IM3014 Linux System Administration Practice
Homework 5: Wiki Assistant

Introduction

After the launch of ChatGPT in late 2023, the new wave of generative AI has become one of the most important technological trends worldwide. To keep pace with this development, **IM Corp** has decided to deploy an internal ChatGPT-like service within the organization. As a senior system administrator, your task is to containerize the application using Docker and make it available to internal users.

By completing this assignment, you will gain hands-on experience in containerizing applications and hosting services in a production-like environment.

Wiki Assistant

This section requires a basic understanding of Python and the development of Generative AI applications, including topics such as LLMs and AI agents. If you are not familiar with these concepts, it is strongly recommended that you start early and explore online learning resources. For example, the *Introduction to GenAI and ML* course offered by the EE Department is available on YouTube and provides a helpful foundation.

Large Language Models (LLMs) acquire knowledge through techniques such as pretraining and supervised fine-tuning (SFT). However, relying solely on an LLM to memorize all knowledge has two major limitations: (1) it is prone to errors such as hallucinations, and (2) the stored knowledge cannot be updated frequently due to the enormous computational cost of retraining. For example, answers to questions like “Who is the current president of Taiwan?” may change over time, but the model’ s internal knowledge may not.

To address this issue, a different approach allows LLMs to search for relevant documents, retrieve the most useful text chunks, place them into the model’ s context window, and then generate an accurate response. This technique is known as **Retrieval-Augmented Generation (RAG)**, which combines LLMs with traditional information retrieval methods. Figure 1 illustrates a typical RAG system.

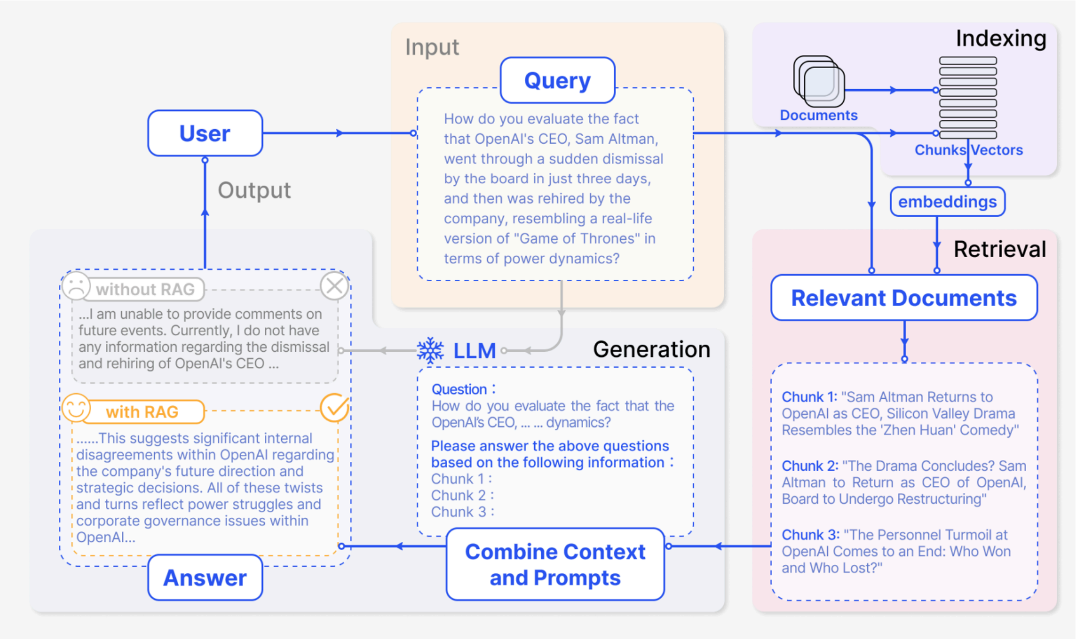
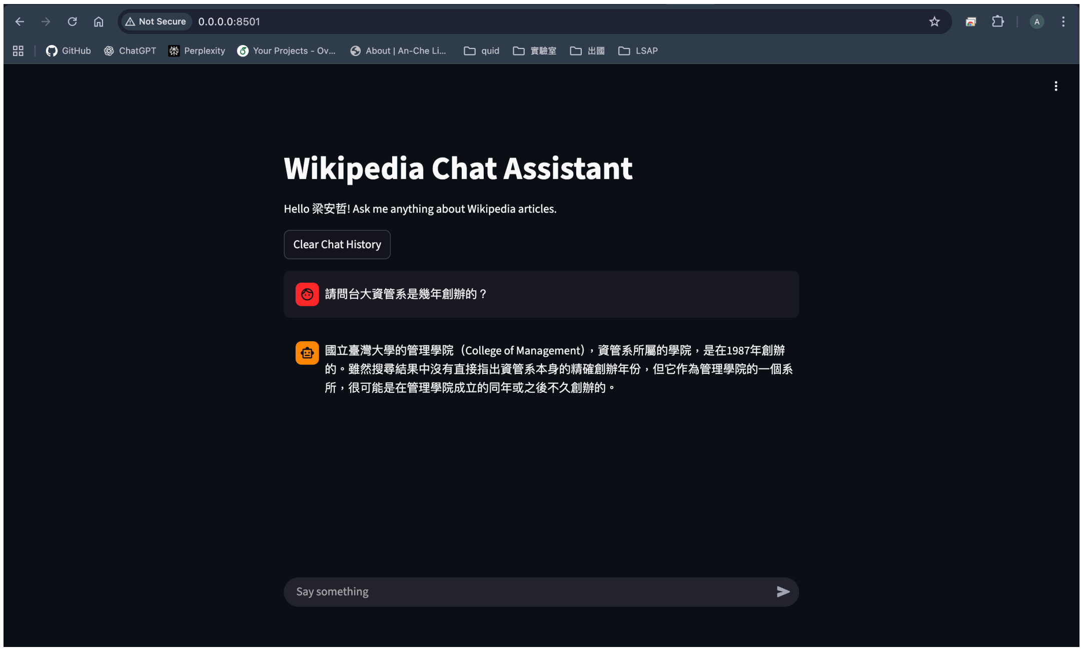

Input
Query
How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?
Indexing
Documents
Chunks
Vectors
embeddings
User
Output
Retrieval
Relevant Documents
without RAG
...I am unable to provide comments on future events. Currently, I do not have any information regarding the dismissal and rehiring of OpenAI's CEO ...
LLM
Generation
Question :
How do you evaluate the fact that the OpenAI's CEO, ... dynamics?
Please answer the above questions based on the following information :
Chunk 1 :
Chunk 2 :
Chunk 3 :
with RAG
......This suggests significant internal disagreements within OpenAI regarding the company's future direction and strategic decisions. All of these twists and turns reflect power struggles and corporate governance issues within OpenAI...
Chunk 1: "Sam Altman Returns to OpenAI as CEO, Silicon Valley Drama Resembles the 'Zhen Huan' Comedy"
Chunk 2: "The Drama Concludes? Sam Altman to Return as CEO of OpenAI, Board to Undergo Restructuring"
Chunk 3: "The Personnel Turmoil at OpenAI Comes to an End: Who Won and Who Lost?"
Answer
Combine Context and Prompts

Figure 1: RAG system architecture. Source: HuggingFace

The project **Wiki Assistant** is a prototype developed by the R&D department of IM Corp to evaluate the capabilities of a RAG-based assistant. It consists of two main components:

- **Frontend:** Built with `streamlit` for the chat user interface and `DSPy` for the agent logic. It uses Google's `gemini-2.5-flash` model as the default LLM.
- **Backend:** Implemented with `FastAPI` to search for relevant Wikipedia pages, retrieve relevant text chunks using semantic search, and store chat history.
  Note: Semantic search uses embeddings to convert text chunks into high-dimensional representation vectors. Relevant information can then be retrieved through mathematical similarity measures such as cosine similarity. This method is considered a successor to traditional keyword-based search.



Wikipedia Chat Assistant

Hello 梁安哲! Ask me anything about Wikipedia articles.

請問台大資管系是幾年創辦的？

國立臺灣大學的管理學院（College of Management），資管系所屬的學院，是在1987年創辦的。雖然搜尋結果中沒有直接指出資管系本身的精確創辦年份，但它作為管理學院的一個系所，很可能是在管理學院成立的同年或之後不久創辦的。

Figure 2: The UI of Wiki Assistant

The source code for this project is hosted on GitHub and is available for download. For reference, the UI of Wiki Assistant is shown in Figure 2 when the application is set up correctly. To ensure that the application can be deployed reliably and maintained for future development, follow the steps outlined below:

1. **Clone the Repository from GitHub.**
   Clone the repository to your VM from GitHub
2. **Run the application locally**
   Install the required packages, then run both the frontend and backend servers locally to verify that the application starts correctly.
   `Verification:` Provide screenshots showing both services running without errors.
3. **Obtain a Google AI Studio API Key**
   Visit Google AI Studio and register for an API key to access the Gemini models. Google provides a generous free usage quota.
   `Verification:` Provide a screenshot confirming that you have generated an API key on Google AI Studio. **DO NOT** reveal the actual key.
4. **Modify the application**
   Follow the instructions in the README to add your API key to the project. Then, update the greeting

message from "Hello 梁安哲! Ask me anything about Wikipedia articles." to display your own name in Mandarin (note: 梁安哲 is the name of the TA).

5. **Containerize the application**
   Use docker to build a single container image that includes both the frontend and backend services.
   Verification: Provide the Dockerfile.

6. **Run the container**
   Use docker to build and run the container image.
   Verification: Provide screenshots showing the output of both docker build and docker run.

7. **Use Nginx as a Reverse Proxy**
   Configure the docker run command so that the frontend service is correctly mapped to a port on your VM's localhost. Then, set up nginx as a reverse proxy to expose the application on port 80, allowing it to function as a standard HTTP service. For reference, check this discussion thread, which highlights several hidden pitfalls when hosting streamlit applications with docker and nginx.
   Verification: Provide the configuration file of nginx.

8. **Verify External Accessibility.**
   Verify that you can access the application from your PC. Note that the port forwarding rules on your VM are: 80 → 80XY and 8000 → 90XY.
   Verification: From your PC, access http://<domain-of-the-VM>:<assigned-port-of-the-VM> in a browser. Provide screenshot of the application loaded successfully.

9. **Compare Linux distributions**
   Modify your Dockerfile to use two different base images: (1) ubuntu:22.04 and (2) alpine:3.22. Compare the build time and startup time of the two resulting images, and discuss the factors that may contribute to the differences in performance.
   Verification: Provide Dockerfile-ubuntu and Dockerfile-alpine.

10. **Compare container build speed**
    When writing a Dockerfile, it is common practice to install dependencies first and copy the source code afterward. Modify your Dockerfile so that the source code is copied as the first step instead. Then, compare the container build time when making changes to the source code. Discuss the reasons behind the differences in build time.

11. **Persistent chat history**
    Currently, the backend server stores chat history in memory. To ensure that data persists across container restarts, modify the backend to use local storage (either plain text files or SQLite). Then, update the docker run command to mount a directory from the host file system into the container.
    Verification:

    - Record a video demonstrating the following steps: 1. Chat with the Wiki Assistant; 2. Stop the running container on your VM; 3. Remove the container; 4. Start a new container using the same image; 5. Verify that the chat history is still displayed in the UI.
    - The modified server.py file and the full docker run command.

12. **Multiple containers**
    To improve maintainability, a common best practice is to separate different application functionalities into individual containers. Split the frontend and backend services into two separate containers, and then deploy the application using either docker-compose or Kubernetes. Due to the limited computing resources of the course VM, you may run the application on your local machine instead.
    Verification: Provide Dockerfile-frontend, Dockerfile-backend, the configuration YAML files for either docker-compose or Kubernetes, and screenshots demonstrating that the application is running correctly (e.g., output from docker compose ps or k9s.

13. **Improve the assistant (bonus)**
    This section is optional; you may leave it blank and still receive full credit.

    As an engineer, you may encounter research or product-related tasks in the future. A common approach to improving an existing system is to formulate a user story. For a more detailed introduction, refer to this blog post from Atlassian.
    Your user story should include the following four steps:

    (a) Define the persona of the user and describe the task he or she aims to accomplish.

    (b) Identify why the current system fails to complete or support this task.

    (c) Propose modifications to the system that would enable the task.

    (d) Evaluate whether the modified system meets the needs of the user.

    Write the user story and describe the corresponding improvements you implemented in the assistant, supplemented with screenshots demonstrating the new or enhanced functionality.

Deliverables

You are required to submit the following materials:

1. **Video:** Submit one video file named `<student-id>-demo.mp4` as described in the first section.

2. **System Setup Report:** Prepare a system setup report (maximum length: 10 pages) named `<student-id>-setup.pdf` that includes:

    - A description of how you containerized the application, along with the corresponding Dockerfiles.
    - The discussions requested in the first section.
    - Screenshots or command outputs demonstrating that the system is functioning correctly.

3. **Improvement Report:** If you choose to complete the bonus portion of the assignment, prepare an improvement report (maximum length: 2 pages) named `<student-id>-bonus.pdf` that includes all materials required in the bonus section.

Your submission should be concise, technically clear, and sufficient to allow another system administrator to reproduce your setup.