# End-to-End Retrieval-Augmented Generation (RAG) System

**DSAN 5800 Project Report**

Yuting Fan Peng Li Yiwei Qi

2024-12-10

## 1. Executive Summary

This report details the development and deployment of a Retrieval-Augmented Generation (RAG) system that combines dense retrieval techniques and generative language modeling to provide accurate, grounded, and explainable responses to user queries. The system includes a document retriever using FAISS, integration with GPT for response generation, and a user-friendly Streamlit web application with features like document upload and source attribution.

## 2. Objectives

The primary goals of this project were:
1. To implement a **retrieval system** using dense retrieval (e.g., FAISS).
2. To integrate a **language model** (GPT) with the retriever for accurate response generation.
3. To build a web application with: - A **chat interface** for user queries. - A **document upload** feature to expand the knowledge base. - **Source attribution** for transparency. 4. To evaluate the system's retrieval and response generation quality using quantitative and qualitative metrics.

## 3. System Architecture

### 3.1 Overview

The system architecture consists of three main components:
1. **Retriever**: Utilized FAISS for dense retrieval, processing user queries and fetching top-k relevant documents from a preprocessed knowledge base.

2. **Generator**: Integrated GPT for natural language generation, synthesizing retrieved document content into coherent and accurate responses.
3. **Web Application**: A React-based frontend connected to a Python backend (Streamlit) for handling user interactions, retrieval, and response generation.

## 4. Implementation Details

### 4.1 Knowledge Base

- Knowledge Base Sources: Custom Uploads.
- Preprocessing:
  - Text cleaning, tokenization, and embedding generation using Hugging Face model SentenceTransformers.
  - Indexed using FAISS for fast retrieval.

### 4.2 Retrieval

- Model: Dense Passage Retrieval (DPR) with FAISS backend.
- Strategy:
  - Queries converted to embeddings.
  - Retrieved top-5 documents based on cosine similarity.

### 4.3 Response Generation

- Model: GPT-3 fine-tuned on conversational datasets.
- Input: Query + Retrieved document snippets.
- Output: A synthesized, natural language response grounded in retrieved content.

### 4.4 Web Application

- **Frontend**: React-based, featuring:
  - Real-time chat interface.

  - Drag-and-drop document upload.
  - Source attribution UI for retrieved documents.

- **Backend**: Streamlit API handling retrieval, GPT generation, and database updates.

## 5. Evaluation

### 5.1 Retrieval Quality

- **Metrics**:
  - Precision: xxx
  - Recall: xxx
- Observations: The retriever efficiently fetched relevant documents for most queries.

### 5.2 Generation Quality

- **Metrics**:
  - BLEU Score: xxx
  - ROUGE-L: xxx
- Human Evaluation:
  - Responses were judged **factually accurate** (xx%), **coherent** (xx%), and **contextually relevant**
- Comparison between `distilgpt2` and `gpt-3.5-turbo`
  - `distilgpt2`: xxxxxxxx
  - `gpt-3.5-turbo`: xxxxxxx

### 5.3 Sample Documents Test

## 6. Challenges and Solutions

1. **Challenge**: Initial issues with irrelevant retrieval results.
   - **Solution**: Fine-tuned the retriever embeddings using domain-specific data.
2. **Challenge**: Latency in document retrieval and response generation.
   - **Solution**: Optimized FAISS index and batch processing in GPT integration.

## 7. Key Features

1. **Real-time Chat Interface**: Users can submit queries and receive responses within seconds.

2. **Document Upload**: Expandable knowledge base to include new documents dynamically.

3. **Source Attribution**: Transparency by displaying document sources used for each response.

## 8. Conclusion

This project successfully developed a RAG system that combines advanced retrieval and generative capabilities, providing accurate and explainable responses. The system was evaluated positively for both performance and usability, with potential applications in customer support, education, and research domains.

## 9. Future Improvements

- **Knowledge Base Expansion**: Incorporate multimodal data (e.g., images, PDFs).

- **Fine-Tuning**: Further optimize GPT for specific domains.

- **Improved Evaluation**: Use a larger dataset and more diverse user testing scenarios.

## 11. Appendix

- Code Repository: https://github.com/pengleee/Project-5800-RAG-2024Fall
- Knowledge Base Sources: Custom Uploads.

- Evaluation Dataset: User Queries.