Retrieval-Augmented Generation With Langchain Report

1. Introduction

Retrieval-Augmented Generation (RAG) is an approach that combines retrieval-based methods with generative models. It involves retrieving relevant documents or knowledge and then generating responses or content based on that retrieved information. This approach is particularly useful for tasks like question answering, summarization, and more.

2. LangChain Overview

LangChain is a library that facilitates the integration of various components in a natural language processing (NLP) pipeline, including retrieval systems, language models, and data handling. It helps in building applications that require complex workflows, such as RAG systems.

3. Components of RAG with LangChain and OpenAI

A. Document Retrieval System

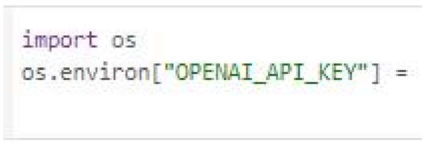
1. **Indexing and Search**: Use an indexing system (e.g., FAISS) to index your documents or knowledge base. This allows efficient retrieval of relevant documents based on queries.
2. **LangChain Integration**: Integrate the retrieval system with LangChain, allowing for easy querying and management of the documents.

B. Generative Model

1. **OpenAI API**: Use OpenAI's GPT-3 or GPT-4 model as the generative component. You can access this through the OpenAI API using your API key.
2. **Prompt Engineering**: Craft prompts that effectively utilize the retrieved information to generate accurate and contextually relevant responses.

C. Workflow

1. **Query Handling**: When a query is received, use LangChain to first retrieve relevant documents or information.
2. **Contextual Generation**: Feed the retrieved information into the generative model via the OpenAI API. Use the information to guide the model's output, ensuring that responses are accurate and relevant.



```
import os
os.environ["OPENAI_API_KEY"] =
```

3. **Response Generation**: The system outputs a response that combines the retrieved information with the generative model's capabilities.

4. Implementation Details

A. Setting Up LangChain and OpenAI API

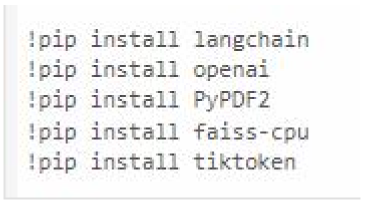
1. **LangChain Installation**: Install LangChain and any necessary dependencies.
    - pip install langchain
2. **OpenAI API Key**: Set up your OpenAI API key in your environment or code to access the OpenAI models.

B. Building the RAG Pipeline

1. **Document Retrieval**:
   - Index your documents using a retrieval system.
   - Use LangChain to integrate and query this system.
2. **Generative Response**:
   - Use the retrieved documents to craft prompts.
   - Send these prompts to the OpenAI API and get the generated response.
3. **Handling Responses**:
   - Process and format the response as needed for your application.

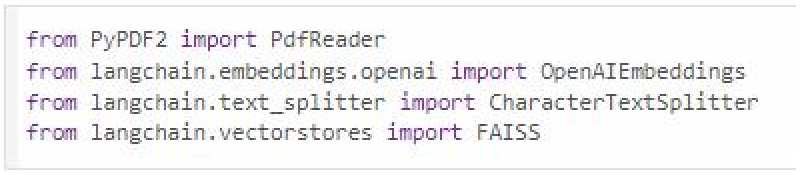
5. Challenges and Considerations

- **Scalability**: Ensure that the retrieval system can handle large volumes of documents and queries efficiently.
- **Prompt Design**: Crafting effective prompts is crucial for getting high-quality responses from the generative model.
- **Cost Management**: Monitor API usage and manage costs, especially if using a paid model like OpenAI's.



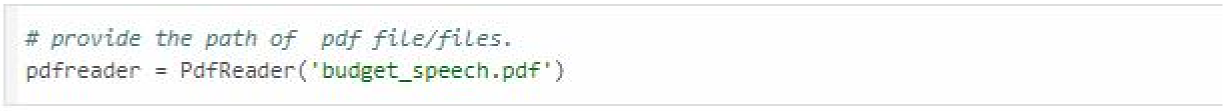
```
!pip install langchain
!pip install openai
!pip install PyPDF2
!pip install faiss-cpu
!pip install tiktoken
```

Import Required Packages,

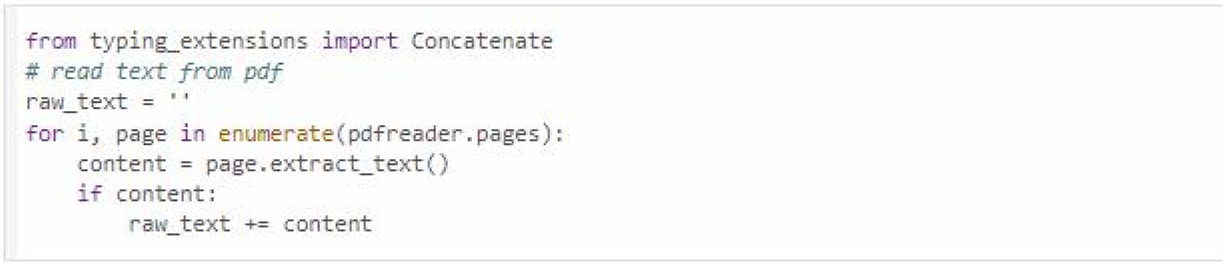


```
from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
```

To insert the document loader,

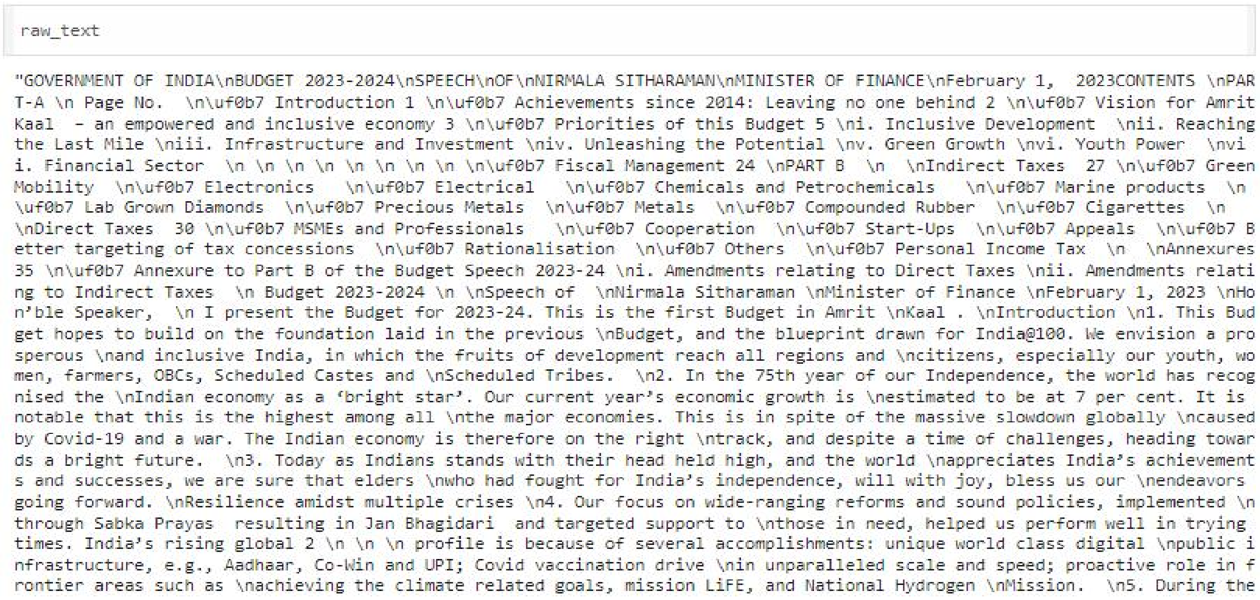


```
# provide the path of  pdf file/files.
pdfreader = PdfReader('budget_speech.pdf')
```



```
from typing_extensions import Concatenate
# read text from pdf
raw_text = ''
for i, page in enumerate(pdfreader.pages):
    content = page.extract_text()
    if content:
        raw_text += content
```

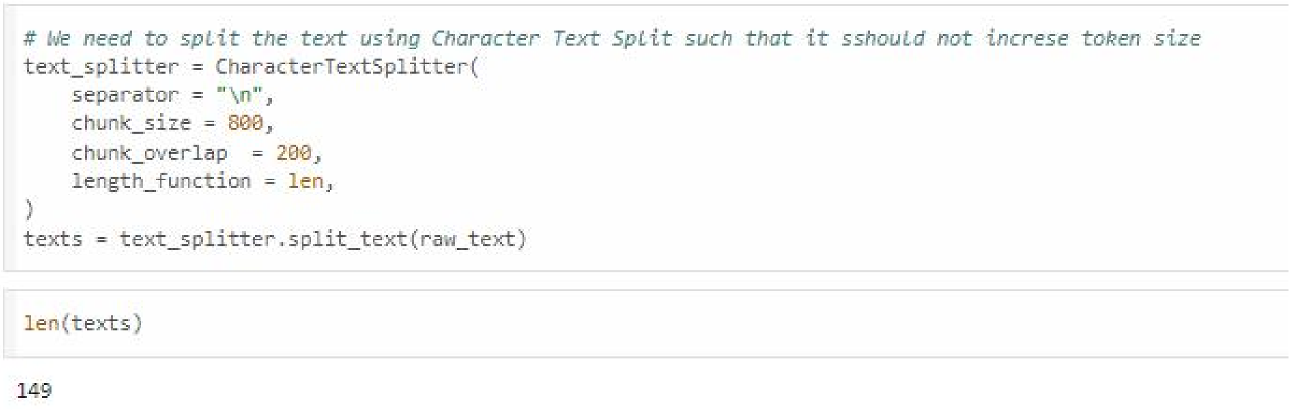
I got raw text from the document,



```
raw_text
```

```
"GOVERNMENT OF INDIA\nBUDGET 2023-2024\nSPEECH\nOF\nNIRMALA SITHARAMAN\nMINISTER OF FINANCE\nFebruary 1,  2023CONTENTS \nPAR
T-A \n Page No.  \n\uf0b7 Introduction 1 \n\uf0b7 Achievements since 2014: Leaving no one behind 2 \n\uf0b7 Vision for Amrit
Kaal  - an empowered and inclusive economy 3 \n\uf0b7 Priorities of this Budget 5 \ni. Inclusive Development  \nii. Reaching
the Last Mile \niii. Infrastructure and Investment \niv. Unleashing the Potential \nv. Green Growth \nvi. Youth Power  \nvi
i. Financial Sector  \n \n \n \n \n \n \n \n \n\uf0b7 Fiscal Management 24 \nPART B  \n  \nIndirect Taxes  27 \n\uf0b7 Green
Mobility  \n\uf0b7 Electronics   \n\uf0b7 Electrical   \n\uf0b7 Chemicals and Petrochemicals   \n\uf0b7 Marine products  \n
\uf0b7 Lab Grown Diamonds  \n\uf0b7 Precious Metals  \n\uf0b7 Metals  \n\uf0b7 Compounded Rubber  \n\uf0b7 Cigarettes  \n
\nDirect Taxes  30 \n\uf0b7 MSMEs and Professionals   \n\uf0b7 Cooperation  \n\uf0b7 Start-Ups  \n\uf0b7 Appeals  \n\uf0b7 B
etter targeting of tax concessions  \n\uf0b7 Rationalisation  \n\uf0b7 Others  \n\uf0b7 Personal Income Tax  \n  \nAnnexures
35 \n\uf0b7 Annexure to Part B of the Budget Speech 2023-24 \ni. Amendments relating to Direct Taxes \nii. Amendments relati
ng to Indirect Taxes  \n Budget 2023-2024 \n \nSpeech of  \nNirmala Sitharaman \nMinister of Finance \nFebruary 1, 2023 \nHo
n’ble Speaker,  \n I present the Budget for 2023-24. This is the first Budget in Amrit \nKaal . \nIntroduction \n1. This Bud
get hopes to build on the foundation laid in the previous \nBudget, and the blueprint drawn for India@100. We envision a pro
sperous \nand inclusive India, in which the fruits of development reach all regions and \ncitizens, especially our youth, wo
men, farmers, OBCs, Scheduled Castes and \nScheduled Tribes.  \n2. In the 75th year of our Independence, the world has recog
nised the \nIndian economy as a ‘bright star’. Our current year’s economic growth is \nestimated to be at 7 per cent. It is
notable that this is the highest among all \nthe major economies. This is in spite of the massive slowdown globally \ncaused
by Covid-19 and a war. The Indian economy is therefore on the right \ntrack, and despite a time of challenges, heading towar
ds a bright future.  \n3. Today as Indians stands with their head held high, and the world \nappreciates India’s achievement
s and successes, we are sure that elders \nwho had fought for India’s independence, will with joy, bless us our \nendeavors
going forward. \nResilience amidst multiple crises \n4. Our focus on wide-ranging reforms and sound policies, implemented \n
through Sabka Prayas  resulting in Jan Bhagidari  and targeted support to \nthose in need, helped us perform well in trying
times. India’s rising global 2 \n \n \n profile is because of several accomplishments: unique world class digital \npublic i
nfrastructure, e.g., Aadhaar, Co-Win and UPI; Covid vaccination drive \nin unparalleled scale and speed; proactive role in f
rontier areas such as \nachieving the climate related goals, mission LiFE, and National Hydrogen \nMission.  \n5. During the
```

We need to split the text using character text split , and find length of characters,

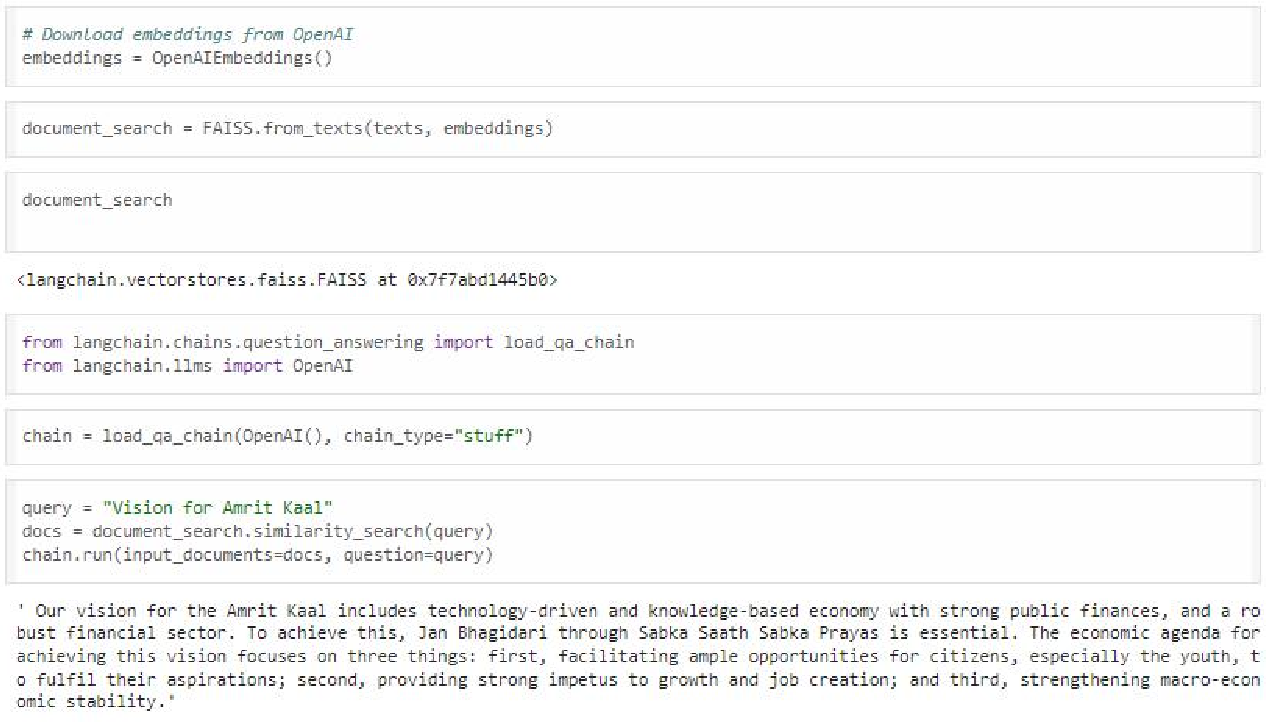


```python
# We need to split the text using Character Text Split such that it sshould not increse token size
text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 800,
    chunk_overlap  = 200,
    length_function = len,
)
texts = text_splitter.split_text(raw_text)
```

```python
len(texts)
```

```
149
```

To create query like a question, to get the answers from the document.



```
# Download embeddings from OpenAI
embeddings = OpenAIEmbeddings()
```

```
document_search = FAISS.from_texts(texts, embeddings)
```

```
document_search
```

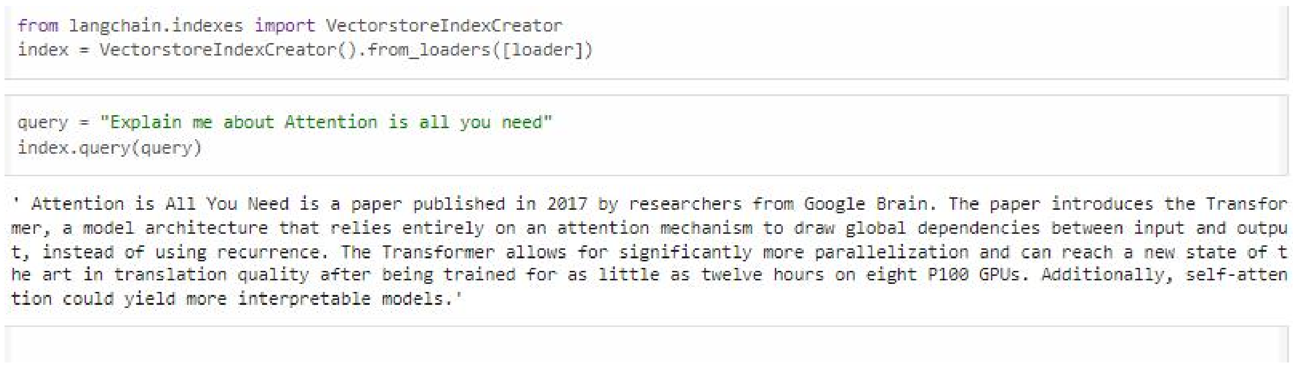
```
<langchain.vectorstores.faiss.FAISS at 0x7f7abd1445b0>
```

```
from langchain.chains.question_answering import load_qa_chain
from langchain.llms import OpenAI
```

```
chain = load_qa_chain(OpenAI(), chain_type="stuff")
```

```
query = "Vision for Amrit Kaal"
docs = document_search.similarity_search(query)
chain.run(input_documents=docs, question=query)
```

```
' Our vision for the Amrit Kaal includes technology-driven and knowledge-based economy with strong public finances, and a ro
bust financial sector. To achieve this, Jan Bhagidari through Sabka Saath Sabka Prayas is essential. The economic agenda for
achieving this vision focuses on three things: first, facilitating ample opportunities for citizens, especially the youth, t
o fulfil their aspirations; second, providing strong impetus to growth and job creation; and third, strengthening macro-econ
omic stability.'
```



```
from langchain.indexes import VectorstoreIndexCreator
index = VectorstoreIndexCreator().from_loaders([loader])
```

```
query = "Explain me about Attention is all you need"
index.query(query)
```

```
' Attention is All You Need is a paper published in 2017 by researchers from Google Brain. The paper introduces the Transfor
mer, a model architecture that relies entirely on an attention mechanism to draw global dependencies between input and outpu
t, instead of using recurrence. The Transformer allows for significantly more parallelization and can reach a new state of t
he art in translation quality after being trained for as little as twelve hours on eight P100 GPUs. Additionally, self-atten
tion could yield more interpretable models.'
```

6. Conclusion

By combining LangChain and OpenAI's generative models, you can create a powerful RAG system that enhances the capabilities of both retrieval and generation. This approach can be applied to various applications, from customer support to content creation.