sbBot : Sunbird Guide Chatbot, Refactor and Similar Case Query Tools

專題編號：114-1-CSIE-S003
執行期限：113年第1學期至114年第1學期
指導教授：郭忠義
專題參與人員： 111590060 鄭秀心
110830009 黃音慈

1. 摘要

This project uses React.js, Tailwind CSS and Flask framework, develops an interactive website query and support system that integrates Large Language Models (LLM) with a Retrieval-Augmented Generation (RAG) framework, using techniques such as query cleaning and rewriting, FAISS vector retrieval, and dynamic prompting, along with code analysis features including code cleaning and tree-based structures. One core function is a chatbot application for Sunbird DCIM test automation, which provides multiple functions to help team members quickly become familiar with website operations.

關鍵詞：RAG, Semantic Search, FAISS Vector Retrieval

2. 緣由與目的

Before writing Sunbird test cases, team members must follow requirement steps and operate directly on the website, which is often time-consuming due to unfamiliarity with its features. Similarly, keyword refactoring requires manually finding and testing all related cases, increasing workload and the risk of missing certain use cases. To resolve these issues, the team decided to develop supporting tools, including a keyword refactoring tool, a topic-based case query tool, and a chatbot that enables members to quickly query and understand website functions.

3. 研究範圍

(a) Guide Chatbot

A local chatbot is developed using a structured RAG workflow. Operation guides are segmented, embedded with Sentence-Transformers, and stored in a FAISS database for retrieval. User queries are vectorized and matched against the database, and a local LLM generates coherent, context-aware responses.

(b) Refactor Keyword Query Tool

This tool converts all test case code into JSON, defines excluded and parent–child cases, and uses a tree structure to trace all test cases referencing a specific keyword, ensuring accurate and efficient keyword dependency tracking.

(c) Similarity Case Query Tool

This tool extracts case data into JSON, embeds it into a FAISS database, and retrieves semantically similar cases by comparing query vectors, enabling quick identification of related test scenarios.

4. 架構流程

The system shown in Figure 1 consists of three layers: Frontend, Backend, and Application layer. The frontend built with React.js and Tailwind CSS, provides an interactive interface. It communicates with the Flask backend through a JSON-based Fetch API. The backend handles API routing and data exchange between the frontend and core modules. The application layer includes three modules, Refactor Keyword Query Tool, Similar Case Query Tool, and User Guide Chatbot, which use Sentence

Transformers, FAISS, PyMuPDF, RAG, and LLM technologies for parsing, semantic retrieval, and response generation.

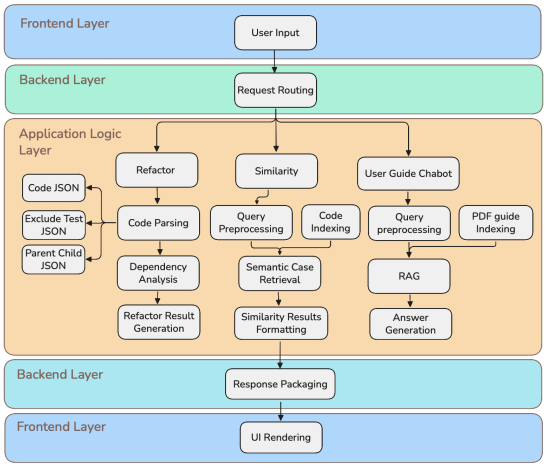

Figure 1. The architecture of the system

5. 使用技術方法

(a) Large Language Model (LLM) - Ollama

Serves as the core generation engine for producing context-aware responses. The LLM runs locally through Ollama, ensuring secure and efficient on-device interaction and understanding of test procedures.

(b) Retrieval-Augmented Generation (RAG)

Combines semantic retrieval with language generation. Relevant document segments are first retrieved from the FAISS database and then passed to the LLM.

(c) FAISS Vector Retrieval

Enables high-performance semantic search by storing document embeddings and retrieving relevant results through similarity computation using the "all-mpnet-base-v2" model.

(d) Semantic Search and Query Cleaning

User queries are cleaned and rewritten to remove noise, then embedded and compared in vector space using "all-mpnet-base-v2" to ensure accurate and contextually relevant search results.

6. 實驗結果

Figure 2 shows that our Refactor Query Tool performs much faster than manual work in all cases, completing each task in under 30 seconds and reducing the average processing time by about 36 times. Additionally, the user guide chatbot and similar case query tools provide relevant references and context, helping the team design more precise and consistent test cases while greatly reducing manual effort.

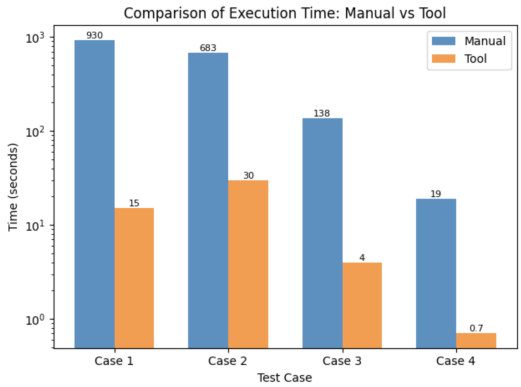

Figure 2. Comparison manual and using Refactor Query Tool for finding cases

7. 結論

In conclusion, those provided functions have significantly accelerated the development process, allowing team members to generate test cases more efficiently. Furthermore, their integration has improved the overall consistency and reliability of the testing framework, ensuring that test case generation follows the requirements.

8. 參考文獻

[1] PyMuPDF, "PyMuPDF Documentation," PyMuPDF ReadTheDocs, [Online]. Available:https://pymupdf.readthedocs.io/en/latest/.

[2] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint*, arXiv:2312.10997v5, Mar. 2024.