Harnessing RAG Fusion for AI-Powered PDF Query Systems

Project Report submitted to

Chhattisgarh Swami Vivekanand Technical University Bhilai (India)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

Aayush Sahu (301602221003)

Sumeet Pratap Singh (301602221010)

Komendra Sahu(301602221015)

Under the Guidance of

Ms. Aakash Sonkar

Asst. Prof.

Department of Computer Science and Engineering

Government Engineering College Raipur

Session 2024-2025

DECLARATION BY THE CANDIDATES

We, the undersigned, solemnly declare that the report of the project work entitled "**Harnessing RAG Fusion for AI-Powered PDF Query Systems"** is based on our own work carried out during our study under the supervision of **Asst. Prof. Mr. Aakash Sonkar**

We assert that the statements made and conclusions drawn are an outcome of the project work. We further declare, to the best of our knowledge and belief, that the report does not contain any part of any work that has been submitted for the award of any other degree/diploma/certificate in this University/deemed the University of India or any other country. All help received and citations used for the preparation of the project work have been duly acknowledged.

(Signature of the candidate)
Name: Aayush Sahu
Roll No: 301602221003
Enrollment No.: CA9144

Signature of the candidate)
Name: Sumeet Pratap Singh
Roll No: 301602221010
Enrollment No.: CA9139

Signature of the candidate)
Name: Komendra Sahu
Roll No: 301602221015
Enrollment No.: CA9150

(Signature of the Supervisor)
Ms. Aakash Sonkar
Asst. Prof.
Dept. Of Computer Science & Engineering,
Government Engineering College
Sejbahar, Raipur,

CERTIFICATE OF THE SUPERVISOR

This is to certify that the report titled "**Harnessing RAG Fusion for AI-Powered PDF Query Systems**" is a record of genuine research work carried out by Aayush Sahu (Roll No: 301602221003), Sumeet Pratap Singh (Roll No: 301602221010), and Komendra Sahu(Roll No: 301602221015) under my supervision and guidance. This work is submitted in partial fulfillment of the requirements for the Minor Project as part of the Bachelor of Technology degree program in Computer Science and Engineering by Chhattisgarh Swami Vivekanand Technical University, Bhilai (C.G.), India.

To the best of my knowledge and belief the Project Work:

- Embodies the work of the candidate himself,
- Has duly been completed,
- Fulfills the requirement of the ordinance relating to the BTech degree of the university.
- Is up to the desired standard both in respect of contents and language for being referred to the examiners.

(Signature)

Ms. Aakash Sonkar
Asst. Prof.
Dept. of Computer Science and Engineering

Prof. Dr. R H Talwekar
(Head of Department)
Dept. of Computer Science and Engineering
Government Engineering Collage, Raipur
(C.G)

,

CERTIFICATE BY THE EXAMINERS

The thesis entitled **“Harnessing RAG Fusion for AI-Powered PDF Query Systems”** submitted by **Aayush Sahu** (Roll No.: **301602221003** & Enrollment No.: CA9144), **Sumeet Pratap Singh** (Roll No.: **301602221010** & Enrollment No.: CA9139)and **Komendra Sahu**(Roll No.: **301602221015** & Enrollment No.:   CA9150) has been examined by the undersigned as a part of the examination and is hereby recommended for the award of the degree of **Bachelor of Technology** in the field of **Computer Science and Engineering** of **Chhattisgarh Swami Vivekanand Technical University, Bhilai.**

________________	________________
(Signature)	(Signature)
Internal Examiner	External Examiner
Date:	Date:

ACKNOWLEDGEMENT

I would like to thank our guide as well as the Head of Department, **Dr. R H Talwekar**, Department of Computer Science and Engineering, for his immense support and enlightened guidance for our project, which we have developed as B.Tech. students.

I also very thankful to **Mr. Pushpendra Dhar Dwivedi**, Assistant Professor and Project Coordinator for his continuous support and guidance. We are very grateful for the inspiring discussions with all our faculties, their valuable support and path-guiding suggestions have helped us to develop this project. Alongside this, we would like to take this opportunity to express our thanks to everyone in the Computer Laboratory of Government Engineering College, Sejbahar, and Raipur (C.G.).

I highly thankful to Prof. **Dr. R H Talwekar**, Head of the Computer Science Department, and Principal **Dr. M R Khan** for providing us necessary facilities cooperation, and constant encouragement throughout the project work and course

We especially thank our colleagues, batch mates, and friends for the support and help that they offered us during the development of this project.

____
(Signature of the candidate)
Name: Aayush Sahu
Roll No: 301602221003
Enrollment No.:CA9144

____
(Signature of the candidate)
Name: Sumeet Pratap Singh
Roll No: 301602221010
Enrollment No.:CA9139

____
(Signature of the candidate)
Name: Komendra Sahu
Roll No: 301602221015
Enrollment No.:CA9150

ABSTRACT

This study explores advancements in Retrieval-Augmented Generation (RAG) systems tailored for PDF-based question answering . By leveraging domain-specific PDFs as input data, the proposed framework incorporates advanced retrieval techniques, including multi-query generation and RAG fusion, to enhance the relevance and contextual accuracy of generated responses.

Utilizing LangChain for orchestrating interactions between the language model and vector database (ChromaDB), the system effectively processes, embeds, and retrieves information from large-scale PDF collections. ChromaDB ensures efficient storage and similarity searches for vectorized document embeddings.

The integration of advanced retrieval strategies, coupled with the dynamic query-handling capabilities of LangChain, significantly improves precision, reduces hallucinations, and enriches response quality for knowledge-intensive tasks. Preliminary results demonstrate the effectiveness of this approach in delivering timely and contextually accurate answers, showcasing its potential in addressing the challenges of traditional LLMs.

Table of Contents

List of Figures

CHAPTER 1-INTRODUCTION

1.1 Introduction

The rise of Large Language Models (LLMs) has transformed the field of natural language processing (NLP), enabling tasks such as question answering, summarization, and content generation with remarkable precision. LLMs like **Mistral** and **LLaMA** represent state-of-the-art architectures capable of understanding and generating human-like text. However, the effectiveness of these models depends significantly on how they process and retrieve contextual information .

Retrieval-augmented generation (RAG) presents a powerful solution to this problem. RAG systems retrieve information from external data sources, such as PDFs, databases, or websites, grounding the generated content in accurate and current data. This makes them particularly effective for knowledge-intensive tasks . By informing LLMs of reference materials related to questions in advance, RAG reduces hallucinations and enables the generation of more accurate and reliable answers. As a result, RAG architecture resolves the information shortage problem of LLMs, providing high-quality responses without requiring additional data training . To implement efficient retrieval mechanisms in RAG pipelines, tools like **FAISS** (Facebook AI Similarity Search) are employed. FAISS facilitates the storage and rapid retrieval of large document embeddings, enabling systems to handle massive datasets without compromising performance. Advanced configurations, such as **RAG Fusion** and **RAG Multi-query** further enhance retrieval

1.2 Objective

The objective of this project is to implement an advanced Retrieval-Augmented Generation (RAG) pipeline that enables intelligent interaction with PDF documents by combining retrieval and generation capabilities to extract relevant information and generate precise responses to user queries. The process begins by loading and preprocessing a PDF document using UnstructuredPDFLoader, which splits the document into manageable chunks for effective handling. These chunks are then embedded into a vector database created using Chroma, leveraging OllamaEmbeddings for semantic similarity search. To enhance retrieval accuracy, the pipeline employs a multiquery generation step, where an LLM like ChatOllama generates multiple variations of a user query. The retrieved results from these queries are ranked and aggregated using the Reciprocal Rank Fusion (RRF) method, ensuring that the most contextually relevant information is selected. A final RAG chain integrates the retrieved information with a prompt template and the LLM, generating user-specific responses. The system is designed for interactivity through the chat_with_pdf() function, allowing users to query and receive detailed, context-aware answers about the content of the PDF. This pipeline exemplifies efficient document-based question answering, incorporating state-of-the-art retrieval and fusion techniques for optimal performance.

CHAPTER 2-LITERATURE REVIEW

2.1 Literature Survey

no.	**Research Paper Title**	**Authors**	**Limitations**
1	Enhancing LLM Factual Accuracy with RAG to Counter Hallucinations	jiarui Li, Ye Yuan, and Zehua Zhang	Hallucination in Generative Mode ls
2	Where is the Answer? Investigating Positional Bias in Language Model Knowledge Extraction	lin Zu Kuniaki Saito Kihyuk Saito Kihyuk Sohn Chen-Yu Lee	Positional Bias in Information Extr action
3	CuriousLL M: Elevating Multi-Document QA with Reasoning-Infused Knowledge Graph Prompting	Haclin Zu Kuniaki Saito Kihyuk Saito Kihyuk Sohn Chen-Yu Lee	Real-Time Multi-Document Handling
4	Seven Failure Points When Engineering a Retrieval-Augmented Generation System	Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu,	Query Ambiguity and Retrieval Quality
5	Chat With Documents Using Large Language Model (LLM	Pathan Simran, Raykar Vaishnavi, Pandarka r Vaibhav	: Contextual Relevance and Response Accuracy

6	Loong: A Long-Context Benchmark for Evaluating Large Language models	Minzheng Wang, Longze Chen, Cheng Fu, Shengyi Liao,	Knowledge Representation for Long Contexts
7	INDIC QA BENCHMARK: A Multilingual Benchmark to Evaluate Question Answering Capability of LLMs for Indic Languages	Abhishek kumar singh⋄ , Rudra Murthy , Vishwajeet Kumar	Limited Multi-Lingual Retrieva
8	A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models	Umar Wenqi Fan,Yujuan Ding,Liangbo Ning	Scaling Vector Databases
9	Chat With Documents Using Large Language Model (LLM)	Pathan Simran, Raykar Vaishnavi, Pandarkar Vaibhav	Generalization to Unseen Queries

CHAPTER 3-PROBLEM IDENTIFICATION

3.1 Problem Statement

The issue revolves around handling large, unstructured technical documents to provide precise, context-aware answers to user queries. Traditional methods like keyword-based searches are insufficient for extracting insights from such voluminous and dense technical data. This project addresses the challenge of implementing a retrieval-augmented system using LangChain and vector databases to effectively split, embed, and retrieve knowledge from technical manuals

3.2 Challenges

Developing a scalable Retrieval-Augmented Generation (RAG) system for processing large technical documents presents several challenges across different stages of the project. These challenges can be broadly categorized into technical, implementation, and optimization aspects:

1. **Document Processing**: Handling diverse document formats, ensuring accurate chunking, and generating embeddings that retain semantic context.
2. **Vector Database**: Managing storage and retrieval efficiency in ChromaDB, especially for large datasets, while ensuring contextual relevance.
3. **Query Expansion**: Balancing precision and recall when implementing RAG fusion for diverse and complex user queries.
4. **Fine-Tuning the LLM**: Adapting a general-purpose language model for domain-specific content while managing resource constraints and ensuring response quality.
5. **Scalability and Deployment**: Designing a system architecture that is both scalable and efficient on local hardware, with seamless integration of all components.
6. **User Experience**: Ensuring natural language understanding for varied user queries and implementing robust error hand

3.3 Proposed System

The primary objective of this project is to develop a scalable Retrieval-Augmented Generation (RAG) system capable of processing large-scale technical documents to deliver efficient, contextually accurate question-answering capabilities. The proposed system aims to bridge the gap between extensive technical information and user accessibility by enabling quick, precise, and context-driven responses.

This project focuses on creating a robust framework that integrates cutting-edge technologies in Natural Language Processing (NLP) to enhance document comprehension, information retrieval, and contextual reasoning. The system will serve as a valuable tool for professionals, researchers, and learners navigating complex technical documentation

.

CHAPTER 4-SYSTEM DESIGN

4.1 System Overview

The Retrieval-Augmented Generation (RAG) Fusion pipeline is a robust system designed to query document-based data by combining advanced retrieval mechanisms with intelligent language generation. It is optimized for knowledge-intensive tasks such as question answering, summarization, and data extraction, providing accurate, contextually enriched responses. Its modular design ensures scalability, adaptability, and reliability across diverse applications.

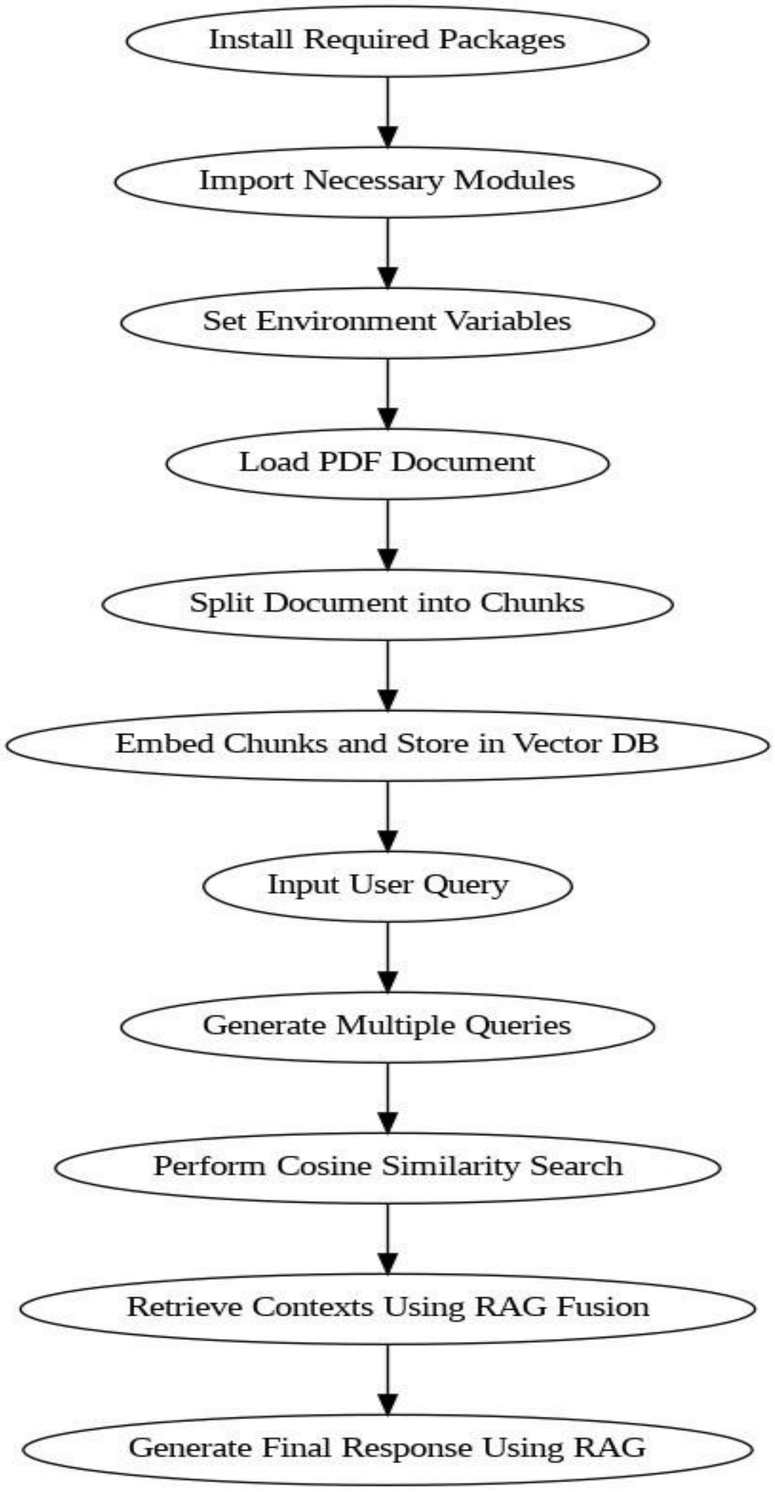
The process begins with a document processing module that ingests and extracts text from raw inputs, including PDFs and other unstructured formats. The extracted text is divided into manageable chunks using a text-splitting mechanism, ensuring each segment is coherent and well-suited for embedding and retrieval. Overlap between chunks maintains contextual flow, critical for questions that span multiple sections.

These text chunks are then transformed into dense vector representations by the embedding generation module, which encodes semantic meaning. This allows the system to retrieve information based on contextual relevance rather than simple keyword matching. The vector representations are stored in a vector database optimized for rapid similarity searches, ensuring efficient handling of large datasets.

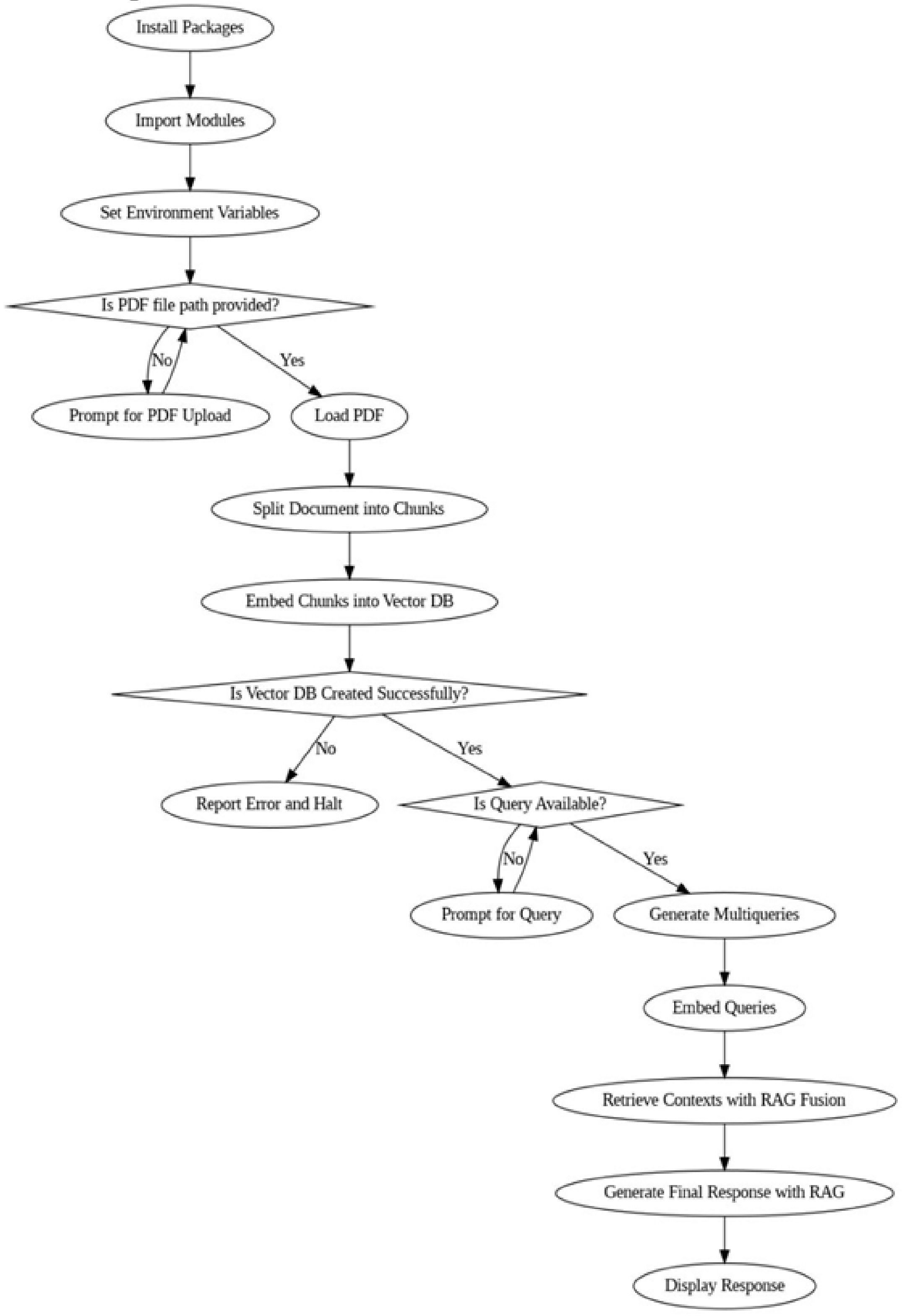
To enhance retrieval, the query generation module dynamically reformulates user queries into multiple variations, improving the system’s ability to capture diverse interpretations of the input. This multi-query retrieval approach broadens the scope of information retrieval, ensuring all relevant document segments are considered.

The retrieved results from multiple queries are consolidated through a ranking and fusion mechanism, which scores and prioritizes the most relevant documents. Reciprocal Rank Fusion (RRF) is commonly used to combine results from different queries, ensuring the final set of retrieved contexts is comprehensive and high-quality.The language generation module synthesizes responses by integrating retrieved contexts with the user’s original query. This ensures that the generated output is coherent, grounded in the provided information, and free from hallucinations—factually incorrect but plausible-sounding responses**.**

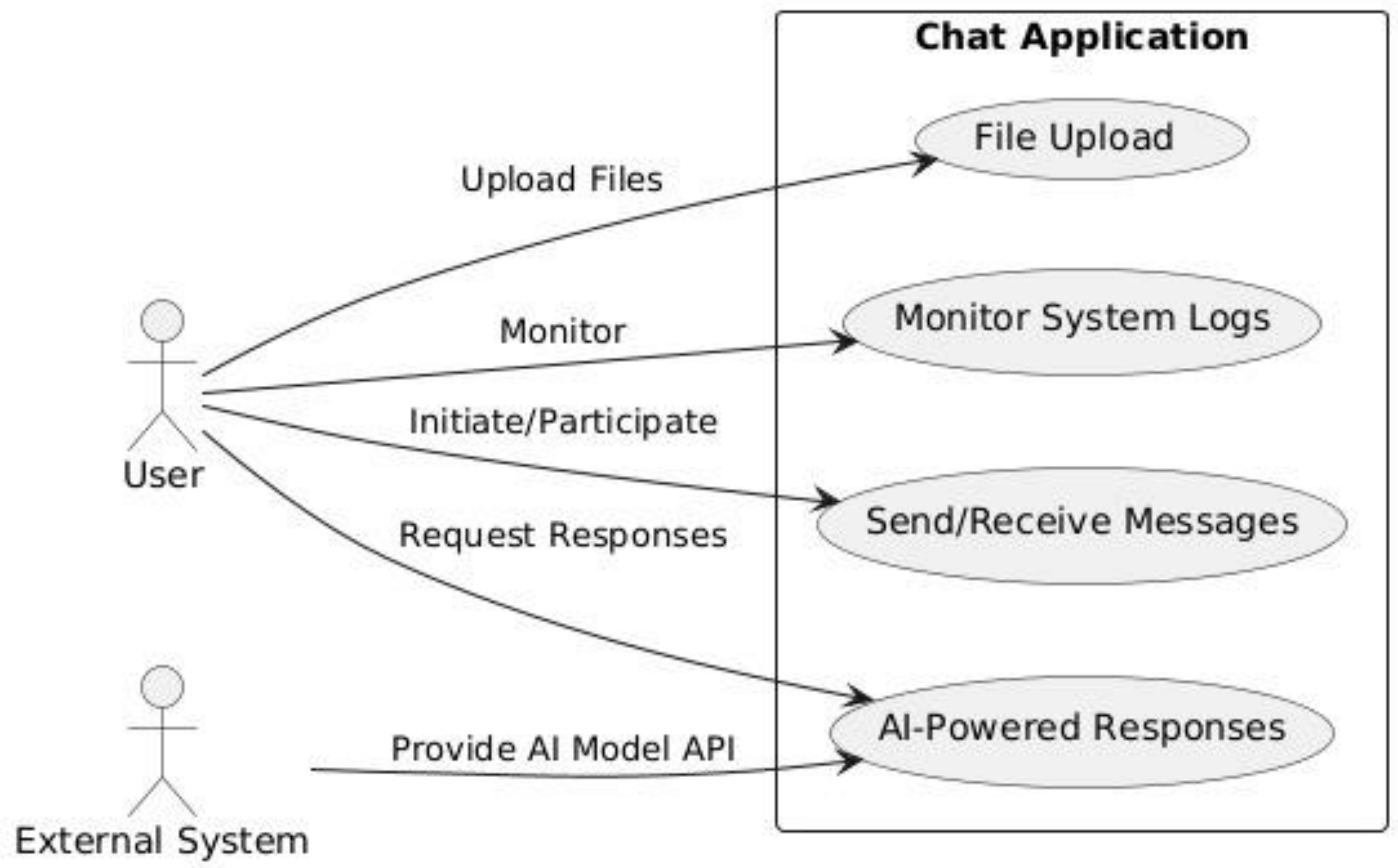
4 .2 Flow Chart


Install Required Packages
Import Necessary Modules
Set Environment Variables
Load PDF Document
Split Document into Chunks
Embed Chunks and Store in Vector DB
Input User Query
Generate Multiple Queries
Perform Cosine Similarity Search
Retrieve Contexts Using RAG Fusion
Generate Final Response Using RAG

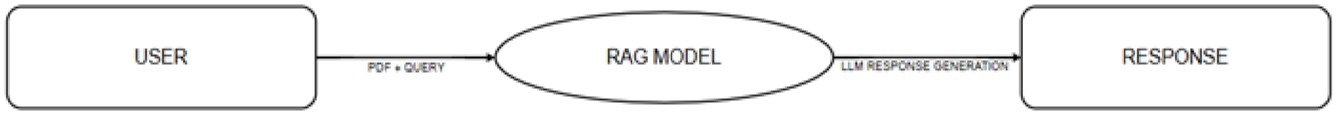
4.2 Decision Diagram


Install Packages
Import Modules
Set Environment Variables
Is PDF file path provided?
No
Yes
Prompt for PDF Upload
Load PDF
Split Document into Chunks
Embed Chunks into Vector DB
Is Vector DB Created Successfully?
No
Yes
Report Error and Halt
Is Query Available?
No
Yes
Prompt for Query
Generate Multiqueries
Embed Queries
Retrieve Contexts with RAG Fusion
Generate Final Response with RAG
Display Response

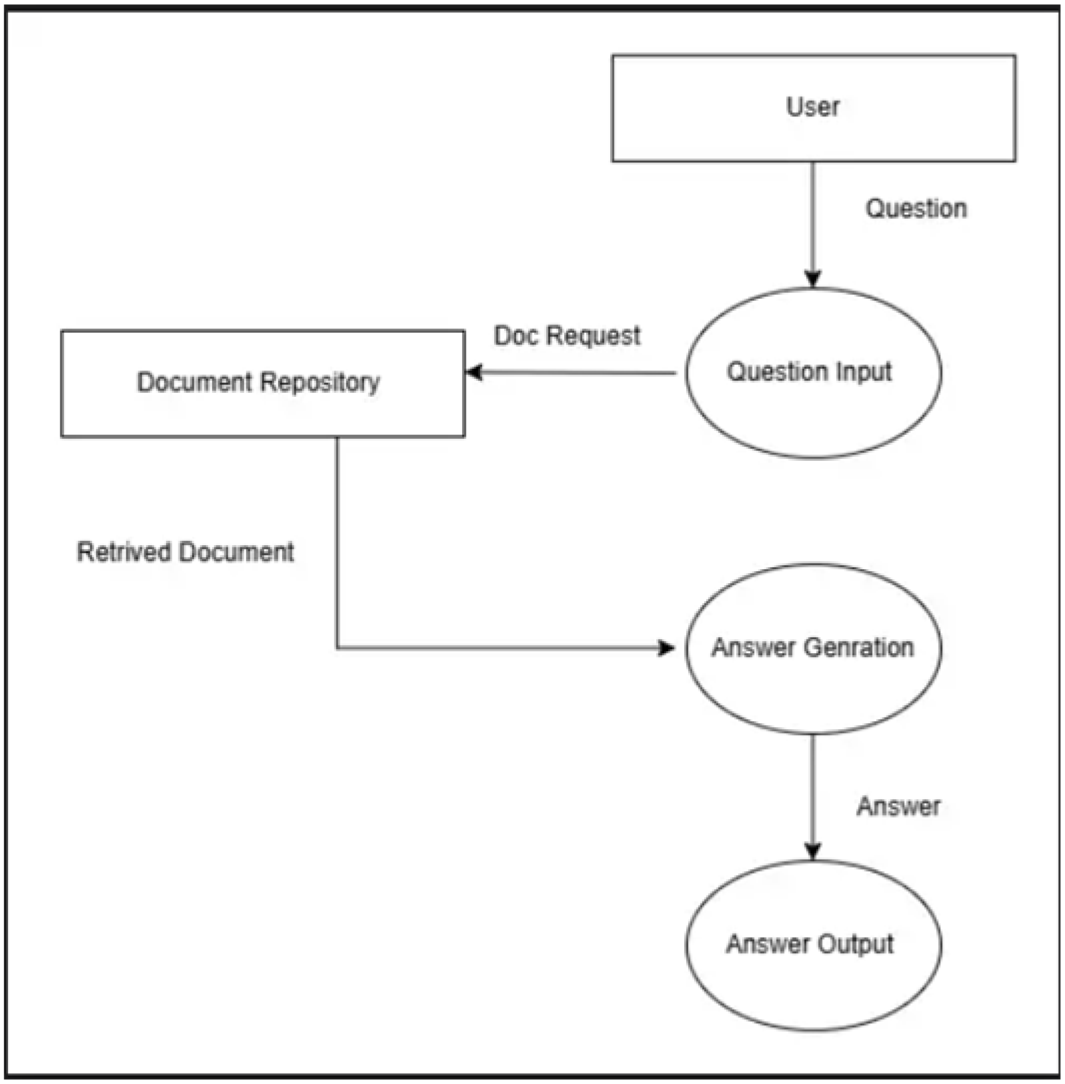
4.3 Use Case Diagram



4.4 DFD LEVEL-0



4.5 DFD LEVEL-1



CHAPTER 5-METHODOLOGY

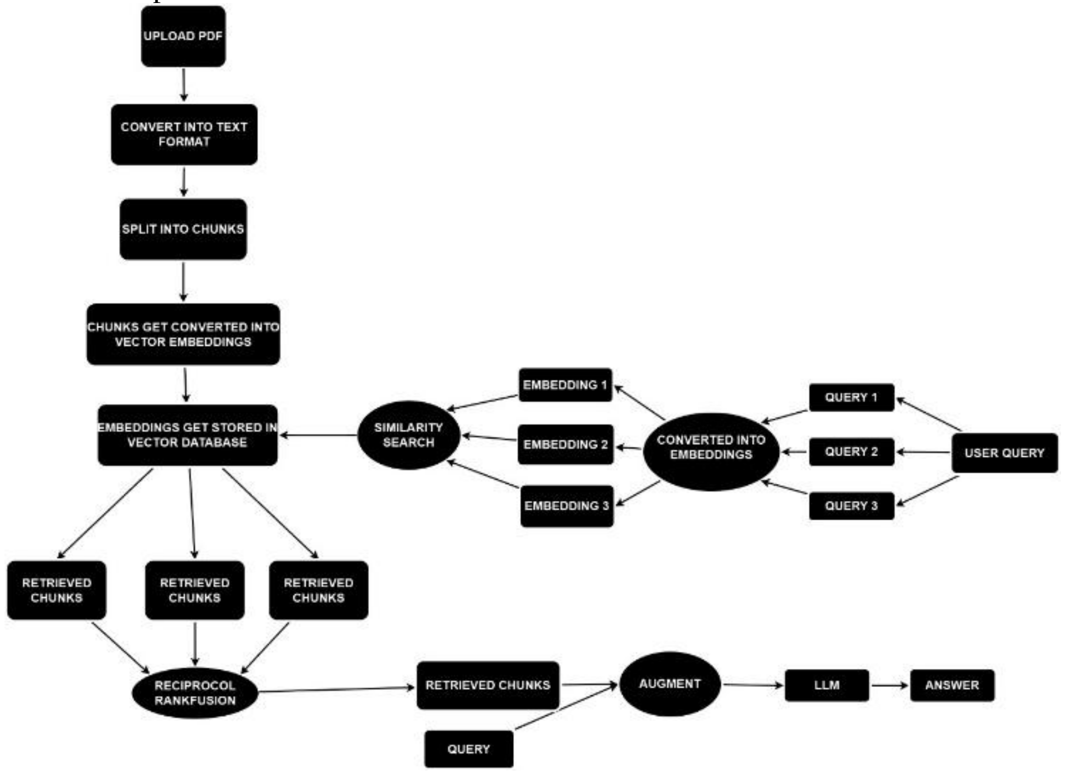
5.1 **Understanding Retrieval-Augmented Generation (RAG)**

Retrieval-Augmented Generation (RAG) models represent a paradigm shift in natural language processing, particularly in the domain of question-answering systems. These models combine two essential components: **Information Retrieval (IR)** and **Natural Language Generation (NLG)**, resulting in outputs that are both contextually relevant and factually accurate. The framework integrates dense retrieval methods with large-scale generative models to provide more accurate and up-to-date answers.

RAG models retrieve relevant passages from large corpora, augmenting the generative process with this data to enhance the factual grounding of responses. This dynamic integration ensures that the model's output is based on the most current and relevant information, bridging the gap between retrieving knowledge and generating contextually rich, coherent responses

5.2 **Architectural Overview of RAG**

RAG systems are structured around two key components: the **Retriever** and the **Generator**. These modules work together to retrieve relevant information from a knowledge base and then use that information to generate an accurate and contextually relevant response.



5.2.1 Retriever Module

The retriever module's primary function is to **identify relevant information** from a vast corpus of data. Using advanced **embedding techniques**, it transforms textual data into vector representations, allowing for quick similarity searches. By leveraging methods like **BM25**, **neural embeddings**, or transformer-based models, the retriever can efficiently find documents that are semantically related to the user's query.

The retriever focuses on **retrieving only relevant data**, ensuring efficiency and accuracy. This approach is particularly effective in situations where the available knowledge base is massive, as it narrows down the dataset to only the most pertinent information, enhancing the overall system's speed and precision.

5.2.2 Generator Module

Once the retriever has selected the most relevant documents, the **generator module** takes over to synthesize the information into a coherent response. The generator is based on **large generative models** like transformers, which create human-like text by drawing upon the retrieved data. This module doesn't just repeat the information; instead, it uses context from the retrieved documents to generate nuanced, fluent, and contextually relevant answers.
The generator ensures that the final output is not only factually correct but also semantically aligned with the user's query, producing answers that feel natural and informed by the retrieved context

5.3 Preprocessing Data for RAG

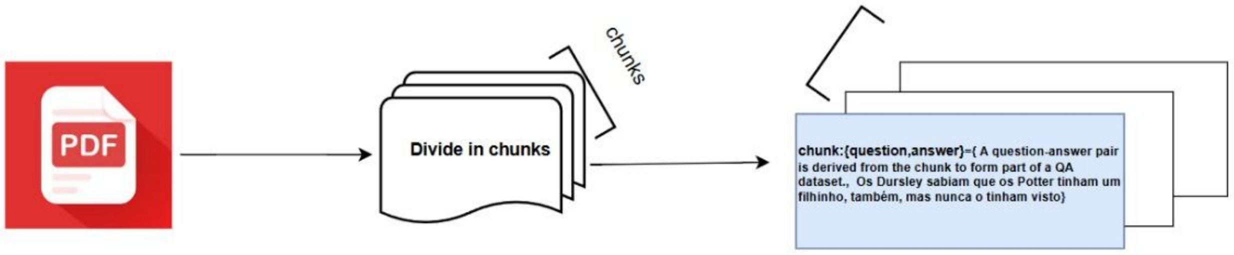
Preprocessing the data is a critical step in ensuring that the Retrieval-Augmented Generation (RAG) model can effectively retrieve relevant information from the provided PDF document and generate accurate answers. This section outlines the specific steps taken to preprocess the data in the context of this project.

1. **Loading the PDF Document**
   The first step in the preprocessing pipeline is to load the content of the PDF document. This is accomplished using the UnstructuredPDFLoader class, which extracts raw text from the PDF file

```
loader = UnstructuredPDFLoader(file_path=local_path)
data = loader.load()
```

2. **Splitting the Text into Chunks**
   Once the document is loaded, the next task is to split the raw text into smaller, manageable chunks. This is necessary because large documents are often too big to process as a single unit, and splitting them into smaller chunks allows for more efficient processing. To achieve this, the RecursiveCharacterTextSplitter is used, which splits the document into text chunks of up to 1000 characters, with an overlap of 200 characters between adjacent chunks to maintain context

```
text_splitter =
RecursiveCharacterTextSplitter(chunk_size=1000,chunk_overlap=200)
chunks = text_splitter.split_documents(data)
```


PDF
Divide in chunks
chunks
chunk:{question,answer}={ A question-answer pair is derived from the chunk to form part of a QA dataset., Os Dursley sabiam que os Potter tinham um filhinho, também, mas nunca o tinham visto}

3. **Embedding the Text Chunks**
   To enable efficient retrieval and similarity-based search, the text chunks are converted into vector representations. This is done using **OllamaEmbeddings**, which leverages a pre-trained model to map each text chunk into a high-dimensional vector space The **Chroma** vector store is used to index these vectors, making it easy to perform similarity searches during the retrieval phase.

```
vector_db = Chroma.from_documents(
    documents=chunks,
    embedding=OllamaEmbeddings(model="nomic-embed-text"),
    collection_name="local-rag"
)
```

4. **Multi-Query Generation for RAG**
   To enhance the retrieval process and ensure that a broad range of relevant information is captured, multiple queries are generated from the original question. This is achieved by using the MultiQueryRetriever class, which generates alternative queries based on the original question. Generating multiple queries helps capture different ways the relevant information might be phrased in the document..

```
retriever = MultiQueryRetriever.from_llm(
    vector_db.as_retriever(),
    llm,
    prompt=QUERY_PROMPT
)
```

5. **Reciprocal Rank Fusion (RRF)**
   To enhance the retrieval process and ensure that a broad range of relevant information is captured, multiple queries are generated from the original question. This is achieved by using the MultiQueryRetriever class, which generates alternative queries based on the original question. Generating multiple queries helps capture different ways the relevant information might be phrased in the document**.**

```
        def reciprocal_rank_fusion(results: list[list], k=60):
            fused_scores = {}
            for docs in results:
                for rank, doc in enumerate(docs):
                    doc_str = dumps(doc)
                    fused_scores[doc_str] = fused_scores.get(doc_str, 0) + 1 / (rank + k)

            reranked_results = [
                loads(doc) for doc, score in sorted(fused_scores.items(), key=lambda x: x[1],
        reverse=True)
            ]
    return reranked_results
```

5.4 **Tools and Frameworks Used**

- **LangChain:**

    1. Document Loading: UnstructuredPDFLoader for extracting text from PDFs.
    2. Text Splitting: RecursiveCharacterTextSplitter for splitting text into smaller chunks.
    3. Embeddings: OllamaEmbeddings for converting text into vector representations.
    4. Retrievers: MultiQueryRetriever for generating multiple queries to enhance retrieval.
    5. Vector Stores: Chroma for storing and querying document embeddings.
    6. RAG Pipeline: RunnablePassthrough and StrOutputParser to manage data flow and parse results

- **Ollama**

    1. OllamaEmbeddings: For generating vector embeddings from text.
    2. ChatOllama: For interacting with language models to generate answers.
       Vector Stores: Chroma for storing and querying document embeddings.

- **ChromaDb**

Chroma is a vector database used to store document embeddings and perform similarity searches, enabling efficient retrieval of relevant context during the RAG process.

- **Python**
  Python was the main programming language used, with libraries like

:

1. **pip**: For managing dependencies.
2. **os**: For environment variable management.
3. **IPython.display**: For displaying results in Jupyter notebooks.

5.4.1 Ollama Models

1. **Nomic-embed-text**
   - **Purpose**: Converts text into high-dimensional vector embeddings that capture semantic meaning.
   - **Role in Project**: Used to create embeddings for document chunks, enabling efficient similarity searches in the vector database.
2. **Mistral**
   - **Purpose**: A large language model designed for natural language understanding and generation.
   - **Role in Project**: Utilized in the Retrieval-Augmented Generation (RAG) pipeline to generate accurate and context-aware responses based on retrieved document data.

5.4.2 Vector Databases

Vector database is a new type of database developed to solve the problem of long term memory deficiency in LLM. This database is specialized in efficiently storing and managing high-dimensional real vector indexes. Unlike traditional databases, vector databases express queries in the form of real vectors (embedding) and extract data with similar vectors. Supports the method. Therefore, it is optimized for storing and utilizing vectors obtained as a result of AI models. A typical vector pipeline for a vector database goes through the following three steps

1. **Indexing :**Vector databases index vectors using algorithms such as PQ, LSH, or HNSW. This step maps the vectors to a data structure that allows for faster retrieval. The final proximity data is retrieved and post-processed to return the final result. This step may involve re ranking the closest data using a different similarity measure.

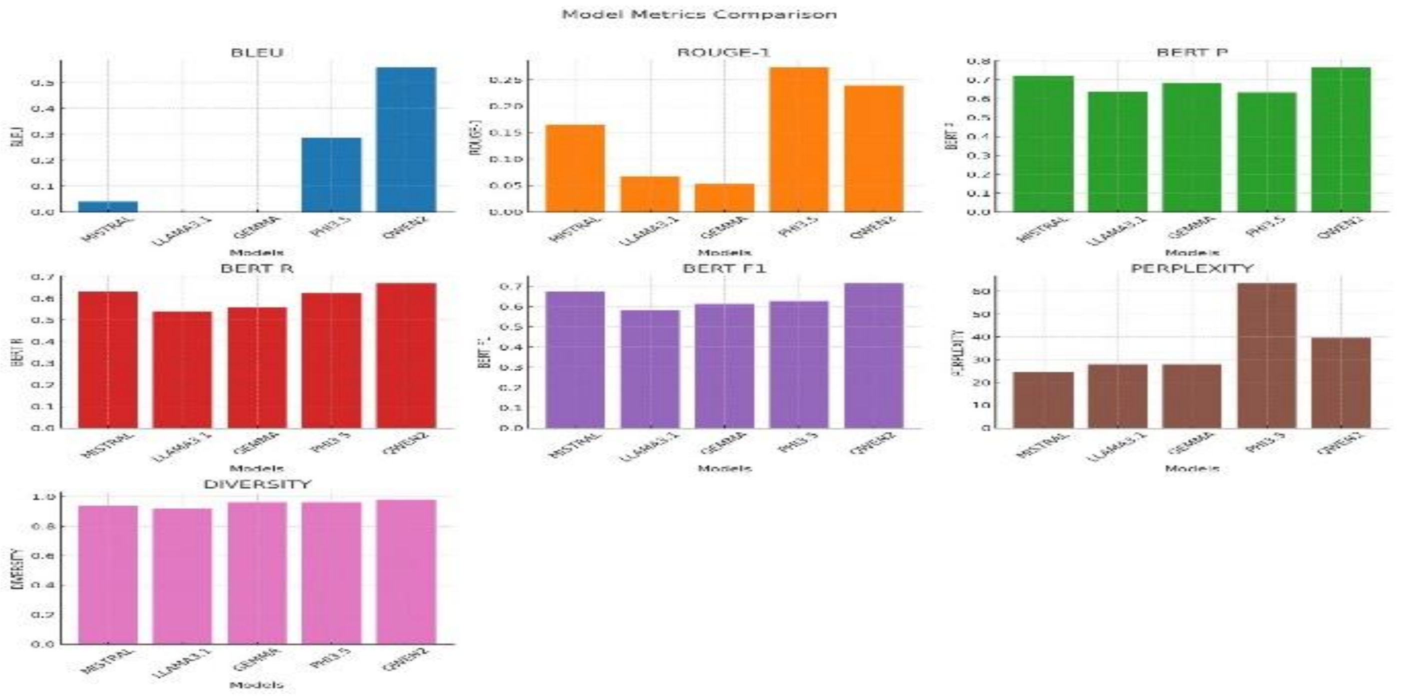
2. **Querying :** A vector database compares the indexed query vector with the indexed vector in the data set to find the closest data. At this time, the similarity metric used in the corresponding index is applied

3. **Post Processing:** In some cases, the vector database is the closest to data set The final proximity data is retrieved and post-processed to return the final result. This step may involve re ranking the closest data using a different similarity measure

CHAPTER 6-EVALUATION METRICS

6.1 Comparision between LLMs

MODEL	BLEU	ROUGE-1	BERT P	BERT R	BERT F1	PERPLEXITY	DIVERSITY
MISTRAL	0.040933	0.164179	0.723513	0.631149	0.674182	24.53725	0.9375
LLAMA3.1	5.97E-09	0.068003	0.637558	0.539186	0.58426	27.99945	0.92
GEMMA	3.29E-07	0.05317	0.683907	0.557536	0.61429	27.93233	0.962963
PHI3.5	0.285397	0.273458	0.633039	0.624801	0.628893	63.4351	0.962433
QWEN2	0.559377	0.239521	0.766305	0.66944	0.714605	39.47182	0.981481



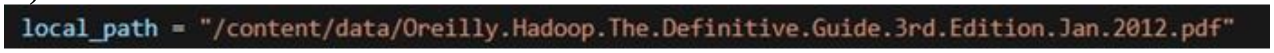
In this table ,the evaluation of several language models—Qwen 2, Phi 3.5, Gemma, Mistral, and LLaMA 3.1—was conducted across multiple performance metrics, including BLEU, ROUGE-1, BERT scores, perplexity, and diversity. The BLEU score, which measures the precision of n-grams between the generated responses and reference texts, was calculated using the sacrebleu library, with Qwen 2 achieving the highest BLEU score of 0.5594, indicating superior surface-level accuracy. The ROUGE-1 score, reflecting the overlap of unigrams between generated and reference responses, was assessed using the rouge_score library, with Phi 3.5 performing well at 0.2735, highlighting its strength in surface-level matching but also revealing challenges in fluency due to its high perplexity. BERT scores, evaluating the alignment of generated and reference texts at the semantic level, were computed using the bert_score library, with Qwen 2 also excelling in BERT Precision (0.7663), indicating strong semantic alignment. Perplexity, an indicator of fluency, was computed using GPT-2's log-likelihood estimation, with Mistral showing the lowest perplexity, reflecting its fluency and coherence in text generation. Lastly, diversity was measured by analyzing the uniqueness of bigrams in the generated text, with Qwen 2 achieving a high diversity score of 0.9815, suggesting a well-balanced ability to generate varied and creative responses. Overall, Qwen 2 demonstrated a well-rounded performance across all metrics, emerging as the top performer, while other models such as Phi 3.5, Gemma, Mistral, and LLaMA 3.1 each exhibited strengths in specific areas, such as diversity, fluency, and semantic precision, depending on the use case.

CHAPTER 7-RESULT AND CONCLUSION

7.1 Result

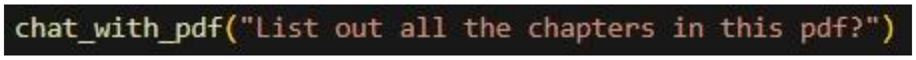
INPUT:

i)PDF Path:



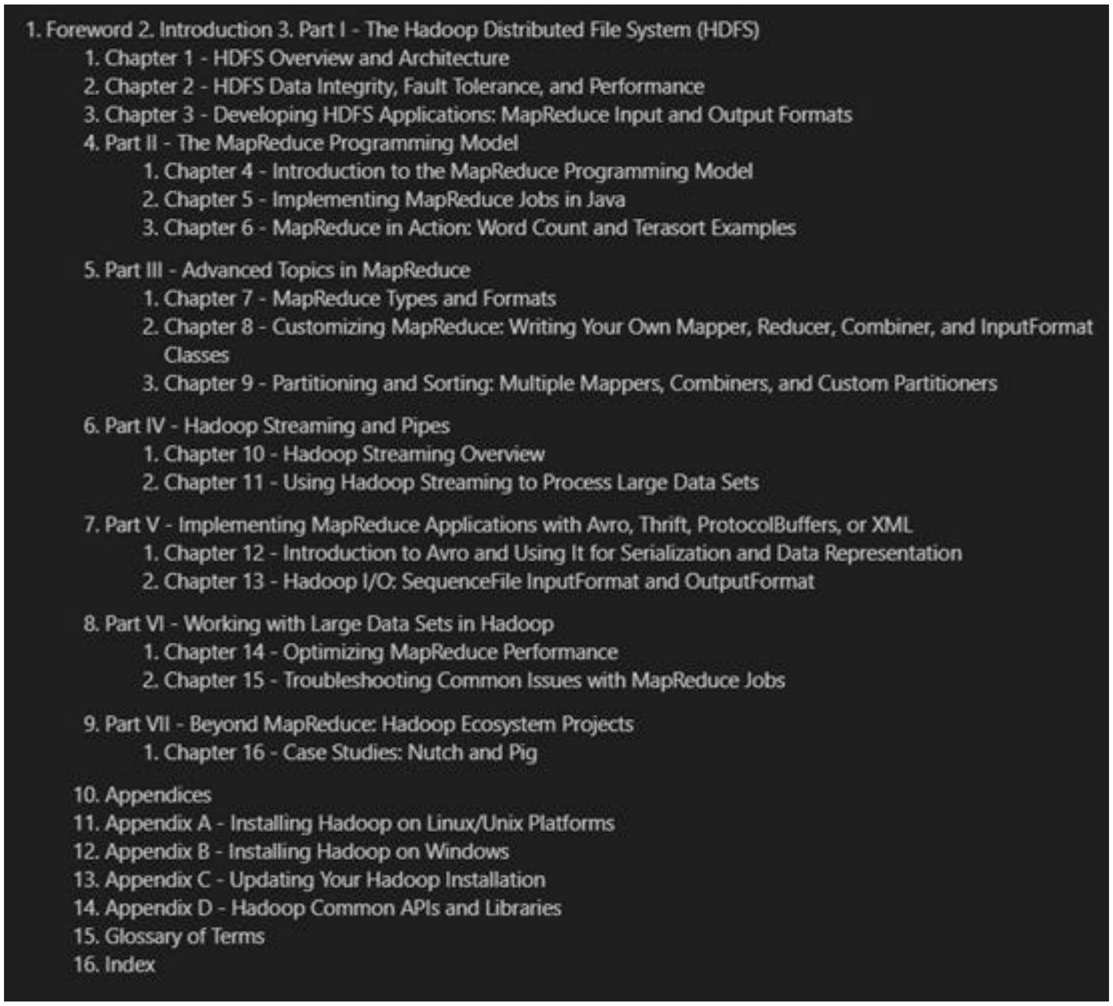
```
local_path = "/content/data/Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.pdf"
```

ii)User Query



```
chat_with_pdf("List out all the chapters in this pdf?")
```

OUTPUT:



```
1. Foreword 2. Introduction 3. Part I - The Hadoop Distributed File System (HDFS)
    1. Chapter 1 - HDFS Overview and Architecture
    2. Chapter 2 - HDFS Data Integrity, Fault Tolerance, and Performance
    3. Chapter 3 - Developing HDFS Applications: MapReduce Input and Output Formats
    4. Part II - The MapReduce Programming Model
        1. Chapter 4 - Introduction to the MapReduce Programming Model
        2. Chapter 5 - Implementing MapReduce Jobs in Java
        3. Chapter 6 - MapReduce in Action: Word Count and Terasort Examples

    5. Part III - Advanced Topics in MapReduce
        1. Chapter 7 - MapReduce Types and Formats
        2. Chapter 8 - Customizing MapReduce: Writing Your Own Mapper, Reducer, Combiner, and InputFormat
           Classes
        3. Chapter 9 - Partitioning and Sorting: Multiple Mappers, Combiners, and Custom Partitioners

    6. Part IV - Hadoop Streaming and Pipes
        1. Chapter 10 - Hadoop Streaming Overview
        2. Chapter 11 - Using Hadoop Streaming to Process Large Data Sets

    7. Part V - Implementing MapReduce Applications with Avro, Thrift, ProtocolBuffers, or XML
        1. Chapter 12 - Introduction to Avro and Using It for Serialization and Data Representation
        2. Chapter 13 - Hadoop I/O: SequenceFile InputFormat and OutputFormat

    8. Part VI - Working with Large Data Sets in Hadoop
        1. Chapter 14 - Optimizing MapReduce Performance
        2. Chapter 15 - Troubleshooting Common Issues with MapReduce Jobs

    9. Part VII - Beyond MapReduce: Hadoop Ecosystem Projects
        1. Chapter 16 - Case Studies: Nutch and Pig

   10. Appendices
   11. Appendix A - Installing Hadoop on Linux/Unix Platforms
   12. Appendix B - Installing Hadoop on Windows
   13. Appendix C - Updating Your Hadoop Installation
   14. Appendix D - Hadoop Common APIs and Libraries
   15. Glossary of Terms
   16. Index
```

7.2 Conclusion

The RAG Fusion system, designed using LangChain, Ollama embeddings, and FAISS-based vector retrieval, provides a highly effective solution for document-based question answering. By employing multi-query generation, reciprocal rank fusion, and an efficient retrieval chain, the system delivers accurate, contextually relevant answers from unstructured documents. This architecture ensures optimal performance in retrieving relevant information while offering scalability and adaptability to large datasets. The modular design, combined with the integration of Ollama models and vector databases, positions RAG Fusion as a powerful tool for diverse AI-driven applications, including chatbots, knowledge extraction, and document summarization. Its robustness and flexibility make it an invaluable asset for information retrieval tasks in complex real-world scenarios..

CHAPTER 8-FUTURE SCOPE

Future Scope:

1. **Enhanced Multimodal Integration:** Extend the current text-based system to support multimodal inputs such as images, videos, or tables. This is particularly useful for documents containing visual elements like graphs, charts, or illustrations.

2. **Real-Time Document Updates:** Implement mechanisms for real-time updates to the vector database, allowing the system to adapt dynamically to changes in the underlying data without requiring a full rebuild..

3. **Advanced Ranking Algorithms:** Explore ranking strategies beyond Reciprocal Rank Fusion, such as learning-to-rank models, which can prioritize results based on context relevance and user feedback..

4. **Support for Multiple Languages:** Enable multilingual support by integrating language-specific embeddings or translation layers, making the system applicable to a broader audience.

5. **Robustness to Noisy Data**: Improve preprocessing steps and embeddings to handle noisy, unstructured data, ensuring high performance even when documents are inconsistent or poorly formatted.

6. **Security and Privacy Enhancements:** Enhance privacy measures for sensitive data by integrating local inference capabilities, secure enclaves, or federated learning to ensure that the system complies with data protection regulations.

CHAPTER 9-REFERENCE

Reference:

1. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. 2023. Retrieval Augmented Generation for Large Language Models: A Survey.arXiv preprint arXiv:2312.10997 (2023)

2. Q. Ai, T. Bai, Z. Cao, Y. Chang, J. Chen, Z. Chen, Z. Cheng, S. Dong, Z. Dou, F. Feng, S. Gao, J. Guo, X. He, Y. Lan, C. Li, Y. Liu, Z. Lyu, W. Ma, J. Ma, and Z. Ren. 2023. Information Retrieval meets Large Language Models: A strategic report from Chinese IR community. FundamentalResearch4(2023),80

3. C. Jeong, "Implementation of generative AI service using LLM application architecture: based on RAG model and LangChain framework," J. Intell. Inform. Syst., vol. 29, no. 4, pp. 129-164, Dec. 2023..

4. S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara. 2022. Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. Electronics 11, 20 (2022), 3388

5. S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek, "Seven Failure Points When Engineering a Retrieval Augmented Generation System," in 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN), Geelong, Australia, 2024.

6. X. Zheng, F. Che, J. Wu, S. Zhang, S. Nie, K. Liu, and J. Tao, "KS-LLM: Knowledge Selection of Large Language Models with Evidence Document for Question Answering," arXiv preprint arXiv:2404.15660v1, 2024

7. M. Shanahan, "Talking about Large Language Models," Communications of the ACM, vol. 67, no. 2, pp. 68-79, 2024.

8. Shervin Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large Language Models: A Survey," arXiv preprint arXiv:2402.06196v2,Feb. 20, 2024