ChatDKU Candidate Task

Please build an **Agent RAG system** based on **DSPy** and **LlamaIndex**. The detailed steps are as follows:

1. First, you need to parse, embed, and store the data in a vector database. In this step, your input should be the PDF file "**DKU bulletin**", and the output should be a **persisted** vector database that can be reloaded without re-embedding. In this step, you can refer to the Loading / Indexing / Storing parts in LlamaIndex documentation.
2. After getting a vector database, please proceed by referring to the **Querying** part in LlamaIndex documentation: when a user asks a query, search the vector database for documents relevant to that query and use them as part of the prompt input to the LLM in order to generate higher-quality responses. Once this step is completed, you will have built a simple but complete RAG system using LlamaIndex. (You are encouraged to return citations/evidence for key claims.)
3. Now that you have built a basic RAG system, the next step is to upgrade it into an **Agent RAG system** using **DSPy** to make it more intelligent. You can refer to DSPy for guidance.
4. Optimize your Agentic RAG system through changing some hyperparameters such as chunk size, chunk overlap, retriever top K, and (if used) reranker top N. Please note that DSPy primarily helps optimize prompts/parameters—the actual agent logic (control flow, multi-step retrieval, tool usage) needs to be implemented by you manually in code.

In your Agent RAG system, you should incorporate ALL of the following enhancements :

- **Query Rewrite**: rewrite the user's question (especially useful for typos / abbreviations). Log the original query, rewritten query candidates, and the final rewrite used.

- **Prompt Compression**: if the retrieved context exceeds the LLM context length, compress/summarize/trim the context while preserving key evidence (and keep traceability to original chunks).

- **Planner** : for complex user queries, implement multi-step planning (decompose into sub-questions / multi-hop retrieval) and execute iterative retrieval before final answering.

Reminder: Whether it's a standard RAG system or an Agent RAG built with DSPy, both require a large language model to operate. Considering ease of setup, we recommend running Qwen3 locally using Ollama. You can refer to the LlamaIndex local starter tutorial and Ollama's Qwen3 page：
https://docs.llamaindex.ai/en/stable/getting_started/starter_example_local/ ；
https://dspy.ai/learn/programming/language_models/

By the end, you should deliver an Agent-RAG demo that meets the above requirements and be thoroughly familiar with your code and its mechanics and make sure it can be run in our meeting. You are strongly encouraged to change key hyperparameters (e.g., chunk size, top-k retrieval) to observe their impact on responses.

Evaluation

Besides "it runs", we care a lot about your **thinking and evaluation**. You must include:

- Quantitative evaluation: create a small dev/eval set (e.g., 30–50 queries) and report at least a few **retrieval + generation** quality metrics (you may design your own metric, but it must be consistent and reproducible).

- Latency & trade-offs: report end-to-end latency, and ideally module-level latency (rewrite / plan / retrieve / rerank / compress / generate).

- Failure analysis : provide at least 5 poor cases, and for each case: (1) what went wrong (retrieval? rewrite? planning? compression? hallucination?), (2) what you changed to fix it, (3) before/after impact.

Extra Credit (Must choose at least ONE)

Choose at least one of the following:

- Bonus A — Try BGE-Reasoner and evaluate impact:

  Use BGE-Reasoner as the **first-stage retriever model** (i.e., query encoder / retrieval

model used to fetch top-K candidates).

Report before/after answer quality + retrieval quality + end-to-end latency changes under the same evaluation setup.

https://github.com/FlagOpen/FlagEmbedding/tree/master/research/BGE_Reasoner

- Bonus B — Try TWO rerankers and compare: integrate both rerankers below, evaluate under the same setup, and compare quality + latency + what you would choose for ChatDKU:

  - BAAI/bge-reranker-v2-m3

  - mixedbread-ai/mxbai-rerank-base-v2