Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automation Frameworks for End-to-End Testing of Large Language Models (LLMs)

Reena Chandra*[1], Karan Lulla[2], Karthik Sirigiri[3]
Corresponding Author: reenachandra11@gmail.com

ARTICLE INFO

Received: 16 March 2025
Revised: 12 April 2025
Accepted: 30 April 2025
Published: 12 May 2025

ABSTRACT

Building and delivering high-quality software is critical in software engineering and requires verification and validation processes for end-to-end testing that are reliable, robust, and deliver correct results fast. Manual testing of LLM models, while feasible, is very time-consuming and inefficient and has scalability issues depending on how big the model under test (MUT) is. Recent research and cutting-edge technology innovations in LLM models have deeply influenced software engineering. We need to integrate its impact robustly in areas of model analysis, test automation, model execution, debugging, and report generation.

This paper focuses on a framework approach for automated software testing of the LLM models to reduce human interactions and has improved results in a fast, cost-efficient, and time-efficient manner for automated testing methods for industries.

The proposed Automation Framework (LLMAutoE2E) leverages and integrates LLMs for testing of different LLM models (like BERT, BART, Hugging Face, and multiple models available to test) to automate the end-to-end execution lifecycle of the LLM models. By leveraging LLMs, companies and industries can generate automated test cases, automated unit test codes, automated integration and end-to-end tests, and automated reporting of the LLM model's execution results.

This research emphasizes the potential of the Automation Framework (LLMAutoE2E) for LLM to automate and streamline the overall execution and result generation of the LLM models and the overall testing workflows while addressing challenges in current LLM models testing, its accuracy and scalability for deployments, and reporting. The proposed Automation Framework (LLMAutoE2E) can also automate defect analysis, which improves the software reliability by a manifold and reduces the development cycles for companies. This Research paper details the role of Automation Frameworks for LLMs and how it is transforming QA processes, key methodologies, improving reliability and efficiency, addressing current challenges like model safety, bias detection, and continuous monitoring, and future trends in AI-driven software testing.

Keywords: Software engineering, LLM, Large Language, models, Software testing of models, Automation, Frameworks, Quality Assurance, Automated Documentation, Datasets, Compilation, Training, Automated Reporting, CI/CD, Continuous Integration and Continous Deployments.

I. INTRODUCTION

As with other sectors, a crucial component of software engineering is making sure that models or software products used by clients fulfill high quality requirements and carry out activities as planned. Software engineers have an obligation to guarantee that the development and implementation of LLM models are fault-free and that the software system or model meets the given business needs for the consumers [1]. The complexities of technical and business requirements are addressed during the software testing and automation phase. Here, a thorough verification and validation process aligns the implementation results with the desired user needs. This phase effectively showcases the overall quality of the software, ensuring a precise reflection of its performance. Achieving completely error-free code, devoid of any defects now or in the future, is no simple task. The objective is to identify, tackle, and rectify as many detectable errors as possible before the product reaches our valued customers [6], [9].

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Since the beginning of the 1970s, the software engineering industries and any industry using software for operations to deliver to its customers have focused on testing, and automation has grown consistently, accompanied by ongoing advancements in technologies, algorithms, methods, operations, and best practices used in the field. This domain is presently considered one of the most significant areas of inquiry within the realm of software engineering [7]. The challenge of manual testing is that it is time consuming and costly in nature, particularly for regression testing and LLM models that are long-running; the need for effective and efficient automated testing frameworks and approaches has increasingly emerged.

Large Language Models are revolutionizing the software engineering field and industries today [3]. Owing to the increasing capabilities of computing power, the accessibility of extensive training datasets, and significant advancements in the fields of machine learning and natural language processing, large language models have experienced rapid growth [1]. The volume and intensity of work required in industries are driving an ever-growing demand for automation; hence, testing is fast benefiting from the generative powers of LLMs. As reported in [21], LLMs have been used in many testing chores, including data input generation, test case generation, debugging, and program repair. The capacity of these models to generate text, audio, and video that closely mimic human attributes results in a significant blurring of the lines between content produced by machines and that created by humans. Generating automated test cases for testing LLM models; software implementation like code completion, unit, integration, and end-to-end test generating; summarizing; software maintenance; quality assurance; and requirements engineering [1] has shown to be quite helpful in many spheres of software engineering.

Component testing assesses all components individually to ensure the product works correctly, and end-to-end testing assesses all integrated components of the application to ensure they work together correctly. End-to-end testing for LLM models assesses if the model is getting the right token and datasets and is compiling and executing the model correctly. It checks for latency and accuracy as well. End-to-End testing encompasses everything from user interactions, back-end database checks, latency and accuracy for the model and the functionality overall. Thus, end-to-end (E2E) testing involves testing the model or application comprehensively, from the user's point of view [2]. Historically, developers engaged in the manual evaluation of application features and the scripting of customer-facing scenarios utilizing frameworks such as Selenium [11]. However, the predominant methodology for establishing end-to-end (E2E) tests relied heavily on documentation and human interaction [11]. Efforts to automate E2E test generation (both unit and integration tests) for software and LLM models have explored reinforcement learning (RL) [12] and model-based approaches [4], [5], [13]. Its use in a wide range of testing activities has recently skyrocketed, thanks to the proliferation of large language models (LLMs). The application of this technology in various testing activities has recently experienced a significant increase, attributed to the widespread emergence of large language models (LLMs). Improvements to the methodologies employed in test generation remain necessary, despite the demonstrated capabilities of large language models in producing unit tests across various applications [14], [15], [16], [17], [18], as well as in the realms of online and mobile testing [19], [20].

Building on the capabilities of LLM and its recent technological advancements, this research work focuses on leveraging it to create Automation for LLM models. This research paper proposes an Automation Framework (LLMAutoE2E) for the software end-to-end testing approach of different LLM models available today and for any upcoming LLM models in the future too. Our approach utilizes LLMs to generate test cases on the fly, significantly reducing both time and costs associated with testing and executing LLM models. This stands in stark contrast to traditional automated methods that depend on testers or software engineers to create the test cases. The primary goal is to reduce human errors and get robust and reliable results while delivering a comprehensive end-to-end automation framework, testing results, and reporting.

This research paper showcases the transformative power of an automation system in enhancing and streamlining software testing for various LLM models. By shifting from a manual, reactive methodology to a proactive, efficient, automated quality testing approach, we can significantly reduce costs, conserve resources, and save time. This innovative method generates test cases, identifies edge scenarios, and performs End-to-End testing of the LLM models in an effort to tackle fundamental issues in software testing, including test case creation, dataset testing, compilation, execution, and thorough coverage.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The paper has been organized in the following structure. Section II provides challenges in testing LLMs. Section III outlines key components of an LLM testing framework of LLM models. Section IV details implementation of Automation Framework (LLMAutoE2E) and strategies for LLM-based automated software testing of LLM models. Section V covers the discussion, and finally, Section VI provides the conclusion of the paper.

II. CHALLENGES

1. **Data Hallucination, Bias and Reliability**
   - Mitigating biases and assumptions in AI-generated test cases.
   - Ensuring the automation framework integrated with AI and LLMs produces consistent, clear, effective, and reliable results.
2. **Model Explainability and Trust**
   - Ensuring LLMs provide transparent, detailed, and verifiable test outputs.
   - Developing very detailed, human Explainable solutions and results for software QA.
3. **Security and Ethical Concerns**
   - Addressing AI-generated test script defects and vulnerabilities.
   - Establishing ethical guidelines for AI and LLM assisted testing.

III. Key Components of an LLM Testing Framework of LLM models

Here are the key components of the LLM testing framework for multiple LLM models available today:

Table I. Key Components of the proposed Automation Framework

Sr no.	Component	Details
1	Automated Test Case generation	• LLMs create thorough test cases for the model after analyzing the requirements from the published documents.
2	Execution of intelligent automated tests for every stage of the LLM models	• Several model execution test phases can be dynamically executed when LLMs are integrated with automation frameworks. • Automated test execution from dataset testing to the training stage.
3	Code Review and Defect Detection	• LLMs find mismatches in requirements from published docs, inconsistencies, defects, and security issues in the code. • AI-driven defect categorization to improve and streamline the time-consuming debugging by providing detailed error message logs with the type of defect found, severity, and category it belongs to.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4	Automated Test Documentation and Reporting	• LLMs automated test documentation, reducing the manual churn and saving the developer • LLMs automated tests and execution Reporting AI-generated summaries to enhance report readability.
5	CI/CD	• Automates workflow in the proposed framework (LLMAutoE2E) to run at regular or specific time intervals based on changes without human interference.
6	Database Storage	• Store all the stages data in the database along with test results of the execution of the LLM model automation and logs captured for each stage of the model execution.

There are small models like BERT and Llama 3 to large models that can leverage the above components in Table I. above to test different models and its End-to-End execution stages along with proper reporting in totality and generating the results in an automated, time and cost-efficient manner consistently. This component categorization will help in efficient and fast defect creation, adding the appropriate error and exception messages for the defects or vulnerabilities found in particular stages of the LLM model automated testing approach in the framework.

IV. PROPOSED METHODOLOGY

Implementation of Automation Framework (LLMAutoE2E) and Strategies for LLM-Based Automated Software Testing of LLM Models. There are different stages that will be Automated and Tested in this Automation Framework (LLMAutoE2E) approach -

1. Published docs for the LLM model for data Ingestion in the End-to-End Automation Framework
2. Automated Test case generation
3. Dataset Automated Testing
4. Pre-Compilation Automated Testing
5. Compilation of the LLM Model Automated Testing
6. Training of the LLM Model Automated Testing,
7. Code review and defect detection, and
8. Automated Documentation and Reporting of the results

Eg: BERT, DistillBERT, Llama 3.18b , GPT4.0 mini etc

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

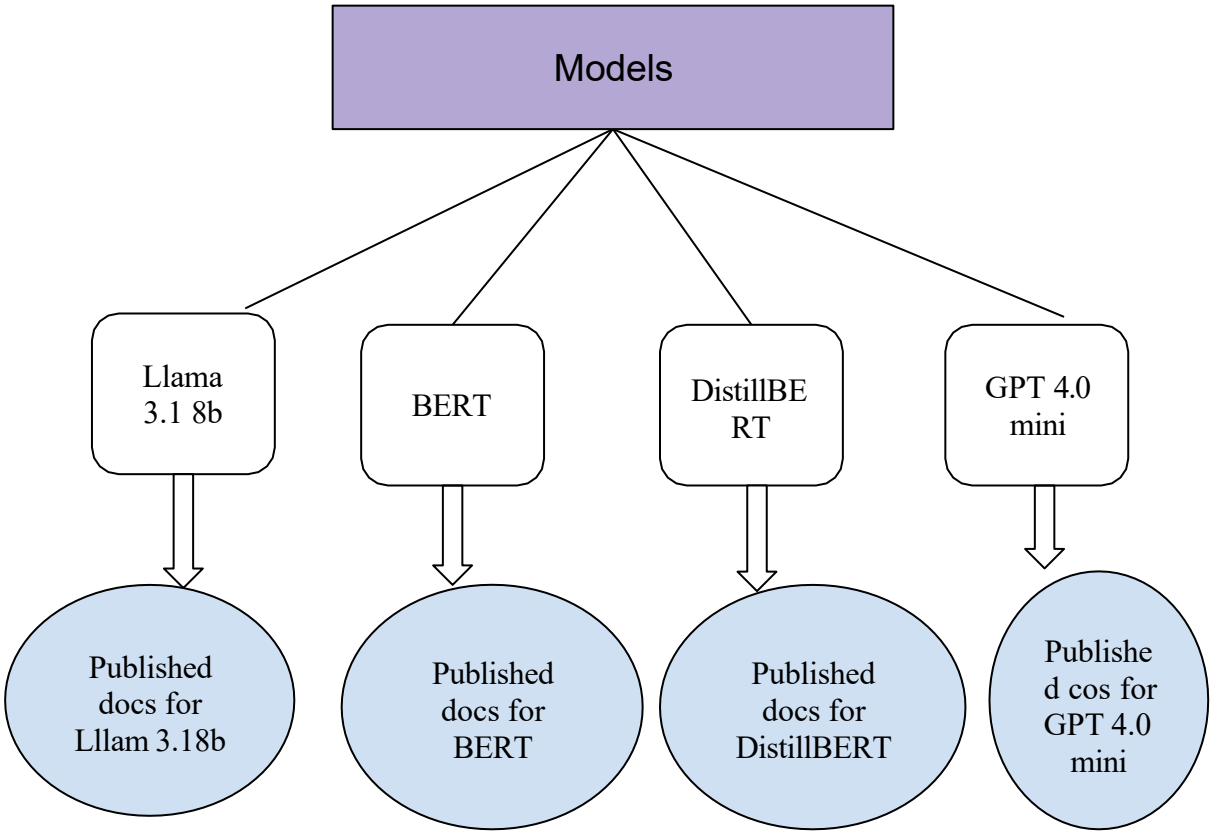

Figure1. Different LLM models to get the published docs.

The published docs from Figure 1, which will serve as input to the Automation Framework and process it further is explained in Figure 2 below.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

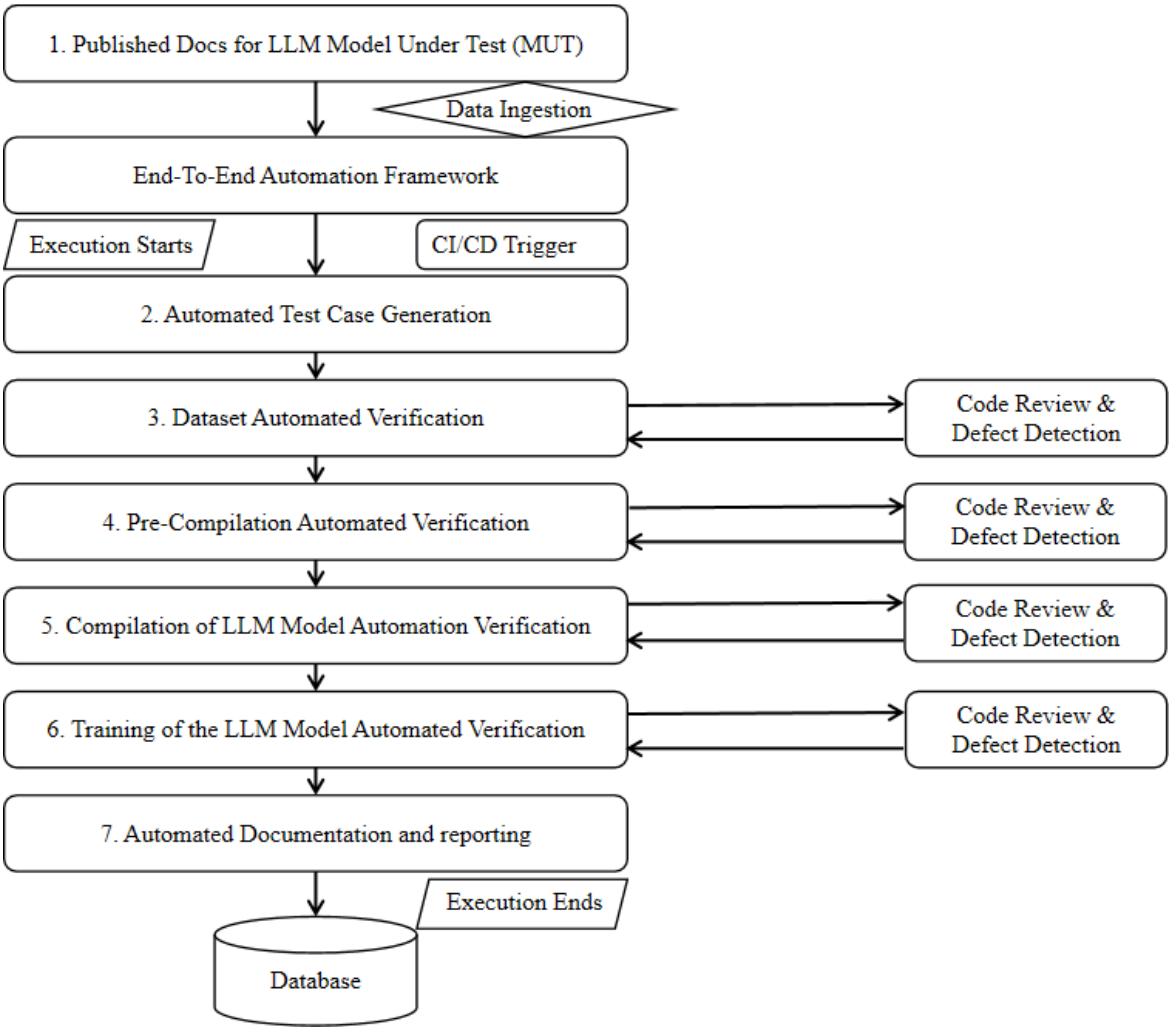

Figure 2. End-To-End Automation Framework for the LLM models

The explanation of the above Figure 2 goes as follows. The published docs of the LLM model under Test (MUT) are used as the source for evaluating all the stages and steps that the LLMAutoE2E should run for end-to-end verification of that particular model. The published docs cover what will execute in setups, dependencies, datasets, compilation, and/or training stages. The LLMAutoE2E uses this data to generate automated test cases for the MUT. The dataset automated testing runs and tests the commands for token created or available, usage, any other configs, libraries, tools required and running the dataset execution 100% successfully. Then the LLMAutoE2E runs pre-compilation execution steps, which are then followed by compilation steps execution. This step reports the compilation status in the form shown below in Figure 3.

Figure 3. Compilation Result status snippet for a Sample model

```
"outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Compiler status PASS",
```


Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS",
"Compiler status PASS "
] }

Finally, the training steps are executed by the automation framework, and verification is performed on it according to the test cases generated in step 2 of Figure 2. This step reports the training status in the form of the below results shown in Figure 3.

The results of the training steps are reported in the form of the below status shown in Figure 4.

Figure 4. Training Result status snippet for Sample model

"Filename: MUT_Result_1.pt",
"Batch Size: 1",
"Batches: 2000",
"Inferences: 2000",
"Threads: 2",
"Models: 2",
"Duration: 0.924",
"Throughput: 2164.0254",
"Latency P50: 0.924",
"Latency P95: 0.940",
"Latency P99: 0.950",
"--",
"Filename: MUT_Result_2.pt",
"Batch Size: 2",
"Batches: 2000",
"Inferences: 4000",
"Threads: 2",
"Models: 2",
"Duration: 1.468",
"Throughput: 2727.086",
"Latency P50: 1.473",

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(43s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

"Latency P95: 1.479",

"Latency P99: 1.488",

Furthermore, detailed documentation and a report are created automatically for the LLM model executed by the framework, along with code review and defect detection details, and all overall run results with detailed data, logs, and errors (if any) for this End-To-End execution are stored in the database storage. This helps in debugging, finding trends in the data stored, analysis and closure of the defects in an effective manner.

V. DISCUSSION

The framework developed in this work, particularly the execution of stages from Section IV, Figure 2, is a way to automate LLM model execution, testing, and reporting to embrace a time-efficient and cost-efficient approach, boosting the overall productivity of the organization. After each stage of the LLM model Execution is completed via LLMAutoE2E, the execution and verification of Testing results are saved in logs with all details of the execution steps, along with any errors, exceptions logged to help in defect debugging in the Database storage for future references. The introduction of LLMAutoE2E provides a standardized, consistent, and automated mechanism for evaluating End-to-End execution techniques for the LLM models. This means that the underlying principles of the LLMAutoE2E can be extended to automate the generation of test cases and the execution of new LLM models released by companies according to the latest trends. Additionally, this Automation Framework can be extended to automate performance and benchmarking, generation of test cases, and execution of existing and new LLM models released as well.

VI. CONCLUSION

This research paper highlights the significant potential of Automation Framework (LLMAutoE2E) in automating various End-to-Test activities of the LLM models execution and reporting, such as test case generation and execution of all stages of the LLM models, covering everything from dataset to training of the models, code review and defect detection, reporting, CI/CD pipelines, and database storage. The proposed framework (LLMAutoE2E) provides explicit error and exception messages for further debugging if there are any defects found in the end-to-end execution of the Model under Test (MUT). This Automation framework can be further expanded to support any new LLM models, and its End-to-End execution can also be adapted for execution of performance and benchmarking verification of the respective LLM models. This research lays a solid foundation for LLM model execution automated verifications. Overall testing via the LLMAutoE2E Automation Framework, demonstrating the potential of minimizing human intervention and execution of all stages of the model, with automated test case generation and reporting, boosts efficiency and enhances overall testing quality.

REFERENCES

[1] Sherifi, B., Slhoub, K., & Nembhard, F. (2024). The Potential of LLMs in Automating Software Testing: From Generation to Reporting. arXiv preprint arXiv:2501.00217.

[2] Alian, P., Nashid, N., Shahbandeh, M., Shabani, T., & Mesbah, A. (2025, March). Feature-Driven End-To-End Test Generation. In 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE) (pp. 678-678). IEEE Computer Society.

[3] Wang, S., Yu, Y., Feldt, R., & Parthasarathy, D. (2025). Automating a Complete Software Test Process Using LLMs: An Automotive Case Study. arXiv preprint arXiv:2502.04008.

[4] Yu, S., Fang, C., Li, X., Ling, Y., Chen, Z., & Su, Z. (2024). Effective, Platform-Independent GUI Testing via Image Embedding and Reinforcement Learning. ACM Transactions on Software Engineering and Methodology, 33(7), 1-27.

[5] Yandrapally, R. K., & Mesbah, A. (2022). Fragment-based test generation for web apps. IEEE Transactions on Software Engineering, 49(3), 1086-1101.

[6] Bäckström, K. (2022). Industrial Surveys on Software Testing Practices: A Literature Review. University of Helsinki.

[7] Sánchez-Gordón, M., Rijal, L., & Colomo-Palacios, R. (2020, June). Beyond technical skills in software testing: Automated versus manual testing. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 161-164).

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Research Article

[8] Morimoto, T., & Haraguchi, H. (2025). A Study on the Improvement of Code Generation Quality Using Large Language Models Leveraging Product Documentation. arXiv preprint arXiv:2503.17837.
[9] Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2016, November). Software testing techniques: A literature review. In 2016 6th international conference on information and communication technology for the Muslim world (ICT4M) (pp. 177-182). IEEE.
[10] Svensson, Johan & Stephen, Michael. (2025). Automating Software Testing with Large Language Models (LLMs): A QA Revolution. Computer Software.
[11] "Selenium," https://www.selenium.dev, 2024, accessed: 2024-06-28.
[12] Chang, X., Liang, Z., Zhang, Y., Cui, L., Long, Z., Wu, G., ... & Huang, T. (2023, May). A reinforcement learning approach to generating test cases for web applications. In 2023 IEEE/ACM International Conference on Automation of Software Test (AST) (pp. 13-23). IEEE.
[13] Biagiola, M., Stocco, A., Ricca, F., & Tonella, P. (2020, October). Dependency-aware web test generation. In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST) (pp. 175-185). IEEE.
[14] Xie, Z., Chen, Y., Zhi, C., Deng, S., & Yin, J. (2023). ChatUniTest: a ChatGPT-based automated unit test generation tool. arXiv e-prints, arXiv-2305.
[15] Kang, S., Yoon, J., & Yoo, S. (2023, May). Large language models are few-shot testers: Exploring llm-based general bug reproduction. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2312-2323). IEEE.
[16] Lemieux, C., Inala, J. P., Lahiri, S. K., & Sen, S. (2023, May). Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 919-931). IEEE.
[17] Nashid, N., Sintaha, M., & Mesbah, A. (2023, May). Retrieval-based prompt selection for code-related few-shot learning. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2450-2462). IEEE.
[18] Schäfer, M., Nadi, S., Eghbali, A., & Tip, F. (2023). Adaptive test generation using a large language model. arXiv preprint arXiv:2302.06527, 1-21.
[19] Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., ... & Wang, Q. (2024, April). Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (pp. 1-13)..
[20] Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Tian, Z., ... & Wang, Q. (2024, April). Testing the limits: Unusual text inputs generation for mobile app crash detection with large language model. In Proceedings of the IEEE/ACM 46th international conference on software engineering (pp. 1-12).
[21] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.