Use a local LLaMA-based model (or any local large language model) for **RAG-style (retrieval-augmented generation)** answers — instead of just showing similar chunks.

We'll combine:

- FAISS for retrieval
- LLaMA or another local model via **`llama-cpp`** or Hugging Face
- LangChain's `RetrievalQA` interface

To download `llama-2-7b.Q4_K_M.gguf`, follow these steps. This model is available on **Hugging Face**, and you can download it manually or using Python.

Option 1: Manual Download via Browser

1. Go to the model page:
   TheBloke/Llama-2-7B-GGUF

https://huggingface.co/matrixportal/Llama-2-7b-chat-hf-Q4_K_M-GGUF/resolve/main/llama-2-7b-chat-hf-q4_k_m.gguf

2. Scroll down to **Files and versions**.
3. Find the file:
   `llama-2-7b.Q4_K_M.gguf`
4. Click the **Download** button next to it.
5. Create a `models/` directory in your project folder, and move the `.gguf` file there:
6. Create a `models/` directory in your project folder, and move the `.gguf` file there:

```
your_project/
├── models/
│   └── llama-2-7b.Q4_K_M.gguf
└── your_script.py
```

Option 2: Download via `huggingface_hub` (Python script)

1. Install the Hugging Face hub CLI:

```
pip install huggingface-hub
```

2. Use the following script to download the model programmatically:

Python code:-

```
from huggingface_hub import hf_hub_download

model_path = hf_hub_download(
    repo_id="TheBloke/Llama-2-7B-GGUF",
    filename="llama-2-7b.Q4_K_M.gguf",
    local_dir="models"
)

print("Downloaded to:", model_path)
```

Notes

- The file is **~3.8 GB**. Make sure you have enough space.
- If you're behind a proxy or firewall, use manual download.
- To use the model with `llama-cpp`, you don't need to extract anything — just point `model_path` to the `.gguf` file.

Required Packages

Install everything you need:

pip install langchain-core langchain-community langchain-huggingface sentence-transformers faiss-cpu torch PyPDF2 llama-cpp-python

Python Code:-

```
from langchain_community.vectorstores import FAISS
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_core.documents import Document
from langchain_community.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.llms import LlamaCpp
from langchain.chains import RetrievalQA

# 1. Load the local embedding model
embedding = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")

# 2. Load and split the PDF
pdf_path = "my-notes.pdf"
loader = PyPDFLoader(pdf_path)
pages = loader.load()

text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=50)
documents = text_splitter.split_documents(pages)

# 3. Create the FAISS vector store
vectorstore = FAISS.from_documents(documents, embedding)

# 4. Load local LLaMA model
llm = LlamaCpp(
    model_path="models/llama-2-7b.Q4_K_M.gguf",  # 🔁 Change to your downloaded model path
    n_ctx=2048,
    temperature=0.5,
    top_p=0.95,
    verbose=True
)

# 5. Create RetrievalQA chain (RAG)
qa_chain = RetrievalQA.from_chain_type(
    llm=llm,
    retriever=vectorstore.as_retriever(search_kwargs={"k": 2}),
    return_source_documents=False  # Set True if you want to see source chunks
)

# 6. Chat loop
print("\n📄 PDF Chatbot using LLaMA is ready! Ask questions (type 'exit' to quit)\n")
```

```
while True:
    query = input("❓ You: ")
    if query.lower() in ["exit", "quit"]:
        print("👋 Goodbye!")
        break

    response = qa_chain.run(query)
    print(f"💡 Answer: {response}\n")
```