Student Assignment: Building a Document Retrieval and Question-Answering System with LangChain

1 Objective

The goal of this assignment is to build a document retrieval and question-answering system using LangChain's capabilities in the Retrieval Augmented Generation (RAG) framework. You will be working with a specific use case to demonstrate the system's capabilities.

2 Use Case

2.1 Document Type: Business Strategy Articles

You will be working with a collection of articles related to business strategy. These articles will cover topics like Business Ecosystems, Core Strategy Tasks, Digital Platforms, and more.

2.2 Test Questions

1. What is a Business Ecosystem?
2. What are the core tasks of a Business Ecosystem Strategy?
3. Do Business Ecosystems always include digital platforms?
4. Why are Business Ecosystems attractive for companies today?
5. How do Business Ecosystems require a new mindset compared to traditional business strategies?

3 Prerequisites

- Basic understanding of Python programming
- Familiarity with Google Colab (or Jupyter Notebooks)
- Basic understanding of Natural Language Processing (NLP)

4 Tools Required

- Google Colab or Jupyter Notebook
- Python Libraries: `python-dotenv`, `openai`, `langchain`, `pypdf`, `tiktoken`, `chromadb`, `lark`, `docarray`, `unstructured`

5 Steps

5.1 Step 1: Environment Setup

1. Google Colab Users: Mount your Google Drive.
2. API Key Configuration: Set up your OpenAI API key. Make sure to handle exceptions and errors effectively.

5.2 Step 2: Document Loading

1. Directory Setup: Set up the directory where your business strategy articles are stored.
2. Document Loader: Use LangChain's `DirectoryLoader` to load documents from the directory.

5.3 Step 3: Document Splitting

1. Text Splitter: Use LangChain's `RecursiveCharacterTextSplitter` to split the documents into smaller chunks.
2. Chunk Size: Decide on the size of the chunks and the overlap between them.

5.4 Step 4: Vector Stores and Embeddings

1. Embedding: Use OpenAI for creating embeddings.
2. Vector Store: Use Chroma to store these embeddings.

5.5 Step 5: Retrieval

1. Retrieval Methods: Implement retrieval methods like Maximum Marginal Relevance (MMR), Metadata Filtering, etc.

5.6 Step 6: Question Answering

1. Language Model: Use LangChain's `ChatOpenAI` for the language model.
2. Prompt Template: Create a prompt template for the questions.
3. RetrievalQA Chain: Use LangChain's `RetrievalQA` to combine the language model with the vector database for answering questions.

5.7 Step 7: Testing

1. Questions: Test your system by asking it the test questions mentioned in the Use Case section and evaluate its performance.

6 Deliverables

- A Google Colab or Jupyter Notebook containing the implemented code.
- A report explaining your code, the use case, and any challenges you faced.
- Test results showing the system's performance in the context of the use case.

7 Evaluation Criteria

- Code Quality: 40%
- Functionality: 30%
- Use Case Understanding: 10%
- Report: 10%
- Test Results: 10%

Good luck! Feel free to reach out if you have any questions.