Debugging Problems

Patrick Farmer

Supervisor: Dr. Jonathan Dukes

January 14, 2025

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

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

Declaration

I hereby declare that this Progress Report 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 / do not 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 thesis 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. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed: Patrick Farmer Date: January 14, 2025

Contents

1 Introduction

Due to the developments in LLM capabilities over the past few years have introduced a couple of issues with the classic approach for teaching program. The first of which is the issue of plagiarism from students using AI generated code. Tasks that are often given to novice programmers can be completed within moments by chat GPT so a lot of students will choose this easier alternative.

There is then on the other hand the issue that AI assisted programming will likely exist in every job in the coming years, meaning that being able to work effectively with AI is also a very valuable skill. This project aims to address both of these issues by creating a tool that can generate defective code which the student must then debug.

This solves the issue of plagiarism as AI will struggle a lot more to fix broken code than it will to generate it.

It also teaches the student how to debug which is the most important tool when working with AI generated code.

2 Goals and Objectives

2.1 Primary Goal

The primary goal of this project is to create a tool that can generate code with bugs in it that provide meaningful lessons for the students that are debugging it.

2.2 Secondary Goals

Some more specific goals for this project are:

- Have the tool generate varying types of programs and types of bugs so that no two

students get the same problem and if the student were to run the tool a number of times they would keep getting different problems.

- To the greatest extent possible prevent or detect and correct hallucinations from the model whether that be related to the intiial code creation or the bug injection.
- To have the ability to generate test cases for the working code to check if the student has fixed the bug.
- Attempt the bug insertion using a more complex approach using abstract syntax trees.

3 Literature Review

Debugging has always been a crucial part of programming but many universities do not actually directly teach it. This is a problem outlined in Towards a Framework for Teaching Debugging [2]. Also as previously mentioned the need for debugging skills is much more important now that AI is becoming more prevalent in the workplace as discussed in A New Programming Exercise for the Generative AI Era [8]. The skill of debugging also has a number of sub-domains that need to be taught too as outlined in [2] the key 3 are knowledge of the language, knowledge of the specific program being debugged and knowledge of how to debug. The latter is the skill that this project aims to teach.

When novice programmers write code they will inevitably write bugs into their code. The approach they take to solve the bug though is almost always to make small changes to syntax and run the code again straight away and do not take the time to understand the problem as is highlighted in Methods and Tools for Exploring Novice Compilation Behaviour [1]. This is another issue the tool helps to solve, by injecting bugs that create underlying

problems it will force the student to try to understand the code.

Another aspect of the debugging process that this tool aims to teach is the use of debugging tools such as python debugger. This is mentioned as one of the sub-domains of debugging in [2], developing the skills to use these debuggers will inevitably help the student debug in the future. However, The Debugging Mindset [3] disagrees and believes that the best way to teach debugging is to force the student to think about the program without any help from tools. The choice of whether to include debugging software can be used is in my opinion best left to the professor who knows what they want to teach the students with the task they are giving.

4 Current state of the project

Through the early stages of the project it quickly became apparent that although clear progress was being made there was no metrics to measure this progress. Due to the nature of the output it is also a little difficult to measure the quality of the output. However, a couple of metrics were selected and used to measure to progress of the project and the impact of the features added. These metrics were:

- The cognitive complexity of the code generated. This was measured using the cognitive complexity metric from radon.
- The cyclomatic complexity of the code generated. This was also measured using radon.
- The retry count for the generation of working code.
- The retry count for the generation of the bug.

- The similarity of the generated code to the rest of the programs generated.

There is no necessary ideal for the complexity. That would depend on the target for the task but it is a clear indicator of whether the code is overly simplistic or massively complex. The retry count and similarity score are all better the lower that they are. Below is a graph of the difference in the metrics as features are added to the tool and the impact of each can be seen.

It will take a while to compile all of the saved results and get a graph but I will have the graph and a short discussion before the due date (24th Jan)

There is also a pipeline setup for the benchmarking of the project that is run on a schedule via a cron job through github workflows. This allows for the project to be benchmarked regularly and the results to be stored as artifacts. The code is then run on my local machine to avoid the inevitably excessive API costs that would be incurred running the full benchmarking suite through the openAI API. There is however support for the openAI API which can be used with the flick of a switch in the command line arguments of the batch file.

Currently the program is able to generate working code using an LLM, create one default test case, inject a bug into the code also using an LLM and ensure that the test case fails. It is the plan in the future to enable some more complex bug injection using Abstract Syntax Trees but that is not yet implemented, that is one of the significant goals for the second half of the year. Each query to the model will self reflect on itself to improve accuracy. There is also some more targeted manual checks such as making sure that the test case fails when the bug is inserted.

5 Project Management

There is few risks in the project as it is already in a working state just is not yet in the most mature possible state. One potential risk is the inability to add the more complex bug injection using ASTs. This is a risk because as the graph above shows the LLM has some difficulties performing the bug insertion and it would greatly increase the customisability of the bugs and the runtime of the program as it would not have to retry the bug insertion as many times.

6 Conclusion

7 Acknowledgements

8 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