Codebase explorer :RAG Based AI Chatbot

Muhammad Umair Imran[1], **Najam Ul Islam Saeed**[2],

This project presents the development of an advanced NLP-based chatbot system that leverages Retrieval-Augmented Generation (RAG) to deliver context-aware and accurate responses. The chatbot integrates CodeBERT and Gemini APIs for enhanced language understanding and response generation. The backend is built using FastAPI to ensure scalability and performance, while Streamlit powers the user-friendly front end. Core components include LlamaIndex for document chunking and retrieval, and RankPass's CodeParser for structured code interpretation. By combining modern large language models with retrieval mechanisms and modular deployment tools, the system achieves both precision and real-time interaction efficiency. This report details the architecture, implementation, and performance of the solution, highlighting its potential for code-based query assistance and domain-specific conversational applications.

Introduction

Programmers often struggle to understand large and complex codebases, especially when joining new projects or working with unfamiliar code. Navigating through thousands of lines of code to answer simple questions can be time-consuming and frustrating. Existing tools like IDEs or static documentation provide limited support and lack intelligent, context-aware assistance.

This project addresses this challenge by developing an AI-powered chatbot using a Retrieval-Augmented Generation (RAG) approach. Leveraging CodeBERT, Gemini APIs, LlamaIndex, and CodeParser, the system provides conversational answers to code-related queries. With a FastAPI backend and Streamlit frontend, the chatbot aims to simplify code comprehension and boost developer productivity.

Literature Review

Recent advancements in language models such as CodeBERT, GraphCodeBERT, and OpenAI's Codex have significantly improved machine understanding of source code. These models are widely used for tasks like code completion, summarization, and question answering. Retrieval-Augmented Generation (RAG) has further enhanced performance in these tasks by enabling models to retrieve relevant context before generating responses. Tools like GitHub Copilot and Amazon CodeWhisperer use such models to assist developers, but they are often limited to IDE integration and lack deep retrieval from custom codebases. Frameworks like LlamaIndex and LangChain support efficient chunking and indexing for RAG pipelines, while libraries such as RankPass CodeParser allow for structured parsing of code. However, there is still a gap in solutions that combine these tools into a conversational system tailored for codebase understanding. This project builds upon existing work to create a domain-specific chatbot that bridges this gap with a fully integrated, RAG-based architecture.

Mathodology

Recent advancements in language models such as CodeBERT, GraphCodeBERT, and OpenAI's Codex have significantly improved machine understanding of source code. These models are widely used for tasks like code completion, summarization, and question answering. Retrieval-Augmented Generation (RAG) has further enhanced performance in these tasks by enabling models to retrieve relevant context before generating responses. Tools like GitHub Copilot and Amazon CodeWhisperer use such models to assist developers, but they are often limited to IDE integration and lack deep retrieval from custom codebases. Frameworks like LlamaIndex and LangChain support efficient chunking and indexing for RAG pipelines, while libraries such as RankPass CodeParser allow for structured parsing of code. However, there is still a gap in solutions that combine these tools into a conversational system tailored for codebase understanding. This project builds upon existing work to create a domain-specific chatbot that bridges this gap with a fully integrated, RAG-based architecture.

Implementation details

The chatbot system was built using a modern and modular tech stack optimized for performance, scalability, and developer experience. The backend is developed using FastAPI, which offers high-speed asynchronous APIs suitable for real-time query handling. The frontend interface is implemented with Streamlit, providing an interactive and user-friendly experience for developers to input queries and view responses.

For language modeling and embeddings, CodeBERT is used to convert code chunks into dense vector representations. These vectors are stored in ChromaDB, a vector database optimized for similarity search, enabling efficient retrieval of semantically related code snippets.

The RankPass CodeParser library plays a key role in splitting raw code into meaningful chunks, along with extracting metadata like start/end line numbers and file descriptions. These chunks are enriched with additional context before being embedded.

To enhance query understanding and generation, the Gemini API is used. It refines user queries by adding relevant keywords and later generates the final response using the retrieved code context. The overall architecture is modular and scalable, allowing easy integration with additional code analysis tools or language models in the future.

Results and discussion

The chatbot system successfully demonstrated its ability to assist developers in navigating and understanding large codebases. By combining semantic search with code-aware generation, it provided relevant and coherent responses to code-related queries. The use of CodeBERT for embedding generation enabled effective retrieval of contextually appropriate code chunks, while the Gemini API produced accurate, natural language explanations based on that retrieved content.

In practical tests, the system handled a variety of developer questions—such as asking about the purpose of a function, understanding how modules interact, or locating specific logic—without requiring the user to manually search through files. The use of metadata (file name, line numbers, and descriptions) in the retrieval process further improved the precision of results.

While the system performs well for mid-sized projects, response quality can depend on the chunking granularity and the relevance of the retrieved context. In some cases, overly broad or vague queries resulted in less precise answers. Incorporating more advanced query refinement or integrating additional ranking strategies could enhance performance further.

Overall, the project demonstrates the potential of combining retrieval-augmented generation with code-specific tools to improve code comprehension. It serves as a valuable assistant for developers, particularly during onboarding or when exploring unfamiliar repositories.

Future work

This project successfully implemented a Retrieval-Augmented Generation (RAG)-based AI chatbot tailored to help developers understand large codebases. By integrating CodeBERT, Gemini API, and supporting tools like LlamaIndex, CodeParser, and ChromaDB, the system was able to deliver relevant, context-aware responses to natural language queries. The use of FastAPI and Streamlit ensured smooth user interaction and backend efficiency.

The chatbot demonstrated its potential in simplifying code comprehension, reducing manual effort, and enhancing developer productivity. It effectively bridged the gap between traditional documentation tools and intelligent code assistance.

For future work, several improvements can be explored. Integrating feedback loops for continuous learning could help fine-tune responses over time. Expanding support for multi-language codebases, improving chunking strategies, and adding source-aware debugging features are promising directions. Additionally, incorporating real-time collaboration features and IDE plugins could further enhance its usability in practical development environments.

References

[1] M. Chen, Z. Zhu, Z. Lu, and Y. Zhang, "CodeBERT: A Pre-trained Model for Code and Natural Language," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1237-1246. Available: https://doi.org/10.1145/3397271.3401071

[2] M. Lewis, B. Oguz, N. Goyal, N. Stiennon, E. Choi, and S. Levine, "Retrieval-augmented Generation for Knowledge-intensive NLP Tasks," in *Proceedings of NeurIPS 2020*, 2020. Available: https://arxiv.org/abs/2005.11401

[3] Hugging Face, "Transformers Library," 2021. Available: https://huggingface.co/transformers/

[4] RankPass, "RankPass CodeParser," 2023. Available: https://www.rankpass.com/codeparser/

[5] Chroma, "Chroma: A Vector Database for Embeddings," 2023. Available: https://www.trychroma.com/

[6] Streamlit, "Streamlit: The Fastest Way to Build Data Apps," 2023. Available: https://streamlit.io/

[7] FastAPI, "FastAPI Documentation," 2023. Available: https://fastapi.tiangolo.com/