

DeepLogic AI Intern Assignment

Development of a Retrieval Augmented Generation (RAG) Based Search Tool

Ayushi

Pre-final year undergraduate
Indian Institute of Technology, Kharagpur

February 28, 2024

1 Introduction

The aim of this project is to develop a Retrieval-Augmented Generation (RAG) model capable of comprehending and generating responses based on content extracted from a collection of PDF documents. The model leverages advanced natural language processing (NLP) techniques, including embeddings and vector search, to facilitate question-answering tasks.

2 Libraries Used

cassio - Cassio is employed for interacting with the database. It provides functionalities to store and retrieve vectors efficiently.

Langchain - Langchain is a framework that offers tools for working with natural language data. It provides modules for vectorization, indexing, and language model interactions.

Open AI - The OpenAI API is integrated to access language model capabilities and generate embeddings for text.

tiktoken - Tiktoken is designed to be fast, efficient, and easy to use when it comes to tokenizing text and managing tokenized data. It's particularly useful for scenarios where we need to count tokens without allocating memory for the actual token strings.

PyPDF2 - This library is utilized for parsing PDF documents and extracting text content.

3 RAG Architecture

Retrieval-Augmented Generation, or RAG, represents a cutting-edge approach to artificial intelligence (AI) and natural language processing (NLP). At its core, RAG is an innovative framework that combines the strengths of retrieval-based and generative models, revolutionizing how AI systems understand and generate human-like text. The development of RAG is a direct response to the limitations of Large Language Models (LLMs) like GPT. While LLMs have shown impressive text generation capabilities, they often struggle to provide contextually relevant responses, hindering their utility in practical applications. RAG aims to bridge this gap by offering a solution that excels in understanding user intent and delivering meaningful and context-aware replies.

The RAG architecture combines retrieval and generation approaches to enhance question-answering capabilities. In the context of this project:

Retrieval : The Cassandra database serves as the retrieval component, storing precomputed embeddings of PDF document content. These embeddings are indexed for efficient retrieval based on query vectors.

Generation : OpenAI's language model, integrated through Langchain, facilitates response generation based on the retrieved document embeddings. The model utilizes the semantic information captured in the embeddings to produce coherent and relevant answers to user queries.

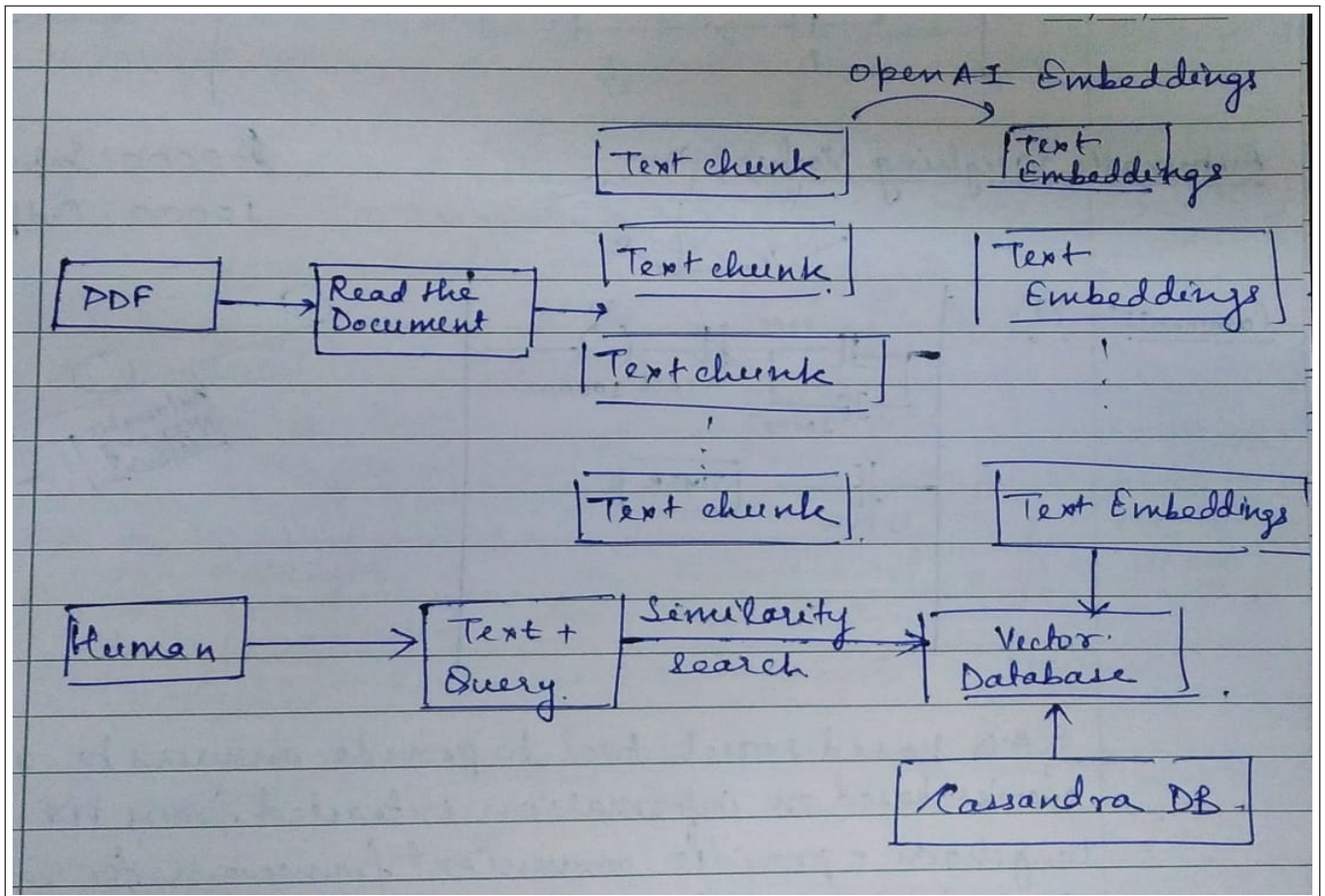


Figure 1: Flowchart illustrating the RAG Architecture

4 Database Connection

The project utilizes Cassandra as the database system, with specific parameters such as application tokens and database identifiers configured for access. Cassio is used to initialize the connection to the Cassandra database, enabling seamless interaction for storing and retrieving text vectors.

5 Vector Search for Question Answering

Vector search is employed as the primary mechanism for retrieving answers to user-input questions. This involves the following steps:

Text Extraction : PDF documents are parsed using PyPDF2 to extract raw text content.

Embedding Generation: OpenAI API is employed to generate embeddings for the extracted text. These embeddings represent the semantic meaning of the text content.

Vector Storage : The generated embeddings are stored efficiently in the Cassandra database using Cassio.

Indexing : Langchain's VectorStoreIndexWrapper is utilized to create an index for the vector store, enabling fast and accurate retrieval of relevant documents based on query vectors.

Question Answering : User prompts are accepted as input queries. The RAG model utilizes the vector index to search for the most relevant documents and generate responses based on their content.

6 Langchain Framework for Question Answering

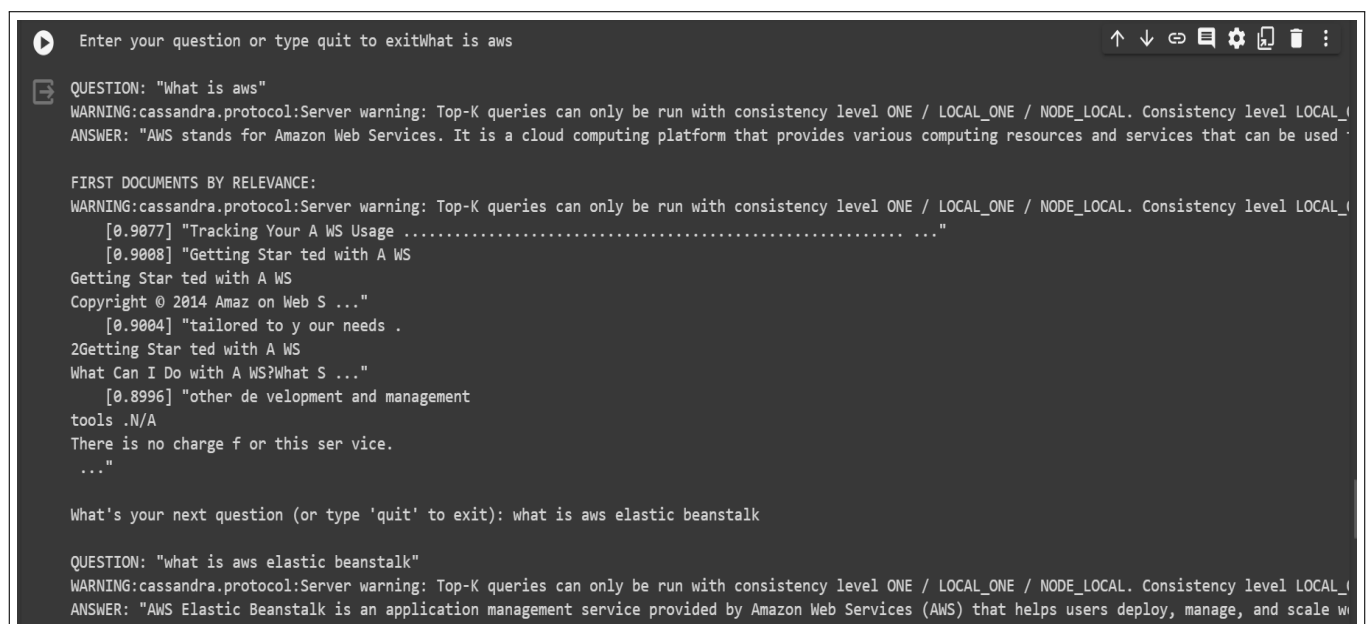
Langchain plays a crucial role in enabling question-answering functionalities for PDF documents. It provides the following functionalities:

Embeddings : Langchain leverages the OpenAI API to generate embeddings for text data, capturing semantic representations.

Vector Storage and Indexing : Through integration with Cassandra and VectorStoreIndexWrapper, Langchain facilitates efficient storage and indexing of text embeddings, enabling fast retrieval of relevant documents.

Querying and Answer Generation : Langchain's vector search capabilities allow for querying the indexed documents based on input questions and generating answers based on the retrieved content.

7 Final Model



```
Enter your question or type quit to exitWhat is aws

QUESTION: "What is aws"
WARNING:cassandra.protocol:Server warning: Top-K queries can only be run with consistency level ONE / LOCAL_ONE / NODE_LOCAL. Consistency level LOCAL_I
ANSWER: "AWS stands for Amazon Web Services. It is a cloud computing platform that provides various computing resources and services that can be used .

FIRST DOCUMENTS BY RELEVANCE:
WARNING:cassandra.protocol:Server warning: Top-K queries can only be run with consistency level ONE / LOCAL_ONE / NODE_LOCAL. Consistency level LOCAL_I
[0.9077] "Tracking Your A WS Usage .....
[0.9008] "Getting Star ted with A WS
Getting Star ted with A WS
Copyright © 2014 Amaz on Web S ..."
[0.9004] "tailored to y our needs .
2Getting Star ted with A WS
What Can I Do with A WS?What S ..."
[0.8996] "other de velopment and management
tools .N/A
There is no charge f or this ser vice.
..."

What's your next question (or type 'quit' to exit): what is aws elastic beanstalk

QUESTION: "what is aws elastic beanstalk"
WARNING:cassandra.protocol:Server warning: Top-K queries can only be run with consistency level ONE / LOCAL_ONE / NODE_LOCAL. Consistency level LOCAL_I
ANSWER: "AWS Elastic Beanstalk is an application management service provided by Amazon Web Services (AWS) that helps users deploy, manage, and scale w
```

Figure 2: Screenshot of the final working model