Validation of Image Content Using LLMs and RAGs: A Strategy for Ensuring Compliance in Software Testing

Wendell Marques
Sidia Institute of Technology
Manaus, Brazil
wendell.marques@sidia.com

Caina Oliveira
Sidia Institute of Technology
Manaus, Brazil
caina.oliveira@sidia.com

Carol Fernandes
Sidia Institute of Technology
Manaus, Brazil
carol.fernandes@sidia.com

Luiz Ribeiro
Sidia Institute of Technology
Manaus, Brazil
luiz.ribeiro@sidia.com

Edluce Veras
Sidia Institute of Technology
Manaus, Brazil
edluce.leitao@sidia.com

Gabriel Sampaio
Sidia Institute of Technology
Manaus, Brazil
gabriel.cruz@sidia.com

Kaua Sanches
Sidia Institute of Technology
Manaus, Brazil
kaua.sanches-e@sidia.com

Renan Peres
Sidia Institute of Technology
Manaus, Brazil
renan.peres-e@sidia.com

ABSTRACT

This article presents an approach for validating content in images using large language models (LLMs) and retrieval-augmented generation (RAGs) as essential tools. The proposal aims to enhance software testing processes by ensuring compliance and accuracy of visual data used. Through the integration of these technologies, the study demonstrates how it is possible to automate content verification in images, identifying inconsistencies and ensuring that data meet established standards. Additionally, the challenges and limitations of this approach, as well as its practical applications in software testing scenarios, are discussed. The results indicate that the combination of LLMs and RAGs offers an efficient and scalable solution for visual content validation, significantly contributing to the quality and reliability of testing processes.

KEYWORDS

RAG, LLM, Software Engineering, Testing

1 Introduction

The use of LLM has become extremely prevalent in recent years, driven by the wide range of possibilities they offer. A language model predicts sequences of characters, words, or sentences based on previous or adjacent context[1]. The best solution to improve LLM accuracy and credibility, avoiding the effort required by complex fine-tuning is represented by RAG[8], which is becoming a popular paradigm in LLM's systems. The underlying idea of the RAG approach is the merging of LLMs knowledge with specialized, vast, and dynamic data coming from external repositories [8]. The initial query triggers external search algorithms to retrieve relevant information. This data is then sent to LLMs prompts for additional context[7]. According to this, the RAG approach combines information retrieval mechanisms with In-Context Learning (ICL) [6] to improve the LLMs performance. In this work, we leverage LLM and RAG to develop a tool for image reviewing in the context of software test validation. The main goal of this research is to create a general RAG system for ensuring compliance in software testing.

2 Contextualization

This study came from necessity to assist the development and requirements validation team of clients from an institute that creates solutions for mobile products in the Latin American market. After implementing software requirements, these need to be tested and validated for approval. One key requirement involves embedding a technical image that displays the product name along with a specific technical code that varies per model. Previously, an experimental tool called Image Test Review (ITR) was developed using computer vision techniques with OpenCV for non-textual elements and OCR for textual elements, demonstrating promising time savings in industrial processes. For this paper, a new tool named ITR-e-LLL (Image Test Review Enhanced by LLMs) is proposed. This tool integrates the use of LLM and RAG concepts to evaluate which LLM performs better in image review within a software test validation context.

3 Proposal

The proposal assumes that a technical image contains specific texts indicating the associated mobile product model, which must be extracted and verified for correctness from an embedded image in the software.

LLM have shown potential in solving text-to-SQL (Structured Query Language) tasks in databases [3], receiving natural language questions as input and, based on a specific database table, returning as output an SQL instance that answers user questions. Our proposal takes as input an initial text composed of [model name, technical code], and based on a CSV database, it returns as output Python code for access, rather than SQL. Our approach will follow the strategy adopted in [2], where a Pretrained Language Model architecture is refined to meet the needs of the text-to-SQL task, and it uses this schema to achieve logical form accuracy and execution accuracy metrics to validate the solution. Within our scope, it is defined a predictive database that includes the code associated with the product to facilitate comparisons of logical and execution accuracy metrics. RAG is employed solely to retrieve

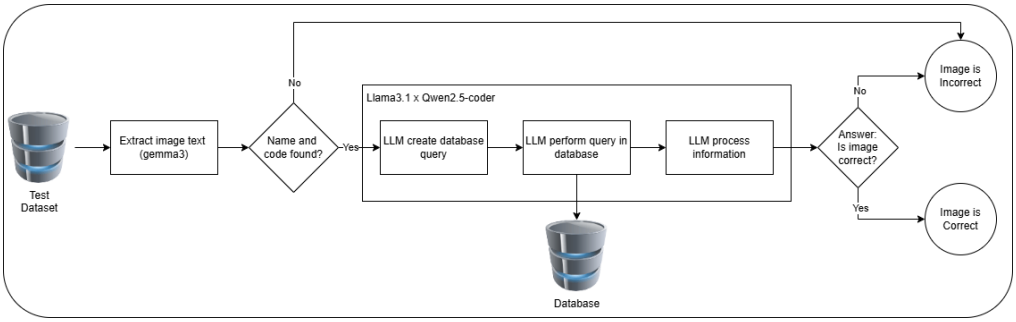

Figure 1: ITR-e-LLM Proposal

the relevant product-code mapping that enables subsequent validation of whether the embedded image content on the device meets the required compliance standards. Additionally, in the context of binary classification, We will evaluate the measure Mathew Correlation Coefficient (MCC) in our proposal. The study proposed in [4] has demonstrated that MCC provides more reliable assessments than accuracy or F1 score, specially when the class distribution is skewed.

The ITR-e-LLM proposal, as illustrated in Figure 1, consists of the following stages: 1) **Test Dataset**: It is composed by a collection of test images containing a variable mix of product models and technical codes as textual content. The collection integrates both correct and incorrect image sets through an image review process; 2) **Predictive database(PD)**: Using RAG concept, a CSV database containing the product name and its certification code is created as a base for evaluating available images for testing; this method ensures that every product, once homologated by its target market's regulatory agency, is assigned a unique code upon successful certification; 3) **Image Text Extraction**: Using a product homologation image, we utilize the multimodal model'gemma3(27B)' to describe as text the image content, specifically the product and code details. The extracted information is analyzed, and if meets the criteria for successful content, it proceeds to a verification process to ensure data integrity; 4) **Validation Architecture (VA)**: Initially, a question is executed to determine whether the certification code belongs to the product. The LLM converts this question into a Python code expression. A parser, component that processes the Python code expression generated by the LLM, then executes the query in PD (Predictive Database) and evaluates the result. Finally, the LLM transforms the result into a natural language answer, providing a binary decision: correct or incorrect. As the goal of this study is to determine which LMM demonstrates the best performance, the VA employs two available LLMs for comparison: llama3.1 and qwen2.5-coder:14b. We intentionally used only our internal server running Ollama with two robustly configured models, omitting any external models to avoid information leakage and ensure controlled integration without internet access.

4 Methodology

With the solution concept well defined, we proceed to the next steps planned to execute the experiments aimed at evaluating the solution, primarily focusing on the performance of LLMs in the verification process. The method proposed in this article is explained below: 1) **Dataset**: It consists of 500 images, with 250 containing correct technical information and the remaining 250 containing incorrect information. This means we have a small dataset, but it is balanced in terms of quantity; 2) **ITR-e-LLM development**: A tool to review image tests embedded in mobile software was developed using the Python programming language, leveraging gemma3:27b, llama3.1, and qwen2.5-coder:14b. To ensure the accuracy of Python code generation for database querying, specific examples were incorporated into the architecture. Additionally, adjustments were made to the prompts, and a more precise context was introduced to minimize the variability in the responses. Five-shot prompting was utilized, employing the Llama3.1:8b and Qwen2.5-coder:14b models. The temperature parameter (random seed) was set to 0 to minimize randomness in the outputs; 3) **Validation and Evaluation**: It consists to execute the tool using each llama3.1 and qwen2.5-coder:14b for test verification. The collected data are then summarized and analyzed. Additionally, some metrics are used to evaluate the performance of each verification LLM.

The machine configuration setup for running LLM verification includes an 8-core CPU, 48GB of RAM, a 40GB GPU, and a 1TB ROM disk. The experiments follow the 5-fold cross-validation procedure described in [5] , where the testing is performed with splits preselected (100 images chosen randomly, not balanced) to eliminate cases of images from the same dataset appearing in the testing set in the same fold. For images splits we execute the VA for each LLM verification.

To evaluate LLMs performance we adopted the metrics used in [2], where logical accuracy is $Acc_{lf} = N_{lf}/N$ and execution accuracy is $Acc_{ex} = N_{ex}/N$, N is the total number of images, N_{lf} is the number of queries that have an exact string match with the ground truth query, and N_{ex} represents the number of queries that, when executed, produce the correct outcome. Additionally, the performance of the binary classification model's results was evaluated by MCC [4].

5 Results

After analyzing experiment results, some technical questions were formulated to determinate the best LLM for image review. **RQ1**.

What is the feedback after applying the selected LLMs to the available dataset for software image review? **RQ2**. Which LLM performed better? **RQ3**. What are the possible ways to improve performance and efficiency for selected LLM?

Table 1: Metrics (%)

Model	Med Acc_{ex}	Med Acc_{lf}	Med MCC
Llama3.1	94.6	92.4	90.5
Qwen2.5-coder(14b)	95.5	93.6	93.6

RQ1: After applying the selected LLMs, the experience indicates that both models performed well in terms of execution and logical accuracy, as shown in Table 1. However, there were variations in performance across different experiments, and after analyzing the logs extracted from each fold, some challenges were observed: **1. Model and code extraction errors**: The highest number of errors was found during the extraction of the model and image code during the fourth fold. This suggests that the LMM used to extract text from images had difficulty interpreting some technical images. **2. Server disconnection issues**: There were instances of server disconnection during some folds, demonstrating a potential reliability issue in the verification process. **3. Syntax Errors in Generated Code**: Two syntax errors were found in the Python code generated solely by LLaMA3.1, indicating that LLaMA3.1 requires refinement to generate syntactically correct code.

RQ2: Based on the metrics, Qwen2.5-coder:14b performed better than Llama3.1. $\mathbf{Acc_{lf}}$: Qwen2.5-coder:14b achieved a median of 93.6%, compared to Llama3.1's 92.4%. $\mathbf{Acc_{ex}}$:Qwen2.5-coder:14b scored a median of 95.6%, while Llama3.1 scored 94.6%. **MCC**: Qwen2.5-coder:14b had a higher median of 93.6%, compared to Llama3.1's 90.5%. This indicates that Qwen2.5-coder:14b demonstrated more reliable and consistent performance across the metrics.

RQ3: The strategy can be: **1. Improve Image Text Extraction**: Invest in more robust technique or fine-tune the model specifically to reduce errors in the fourth fold. **2. Handling Server Disconnections**: Implement mechanism to ensure continues server connectivity during the verification process. **3. Syntax Error Reduction**: Fine-tune Llama3.1 to check syntax errors in generated code, including more examples to generate python code during training phase. **Prompt engineering**: Refine prompts and context to reduce variability and improve code generation and verification results. **Larger Dataset Testing**: Evaluate the ITR-e-LLM performance on larger dataset to ensure representative testing folds, reducing the risk of biased results.

6 Conclusion and Future works

ITR-e-LLM architecture was created to compare two verification LLMs within image review process for mobile software test. The approach involved stratifying text [product, code] into 5-fold cross-validation procedure, converting the text to Python code, and executing this code in database. The database results are then transformed into binary natural language answers, indicating whether test image was correct or incorrect. The experiments were done to determine which model demonstrates superior performance, calculating Logical Accuracy (Acc_{lf}) , Execution Accuracy (Acc_{ex}) and Mathew Correlation Coefficient (MCC) metrics on both selected LLMs. Qwen2.5-coder:14b outperformed Llama3.1 across all metrics, demonstrating high reliability and consistency, achieving median MCC of 93.6%, median Acc_{ex} of 95.5% and median Acc_{lf} of 93.6%. Challenges identified during the experiments such as image extraction errors, server disconnections issue, and syntax issues in generated Python code highlight areas for improvement.

Before this automated approach, manual verification was conducted by testers who were responsible for reviewing images and validating certification codes in a CSV database. With ITR-e-LLM, the process becomes automatic, only requiring updates to the CSV database whenever a new product is homologated, significantly reducing human intervention. Initial experiments were conducted on a limited dataset, without feedback from end-users or additional comparison with OpenCV+OCR process. Additionally, this approach demonstrated the potential of using LLMs for mobile software image review, providing great results and excellent insights into model performance, highlighting areas for continuous optimization. Future work should focus on enhancing image text extraction, ensuring server stability, refining prompt engineering, and expanding dataset testing to optimize performance and scalability.

ARTIFACT AVAILABILITY

ITR-e-LLM cannot be shared due to industrial restrictions and confidentiality constraints.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No.8.387/1991, by having its disclosure and publicity in accordance with art.39 of Decree No.10.521/2020.

REFERENCES

[1] Emily M Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 5185–5198.

[2] Vanessa Câmara, Rayol Mendonca-Neto, André Silva, and Luiz Cordovil-Jr. 2023. DBVinci–towards the usage of GPT engine for processing SQL Queries. In *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web*. 91–95.

[3] Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2024. BEAVER: an enterprise benchmark for text-to-sql. *arXiv preprint arXiv:2409.02038* (2024).

[4] Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics* 21 (2020), 1–13.

[5] André Ramos Fernandes Da Silva, Lucas Marcondes Pavelski, Luiz Alberto Queiroz Cordovil Júnior, Paulo Henrique De Oliveira Gomes, Layane Menezes Azevedo, and Francisco Erivaldo Fernandes Junior. 2022. An evolutionary search algorithm for efficient ResNet-based architectures: a case study on gender recognition. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–10.

[6] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey, 2024. *URL https://arxiv. org/abs/2312.10997* (2024).

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.