Debugging Problems

Patrick Farmer

Supervisor: Dr. Jonathan Dukes

March 3, 2025

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

A Dissertation submitted in partial fulfillment of the requirements for the degree of MAI in Computer Engineering.

Declaration

I hereby declare that this Dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this Dissertation will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed: Patrick Farmer Date: March 3, 2025

Contents

1 Acronyms

- LLM - Language Model
- AST - Abstract Syntax Tree
- AI - Artificial Intelligence

2 Introduction

2.1 Context

In recent years there have been great strides made in the ability of LLMs to generate code. These LLMs are already being integrated into employee workflows in companies all over the world. This will place a much greater emphasis on the debugging process as the code generated by these LLMs much be inspected and fixed when bugs are produced. It has also been noted in many studies that debugging is an extremely important skill as it is the most difficult part of the programming process and is also the most often overlooked when teaching programming. More will be discussed on this in the literature review. This project will aim to create a tool that will help teach debugging to students who will be the future employees of these companies.

2.2 Aims

The aim of this tool is quite simple. To generate problems for students. The language, code topic and bug type should all be customisable.

It was also a goal to create two different methods for inserting the bug which can be compared with each other. The first method being to insert the bug by querying the LLM

and the second method being to insert the bug by walking through the AST of the code and inserting the bug semi manually.

Another aim of the project was to create test cases that the code could be tested against to ensure that the bug was inserted correctly. These same test cases could then be used by a student to test their solution to the problem.

The creation of a frontend was not a goal of this project, the meaningful work of this project is all done in the backend and a frontend could be added at a later date.

2.3 Evaluation Metrics

It was decided early on that the evaluation and measuring of the project success would be done using desktop metrics and that there would not be a user study. The metrics that were decided on were code complexity, code diversity, attempt count and run time. The calculation of these metrics and the reasoning behind them will be discussed in the testing and evaluation section.

2.4 Structure of the Dissertation

The structure of the dissertation will be as follows. The background will go through previous work done in the area of debugging and LLMS. It will also give a brief overview of how LLMs work and more importantly how they can be used to generate code, the difficulties that come with using them and where they excel.

The design and implementation will show how the code is structured and will give a brief description of how each part of the code works.

The testing and evaluation section will show the testing that was done on the project and the results of that testing. It will also discuss the metrics used in testing, why they were chosen and how they were calculated.

The conclusion will summarise the project and discuss the future of the project.

3 Background

3.1 Literature Review

Debugging has always been an essential aspect of programming, yet many universities do not teach it specifically. This is highlighted by Li in [2]. As previously mentioned, the significance of debugging skills has grown markedly now that AI is becoming more prevalent in industry, as noted by Denny in [8]. Debugging itself includes several sub-domains, as outlined in [2] as language knowledge, understanding of the specific program, and skill in the debugging process. This project primarily focuses on developing the debugging skill but there will be some domain knowledge that comes with it.

When novice programmers write code, they inevitably introduce bugs, but as Jadud mentions in [1], they often respond by making minor syntax tweaks and rerunning their code immediately rather than trying to identify and resolve the underlying issue. This is another challenge that this tool addresses: by inserting deeper bugs, students are compelled to scrutinize and understand the code.

Another important aspect of debugging that this tool addresses is the use of debugging tools like Python's debugger, which are part of the sub-domains indicated by Li in [2]. Learning to effectively use these debuggers can considerably improve students' debugging abilities. However, Odell [3] argues that forcing students to think about the program without any tool-assisted help is the most effective instructional approach. Ultimately, each professor can decide whether to allow debugging tools, depending on the objectives of their

particular assignment.

As Nguyen states in [9], significant misunderstandings can easily arise between an AI and its user, which is especially true for novices who may struggle to articulate their problems thoroughly. This situation creates a chance to teach students how to harness LLMs effectively. The future frontend may benefit from incorporating a co-pilot like assistance, allowing professors to enable or disable it depending on the complexity level they want to set. A comparable methodology is described by Denny in [7], where students restricted to using LLMs alone learned to refine their prompts for more accurate output.

3.2 Summary of LLMs

3.2.1 How LLMs work

3.2.2 Using LLMs

3.2.3 Evaluation of LLMs

4 Design and Implementation

4.1 Design Overview

Includes Diagram

4.2 Ollama

4.3 Code generation

4.4 Test case generation

4.5 Bug Insertion

4.5.1 LLM Bug Insertion

4.5.2 AST Bug Insertion

5 Testing and Evaluation

This will discus the testing of the project which will show the improvement of the project over time. This will also include an evalution of the final state of the project.

5.1 Metrics

5.1.1 Code Complexity

5.1.2 Code Diversity

5.1.3 Attempt Count

5.1.4 Run Time

6 Conclusion

This will briefly summarise the project and discuss the future of the project.

7 References

References

[1] Jadud, M. C. (2006). Methods and tools for exploring novice compilation behaviour. Proceedings of the Second International Workshop on Computing Education Research, 73–84. https://doi.org/10.1145/1151588.1151600

[2] Li, C., Chan, E., Denny, P., Luxton-Reilly, A., & Tempero, E. (2019). Towards a Framework for Teaching Debugging. Proceedings of the Twenty-First Australasian Computing Education Conference, 79–86. https://doi.org/10.1145/3286960.3286970

[3] O'Dell, D. H. (2017). The Debugging Mindset: Understanding the psychology of learning strategies leads to effective problem-solving skills. Queue, 15(1), 71–90. https://doi.org/10.1145/3055301.3068754

[4] Parkinson, M. M., Hermans, S., Gijbels, D., & Dinsmore, D. L. (2024). Exploring debugging processes and regulation strategies during collaborative coding tasks among elementary and secondary students. Computer Science Education, 0(0), 1–28. https://doi.org/10.1080/08993408.2024.2305026

[5] Whalley, J., Settle, A., & Luxton-Reilly, A. (2021). Analysis of a Process for Introductory Debugging. Proceedings of the 23rd Australasian Computing Education Conference, 11–20. https://doi.org/10.1145/3441636.3442300

[6] Whalley, J., Settle, A., & Luxton-Reilly, A. (2023). A Think-Aloud Study of Novice Debugging. ACM Transactions on Computing Education, 23(2), 1–38. https://doi.org/10.1145/3589004

[7] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A.,

& Reeves, B. N. (2023). Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators. https://doi.org/10.48550/ARXIV.2307.16364

[8] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024). Prompt Problems: A New Programming Exercise for the Generative AI Era. Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 296–302. https://doi.org/10.1145/3626252.3630909

[9] Nguyen, S., Babe, H. M., Zi, Y., Guha, A., Anderson, C. J., & Feldman, M. Q. (2024). How Beginning Programmers and Code LLMs (Mis)read Each Other. Proceedings of the CHI Conference on Human Factors in Computing Systems, 1–26. https://doi.org/10.1145/3613904.3642706