Automated Program Debugging with Large Language Models

Fatemeh Ghassemi

University of Tehran

Abstract Large Language Models (LLMs) will revolutionize the process of automated program debugging through interpreting source code, identifying bugs, and suggesting their corrections in a way really similar to human reasoning. Debugging is intrinsically a complex task because it requires deep logical analysis and understanding of the flow of executions, especially in high-stakes settings like competitive programming, where solutions must pass comprehensive test cases under strict time constraints. Traditional debugging involves both static and dynamic analysis methods, but large language models propose a new paradigm in the use of natural language understanding with code synthesis for automation and enhancement of the debugging process.

A key development in LLM-driven debugging is the integration of mechanisms for iterative feedback. Techniques such as Chain of Thought reasoning allow the model to break down the debugging process into an incremental series of steps, thereby enhancing its capabilities to analyze, hypothesize, and iteratively refine corrections. The LLMs continuously assess the test case results and adjust the fixes accordingly, while emulating a trial-and-error approach employed by human programmers. This iterative process of refinement gives automated debugging accuracy and reliability so that models could make more precise and contextually sensitive corrections.

This research aims to further explore the potential of LLMs in automated debugging, investigating optimization strategies such as prompt engineering and hybrid approaches that combine LLM reasoning with traditional techniques. By benchmarking our methods against existing debugging tools and human performance, we seek to assess the practical viability of LLM-driven debugging in real-world software development scenarios. The results can also contribute to the general pool of research in AI-assisted software engineering that could reduce debugging time, improve software quality, and make the task of programming easier for developers of all skill levels.