**Offline Chat with PDF using LangChain & HuggingFace**

**Overview**

This practical project allows students to develop an application that enables chatting with uploaded PDF files entirely offline using local models for both embeddings and language generation.

**Features Used**

* **LangChain**: For chaining retrieval and question-answering
* **Sentence Transformers**: For text embedding
* **FAISS**: For vector storage and similarity search
* **Hugging Face Transformers**: For offline text generation
* **Streamlit**: For the user interface

**Step-by-Step Instructions**

**1. Install Python and Git**

* Download and install Python (recommended 3.10+) from: <https://www.python.org/downloads/>
* Install Git from: <https://git-scm.com/downloads>

**2. Create Project Directory**

mkdir RAG\_LocalChatPDF

cd RAG\_LocalChatPDF

**3. Create Virtual Environment**

python -m venv .venv

**4. Activate Environment (Windows/Linux/Mac)**

.venv\Scripts\activate

**5. Create requirements.txt File**

Create a file named requirements.txt and paste the following:

streamlit

PyPDF2

sentence-transformers

langchain

langchain-community

transformers

accelerate

torch

faiss-cpu

**6. Install Required Libraries**

pip install -r requirements.txt

**7. Prepare Local Models**

**7.1. Sentence Transformer (Embeddings)**

Download model [all-MiniLM-L6-v2](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2)

* Create folder: local\_model
* Unzip the downloaded model files into ./local\_model/

**7.2. LLM Generator (LaMini-T5)**

Download model [LaMini-T5](https://huggingface.co/MBZUAI/LaMini-T5-738M)

* Create folder structure: models/LaMini-T5
* Unzip the downloaded files inside it.

Now, your folder structure should look like:

**📁 Directory Structure**

RAG\_LocalChatPDF/

├── chatpdf.py # Main app code

├── requirements.txt # Python package requirements

├── local\_model/ # Sentence Transformer model files

├── models/

│ └── LaMini-T5/ # Text generation model files

└── .venv/ # Virtual environment

**chatpdf.py (Main Application Code)**

import streamlit as st

from PyPDF2 import PdfReader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain\_community.vectorstores import FAISS

from langchain\_community.embeddings import HuggingFaceEmbeddings

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

from langchain\_community.llms import HuggingFacePipeline

from transformers import pipeline

# Load offline QA model

def load\_qa\_model():

return pipeline(

"text2text-generation",

model="./models/LaMini-T5",

tokenizer="./models/LaMini-T5",

device=0 # GPU if available

)

# Extract text from PDF

def load\_pdf\_text(pdf\_docs):

text = ""

for pdf in pdf\_docs:

reader = PdfReader(pdf)

for page in reader.pages:

page\_text = page.extract\_text()

if page\_text:

text += page\_text

return text

# Split text into chunks

def get\_text\_chunks(text):

splitter = RecursiveCharacterTextSplitter(chunk\_size=1000, chunk\_overlap=200)

return splitter.split\_text(text)

# Create vectorstore with embeddings

def get\_vectorstore(chunks):

embeddings = HuggingFaceEmbeddings(model\_name="./local\_model")

return FAISS.from\_texts(texts=chunks, embedding=embeddings)

# Answer user query

def ask\_question(vectorstore, query):

retriever\_docs = vectorstore.similarity\_search(query, k=3)

qa\_llm = HuggingFacePipeline(pipeline=load\_qa\_model())

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

return chain.run(input\_documents=retriever\_docs, question=query)

# Streamlit Interface

def main():

st.set\_page\_config(page\_title="Offline PDF Chat")

st.title("Ask Questions from Your PDF (100% Offline)")

pdf\_docs = st.file\_uploader("Upload PDF files", type="pdf", accept\_multiple\_files=True)

if pdf\_docs and st.button("Process PDFs"):

with st.spinner("Processing PDFs..."):

text = load\_pdf\_text(pdf\_docs)

chunks = get\_text\_chunks(text)

vectorstore = get\_vectorstore(chunks)

st.success("PDFs processed successfully!")

query = st.text\_input("Ask a question from your PDFs:")

if query:

with st.spinner("Generating answer..."):

result = ask\_question(vectorstore, query)

st.write("\*\*Answer:\*\*", result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Demo Questions You Can Ask**

* What is the summary of the document?
* What are the key takeaways?
* Explain concept X in simple terms.
* List the main sections discussed.

**Key Concepts to Explain in Seminar**

* **Offline capability**: No need for internet once models are downloaded.
* **RAG Architecture**: Combines retrieval (vector search) and generation (LLM).
* **Open-source models**: Cost-effective, customizable.
* **Streamlit UI**: Enables easy interaction for non-technical users.

**Final Tips**

* Ensure your .venv is activated during setup.
* Do not delete model folders after download.
* GPU will significantly improve response time.

Let me know if you want slides or a video version next!