Project 81: Chat with PDF (One File)

Description:
This tool allows users to upload a PDF and ask questions about its content. It reads the file, chunks it intelligently, and uses embedding + retrieval to power a mini chatbot that can answer questions contextually from the document.

chat_with_pdf.py

```python
import os
import openai
import gradio as gr
import PyPDF2
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer
import numpy as np

# Load OpenAI key
openai.api_key = os.getenv("OPENAI_API_KEY")

# Load embedding model
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")

# Function to extract and chunk text from PDF
def extract_text_chunks(pdf_file, chunk_size=500):
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    text = ""
    for page in pdf_reader.pages:
        text += page.extract_text()

    # Split into chunks
    chunks = [text[i:i+chunk_size] for i in range(0, len(text), chunk_size)]
    return chunks

# Embed all chunks
def embed_chunks(chunks):
    return embedding_model.encode(chunks)
```

```python
# Search for relevant chunks based on query
def search_chunks(query, chunks, chunk_embeddings):
    query_embedding = embedding_model.encode([query])[0]
    similarities = cosine_similarity([query_embedding], chunk_embeddings)[0]
    top_indices = np.argsort(similarities)[-3:][::-1]  # Top 3 most similar
    return [chunks[i] for i in top_indices]

# Generate answer using OpenAI and relevant chunks
def ask_pdf_question(pdf_file, query):
    chunks = extract_text_chunks(pdf_file)
    chunk_embeddings = embed_chunks(chunks)
    top_chunks = search_chunks(query, chunks, chunk_embeddings)

    context = "\n\n".join(top_chunks)

    prompt = (
        f"You are an assistant who answers questions based on the following PD
        f"{context}\n\n"
        f"Question: {query}\nAnswer:"
    )

    try:
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[{"role": "user", "content": prompt}]
        )
        return response['choices'][0]['message']['content'].strip()
    except Exception as e:
        return f"Error: {str(e)}"

# Gradio interface
iface = gr.Interface(
    fn=ask_pdf_question,
    inputs=[
        gr.File(label="Upload PDF", file_types=[".pdf"]),
        gr.Textbox(label="Ask a question about the PDF")
    ],
    outputs="text",
    title="📄 Chat with PDF",
    description="Upload a PDF file and ask questions about its content. The AI
)

# Launch the app
```

```
iface.launch()
```