CS4221 Tutorial 5: Retrieval-Augmented Generation (RAG)

Yao LU

2024 Semester 2

Objective

By the end of this tutorial, you will:

- ➢ Set up local large language model (LLM).
- ➢ Prepare the virtual environment for mini-RAG application.
- ➢ Understand the basic pipeline in RAG application.
- ➢ Learn to implement a mini-RAG under *LangChain* framework.

Setup

Step 1: Install ollama

- ➢ Download the llama docker image from dockerhub.
- ➢ Choose one specific model and start up the model service following README .

Step 2: Prepare the environment

- ➢ Make sure you have installed the *anaconda.*
- ➢ Download the provided environment file.
- ➢ Set up the virtual environment (may take a few minitutes)

NOTE: If the virtual environment exists, delete it and create a new one.

Prerequisites

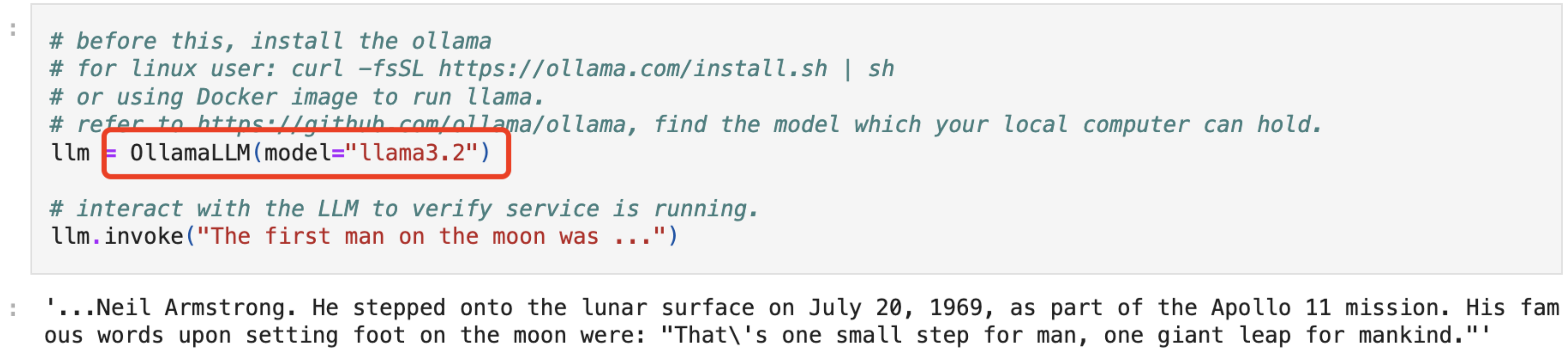
Before starting, ensure you have gone through :

- ➢ Ollama overview:
- https://github.com/ollama/ollama

- ➢ LangChain
- RAG Concept: https://python.langchain.com/docs/concepts/rag/
- RAG Guide Part 1: https://python.langchain.com/docs/tutorials/rag/
- RAG Guide Part 2: https://python.langchain.com/docs/tutorials/qa_chat_history/

Hello-World

NOTE: model variable should match your local deployed LLM.



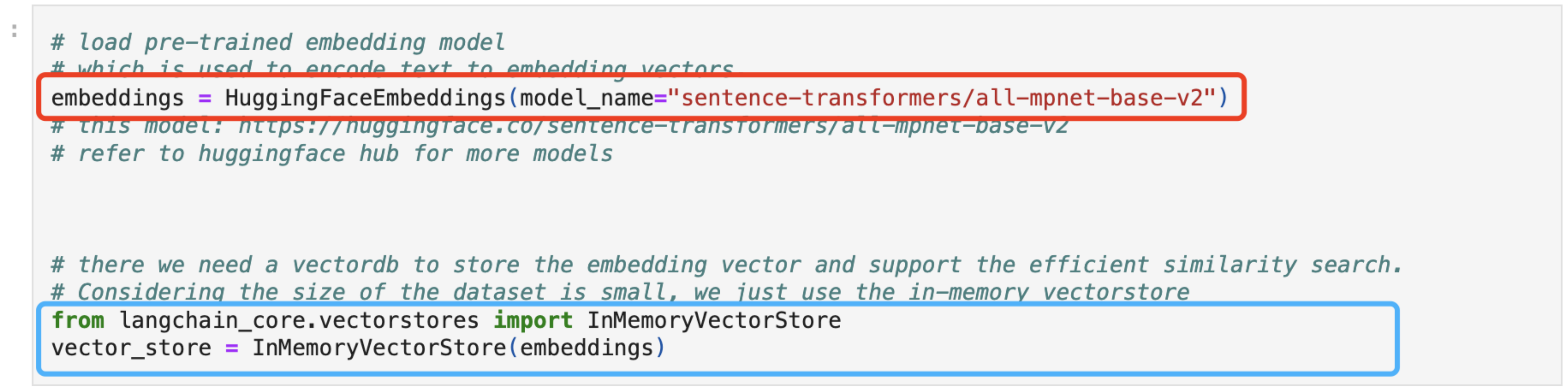
```
# before this, install the ollama
# for linux user: curl -fsSL https://ollama.com/install.sh | sh
# or using Docker image to run llama.
# refer to https://github.com/ollama/ollama, find the model which your local computer can hold.
llm = OllamaLLM(model="llama3.2")

# interact with the LLM to verify service is running.
llm.invoke("The first man on the moon was ...")
```

```
'...Neil Armstrong. He stepped onto the lunar surface on July 20, 1969, as part of the Apollo 11 mission. His fam
ous words upon setting foot on the moon were: "That\'s one small step for man, one giant leap for mankind."'
```

Hello-World

NOTE:

- The first execution will install the embedded model from the huggingface.
- Try to replace the in-memory vector store to the Milvus deployed previously

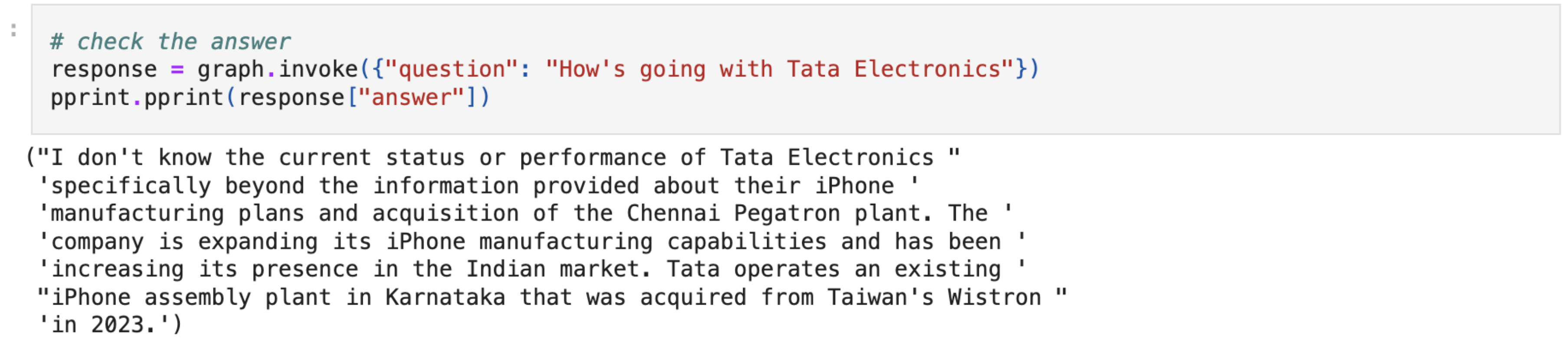


```
# load pre-trained embedding model
# which is used to encode text to embedding vectors
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")
# this model: https://huggingface.co/sentence-transformers/all-mpnet-base-v2
# refer to huggingface hub for more models



# there we need a vectordb to store the embedding vector and support the efficient similarity search.
# Considering the size of the dataset is small, we just use the in-memory vectorstore
from langchain_core.vectorstores import InMemoryVectorStore
vector_store = InMemoryVectorStore(embeddings)
```

Hello-World

NOTE:

- Try different questions and verify that you can search for relevant news.



```python
# check the answer
response = graph.invoke({"question": "How's going with Tata Electronics"})
pprint.pprint(response["answer"])
```

```
("I don't know the current status or performance of Tata Electronics "
 'specifically beyond the information provided about their iPhone '
 'manufacturing plans and acquisition of the Chennai Pegatron plant. The '
 'company is expanding its iPhone manufacturing capabilities and has been '
 'increasing its presence in the Indian market. Tata operates an existing '
 "iPhone assembly plant in Karnataka that was acquired from Taiwan's Wistron "
 'in 2023.')
```