiment Results for RQ1

TABLE 3: Summary of Experiment Results for RQ1

	SETU	DeepPrior	LLM-guided
Precesion	0.23	0.31	0.86
Recall	0.26	0.42	0.92
F1-measure	0.24	0.34	0.89
Purity	0.52	0.60	0.89
ARI	-0.01	0.09	0.95
NMI	0.28	0.39	0.95
Time overhead	10.8s	9.9s	5.1s

TABLE 4: Summary of Experiment Results for RQ2

	LLM-guided(T)	LLM-guided(I)	LLM-guided
Precesion	0.74	0.77	0.86
Recall	0.68	0.84	0.92
F1-measure	0.76	0.80	0.89
Purity	0.67	0.69	0.89
ARI	0.82	0.71	0.95
NMI	0.79	0.69	0.95

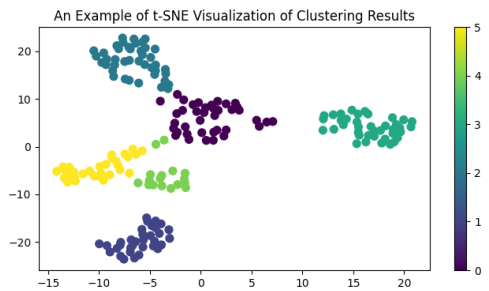

FIGURE 2: Clustering Results Visualization

higher than SETU and 48.3% higher than DeepPrior. To assess the statistical significance of the performance improvement over other methods, we apply the Wilcoxon signed-rank test. The p-values for our method, when compared to SETU and DeepPrior across seven metrics—precision, recall, F1-measure, purity, ARI, NMI, and time overhead—are all on the order of 7×10^{-6}, which is well below the threshold of 0.05. This demonstrates that the improvement in clustering effectiveness achieved by our method is statistically significant. These results show that LLM-guided method has excellent performance in forecasting accuracy, clustering consistency and information capture, and has high practical value and reliability.

For RQ2, we analyze the contributions of different components to the proposed method by comparing the performance of three configurations: LLM-guided(T) (using only text features), LLM-guided(I) (using only image features), and LLM-guided (using both text and image features). The results presented in Table 4 show that integrating both text and image features provides the best performance across all evaluation metrics. For Precision, Recall, and F1-measure, the LLM-guided method (using both features) outperforms the individual feature-based approaches. F1-measure improves from 0.76 (LLM-guided(T)) and 0.80 (LLM-guided(I)) to 0.89 (LLM-guided), indicating that combining the two features leads to more accurate results. Similarly, Purity, ARI, and NMI also show clear improvements when both features are used. Purity rises from 0.67 (LLM-guided(T)) and 0.69 (LLM-guided(I)) to 0.89 (LLM-guided), suggesting better clustering quality. Both ARI and NMI increase to 0.95 in the combined model, compared to 0.82 and 0.79 for LLM-guided(T), and 0.71 and 0.69 for LLM-guided(I). These results demonstrate that integrating text and image features enhances the method's performance, making it more effective at clustering and improving overall accuracy.

For RQ3, different prompt has some effect on the final clustering result. Detailed and specific prompt can significantly improve the quality of feature extraction, semantic binding rule generation and application, and reported clustering processes, thereby improving the accuracy and consistency of the final clustering results. On the contrary, simple and vague prompt can lead to uncertainty in the individual steps and instability in the quality of the results. Therefore, in practical applications, providing as much detailed and specific guidance as possible is critical to ensuring high-quality clustering results.

For RQ4, the LLM-guided crowdsourced test report clustering method has a lower time overhead compared to the benchmark method. While the benchmark methods SETU and DeepPrior both have a time overhead of around 10 seconds, the LLM-guided method takes only 5.7 seconds to classify each report, which is a significant improvement in clustering speed.

E. ERROR ANALYSIS

While the proposed method demonstrates significant improvements over existing approaches, there are several potential failure cases and limitations that should be considered. The method assumes that the semantic similarity between reports is best represented by cosine similarity in the high-dimensional embedding space. However, this metric may not fully capture all nuances of report relationships, particularly in cases where the reports exhibit highly nonlinear or context-dependent similarities. For example, reports with similar content but slightly different wording or those containing non-standard screenshots might be misclustered or split across multiple clusters. Additionally, the proposed method assumes that the input data (text and images) are of sufficient quality. In real-world crowdsourcing environments, however, test reports may often be noisy, incomplete, or contain errors, such as poorly cropped images or ungrammatical text. These issues can significantly degrade the performance of the model, resulting in suboptimal clustering outcomes. Future research should aim to address these limitations by exploring alternative similarity measures that better account for the inherent complexity of test reports, and developing more adaptive clustering mechanisms that can better handle noisy or incomplete data.

F. THREATS TO VALIDITY

When evaluating the crowdsourced test report clustering method based on Large Language Models (LLMs), we must meticulously consider several potential threats that may impact the validity of our research.

Firstly, the representativeness of the dataset can significantly influence the study's outcomes. The dataset employed in this research comprises 847 test reports from 18 different mobile applications, spanning 109 bug categories. While these datasets aim to reflect a wide range of application scenarios, their diversity and representativeness may still fall short of comprehensively representing all possible testing environments. Consequently, future studies should incorporate more diversified and larger-scale datasets to validate the universality and robustness of the proposed method.

Secondly, the randomness inherent in the clustering process can introduce uncertainty. As clustering algorithms exhibit randomness when dealing with large-scale and complex data, we mitigate this by running each algorithm 10 times to eliminate the adverse effects of randomness. This approach reduces biases stemming from single-run random factors, enhancing the stability and reliability of the results.

Thirdly, the annotation quality of crowdsourced test reports poses a crucial validity threat. Annotation relies on testers' professional knowledge and experience, which can lead to variation in inter-annotator agreement. To mitigate this threat, we invited experts with extensive experience in mobile application development and testing to conduct independent annotations and employed cross-validation to ensure consistency. In cases of annotation discrepancy, annotators engaged in discussions to reach a consensus, thereby ensuring data accuracy and reliability.

Furthermore, the language issue of the dataset also constitutes a potential threat. Our dataset primarily comprises Chinese reports, which may affect the model's performance on data in other languages. However, our baseline models (including SETU [1] and DeepPrior [2]) also handle similarly distributed data, featuring primarily Chinese with some interspersed English. Additionally, modern natural language processing techniques excel in multilingual settings, suggesting that substituting the feature extraction with NLP models tailored for other languages would likely have minimal impact on validity. Hence, we did not undertake additional language optimizations for the baseline models.

Lastly, the computational resource requirements and operational efficiency of LLMs are concerns that warrant attention. Large-scale language models typically necessitate substantial computational resources and storage space, potentially encountering performance bottlenecks in practical applications. When processing massive test reports, the model's real-time response and speed directly influence its practicality and scalability. Consequently, optimizing the model's architecture and algorithms to improve computational efficiency represents an essential direction for future research.

G. DISCUSSION ON SCALABILITY

The scalability of our proposed clustering method is a key consideration for its application to large datasets. The method leverages pre-trained large language models for text encoding and visual models for encoding images, both of which are highly optimized for efficient performance. These models can process large volumes of data in parallel during the encoding phase, making them suitable for handling substantial datasets. Once the text and image data are encoded into high-dimensional semantic vectors, the cosine similarity calculation, which is computationally efficient, enables effective clustering even as the dataset size increases.

While the initial encoding process may demand significant computational resources, this challenge can be mitigated by distributing the workload or utilizing cloud-based infrastructure. This parallel processing capability ensures that the method can scale to larger datasets without compromising performance. Furthermore, the simplicity and efficiency of cosine similarity calculations allow for the seamless expansion of the method to handle large-scale applications, ensuring that it remains effective as the dataset grows.

H. REAL-WORLD IMPACT DISCUSSION

To evaluate the impact of Large Language Model (LLM)-based clustering methods for crowdsourced test reports in practical project maintenance, we conducted a user study. Five graduate students were invited to inspect test reports for Application A9, which consisted of 26 reports describing 5 distinct defects. Participants were randomly divided into two groups: one provided with clustered test reports, and the other without. During the experiment, participants were unaware of

the exact number of defects and had no fixed order for report inspection.

Within the experiment, we calculated the total number of reports inspected, inspection time, and number of defects discovered by each group. The three participants in the first group opted to inspect all 26 reports, with an average inspection time of 574 seconds, equating to 22.1 seconds per report. They identified 5, 5, and 4 defects respectively. The three participants in the second group, leveraging the clustering results, selected one representative report from each cluster for inspection, achieving an average inspection time of 93 seconds, or 18.6 seconds per report. They discovered 5, 4, and 5 defects, respectively. To ensure no significant difference in defect detection capabilities between the two groups, we conducted a significance test, which revealed no notable differences in defect inspection abilities among the students. The similarity in average inspection times indicates that all participants required similar timeframes to inspect individual reports.

Participants in the first group, lacking clustering results, had to inspect all reports individually. In contrast, those in the second group utilized the clustering outcomes, necessitating the inspection of just one representative report from each cluster. This is because reports within a cluster typically describe the same defect, allowing for the effective identification of all defects within the cluster through a single representative report. Consequently, despite inspecting only 5 reports, the second group participants discovered the same number of defects as the first group, demonstrating the purpose of clustering technology: to reduce the number of reports inspected while maintaining defect detection capabilities.

The experimental results highlight that employing LLM for clustering can significantly enhance the efficiency of crowdsourced test report inspection. In real-world project maintenance, LLM-guided clustering methods can reduce the inspection workload for developers while preserving efficient defect detection capabilities, thereby greatly improving the efficiency and effectiveness of test report inspection. This research finding underscores the immense potential of LLMs in practical applications, promising significant improvements in software testing and maintenance processes.

For future work, several specific research avenues can be explored. First, the integration of other multi-modal models, which combine textual, visual, and contextual information, could further enhance the clustering process. By incorporating different types of data, such models may capture a wider range of defects and provide more nuanced insights, potentially improving the accuracy of test report inspection. Second, incorporating feedback loops from human testers could refine the clustering results by enabling iterative model updates based on real-world insights. This approach would facilitate continuous improvement of the model's accuracy, ensuring that it adapts to evolving developer practices and project-specific contexts. Third, conducting case studies in industrial settings with diverse test report datasets would validate the scalability and robustness of the proposed method. This effort would also provide an opportunity to assess the practical impact of clustering on real-world software maintenance workflows, further bridging the gap between research and application.

V. RELATED WORK

Crowdsourcing is a method that harnesses the collective power of diverse individuals to efficiently tackle large-scale tasks that are traditionally costly or time-consuming [3], [4]. Crowdsourced testing leverages this concept by engaging testers with varied backgrounds, skills, and global distribution through an online platform [5]. By aggregating the expertise of numerous testers, this approach enables comprehensive software testing at reduced costs and shorter timelines, thereby uncovering more potential issues and defects. However, while crowdsourced testing offers extensive coverage, cost-effectiveness, and efficiency, it also presents challenges such as quality control difficulties, data security risks, and complex management.

In response to the challenges in traditional mobile application testing, numerous researchers have been dedicated to exploring more effective and efficient crowdsourcing testing methods and strategies aimed at enhancing the efficiency of final developer review reports. Drawing on an understanding of testing background, competence, and domain knowledge, Wang et al. [6] introduced a multi-objective crowdsourced worker recommendation method (MOCOM) that suggests a minimal number of crowdsourced workers for detecting the maximum number of vulnerabilities in the task. Additionally, Wang et al. [7] presented iRec, a context-based process for recommending workers dynamically to enhance the efficiency and effectiveness of crowdsourcing testing. Moreover, [8], [9] ISENSE was proposed as an automated decision support method that enhanced management efficiency and cost-effectiveness through incremental sampling and predictive models in crowdsourcing testing. Furthermore, Ge et al. [10] put forward an assistant method for automated Android testing by constructing an annotated Window Transition graph (AWTG) model based on dynamic and static analysis results to improve the quality and efficiency of crowdsourcing testing.

Currently, there is an increasing focus on utilizing multi-modal information and richer classification methods for processing crowdsourced test reports. Wang et al. [1] suggested using screenshots to enhance accuracy in repeatedly detecting GUI system crowdsourced test reports as a solution to existing limitations with text-based detection methods. Meanwhile, DeepPrior [2] was introduced as a prioritization method for analyzing application screenshots and text descriptions deeply to improve processing efficiency and accuracy of test reports. Similarly addressing report prioritization is DivRisk [11], which improved inspection efficiency through diversity- and risk-based strategies; additionally there existed a multi-objective optimization priority processing technology based on image understanding [12]. In terms of report classification, Wang et al. [13] proposed a clustering-based classification

method effectively distinguishing real faults from industrial-scale crowdsource test reports while improving accuracy and efficiency.LOAF [14] achieved superior results compared with existing methods by reducing manual labeling burden when classifying true fault reports. Focusing on classification quality, Chen's TERQAF framework [15] significantly enhanced developers' handling efficiencies by systematically quantifying characteristics within test reports using logistic regression classifiers.

In terms of repeated detection of defect reports, researchers have proposed a variety of methods to improve the accuracy and efficiency of detection. A repeated defect report detection method called DBTM [16] was proposed, which significantly improves detection accuracy by combining IR-based and subject-based features. Alipour et al. [17] proposed a method to improve defect removal by using context information, and its effectiveness was verified in the defect library of the Android ecosystem. The results showed that the context of software engineering cannot be ignored when using information retrieval tools to remove defects. Similarly, Hindle et al. [18] also focused on context information and proposed a method to improve defect removal by using domain-specific context information, and verified its effectiveness in multiple software systems. In addition, Sun et al. [19] proposed an optimized retrieval function (REP) that improves the accuracy of repeated detection of defect reports by combining information from text and non-text fields, and validates its effectiveness in multiple large software defect libraries. Advances in discriminant models have improved the accuracy of repeated defect report detection and demonstrated significant performance improvements in several large software defect libraries [20].

VI. CONCLUSION

This paper proposes a Large Language Model (LLM)-based clustering method for crowdsourced test reports, addressing the issue of redundancy in such reports. By integrating semantic information from both textual descriptions and screenshots, our approach overcomes limitations in existing methods regarding handling duplicate reports and underutilizing multimodal information. Experimental results demonstrate that the LLM-based method significantly enhances clustering accuracy and efficiency, thereby reducing the workload of developers and improving the quality and consistency of test reports. The novelty of this method lies in leveraging LLMs to extract rich semantic features from both text and images, and guiding the clustering process through semantic binding rules. This approach not only achieves theoretical breakthroughs but also showcases its immense potential value in practical applications.

REFERENCES

[1] J. Wang, M. Li, S. Wang, T. Menzies, and Q. Wang, "Images don't lie: Duplicate crowdtesting reports detection with screenshot information," *Information and Software Technology*, vol. 110, pp. 139–155, 2019.

[2] S. Yu, C. Fang, Z. Cao, X. Wang, T. Li, and Z. Chen, "Prioritize crowdsourced test reports via deep screenshot understanding," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 946–956.

[3] J. Howe *et al.*, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 176–183, 2006.

[4] K. Wazny, ""crowdsourcing" ten years in: A review," *Journal of global health*, vol. 7, no. 2, 2017.

[5] 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*, vol. 24, pp. 537–561, 2019.

[6] J. Wang, S. Wang, J. Chen, T. Menzies, Q. Cui, M. Xie, and Q. Wang, "Characterizing crowds to better optimize worker recommendation in crowdsourced testing," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1259–1276, 2019.

[7] J. Wang, Y. Yang, S. Wang, Y. Hu, D. Wang, and Q. Wang, "Context-aware in-process crowdworker recommendation," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 1535–1546.

[8] J. Wang, Y. Yang, R. Krishna, T. Menzies, and Q. Wang, "isense: Completion-aware crowdtesting management," in *2019 IEEE/ACM 41st international conference on software engineering (ICSE)*. IEEE, 2019, pp. 912–923.

[9] J. Wang, Y. Yang, T. Menzies, and Q. Wang, "isense2. 0: Improving completion-aware crowdtesting management with duplicate tagger and sanity checker," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–27, 2020.

[10] X. Ge, S. Yu, C. Fang, Q. Zhu, and Z. Zhao, "Leveraging android automated testing to assist crowdsourced testing," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2318–2336, 2022.

[11] Y. Feng, Z. Chen, J. A. Jones, C. Fang, and B. Xu, "Test report prioritization to assist crowdsourced testing," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 225–236.

[12] Y. Feng, J. A. Jones, Z. Chen, and C. Fang, "Multi-objective test report prioritization using image understanding," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 202–213.

[13] J. Wang, Q. Cui, Q. Wang, and S. Wang, "Towards effectively test report classification to assist crowdsourced testing," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2016, pp. 1–10.

[14] J. Wang, S. Wang, Q. Cui, and Q. Wang, "Local-based active classification of test report to assist crowdsourced testing," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 190–201.

[15] X. Chen, H. Jiang, X. Li, L. Nie, D. Yu, T. He, and Z. Chen, "A systemic framework for crowdsourced test report quality assessment," *Empirical Software Engineering*, vol. 25, pp. 1382–1418, 2020.

[16] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 2012, pp. 70–79.

[17] A. Alipour, A. Hindle, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 183–192.

[18] A. Hindle, A. Alipour, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection and ranking," *Empirical Software Engineering*, vol. 21, pp. 368–410, 2016.

[19] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 253–262.

[20] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010, pp. 45–54.

YING LI was born in Langfang, Hebei, China in 1982. She is an Associate Professor. Her research interests include software testing, software quality assurance, and computer application technology.

YE ZHONG is a bachelor student of DUT School of Software Technology & DUT-RU International School of Information Science and Engineering, Dalian University of Technology. Her research interest is software testing.

LIJUAN YANG was born in Xingtai, Hebei, China in 1981. She is an Associate Professor. Her research interests include software engineering, software testing technology, and database technologies.

YANBO WANG was born in Shijiazhuang, Hebei, China in 1981. He is a senior engineer at the Beijing Aerospace Automation Research Institute. His major is software development and software testing.

PENGHUA ZHU was born in Handan, Hebei, China in 1980. He is currently working as a Senior Experimenter with the North China Institute ofAerospace Engineering. He is also the Principal of the School of Computer, North China Institute of Aerospace Engineering. His research interests include software engineering and computer application technology.