How to Run Your RAG from the Command Line

Great — I inspected your `My First RAG.ipynb` and can now tell you exactly how to use it from the command line and directly in the notebook. Your notebook contains the essential pieces of a Retrieval-Augmented Generation (RAG) pipeline:

- Loading a dataset (e.g., cat facts)
- Embedding chunks using Ollama
- Storing the embeddings
- A retrieval function to get the most relevant chunks
- A chat loop that builds a prompt and queries the model

This document explains how to turn that notebook into a full RAG system you can run from:

1. the **command line**, and
2. directly inside the **notebook**.

1. Preparing a Python RAG Module

To use your RAG pipeline outside the notebook, create a file named `rag.py` containing the logic for embeddings, retrieval, and answering:

rag.py

```
import json
import numpy as np
import ollama

# --------------------------------------
# Cosine similarity WITHOUT sklearn
# --------------------------------------
def cosine_similarity(a, b):
a_norm = a / np.linalg.norm(a)
b_norm = b / np.linalg.norm(b, axis=1, keepdims=True)
return np.dot(b_norm, a_norm)

# Load dataset
```

```
with open("cat-facts.txt") as f:
dataset = f.readlines()

# Embedding using Ollama
def embed(text):
response = ollama.embeddings(model="mxbai-embed-large", prompt=text)
return np.array(response["embedding"])

# Precompute embeddings
embeddings = np.array([embed(chunk) for chunk in dataset])

def retrieve(query, k=5):
query_emb = embed(query)
sim = cosine_similarity(query_emb, embeddings)
top_k_idx = np.argsort(sim)[-k:][::-1]
return [(dataset[i], float(sim[i])) for i in top_k_idx]

def answer(query):
retrieved = retrieve(query)
context = "\n".join([f"- {c}" for c, s in retrieved])

prompt = f"""You are a helpful chatbot.
Use only this context:
{context}

Question: {query}
Answer:"""

response = ollama.generate(model="llama3", prompt=prompt)
return response["response"]
```

2. Creating a Command-Line Interface

Now create a wrapper script `ask.py`:

ask.py

```
from rag import answer

query = input("Ask a question: ")
response = answer(query)
print("\nAnswer:")
print(response)
```

You can now run:

```
python ask.py
```

This launches an interactive RAG chatbot directly from your terminal.

3. Asking Questions Inside the Notebook

Inside the original notebook, define:

```
def ask(query):
retrieved = retrieve(query)
context = "\n".join([f"- {c}" for c, s in retrieved])

prompt = f"""You are a helpful chatbot.
Use only this context:
{context}

Question: {query}
Answer:"""

response = ollama.generate(model="llama3", prompt=prompt)
return response["response"]
```

Then simply call:

```
ask("Tell me an interesting cat fact.")
```

4. Optional: Interactive Chat Loop

Add this cell to the notebook:

```
while True:
q = input("You: ")
if q.lower() in ["quit", "exit"]:
break
print("Bot:", ask(q))
```

Conclusion

You now have:

- a full RAG pipeline,
- a command-line interface,
- and direct notebook interaction.

This structure lets you expand into vector databases (FAISS, Chroma), REST APIs, or a fully interactive UI if needed.