import streamlit as st

from PyPDF2 import PdfReader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain\_community.embeddings import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.chains.question\_answering import load\_qa\_chain

from langchain\_community.chat\_models import ChatOpenAI

OPENAI\_API\_KEY = "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" #Pass your key here

#Upload PDF files

st.header("My first Chatbot")

with st.sidebar:

st.title("Your Documents")

file = st.file\_uploader(" Upload a PDf file and start asking questions", type="pdf")

#Extract the text

if file is not None:

pdf\_reader = PdfReader(file)

text = ""

for page in pdf\_reader.pages:

text += page.extract\_text()

#st.write(text)

#Break it into chunks

text\_splitter = RecursiveCharacterTextSplitter(

separators="\n",

chunk\_size=1000,

chunk\_overlap=150,

length\_function=len

)

chunks = text\_splitter.split\_text(text)

#st.write(chunks)

# generating embedding

embeddings = OpenAIEmbeddings(openai\_api\_key=OPENAI\_API\_KEY)

# creating vector store - FAISS

vector\_store = FAISS.from\_texts(chunks, embeddings)

# get user question

user\_question = st.text\_input("Type Your question here")

# do similarity search

if user\_question:

match = vector\_store.similarity\_search(user\_question)

#st.write(match)

#define the LLM

llm = ChatOpenAI(

openai\_api\_key = OPENAI\_API\_KEY,

temperature = 0,

max\_tokens = 1000,

model\_name = "gpt-3.5-turbo"

)

#output results

#chain -> take the question, get relevant document, pass it to the LLM, generate the output

chain = load\_qa\_chain(llm, chain\_type="stuff")

response = chain.run(input\_documents = match, question = user\_question)

st.write(response)