Task Description: Design and Build an Agentic RAG System for Scientific Documents

Overview

Your task is to design and implement an **agentic Retrieval-Augmented Generation (RAG)** system capable of processing and extracting key insights from scientific documents, specifically **PhD theses** and **peer-reviewed research papers**. The system should employ an agentic workflow, structured prompting and modular reasoning steps, to ensure accurate and context-aware information retrieval.

Requirements

1. **System Capabilities**
    - o The system should be able to **distinguish between a PhD thesis and a single scientific paper**.
    - o For **single papers**, it should answer the following questions:
        - ▪ What are the research questions addressed in the paper?
        - ▪ Which research methods were used?
        - ▪ Is there an evaluation, and if so, what type?
        - ▪ What are some striking results of the paper?
        - ▪ What are the implications for science and practice?
        - ▪ What are the limitations of the paper?
    - o For **PhD theses** (typically a compilation of multiple papers):
        - ▪ Provide a high-level **overview** of the entire thesis.
        - ▪ If prompted, deliver **details of individual papers** contained in the thesis (e.g., titles, questions, methods, results, implications).
    - o Answer user questions related to the document. If the answer is not found in the document, decline to give an answer.
2. **Workflow and Prompting**
    - o Use **agentic workflows** to structure the reasoning process (e.g., role-based prompting, task decomposition, iterative refinement).
    - o You may choose any **open-source or commercial model** (API access is allowed).
    - o Ensure your prompts and system logic guide the LLM to make correct distinctions, retrieve relevant content, and structure responses reliably.
3. **Input Handling**
    - o The system should be able to accept **PDFs or plain text** versions of documents for ingestion.
    - o If preprocessing is necessary (e.g., chunking, metadata extraction), include that logic in your pipeline.
4. **User Interface (Optional but Preferred)**
    - o While a **UI is not required**, a **very basic web interface or CLI** to upload/select a document and pose questions is appreciated.
5. **Technical Flexibility**
    - o You may use frameworks such as LangChain, LlamaIndex, or custom Python scripts.
    - o External vector databases (e.g., FAISS, Pinecone) are allowed for implementing the retrieval backend.
    - o You may use any code that you already have (provided that you wrote it yourself!).
    - o You can deploy the system on a server. Alternatively, you can demonstrate it on your computer during the interview.

Demo (during the interview)

- Explain:
    - o Model choice and reasoning
    - o System architecture
    - o How to run the system
- Show the source code
- Run the demo

Evaluation Criteria

- Use of agentic principles in prompt and workflow design
- Code clarity and modularity
- Working prototype with minimal interface

Test Data
You can find full text of theses here: https://siks.nl/archive/list-of-dissertations/
You can download research papers on any academic database. Most of my papers are open access:
https://scholar.google.com/citations?user=Kd7YYkcAAAAJ&hl=en

Time allocation and partial submissions
Depending on the design and implementation choices, the previous experience of the candidate, this task could take a lot of time. Try to timebox it to an acceptable duration for yourself. It is okay not to complete the task. In case you cannot find the time to finish the task, you may walk us through the partially completed system and your design.

Please reach out for your questions: deniz.iren@ou.nl