Reducing Large Language Model Hallucinations for Test Generation

AUTHOR

Angelika Mentzelopoulou
A.Mentzelopoulou@student.tudelft.nl

RESPONSIBLE PROFESSOR

Andy Zaidman

SUPERVISOR

Amir Deljouyi

1. INTRODUCTION

UTGen combines a search-based software testing[1] tool, EvoSuite[2], with LLMs to improve the understandability of the generated tests.

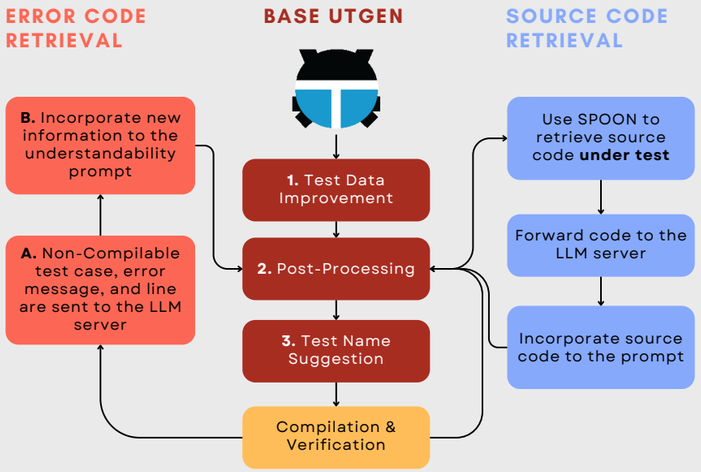
The LLM may "hallucinate" and generate test code that is not compilable or code that is too dissimilar to the initial EvoSuite test. UTGen handles this by re-prompting, making the process inefficient and expensive.

The purpose of our research is to decrease the number of re-prompts, by engineering an enhanced prompt for the LLM using information retrieval.

2. RESEARCH QUESTIONS

1. **RQ1:** Is it possible to reduce the hallucination of LLMs used in UTGen during the Post-Processing phase, to minimise the need for re-prompting, using prompt engineering with *source code retrieval*?
2. **RQ2:** Is it possible to reduce the hallucination of LLMs used in UTGen during the Post-Processing phase, to minimise the need for re-prompting, using prompt engineering with *error code retrieval*?

3. OUR PROPOSED APPROACHES

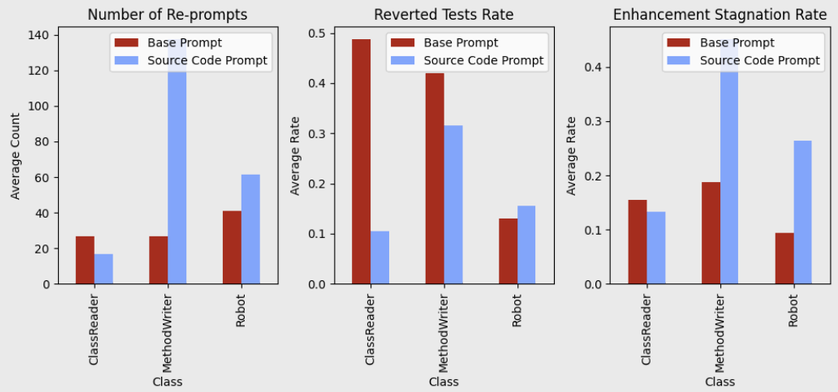


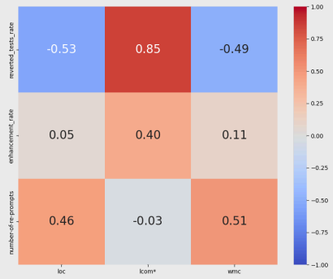
4. METHODOLOGY

- Comparison Study: hallucination performance on same EvoSuite test suite between our respective methods and base UTGen
- Metrics: Number of re-prompts, Enhancement Stagnation rate (LLM did not generate test similar to EvoSuite), Reverted tests rate (LLM did not generate compilable test), Source code complexity.
- Dataset: 4 available benchmarks of UTGen on 204 classes of SF110 of DynaMOSA[4] dataset. From those we select 3 classes according to benchmark enhancement stagnation and reverted test performance.
- LLM: code-llama:7b-instruct model, as provided by Hugging Face

5. RESULTS & CONCLUSIONS

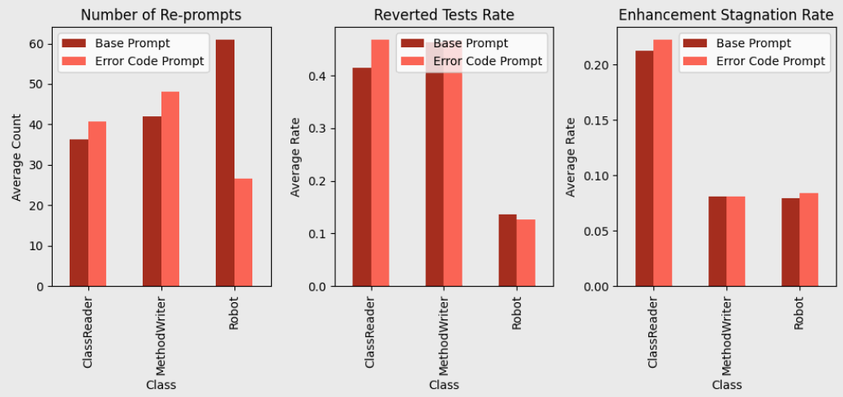
SOURCE CODE RETRIEVAL

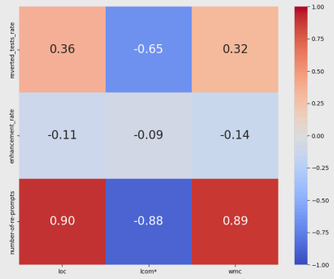




- Non-uniform results
- ClassReader, class with high problematic test cases in the benchmarks, shows improvement in all three areas when using our source code approach
- Our approach decreases the reverted test rate for classes with low *LCOM
- Hypothesis: source code shifts LLM focus to compilability

ERROR CODE RETRIEVAL





- Uniform results, with the exception of re-prompt count drop for Robot (high *LCOM)
- Mirrored correlation between reverted test rate and number of re-prompts
- Expected no correlation with enhancement stagnation rate, since the prompt is not used between similarity evaluations
- Hypothesis: High prompt complexity does not allow for improvement

6. LIMITATIONS

- Lack of Documentation on UTGen Implementation
- Limited technical & financial resources for utilizing the LLM
- Measuring the Number of Re-Prompts effectively

7. FUTURE WORK

- Investigate conciser and more focused prompts to ground the LLM further
- Upgrade to LLM with more training parameters for higher accuracy
- Investigate similarity between EvoSuite and error code approach test suites

8. REFERENCES

[1] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, and Ra- jwinder Kaur Panesar-Walawege. A Systematic Re- view of the Application and Empirical Investigation of Search-Based Test Case Generation. *IEEE Transac- tions on Software Engineering*, 36(6):742-762, Novem- ber 2010.
[2] Gordon Fraser and Andrea Arcuri. EvoSuite: automatic test suite generation for object-oriented software. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ESEC/FSE '11, pages 416-419, New York, NY, USA, September 2011. Association for Computing Machinery.
[3] Renaud Pawlak, Martin Monperrus, Nicolas Petitprez, Carlos Noguera, and Lionel Seinturier. SPOON: A library for implementing analyses and transfor- mations of Java source code. Software: Practice and Experience, 46(9):1155-1179, 2016. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2346.
[4] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets. IEEE Transactions on Soft- ware Engineering, 44(2):122-158, February 2018.