Lecture 4: RAG Based Prompting

Mia Mohammad Imran

Overview

- Talking about Code Translation problem
- Introducing RAG-based prompting

Week 1: Paper Review Feedback

- When you cite, you must cite a different source and not the paper you are reviewing to support your claim, otherwise it may lead to circular dependency problem
- Going forward, follow this format:
  - Spacing: single
  - Font size: 11 pt
  - Font: Times new roman/Calibri/Arial

Week 1: Hands-on Activity Feedback

- **Won’t count to final grade, instead who has done well, I will add as bonus**
- I have made some comments on some submission, reply those as your grades will depend on it. Even if it’s currently showing full mark, it may change unless you answer.

Going forward, make sure:

- Ollama is installed and runs properly.
- Run the demo I show in class in your local pc/cloud as every weekly hands-on activity will depend on those demo.
- Langchain and other relevant libraries that would be demo-ed, runs on your local/cloud. If it does not, google first to figure out common issues. If the issues still persists, let me know asap.

Common Mistakes in your Prompts

- Making the prompts too specifics ({"addr": "88"})
- Taking a specific problem (e.g., JSONProcessor)
- Not developing the ground rules properly on how to translate C++ to Python
- Not done the error analyze properly

Designing a Prompt

- Define what's "Expected Behavior"
- *Establish the core set of rules on how to solve the problem: aka designing the high level algorithm*
- Design initial prompt
- Run and observe error
- Update the prompt
- Repeat the process

So What are the Expected Behavior of Code Translation?

- All test cases must pass
  - (At least reasonable amount)

Run Zero-shot and Observe Errors

- What are the common errors?

Run Zero-shot and Observe Errors

- What are the common errors?
  - Type Conversion
  - Return Errors
  - Class/Struct conversions
  - Map to Dict conversion

What are the rules to translate C++ to Python?

https://lec.inf.ethz.ch/tutorials/cpp-to-py

- How to translate types
- Translate Return types
- Translate c++ class/struct to python class
- Translate fields
- Translate keywords
- Naming conventions
- Anything specific that python needs to taken care of
- Define any specific rules that need to be followed

Revising the prompt

- Develop your initial prompt
- If you satisfied with so far prompt, let’s try to add advance techniques
  - Like few-shot
  - Least-to-Most
  - Active prompt
  - ReAct style
  - **RAG based prompting**
- https://www.promptingguide.ai/techniques

What Is Retrieval-Augmented Generation (RAG)?

Retrieval-Augmented Generation (RAG) is a paradigm in which a language model is augmented with **external knowledge retrieved at inference time** rather than relying solely on its internal, parametric memory.

In standard prompting, the model generates outputs based only on:

- Its pretraining data
- The prompt provided by the user

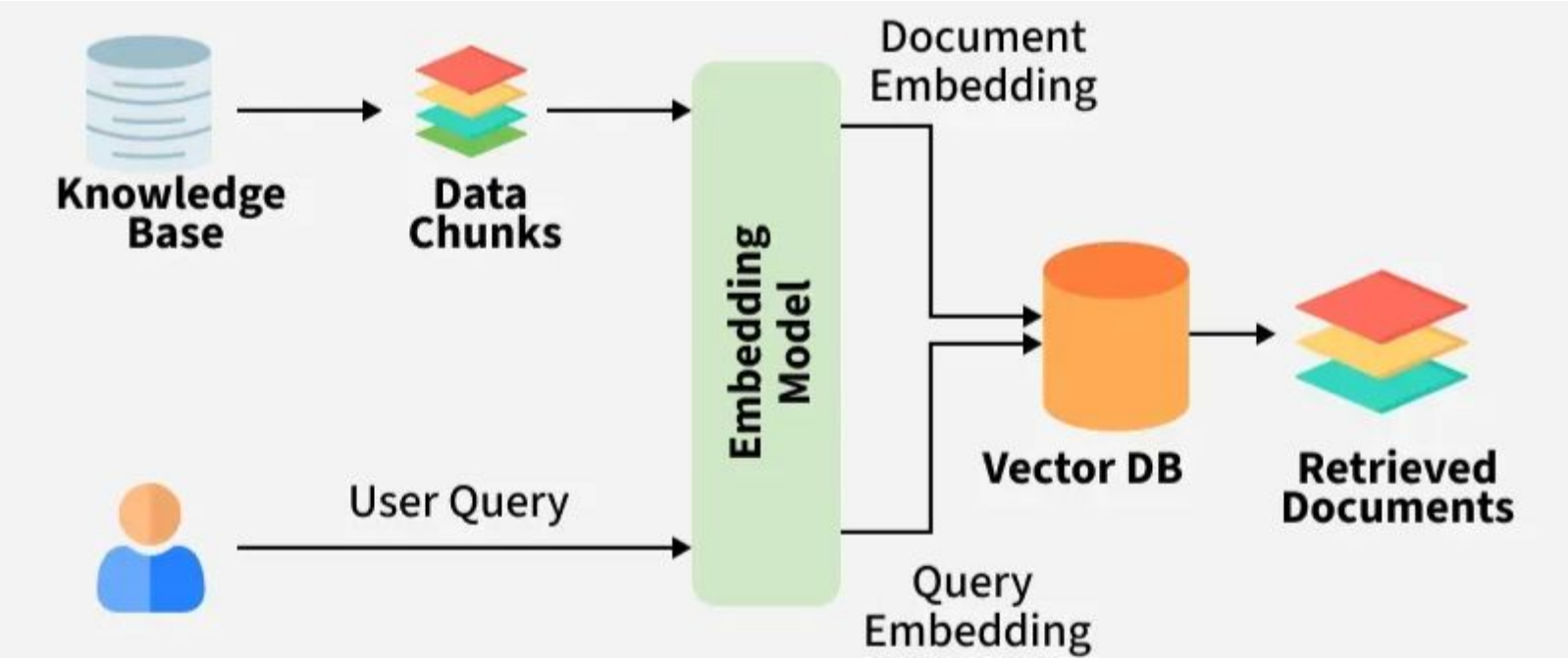
In RAG-based prompting, generation is grounded in **retrieved, task-relevant context**.

Core Definition

RAG combines two components:

1. **Retrieval**: Fetch relevant external documents, rules, or examples
2. **Generation**: Use the retrieved context as part of the prompt to guide the model's output

RAG Architecture Overview



RAG-based Prompting

Core idea:
Prompt = Task Instruction + Retrieved Context + Constraints

How RAG Helps

Limitations of prompt-only LLM usage:

- Knowledge cutoff and staleness
- Hallucination under missing facts
- Contextual brittleness for domain-specific tasks
- Poor traceability to source material

RAG addresses these by grounding generation in **verifiable, up-to-date, and scoped data**.

How RAG Works (Conceptual Pipeline)

User Task

- Example: Translate C++ code to Python.

Query Construction

- Queries are derived from the task, code structure, or observed errors.

Retrieval

- Relevant documents, rules, or examples are fetched from a corpus.

Prompt Assembly

- Retrieved content is injected into a fixed prompt template.

Generation

- The LLM produces output while reasoning over the retrieved context.

Demo

https://github.com/imranraad07/CS-5001-AI-Augmented-SE

Summary

- RAG can improve the knowledge-base and prompt formation
- However, at the same time, it adds additional complexity