# Final Project: Retrieval-Augmented Generation

SHINE-MING WU SCHOOL OF INTELLIGENT ENGINEERING &
SCHOOL OF FUTURE TECHNOLOGY
Autumn 2024

## 1 Overview

In this project, you will work in a group of **two to three** students to complete the tasks. The final project consists of three parts.

In the first part, you are required to build a question answering system that can answer questions in a specific domain (i.e., aircraft). We provide documents in aircraft in materials. The recommended approach for this part is **retrieval-augmented generation**, but you are welcome to use different methods as well.

The second part is constructing a evaluation dataset. The evaluation dataset is constructed collaboratively by all groups. Each group is required to submit a evaluation dataset. The evaluation dataset should contain **20** questions, reference answers, supporting passages, document name and page number. The questions, answers, and supporting passages should be derived from the given documents. The evaluation dataset is then used to assess the systems developed by other groups.

The third part is evaluation. TA will distribute a question set to each group. You are required to take the question set as input and record the responses produced by your system. You are required to submit the responses within 1 hour after the release of the question set. Note that the questions distributing to each group are different.

## 2 QA System Implementation

We recommend using retrieval-augmented generation (RAG) to build the question answering system. Of course, you may also use different methods. In this section, we will provide a detailed introduction to retrieval-augmented generation to facilitate your understanding of this approach. Additionally, we will introduce the dataset for constructing the local knowledge base.

### 2.1 Details of Retrieval-Augmented Generation

In the realm of Large Language Models (LLMs) application, a plethora of challenges confront us, ranging from the paucity of domain-specific knowledge to the quandary of information accuracy and the specter of generated spurious content. Retrieval-Augmented Generation (RAG) emerges as a potent mitigatory strategy for these quandaries by introducing supplementary sources of information like external knowledge bases. RAG proves particularly efficacious in knowledge-intensive scenarios necessitating continual updates or specific domain applications. A notable advantage of RAG over alternative methodologies lies in its obviation of the necessity to retrain LLMs for specific tasks. Recently, RAG has gained significant traction owing to its successful deployment in applications such as conversational assistants.

RAG constitutes a technique amalgamating input with a corpus of pertinent supporting documents. These documents are incorporated into the input prompts and jointly fed into the text generator, thereby yielding the ultimate output. This mechanism of RAG finds particular utility in scenarios demanding adaptability to constantly evolving information landscapes, as the parameterized knowledge upon which LLMs rely is inherently static. Through RAG, language models can directly access the latest information without necessitating retraining, facilitating the generation of reliable, retrieval-based outputs.

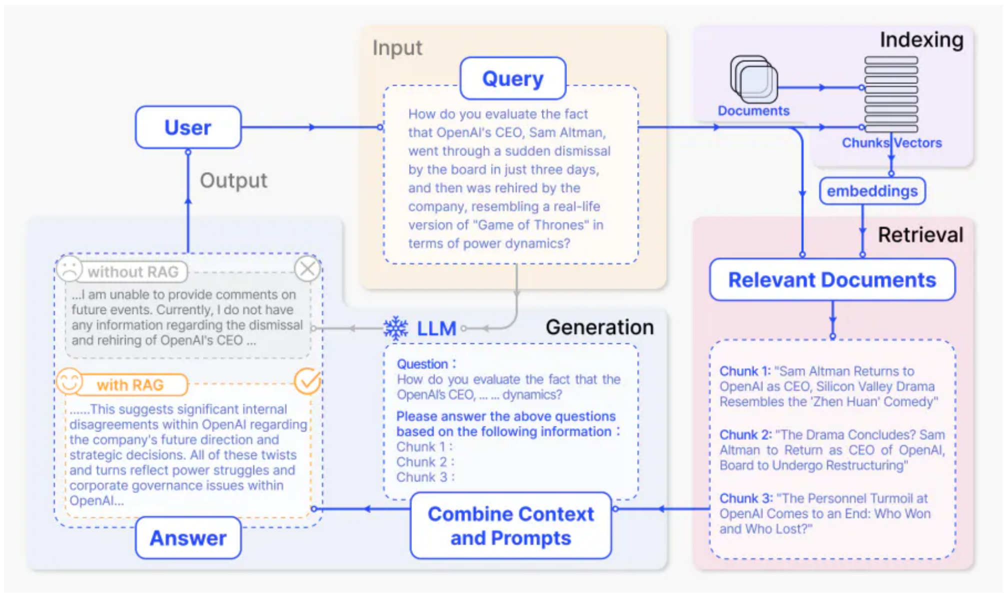
In essence, RAG enhances the accuracy, controllability, and relevance of LLM responses through retrieved evidence, thereby proving invaluable for problem-solving in rapidly evolving environments and effectively mitigating issues of erroneous information generation and performance degradation.

A typical application of RAG is illustrated in figure 1. Here, a user poses a question to ChatGPT about a recent, widely discussed news. Given ChatGPT's reliance on pretraining data, it initially lacks the capacity to provide updates on recent developments. RAG bridges this information gap by sourcing and incorporating knowledge from external databases. In this case, it gathers relevant news articles related to the user's query. These articles, combined with the original question, form a comprehensive prompt that empowers LLMs to generate a well-informed answer.

The typical RAG follows a traditional process that includes indexing, retrieval, and generation, which is also characterized as a "Retrieve-Read" framework.

**Indexing**. Starts with the cleaning and extraction of raw data in diverse formats like PDF, HTML, Word, and Markdown, which is then converted into a uniform plain text format. To accommodate the context limitations of language models, text is segmented into smaller, digestible chunks. Chunks are then encoded into vector representations using an embedding model and stored in vector database. This step is crucial for enabling efficient similarity searches in the subsequent retrieval phase.

**Retrieval**. Upon receipt of a user query, the RAG system employs the same encoding model utilized during the indexing phase to transform the query into a vector representation. It then computes the similarity scores between the query vector and the vector of chunks within the indexed corpus. The system prioritizes and retrieves the top-K chunks that demonstrate the greatest similarity to the query. These chunks are subsequently used as the expanded context in prompt.



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?

...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 ...

......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...

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 :

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?"

Figure 1: A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

**Generation**. The posed query and selected documents are synthesized into a coherent prompt to which a large language model is tasked with formulating a response. The model's approach to answering may vary depending on task-specific criteria, allowing it to either draw upon its inherent parametric knowledge or restrict its responses to the information contained within the provided documents. In cases of ongoing dialogues, any existing conversational history can be integrated into the prompt, enabling the model to engage in multi-turn dialogue interactions effectively.

However, the typical RAG encounters notable drawbacks:

**Retrieval Challenges**. The retrieval phase often struggles with precision and recall, leading to the selection of misaligned or irrelevant chunks, and the missing of crucial information.

**Generation Difficulties**. In generating responses, the model may face the issue of hallucination, where it produces content not supported by the retrieved context. This phase can also suffer from irrelevance, toxicity, or bias in the outputs, detracting from the quality and reliability of the responses.

**Augmentation Hurdles**. Integrating retrieved information with the different task can be challenging, sometimes resulting in disjointed or incoherent outputs. The process may also encounter redundancy when similar information is retrieved from multiple sources, leading to repetitive responses. Determining the significance and relevance of various passages and ensuring stylistic and tonal consistency add further complexity. Facing complex issues, a single retrieval based on the original query may not suffice to acquire adequate context information.

Moreover, there's a concern that generation models might overly rely on augmented information, leading to outputs that simply echo retrieved content without adding insightful or synthesized information.

You can try to challenge these limitations and enhance the performance of the QA system when you build the system using retrieval enhancement generation.

## 2.2 System Features

In conclusion, if you want to implement a local knowledge base QA system based on retrieval-augmented generation, you need to implement the following features:

- Document Extraction: Raw document data is extracted into the system. These data are preprocessed, and functions from the LangChain library can be used for this purpose. LangChain provides various document loaders to handle data in various forms from many different sources.
- Document Preprocessing: After document loading, transformations are typically performed. One method of transformation is text segmentation, which breaks long texts into smaller segments. This is crucial for fitting text into embedding models, such as e5-large-v2, where the maximum token length is 512. While text segmentation may sound straightforward, it can be a nuanced process that requires careful design of text segmentation functions. Of course, you can also use functions from the LangChain library, but you will need to adapt them to the Chinese language context.
- Embedding Generation: When extracting data, it must be converted into a format that the system can efficiently handle. Generating embeddings involves converting data into high-dimensional vectors, representing text in numeric format. This functionality requires the involvement of embedding models. Please note that you should use models with strong Chinese capabilities to avoid suboptimal performance.
- Storing Embeddings in a Vector Database: Processed data and generated embeddings are stored in a dedicated database known as a vector database. These databases are optimized for handling vectorized data to enable fast search and retrieval operations.
- LLM (Large Language Model): LLMs are the foundational generation components of the RAG (Retrieval-Augmented Generation) process. These advanced language models are trained on vast datasets, enabling them to understand and generate human-like text. In the RAG environment, LLMs are used to generate responses based on the context information retrieved from the vector database during user queries and user query periods.

- Querying: When a user submits a query, the RAG system performs efficient searches using indexed data and vectors. The system identifies relevant information by comparing the query vector with the vectors stored in the vector database. Then, LLMs formulate appropriate responses using the retrieved data.

### 2.3 Details of Documents

The documents used to construct local knowledge base comprises eight foundational textbooks on civil aircraft maintenance. These textbooks include "Introduction to Aviation," "Aircraft Maintenance," "Aircraft Structures and Systems," "Helicopter Structures and Systems," "Aviation Turbine Engines," "Piston Engines and Their Maintenance," "Basic Skills for Aircraft Maintenance," and "Aircraft Maintenance Practice."

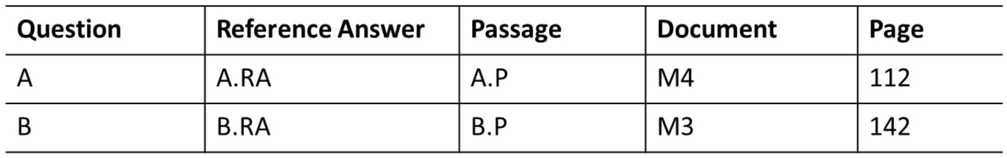
This series is compiled by the China Civil Aviation Maintenance Association based on the Civil Aviation Administration of China (CAAC) regulations CCAR-66R3 "Regulations for the Licensing of Civil Aircraft Maintenance Personnel" and AC-66-FS-002 R1 "Standards for Basic Knowledge and Practical Training in Aircraft Maintenance." The objective of these textbooks is to support the reform of the maintenance personnel licensing system and the high-quality development of the industry. Covering a wide range of content from basic theory to practical operations, these textbooks are primarily used for CCAR-66 licensing examinations, CCAR-147 maintenance training, continuing education for in-service maintenance personnel, and as instructional materials in relevant academic institutions. They provide an essential resource for enhancing the professional knowledge and skills of civil aircraft maintenance personnel.

## 3 Evaluation

The evaluation dataset is constructed collaboratively by all groups. Each group is required to submit a evaluation dataset consists of **20** questions, reference answers, supporting passages, document name, page number. Reference answers are responses that you believe can effectively answer the questions. These reference answers serve as a benchmark for the TA to evaluate the performance of the model. Supporting passages are the original excerpts from the given documents that contain the answers and serve as evidence and source material. The document name and page number of the supporting passage should also be provided. The questions, answers, and supporting passages should be derived from the provided documents. The questions should be diversified and not extracted solely from a single document. It is recommended to utilize multiple documents as sources to ensure a well-rounded and comprehensive set of questions. The evaluation dataset for each group should be recorded in the format shown in Figure 2. This dataset should be sent to the TA in the form of an **Excel file**.

After collecting all evaluation datasets, TA will release a question set to each group. The question set is different for different group. Each group should take the questions as input and record the system's responses in an **Excel file**. Each group should submit the responses within 1 hour after the release of the question set.

Note that the submission filename for the **evaluation dataset** should be `StudentName_1_StudentName_2_GroupNumber_evaluation_dataset.xlsx` and the submission filename for the **evaluation response** should be `StudentName_1_StudentName_2_GroupNumber_evaluation_response.xlsx`. All files are submitted via **Blackboard**.



| Question | Reference Answer | Passage | Document | Page |
|---|---|---|---|---|
| A | A.RA | A.P | M4 | 112 |
| B | B.RA | B.P | M3 | 142 |

Figure 2: Evaluation Results Format

## 4 Grading Breakdown

The grading breakdown for the final project assessment is as follows:

- Report Quality (30%): The quality of the written report constitutes 30% of the overall project grade. This component evaluates the clarity, completeness, and professionalism of the documentation provided.
- Evaluation Dataset Quality (20%): The quality of the evaluation dataset used in the project accounts for 20% of the total grade. This involves assessing the relevance, accuracy, and comprehensiveness of the dataset selected for testing and validation purposes.
- Response Quality (30%): The effectiveness and accuracy of the responses generated by the project make up 30% of the final score. This metric focuses on the quality of responses in the evaluation part.
- Code (20%): The organization, readability, and efficiency of the code developed during the project contribute to 20% of the overall grade. This includes adherence to coding standards, implementation of best practices, and overall code quality.

## 5 Submission

Please finish the tasks outlined in previous sections and complete the **codes** and **report**. The report should include the name and student id of each group member and the contribution of each group member. The leader of a group should submit a zip file entitled `StudentName_1_StudentName_2_GroupNumber_project.zip` to **Blackboard**, which consists of a report entitled `StudentName_1_StudentName_2_GroupNumber_project_report.pdf`, codes entitled `StudentName_1_StudentName_2_GroupNumber_project_code.zip`, and responses entitled `StudentName_1_StudentName_2_GroupNumber_response.xlsx`

| **Task** | **Deadline** | **Time** |
|---|---|---|
| Submit evaluation dataset | Dec 15 | 20:00 |
| Release question (Done by TA) | Dec 16 | 14:00 |
| Submit response | Dec 16 | 15:00 |
| Final report, code, and response submission | Dec 19 | 20:00 |

Table 1: Schedule for Submission

The table outlines the schedule for the submission of evaluation datasets, responses, final report, and code. The responses submitted on December 22nd will be used for evaluation purposes. The instructor will evaluate the quality of response and give marks based on the submissions made on December 23rd. The responses submitted on December 26th will be archived for record-keeping. Please adhere to the deadlines to ensure a smooth and timely submission.