CS 846 Software Engineering for Big Data and AI

The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development

Jun Lim

20870249

Software development is a highly skilled and complex task that demands knowledge, and various tools have been developed to enhance productivity, such as code repositories and collaborative software engineering. The early vision of software-AI assistants (The Programmer's Apprehentice) could not be realized due to technological limitations at that time. However, recent advancements in computational power and AI, particularly models of transformer architecture (or LLMs like GPT-3), have created the possibility of AI assistants engaging in conversational interactions.

The authors introduced a system (Programmer's Assistant), an AI-driven tool to assist in software engineering by providing code and natural language interactions. It combines a code editor with a chat interface and employs OpenAI's Codex model for its prompting mechanism and conversational engine. It is built to respond to queries grounded in the context of the user's source code. It is also able to keep track of user selection state, send messages to the assistant, and ask follow-up questions.

This system is then empirically evaluated with 42 participants of varying levels of programming experience, backgrounds, and genders to assess its value in a software engineering context. They found that their system, the Programmer's Assistant, was capable of conducting extended discussions and enabled additional knowledge and capabilities to emerge from the LLM. Despite initial skepticism, participants were impressed by the breadth of the assistant's capabilities and the quality of its responses. The participants have also highlighted the following:

1. Rated the value at 8.6 out of 10 with the hope of an official release in the future.
2. Commented its conversational interactions grounded in the code as highly useful; of 83% rated ability to ask follow-up questions as useful.
3. It helps engineers improve programming skills, reinforce knowledge gaps, and force people to learn how to communicate code.
4. Pointed out an issue with the correctness of the code produced by the tool and had to rely on participants' own knowledge to judge its correctness.

These studies have demonstrated that the Programmer's Assistant is useful and deemed valuable. The authors then finally conclude that conversational AI can significantly enhance programming by providing context-aware assistance, suggesting the potential for AI systems to transform programming practices. It also suggests that further research should focus on improving human-centered AI design for better human-AI interaction and learning, advocating for systems that not only augment human intelligence but also learn from human input.

Paper Commentary

The paper made an innovative contribution to the field of conversational AI in software development, pushing the boundaries of how developers could interact with AI tools. The authors did not just propose the tool but also conducted an empirical study to evaluate the practicality of the tool, giving the readers context on the usefulness of its proposal. Given the emerging capabilities of AI, this tool or system demonstrated the ability of LLM to engage in discussions beyond code generation.

There are also some downsides. First, the scope of the evaluation involving only 42 participants is too limited; a larger sample size of evaluation would provide a much more comprehensive understanding. Secondly, there could be issues of user privacy, such as AI handling sensitive (or proprietary) code, which have not been highlighted in the paper. Thirdly, I felt the paper lacks details on the expansion of its work; the paper could have provided more in-depth details of its implementation to allow other researchers to build upon the work easily.