UCLA Advanced Research Computing

Learn Prompt Engineering and Retrieval Augmented Generation Using 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 Prompt Engineering and Retrieval Augmented Generation (RAG)
2. Prompt Engineering Basics
3. RAG with Prompt Engineering
4. RAG Web-Based DEMO
5. Conclusion

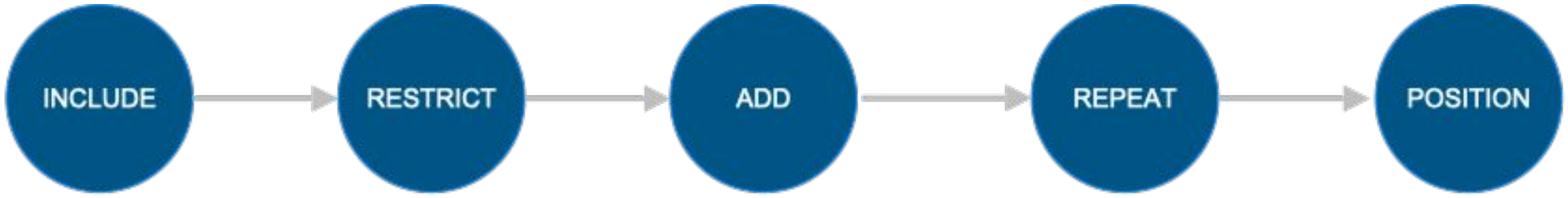
Prompt Engineering Basics

Introduction to Prompt Engineering and RAG

Prompt Engineering is the process of crafting instructions (prompts) to guide generative AI models, ultimately controlling outputs.

RAG is the process of using an external knowledge base in combination with Prompt Engineering to further restrict outputs

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

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
```

Prompts At The Command Line

- Ollama run gemma:2b
- **INCLUDE:** You are friendly story teller that tells stories in the style of Charles Bukowski. Your stories only include factual information about Bombay, India in the 1970s.
- **RESTRICT:** Your output is always a 500 word original short story.
- **ADD:** Your stories will have strong character development and all characters will be Indian Women except for one male protagonist named Chinaski.
- **REPEAT:**You will write these stories as Charles Bukowski would have written them in the 1970s.
- **POSITION:** Your stories must evoke sympathy for the characters and the time and include relevant news information from Bombay, India in the 1970s.

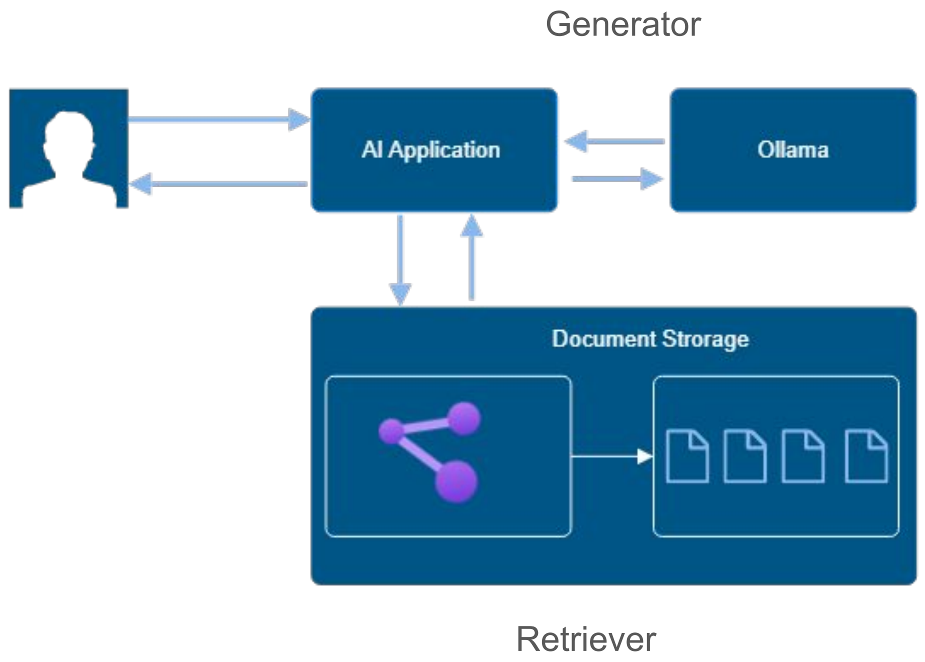
RAG With Prompt Engineering

Leveraging Prompt Engineering Through RAG

- RAG requires a good knowledge base(s)

- RAG allows the developer to create prompts on behalf of a users who may not know how to engineer prompts.
- RAG has two parts
  - Retriever - grabs documents/data from the knowledge base and passes the data to the LLM.
  - Generator - Uses the prompt and the LLM to generate responses based upon result from the retriever.
- RAG use cases - Chatbots, QA systems, Medical Records - Personalized, real time content generation.

RAG 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

RAG Application Framework (Cont)

Sample Prompt Sent to LLM

- **User question** - What is the current weather?
- **Knowledge base context** - Today is bright and sunny, tomorrow is rainy.
- **Rules**
  - Prompt Engineering Steps, i.e., Include, restrict, add, repeat, position
  - Return weather based upon user's input

→

Today's weather is bright and sunny.

UCLA Advanced Research Computing

RAG Application Framework (Cont)

- RAG Prompt Engineering
  - Guide the LLM to produce the desired output
  - Requires creativity, logic, and experimentation
  - Requires understanding of capabilities and limitations of the chosen LLM
  - Is a tool to control hallucinations
  - Is a way to decrease vulnerabilities such as prompt injection

Sample Python Application

```
# Split text into smaller chunks
text_splitter = CharacterTextSplitter(chunk_size=300, chunk_overlap=0)
doc_chunks = text_splitter.create_documents([docs_text])

# Create local embeddings & store in a local vector db. Knowledge is lost on quit
embed_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
vectorstore = FAISS.from_documents(doc_chunks, embedding=embed_model)

# Create a retrieval-based QA chain using Ollama as the LLM
retriever = vectorstore.as_retriever(search_type="similarity", search_kwargs={"k": 2})

CUSTOM_PROMPT = PromptTemplate(
      template="""You are an expert at making desserts. You will only answer questions about making desserts…
      Answer the following question using only the given context: {context}.
      Question: {question}
      Answer:""",
      input_variables=["context", "question"],
)
```

Sample Python Application

```
# Create the chain
ollama_llm = OllamaLLM(
      endpoint_url="http://127.0.0.1:11434", model="llama3.2", temperature=0.7 )

qa_chain = RetrievalQA.from_chain_type(
      llm=ollama_llm, chain_type="stuff", retriever=retriever, chain_type_kwargs={"prompt": CUSTOM_PROMPT})

# Pydantic model for incoming requests
class QueryRequest(BaseModel):
      question: str

@app.post("/query")
def query_llm(request: QueryRequest = Body(...)):
      user_question = request.question
      result = qa_chain.invoke({"query": user_question})
      return result['result']
```

Sample Python Application (Cont)

Langchain compatible Open-source vector databases for RAG applications

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

Using Chunking and Embedding Models

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

Example Command Line Demo

Github Repo:
https://github.com/ucla-oarc-web/rag-for-open-source-llms

Tools Required

- Docker or Python
- Ollama

Prompting with a Web Based RAG Application

- Uses all Open Source Technologies
- Accepts PDF or Web URL Knowledge Base
- Programming Language - Python 3
- Large Language Model - Llama 3
- Python Framework - Langchain
- UI Framework - Streamlit

RAG Web-Based Demo

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

Questions