# Install required packages

# pip install langchain langchain-community chromadb sentence-transformers pypdf gradio transformers torch

import gradio as gr

from langchain\_community.document\_loaders import PyPDFLoader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain\_community.vectorstores import Chroma

from langchain\_community.embeddings import HuggingFaceEmbeddings

from langchain\_community.llms import HuggingFacePipeline

from langchain.chains import RetrievalQA

from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline

import torch

# Step 1: Load the PDF

loader = PyPDFLoader("paper.pdf")

documents = loader.load()

# Step 2: Split the document into chunks

text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=500, chunk\_overlap=50)

chunks = text\_splitter.split\_documents(documents)

# Step 3: Create a vector store with ChromaDB

embedding\_model = HuggingFaceEmbeddings(model\_name="all-mpnet-base-v2")

vectorstore = Chroma.from\_documents(

documents=chunks,

embedding=embedding\_model,

collection\_name="qa\_documents"

)

# Step 4: Set up the retriever

retriever = vectorstore.as\_retriever(search\_kwargs={"k": 3})

# Step 5: Set up the LLM (using Hugging Face's Flan-T5)

model\_name = "google/flan-t5-base"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSeq2SeqLM.from\_pretrained(model\_name)

pipe = pipeline(

"text2text-generation",

model=model,

tokenizer=tokenizer,

max\_length=512,

device=0 if torch.cuda.is\_available() else -1 # Use GPU if available

)

llm = HuggingFacePipeline(pipeline=pipe)

# Step 6: Create the QA chain

qa\_chain = RetrievalQA.from\_chain\_type(

llm=llm,

chain\_type="stuff",

retriever=retriever,

return\_source\_documents=True

)

# Step 7: Define a function to interact with the QA bot

def qa\_bot(query):

result = qa\_chain({"query": query})

answer = result["result"]

source\_docs = [doc.page\_content for doc in result["source\_documents"]]

return answer, source\_docs

# Step 8: Create a Gradio interface

def qa\_interface(pdf\_path, query):

answer, source\_docs = qa\_bot(query)

return f"\*\*Answer:\*\* {answer}\n\n\*\*Source Documents:\*\*\n{source\_docs}"

# Launch the Gradio interface

interface = gr.Interface(

fn=qa\_interface,

inputs=[

gr.File(label="Upload PDF (already loaded as paper.pdf)"),

gr.Textbox(label="Query", value="What this paper is talking about?")

],

outputs=gr.Textbox(label="Response"),

title="QA Bot for PDF Documents"

)

interface.launch()