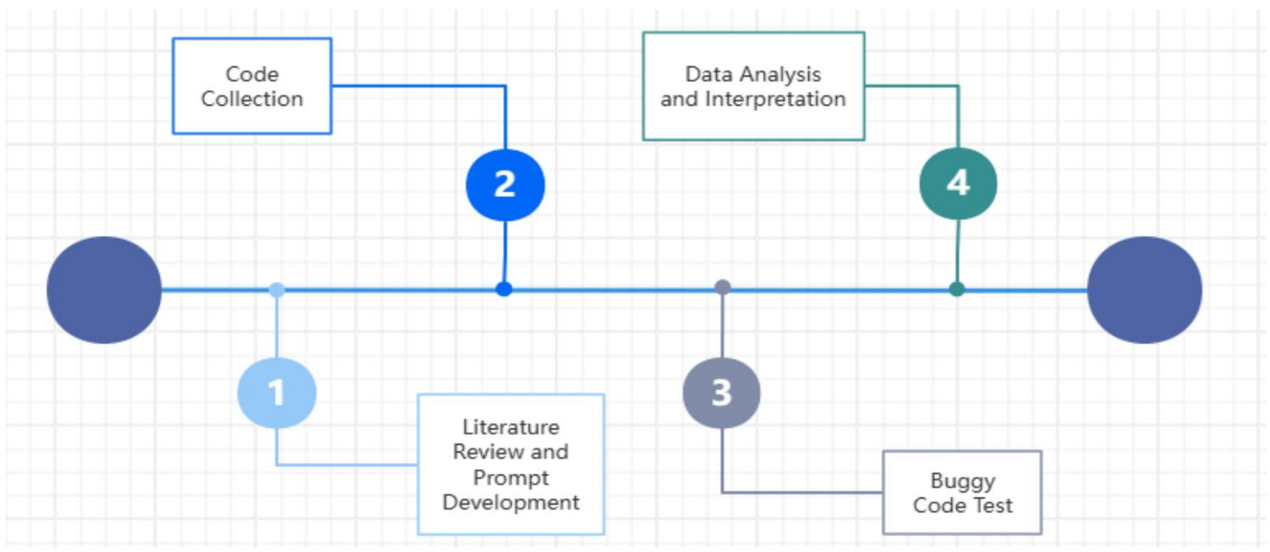
If It Ain't Broke, Don't Fix It: The Unintended Consequences of Large Language Model Code Repairs

Xiaofei Yu East China Normal University & University of Alberta

MOTIVATION

ABSTRACT

The increasing use of large language models, like ChatGPT, for code generation raises concerns about the accuracy and reliability of AI-produced content. This project explores the unintended consequences of code repairs by these models, comparing original correct code with AI-modified versions to assess potential risks. Findings indicate that while AI can be a powerful tool, it may also introduce subtle yet critical bugs that jeopardize software integrity. This research, funded by Professor Lutellier, was independently conducted by me and focused on a specific area of individual interest.



METHODOLOGY

The approach used focuses on the collection of new code generated by ChatGPT and the analysis of the correctness of the new code.

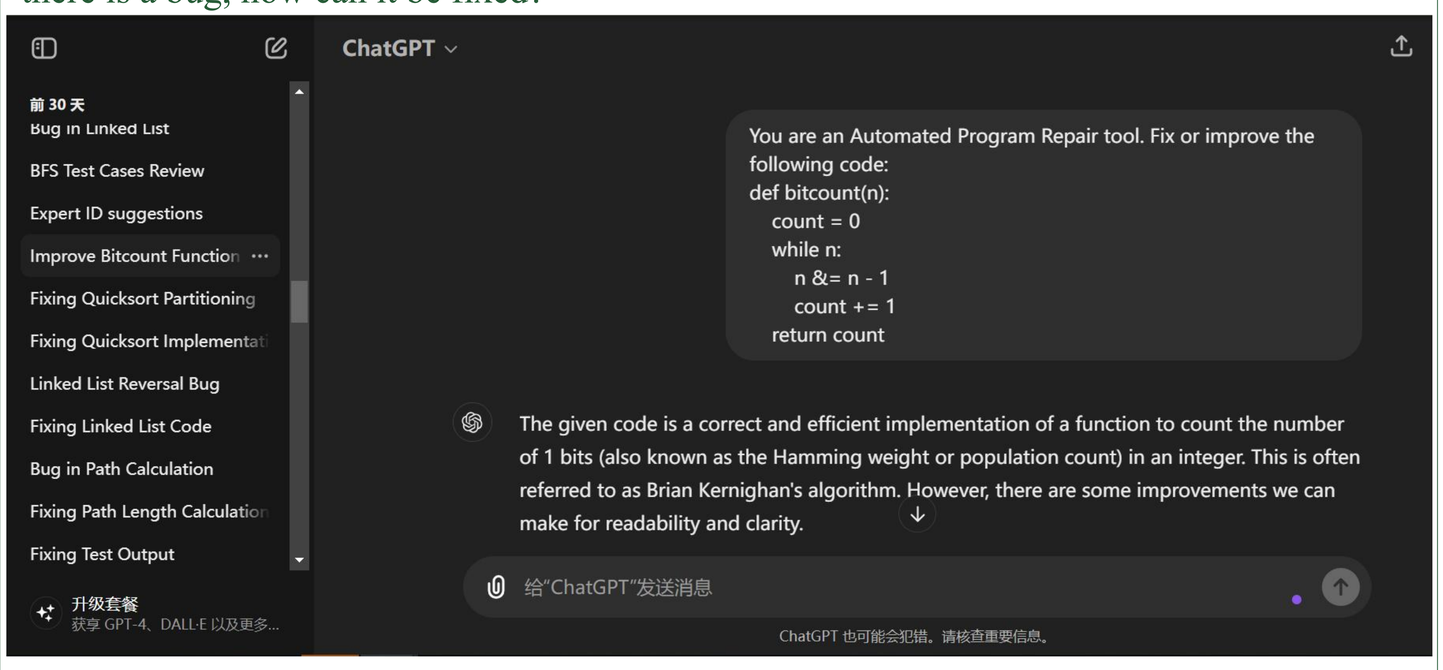
PROMPT

Our approach involves a systematic evaluation of code fixes suggested by a large language model (ChatGPT) extended from the original correct code.

After reading the literature, we selected two different sets of prompts, paired with correct QuixBugs, and fed into ChatGPT.

Prompt1:You are an Automated Program Repair tool. Fix or improve the following code:

Prompt2:You are an Automated Program Repair tool. Is there a bug in this program? If there is a bug, how can it be fixed?



COLLECTION

We compared the code produced by chatgpt with the source code, cleaned the text, and recorded the parts that changed for subsequent research.

FINDINGS

The project is in its mid-stage, with data collection ongoing for over a month. So far, I've gathered a month's worth of data, focusing on the following types.

BugID
Original Program link to the original file on the Github repository
Prompt (P1 or P2)
AI used (chatGPT 4o or 3.5)
Generated Program link to the generated file on the Github repository
Generated Program is Changed 0 = no change 1 = changed
Generated Program is buggy 0 = correct change/benign change 1 = new bug introduced/ bug not fixed ? = I don't know/I'm not sure

RESULTS

Preliminary results indicate that while the language model can correctly identify and fix some issues, it also frequently introduces unintended bugs into otherwise correct code. These errors vary in severity, from minor syntactical mistakes to significant logical flaws that could compromise the functionality of the software.

FUTURE GOAL

The next phase of this research will focus on evaluating the effectiveness of ChatGPT as a code evaluation tool, using pytest to measure the correctness of newly generated code. The performance of ChatGPT in 80 different conversations will be analyzed. The goal is to quantify the accuracy of the AI and assess the potential risks of using it for critical coding tasks.

CONCLUSION

This study demonstrates the potential pitfalls of over-reliance on AI for code repairs. While large language models have their place in the software development process, they are not infallible and can introduce errors that may go unnoticed until they cause significant issues. Future research should focus on enhancing the accuracy of AI models in code-related tasks and developing strategies for effective human-AI collaboration in software development.