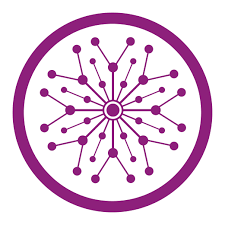
**

**The Superior University**

**Project Title**

Study Assistant RAG App

**Project Details**

1. Course: Artificial Intelligence
2. Instructor: Sir Rasikh Ali
3. Semester: 3rd
4. Section: BSAI-3B
5. Submission Date: (9/12/2024)
6. Group Members: 1

| **Name** | **Roll No** | **Email** | **Contact** |
| --- | --- | --- | --- |
| Sajid Ali | 078 | sajidali1@gmail | 0327848265 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Project Report

## Abstract

This project demonstrates the application of document processing, embedding creation, and information retrieval using state-of-the-art machine learning models. The project employs LangChain, SentenceTransformers, and DocArray to process a PDF document, create vector embeddings for its content, and retrieve relevant text using similarity search. By showcasing the integration of modern NLP techniques, this project highlights efficient information retrieval workflows suitable for educational or organizational purposes.

## Table of Contents

1. Introduction

2. Objectives

3. System Requirements

4. Methodology

5. Implementation

6. Challenges and Solutions

7. Conclusion

## Introduction

### Overview of the project

This project aims to process and analyze text from PDF documents. It uses advanced machine learning models to generate embeddings, enabling efficient storage and retrieval of contextually relevant information.

### Explanation of the selected topic

The project focuses on document embedding and retrieval to tackle the problem of searching through large amounts of text. By leveraging SentenceTransformer and LangChain, the project integrates state-of-the-art methodologies to solve real-world data analysis challenges.

### Relevance to Operating Systems concepts

While the project is more aligned with AI and NLP, it indirectly involves process and memory management through embedding models, reflecting concepts like resource allocation and data segmentation in operating systems.

## Objectives

· To implement a document processing pipeline that extracts and embeds text from PDFs.

· To understand and demonstrate the use of embedding models for information retrieval.

· To enable similarity-based searches to retrieve contextually relevant data.

· To explore integration techniques for modern machine learning models within practical applications.

## System Requirements

### Hardware Requirements

- Processor: Intel Core i5 or equivalent

- RAM: 8GB or more

- Storage: 256GB or more

### Software Requirements

- Programming Language: Python 3.10

- Libraries: LangChain,Ollama, llama3.2, DocArray, PyPDFLoader,

- Environment: Google Colab

## Methodology

1. Document Processing:  
 - Use PyPDFLoader to extract text and images from a given PDF file.  
 - Split the document into manageable chunks using the RecursiveCharacterTextSplitter.  
  
 2. Embedding Generation:  
 - Employ the SentenceTransformer model (all-MiniLM-L6-v2) to create embeddings for the document chunks.  
  
 3. Vector Store Creation:  
 - Use DocArrayInMemorySearch to store document embeddings for similarity-based searches.  
  
 4. Information Retrieval:  
 - Perform cosine similarity calculations between query embeddings and document embeddings to retrieve the most relevant text chunks.

## Implementation

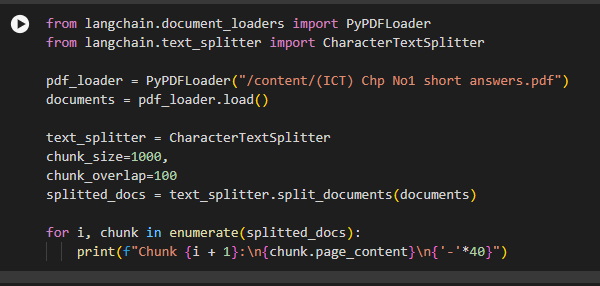
The project follows these key steps:  
 1. Extract text from the PDF using PyPDFLoader.  
 2. Split the text into smaller chunks.  
 3. Create embeddings for the text chunks using the SentenceTransformer model.  
 4. Store embeddings in a DocArrayInMemorySearch vector store.  
 5. Retrieve information using a similarity search.

### Key Code Snippets

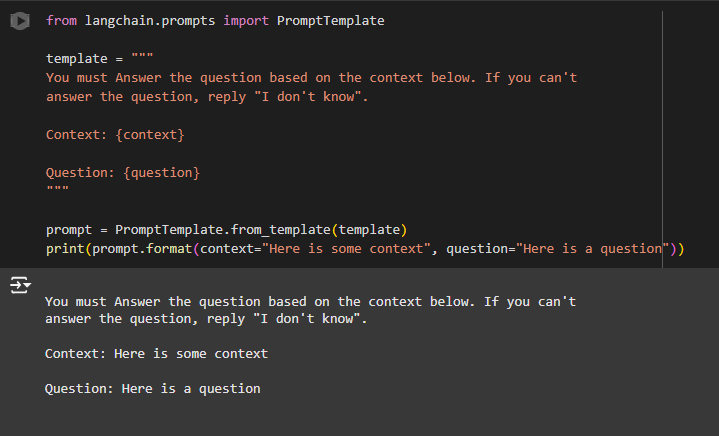
Importing LLm(llama3.2):

## 

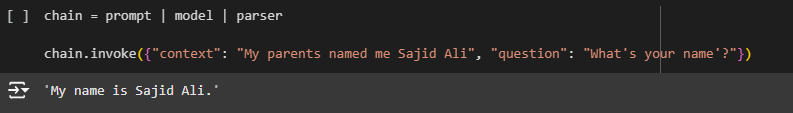
2. Loading and text splitting:



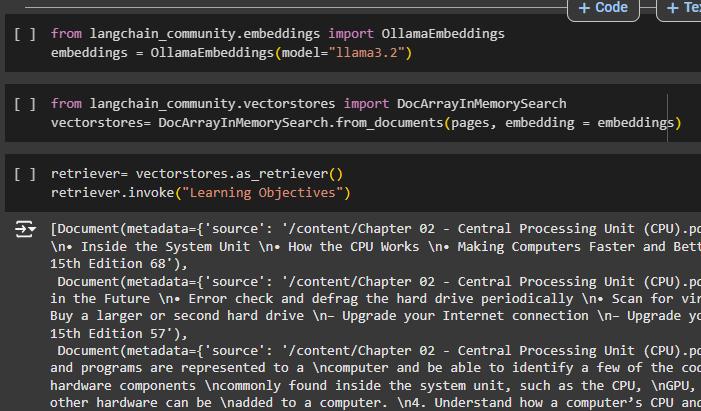
Creating a system prompt:



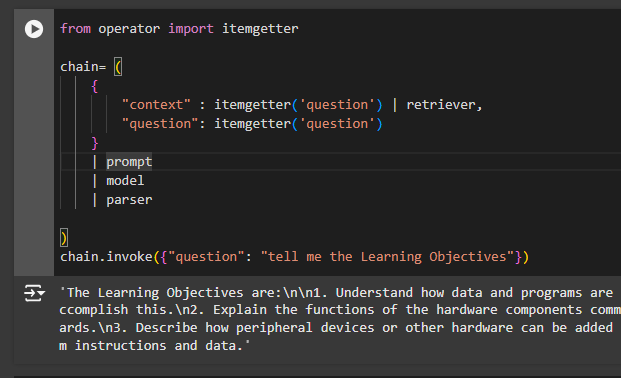
creating chain:



Creating Embeddings, Vector Stores and Retriever:



At the end Completing the chain and invoking



## Challenges and Solutions

### Challenges

1. Embedding Model Integration: Integrating a custom embedding model required thorough understanding and proper adjustments.

2. High Computational Resource Requirements: Processing embeddings for large documents posed challenges in resource-constrained environments.

3. Dependency Conflicts: Installing and managing dependencies (e.g., DocArray, LangChain) introduced compatibility issues.

### Solutions

1. Used pre-trained models like all-MiniLM-L6-v2 for efficient embedding creation.

2. Leveraged Google Colab’s GPU runtime for processing resource-intensive tasks.

3. Resolved dependency conflicts by explicitly managing versions of libraries (e.g., pydantic, docarray).

## Conclusion

This project successfully demonstrates the use of modern NLP techniques for document processing and information retrieval. By utilizing LangChain, SentenceTransformers, and DocArray, the pipeline efficiently retrieves relevant information from large texts. The implementation is scalable and can be extended to other domains such as education, legal analysis, or enterprise document search.

### Future Enhancements

1. Use advanced LLMs like GPT-4 for more context-aware queries.

2. Implement additional storage options like FAISS for large-scale vector storage.

3. Integrate OCR for non-text PDFs to process scanned documents.