Contribution ID: **138** Type: **Poster**

LLM-based Code Documentation, Generation, and Optimization AI Assistant

Recent advancements in large language models (LLMs) have paved the way for tools that can enhance the software development process for scientists. In this context, LLMs excel at two tasks – code documentation in natural language and code generation in a given programming language. The commercially available tools are often restricted by the available context window size, encounter usage limits, or sometimes incur a substantial cost for large scale development/modernization of gigantic scientific codebases. Moreover, there are data privacy/security concerns. Thus, a programmatic framework that can be used from the Linux terminal on a local server is needed. This framework should be less laborious to use by batching code documentation of large codebases and must be able to use the latest open models offline for code generation.

We present a retrieval-augmented generation (RAG)-based AI assistant for code documentation and generation which is entirely local and scalable. The advantage of this setup is the availability of a large context window free of any external recurring costs while ensuring transparency. The code documentation assistant has three components – (a) Doxygen style comment generation for all functions and classes by retrieving relevant information from RAG sources (papers, posters, presentations), (b) file-level summary generation, and (c) an interactive chatbot. This will help improve code comprehension for new members in a research group and enhance the understanding of sparsely documented codebases. The code generation assistant splits the code in self-contained chunks before embedding –this strategy improves code retrieval in large codebases. We compare different text and code embedding models for code retrieval. This is followed by AI generated suggestions for performance optimization and accurate refactoring while employing call graphs knowledge to maintain comprehensiveness in large codebases. Additionally, we discuss the guardrails required to ensure code maintainability and correctness when using LLMs for code generation.

Significance

The impact of text and code based LLMs on scientific software development needs to be critically investigated. Their rapid and continuing growth have made an overwhelming number of options available for domain scientists. To this effect, we develop a local and secure framework and evaluate several LLMs for code documentation and generation. We also discuss the role of test-driven development and guardrails that are required to ensure the quality of generated code.

References

Experiment context, if any

Authors: ATIF, Mohammad (Brookhaven National Laboratory); CHOPRA, Kriti (Brookhaven National Lab); KILIC, Ozgur Ozan (Brookhaven National Laboratory); Dr LEGGETT, Charles (Lawrence Berkeley National Lab (US)); LIN,

Meifeng (Brookhaven National Laboratory (US)); WANG, Tianle (Brookhaven National Lab); DONG, Zhihua

Presenter: ATIF, Mohammad (Brookhaven National Laboratory)

Session Classification: Poster session with coffee break

Track Classification: Track 1: Computing Technology for Physics Research