Assignment: Develop a RAG Application

Objective:
Design and implement a RAG application that ingests content from 5 PDF documents, creates a vector database for semantic retrieval, and powers a conversational bot with memory for the last 4 interactions. The application should answer a series of 10 predefined questions and be evaluated using a framework such as RAGAS, Trulens, or a self-devised metric. A final report (in PDF format) detailing your approach, implementation, and evaluation results is required.

Task Breakdown

1. PDF Ingestion & Data Sourcing (20 Marks)

- **PDF Sources:**
  Use these PDFs as your data sources for ingestion and embedding in your RAG application. They are readily accessible, and provide diverse content for evaluating your system's retrieval and conversational abilities -

  a) https://arxiv.org/pdf/1706.03762.pdf
  b) https://arxiv.org/pdf/1810.04805.pdf
  c) https://arxiv.org/pdf/2005.14165.pdf
  d) https://arxiv.org/pdf/1907.11692.pdf
  e) https://arxiv.org/pdf/1910.10683.pdf

- **Data Extraction** (10 Marks)**:**
  Implement a pipeline to extract text from each PDF. Consider handling layout, tables, or figures if needed, but focus primarily on ensuring that the text is correctly segmented for embedding.

- **Preprocessing & Chunking** (10 Marks)**:**
  Effective preprocessing of text into meaningful chunks for embedding.

2. Vector Database Creation (20 Marks)

- **Requirement:**
  Create a vector database to store and retrieve text embeddings from your extracted PDF content.

- **Suggested Tools:**
  You may use an open-source vector store such as FAISS, or any alternative that meets the assignment requirements.
- **Implementation:**
  - Process the text from the PDFs into meaningful chunks. (5 Marks)
  - Generate embeddings for these chunks using your chosen model. (5 Marks)
  - Proper setup and configuration for vector DB to store these for efficient similarity search. (10 Marks)

3. Open Source Language Model Integration (20 Marks)

- **Model Requirement:**
  Integrate an open-source language model from Hugging Face or Ollama to generate responses based on retrieved context.
- **Usage:**
  - The model should be used to generate answers by combining retrieved text chunks with the user's query.
  - Ensure that your model and embedding generator are compatible.

4. Conversational Bot with Memory (10 Marks)

- **Conversational Memory:**
  The bot should maintain context over the last 4 conversation turns. This means it should "remember" the past interactions and use them to inform its responses.
- **Implementation Hints:**
  - Use techniques such as conversation state management or prompt concatenation.
  - Ensure the memory is updated and only the last 4 interactions are considered.

5. Interaction & Evaluation (20 Marks)

- **Testing with Questions:**
  Create a test suite that interacts with your bot by asking 10 different questions. These questions should cover various aspects of the content in the PDFs and test the bot's ability to recall context.

- **Evaluation Framework:**
    - Evaluate the quality and relevance of the responses using an established framework such as RAGAS or Trulens.
    - Alternatively, you may design your own evaluation metric, but ensure you clearly document the criteria and methodology.
- **Evaluation Aspects to Consider:**
    - Relevance: How well does the answer address the question?
    - Accuracy: Is the information correct based on the source PDFs?
    - Contextual Awareness: Does the bot effectively leverage conversational memory?
    - Response Quality: Is the language clear and informative?

6. Final Report (10 Marks)

- **Format:**
  Submit a final report as a PDF document.
- **Contents:**
  The report should include:
    - Overview: An introduction to your approach and system architecture.
    - Technical Implementation: Details on how you extracted data, generated embeddings, set up your vector database, integrated the language model, and implemented conversational memory.
    - Evaluation: A detailed description of the 10 test questions and the evaluation framework (including any metrics, scoring methods, or qualitative assessments used).
    - Results & Discussion: Analysis of the performance of your RAG application, challenges faced, and potential improvements.
    - Conclusion: A summary of your findings and overall performance.

7. **Bonus:**
   Design and implement an extension to your RAG application that incorporates real-time user feedback into the conversational memory. The goal is to dynamically update and

refine the context used for generating responses based on feedback provided during interactions.

Submission Guidelines

- **Code:**
  Provide all source code used in your implementation, with appropriate comments and documentation.
- **Final Report:**
  Submit your final report in PDF format.
- **Additional Files:**
  Include any configuration files, environment setup instructions, or other materials necessary to run your application.

Resources & References

- **PDF Extraction Libraries:**
  - PyPDF2
  - pdfminer.six
- **Vector Database Tools:**
  - FAISS
- **Open-Source Models:**
  - Hugging Face Transformers
  - Ollama.ai
- **Evaluation Frameworks:**
  - RAGAS (Retrieval-Augmented Generation Assessment Suite)

This assignment is designed to help you integrate multiple aspects of generative AI, from data ingestion to model integration and performance evaluation.

Good luck, and we look forward to seeing your innovative solutions!