Laboratory file

on

AGENTIC AI

School of Engineering and Technology

Department of Computer Science and Engineering

Subject code – **CSCR 3215**

SUBMITTED BY:

Rachna Singh(2023371832)
Syed Maaz Ahmad (2023566381)

SUBMITTED TO:

Mr. Ayush Singh

Sharda University

Greater Noida, Uttar Pradesh

Retrieval Augmented Generation (RAG) System Using Research Paper Corpus

1. Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation. However, they often generate hallucinated or ungrounded responses when answering domain-specific questions. This project implements a **Retrieval Augmented Generation (RAG)** system to address this limitation.

The system retrieves relevant contextual information from a corpus of research papers in PDF format and uses that context to generate grounded and accurate responses using a Large Language Model accessed via the OpenRouter API. The architecture integrates document loading, chunking, embedding generation, vector similarity search (FAISS), and response generation in a structured pipeline.

The implementation demonstrates how combining retrieval mechanisms with generative models significantly improves factual reliability and interpretability in question-answering systems.

2. Introduction

Large Language Models (LLMs) such as GPT and DeepSeek are powerful generative systems trained on massive datasets. However, they suffer from:

- Hallucination (generating incorrect information confidently)
- Lack of domain-specific grounding
- No real-time knowledge updates

To overcome these limitations, **Retrieval Augmented Generation (RAG)** is used.

RAG enhances LLM performance by:

- Retrieving relevant documents from an external knowledge base
- Injecting retrieved context into the prompt
- Generating answers grounded in retrieved evidence

This project builds a complete RAG pipeline using research papers as a knowledge source.

3. Problem Statement

Large Language Models often produce hallucinated or ungrounded responses when answering questions about specific documents or domains.

The objective of this project is to design and implement a Retrieval Augmented Generation (RAG) system that:

- Retrieves relevant information from a corpus of research PDFs.
- Uses retrieved context to generate grounded responses.
- Improves factual reliability in question-answering tasks.

4. Objectives

1. To implement a document-based question-answering system.
2. To integrate vector similarity search using FAISS.
3. To generate embeddings using a transformer-based embedding model.
4. To use an external LLM API (OpenRouter) for answer generation.
5. To evaluate how retrieval improves response accuracy.

5. Dataset / Knowledge Source

Type of Data:

PDF research papers

Source:

Publicly available research papers

Documents Used:

- Attention Is All You Need
- BERT: Deep Bidirectional Transformers for Language Understanding
- ImageNet Classification with Deep Convolutional Neural Networks
- Long Short-Term Memory

Description:

The dataset consists of foundational research papers in deep learning and transformer architectures. These documents form a domain-specific corpus suitable for evaluating RAG-based systems.

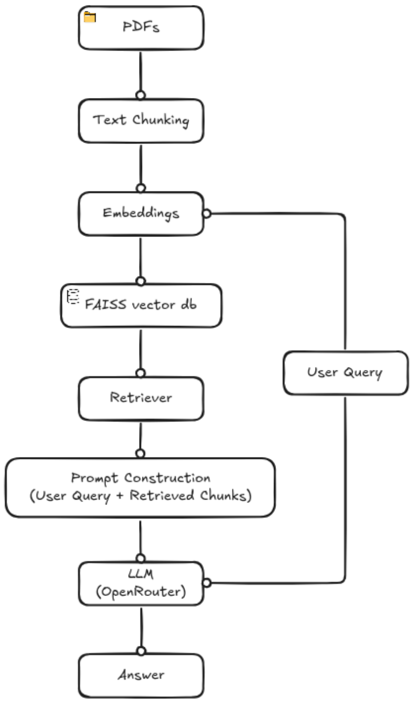
6. System Architecture

Overall RAG Architecture

The system follows this pipeline:

1. User submits a query.
2. Query is embedded using the same embedding model as documents.
3. FAISS performs similarity search to retrieve top-k relevant chunks.
4. Retrieved chunks are injected into a prompt template.

5. LLM generates a final grounded response.



7. Methodology

7.1 Document Loading

PDF documents are loaded using: PyPDFLoader

Each page is treated as an individual document.

7.2 Text Chunking

ToolUsed:
RecursiveCharacterTextSplitter

Parameters:

- Chunk Size: 500 characters
- Chunk Overlap: 100 characters

Reasoning:

- 500 characters provide sufficient semantic context.
- 100-character overlap prevents loss of information across chunk boundaries.
- Improves retrieval accuracy by preserving contextual continuity.

7.3 Embedding Generation

Model Used:
sentence-transformers/all-MiniLM-L6-v2

Reason for Selection:

- Lightweight and efficient
- Strong semantic similarity performance
- Widely adopted in RAG systems
- Suitable for small-to-medium datasets

Each chunk is converted into a dense vector representation.

7.4 Vector Database

VectorStore:
FAISS (Facebook AI Similarity Search)

Why FAISS?

- High-speed similarity search
- Efficient memory usage
- Seamless integration with LangChain
- Suitable for CPU environments

Top-k Retrieval:

search_kwargs={"k": 3}

Top 3 relevant chunks are retrieved for each query.

7.5 Prompt Engineering

Prompt Template:

You are an AI assistant. Use the following context to answer the question.

Context:

{context}

Question:

{question}

Answer:

Purpose:

- Force the model to rely on retrieved context
- Reduce hallucination

- Improve interpretability

7.6 Language Model Integration

LLMUsed:
deepseek/deepseek-r1-0528:free

Accessedvia:
OpenRouter API

Library:
ChatOpenAI from LangChain

8. RAG Pipeline Implementation

Core Function:

def rag_pipeline(query):

Steps:

1. Retrieve relevant chunks:
2. docs = retriever.invoke(query)
3. Combine retrieved content:
4. context = "\n\n".join([doc.page_content for doc in docs])
5. Format prompt:
6. final_prompt = prompt.format(...)
7. Generate response:
8. response = llm.invoke(final_prompt).content
9. Return:
    - Generated answer
    - Retrieved documents (for transparency)

9. Working of Retrieval Augmented Generation

Traditional LLM:

- Relies only on internal training knowledge.
- May hallucinate or produce outdated answers.

RAG:

- Retrieves relevant information from external knowledge.
- Injects context into the generation process.

- Produces grounded, fact-based responses.

Thus, RAG = Retrieval + Generation

This significantly improves:

- Factual reliability
- Transparency
- Explainability

10. Testing and Evaluation

Test Queries:

1. What is a transformer model?
2. How does attention mechanism work?
3. What are the limitations of large language models?

For each query:

- Retrieved sources are printed.
- Page number and source metadata are shown.
- Generated response is grounded in retrieved chunks.

This demonstrates:

- Proper retrieval behavior
- Context-aware generation
- Reduced hallucination

11. Advantages of the System

- Reduces hallucinations
- Domain-specific question answering
- Transparent source attribution
- Modular architecture
- Scalable to larger datasets
- Easy integration with UI tools

12. Limitations

- Fixed-size chunking may break semantic boundaries
- No reranking mechanism

- No hybrid search (BM25 + vector search)
- Dependent on external LLM API
- No evaluation metrics implemented

13. Future Improvements

1. Use semantic chunking instead of fixed-size chunking.
2. Implement cross-encoder reranking.
3. Add hybrid search (BM25 + FAISS).
4. Introduce metadata filtering.
5. Develop UI using Streamlit or Gradio.
6. Add evaluation metrics (Precision@k, Recall@k).
7. Use local LLM for full offline deployment.

14. Applications

- Research paper Q&A systems
- Enterprise knowledge base search
- Legal document analysis
- Medical document assistance
- Academic tutoring systems

15. Conclusion

This project successfully demonstrates the implementation of a Retrieval Augmented Generation (RAG) system using research PDFs as a knowledge base.

By integrating:

- Document loading
- Text chunking
- Embedding generation
- FAISS vector search
- Prompt engineering
- OpenRouter LLM

the system provides grounded, context-aware answers to user queries.

The results show that combining retrieval with generation significantly improves factual reliability compared to standalone LLMs.

This project highlights the practical importance of RAG in building trustworthy AI systems.