UCLA Advanced Research Computing

RAG For Open Source LLMs
Presented By
OARC Research Data and Web Platforms Group

Contact Us: webteam@oarc.ucla.edu

Andrew Browning - Research Data and Web Platforms Manager

Anthony Doolan - Web Developer

Hayk Zakaryan - Software Architect

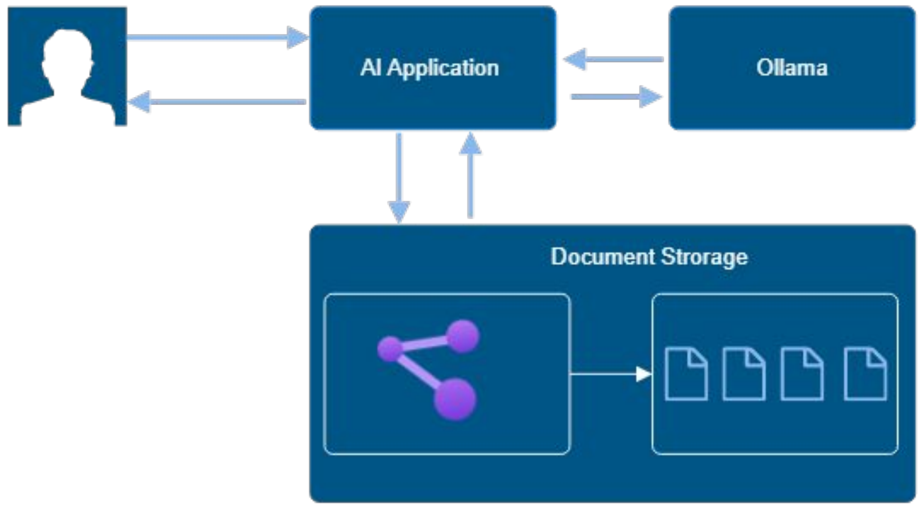
Agenda

1. Introduction to RAG
2. Application Framework
3. Ollama Setup & Installation
4. Prompt Engineering
5. Sample Python Code
6. Sample Application Demonstration

Introduction to RAG

Retrieval-Augmented Generation (RAG) allows for LLM query retrieval against a stored set of documents restricting LLM inferences to domain specific datasets

Application Framework



- User submits query to AI application
- AI application queries document storage and fetches result
- AI application submits original query and document result to Ollama for inference
- AI application returns result to user

Ollama Setup and Installation

Ollama download page: https://ollama.com/download

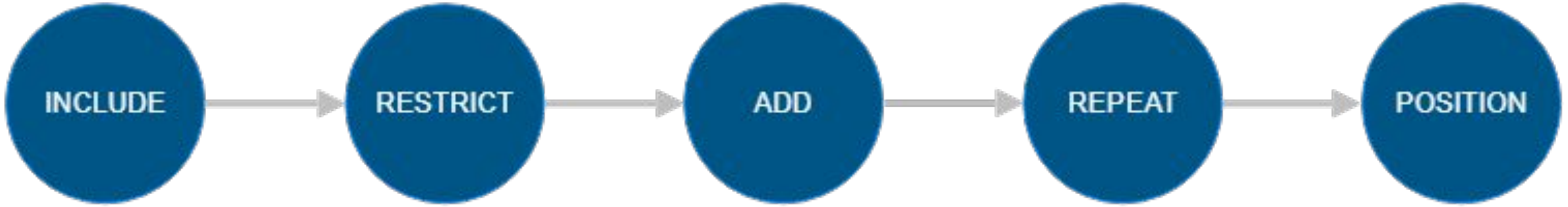
MAC

```
brew install ollama
ollama serve - start ollama
ollama pull mistral - download mistral
ollama list - list available models
pip install ollama - Install Ollama Python package
```

Linux

```
curl -fsSL https://ollama.com/install.sh | sh
ollama serve - start ollama
ollama pull mistral - download mistral
ollama list - list available models
pip install ollama - Install Ollama Python package
```

Prompt Engineering



Include instructions that the model not make anything up and only stick with the facts

Restrict the output to a confined list

Add chain of thought style instructions, e.g., "solve the problem step by step."

Repeat most important instructions in the prompt multiple times

Position most important instructions last

Prompt Engineering

- Prompts guide the LLM to produce the desired output
- Prompt engineering has a huge impact over the LLM output
- Prompt engineering requires creativity, logic, and experimentation
- Prompt engineering requires understanding of capabilities and limitations of the LLM
- Prompt engineering is a tool to control hallucinations
- Prompts are vulnerable to hacking, so meta-prompts must be designed to counteract that

Knowledge Base

- Facebook AI Similarity Search (FAISS)
    - Very easy to setup and experiment with
    - In memory storage
    - Fast
    - Persistent storage is more complicated
- Chroma DB
    - Easy to set up persistent Storage
    - Slower than FAISS for large scale queries

Chunking and Embedding Model

- Ollama and Hugging Face both have embedding models that can be leveraged by langchain.
- Chunking helps to keep the context short and concise.
- Different document types and use cases require different chunking strategies.
- Example Chunking Strategies:
  - Plain Text - Split on character count and new lines
  - CSV - Split on new rows
  - Web Page - Split on headings

Sample Python Code (Prompt)

```
CUSTOM_PROMPT = PromptTemplate(
    template="""
    You are an expert at making desserts. You will only answer questions about making desserts.
    If the user asks you about something else, respond: "I am not able to answer that question."
    If you do not know the answer, say: "I do not know."

    Think about this step by step.

    Answer the following question using only the given context: {context}.
    Question: {question}
    Answer:
    """,
    input_variables=["context", "question"],
)
```

Sample Application Demo

1.) Command line Rag demo - Uses open source tools/packages only

2.) UI based Rag demo - Uses open source tools/packages only

Further Reading - Available Tools and Model Selection

Ollama - Download page and resources

Lanchain - Homepage and resources

Embeddings - Resource to learn more about LLM embeddings

Python3 - Python homepage

ChromaDB - Chroma homepage and resources

Marqo - Marqo home page and resources

Redis - Redis homepage resources