Project Report: Company Quarter Analysis using Retrieval-Augmented Generation (RAG)

Name : PIYUSH SUHAG

Data Science Trainee - Incedo

1. Introduction:

This project focuses on analyzing Zomato's quarterly report using advanced Natural Language Processing (NLP) techniques and building an intelligent chatbot to provide meaningful insights from the report. By leveraging Retrieval-Augmented Generation (RAG), we enable the chatbot to provide detailed responses based on the relevant sections of the report. The chatbot is designed to answer user queries regarding the company's financial performance, operations, and more by analyzing the extracted text from the quarterly report.

2. Goals of the Project:

- **Text Extraction:** Extract valuable data from Zomato's quarterly report (PDF format) for further analysis.
- **Data Chunking and Embedding:** Split the extracted data into chunks and create embeddings for efficient retrieval using Pinecone.
- **Pinecone Integration:** Store the embeddings in Pinecone for similarity search and retrieval.
- **Mistral Model for Generation:** Use the Mistral model to generate responses based on the relevant chunks retrieved from Pinecone.
- **Gradio UI:** Implement a simple user interface for real-time querying of the extracted data.

3. Technologies Used:

- **Python Libraries:**
  - `PyPDF2`, `pdfminer.six`: For basic PDF text extraction.
  - `tabula-py`: For table extraction from PDFs.
  - `langchain`: Used to split large text into smaller, manageable chunks.
  - `sentence-transformers`: To generate embeddings for the extracted chunks.
  - `pinecone`: To index and query the text chunks based on their embeddings.
  - `mistralai`: To generate text responses using the Mistral large language model.
  - `gradio`: For creating a user-friendly interface.
- **External Tools:**
  - **Google Colab:** Used for environment setup and running the Python code.
  - **Pinecone API:** For vector database management.
  - **Mistral API:** For generating answers from retrieved data using the fine-tuned model.

4. Process Flow:

4.1 Step 1: Text Extraction from PDF

The Zomato quarterly report is in PDF format. First, text extraction is performed using the `PyMuPDF` library (also known as `fitz`), which allows for the extraction of plain text from each page of the document.

```
python
Copy code
def extract_text_fitz(file_path):
    try:
        doc = fitz.open(file_path)
        text = ""
        for page_num in range(len(doc)):
            page = doc.load_page(page_num)
            text += page.get_text("text")  # Extract plain
text
        return text
    except Exception as e:
        return f"Error: {e}"
```

4.2 Step 2: Table Extraction

Tables from the PDF are extracted using the `tabula-py` library, which reads the tables from the report and converts them into a DataFrame format. This allows for structured data extraction.

```
python
Copy code
all_tables = tabula.read_pdf(pdf_file_path, pages='all',
multiple_tables=True)
```

4.3 Step 3: Text Chunking

The extracted text is split into smaller chunks using the `RecursiveCharacterTextSplitter` class from `langchain`. This is done to ensure that the text is divided into manageable pieces for the model to process efficiently.

```
python
Copy code
text_splitter =
RecursiveCharacterTextSplitter(chunk_size=500,
chunk_overlap=50)
chunks = text_splitter.split_text(parsed_text_fitz)
```

4.4 Step 4: Embedding Generation

Using the `SentenceTransformer` model, each chunk of text is converted into an embedding that captures the semantic meaning of the content. These embeddings are stored in Pinecone for future retrieval.

```
python
Copy code
model = SentenceTransformer('all-MiniLM-L6-v2')
embeddings = generate_embeddings(chunks)
```

4.5 Step 5: Storing Data in Pinecone

The generated embeddings, along with their respective chunks of text, are upserted into the Pinecone index. This allows for fast and efficient similarity searches.

```
python
Copy code
vectors_to_upsert = []
for i, embedding in enumerate(embeddings):
    vector = {
        "id": f"chunk_{i}",
        "values": embedding.tolist(),
        "metadata": {"chunk_text": chunks[i]}
    }
    vectors_to_upsert.append(vector)

index.upsert(vectors=vectors_to_upsert)
```

4.6 Step 6: Query Handling and Response Generation

When a user submits a query, it is converted into an embedding using the `SentenceTransformer` model. This embedding is then used to query the Pinecone index for the most similar chunks. The relevant chunks are sent to the Mistral model to generate a coherent response.

```
python
Copy code
def generate_response(query):
    query_embedding = model.encode([query])[0]
    results = index.query(vector=query_embedding.tolist(),
top_k=3, include_values=True, include_metadata=True)
    relevant_chunks = [result['metadata']['chunk_text'] for
result in results['matches']]
    messages = [{"role": "user", "content": query}]
    for chunk in relevant_chunks:
        messages.append({"role": "assistant", "content":
chunk})
    messages.append({"role": "user", "content": query})
    chat_response = client.chat.complete(model=mistral_model,
messages=messages)
    return chat_response.choices[0].message.content
```

4.7 Step 7: User Interface

A Gradio interface is implemented to allow users to interact with the model. The interface allows users to input a query and receive a response generated by the model.

```
python
Copy code
iface = gr.Interface(fn=generate_response,
inputs=gr.Textbox(label="Enter your query"),
outputs=gr.Textbox(label="Response"))
iface.launch()
```

5. Future Implementation:

- **Multilingual Support:** Implementing multilingual capabilities to analyze reports in different languages, expanding the accessibility of the chatbot.
- **Fine-Tuning the Model:** Further fine-tune the Mistral model to improve the accuracy and relevance of answers, especially in specific domains like finance or business analysis.
- **Integration with Other Data Sources:** Integrating the system with other reports or live data feeds (e.g., stock data, operational metrics) for real-time insights.
- **Improved UI Features:** Enhance the user interface by adding more advanced query options and a more detailed response breakdown.
- **Optimization:** Optimize Pinecone indexing and retrieval processes to handle large datasets with better performance, especially for larger reports.

6. Challenges Faced:

- **PDF Extraction:** Handling complex PDF layouts (e.g., multiple columns or embedded images) posed challenges for accurate text extraction. This was mitigated by using multiple extraction techniques (PyMuPDF and Tabula).
- **Chunking and Context Preservation:** Maintaining the context between chunks during text splitting and ensuring that the query could still be answered coherently was a challenge, which was addressed by using overlapping chunks.
- **Model Fine-Tuning:** Generating contextually accurate responses using a pre-trained model required experimentation with various configurations to enhance the output's relevance.

7. Conclusion:

This project successfully integrates advanced NLP techniques with modern vector search capabilities to build a powerful chatbot capable of analyzing and extracting insights from Zomato's quarterly reports. By using Pinecone for vector search and Mistral for text generation, the system is able to provide accurate and contextually relevant answers to user queries. With future improvements, this solution can be expanded to handle a wider range of use cases and data sources.