Offline Chat with PDF using LangChain & HuggingFace

Overview

This practical project allows users to chat with uploaded PDF files entirely offline using local models for both embeddings and language generation.

It uses:

- **LangChain** for chaining retrieval and QA
- **Sentence Transformers** for embedding
- **FAISS** for vector storage and similarity search
- **Hugging Face transformers** for text generation
- **Streamlit** for web UI

Directory Structure

```
RAG_LocalChatPDF/
├── chatpdf.py                  # Main app code
├── requirements.txt            # Python package
requirements
├── local_model/                # Local Sentence
Transformer model (e.g., all-MiniLM-L6-v2)
├── models/
│   └── LaMini-T5/              # Local generation model
directory
└── .venv/                      # Python virtual
environment
```

---

requirements.txt

```
streamlit
PyPDF2
sentence-transformers
langchain
langchain-community
transformers
accelerate
torch
faiss-cpu
```

Setup Instructions

```
# 1. Create Virtual Environment
python -m venv .venv

# 2. Activate Environment (Windows)
.venv\Scripts\activate

# 3. Install Requirements
pip install -r requirements.txt

# 4. Place local models in their directories
#    - Sentence Transformer -> ./local_model/
#    - LLM Generator        -> ./models/LaMini-T5/

# 5. Run Streamlit App
streamlit run chatpdf.py
```

chatpdf.py (Main 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 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 questions from PDF chunks
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)

# UI
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.

Emphasize **offline capability**: No internet needed after downloading models

- Explain **RAG architecture**
- Mention use of **open-source models** (LaMini-T5, MiniLM)
- Point out where models are **plug-and-play**
