Double-click (or enter) to edit

Assignment 12: Generative AI and RAG Pipeline

Name: Daniel Muthama

Program: DATA and AI

Date: July 29, 2025

Introduction

This notebook implements a Retrieval-Augmented Generation (RAG) pipeline to process a PDF document, extract relevant information, and generate context-aware answers using a generative model. The pipeline uses `langchain` for document loading and chunking, `sentence-transformers` for embeddings, `faiss-cpu` for vector storage, and `transformers` (FLAN-T5) for answer generation. The goal is to demonstrate how retrieval enhances generative question-answering compared to generic answers.

Step 1: Set Up the Environment

Install Required Libraries:

langchain: Framework for chaining NLP components.

sentence-transformers: For generating embeddings.

faiss-cpu: Efficient vector similarity search.

pypdf: PDF text extraction.

Import Libraries:

Step 2: Load and Preprocess the PDF

Load the PDF:

Split into Chunks:

chunk_size=500: Split text into 500-character segments.

chunk_overlap=50: Ensures context continuity between chunks.

Step 3: Create Embeddings and Vector Store

Initialize Embeddings:

Uses the lightweight all-MiniLM-L6-v2 model for sentence embeddings.

Build FAISS Vector Store:

FAISS enables fast similarity search for retrieval.

Step 4: Initialize the Generative Model

Load FLAN-T5 (Text-to-Text Model):

FLAN-T5 is a powerful open-source model for generative tasks.

Step 5: Implement the RAG Pipeline

Define the Query Function:

Test the Pipeline:

```
# ## Step 1: Install Required Packages
# Uninstall conflicting packages, install numpy first, and restart runtime to avoid binary incompatibilities.

#!pip uninstall -y faiss-cpu faiss-gpu numpy transformers tokenizers huggingface_hub sentence-transformers langchain langch
!pip install --upgrade pip
!pip install numpy==1.26.4 packaging>=24.1  # Pin numpy and upgrade packaging

# After restart, run installations again
!pip install faiss-cpu  # Install latest prebuilt faiss-cpu
!pip install pydantic==2.9.2 langsmith==0.1.13  # Use compatible pydantic version
!pip install sentence-transformers==2.2.2 huggingface_hub==0.23.0  # Use compatible huggingface_hub
!pip install -q langchain==0.1.13 langchain-community==0.0.29
!pip install -q transformers==4.41.1 pypdf==3.17.4 tenacity  # Add tenacity, let pip select tokenizers
```

```
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: huggingface_hub
    Found existing installation: huggingface-hub 0.34.1
    Uninstalling huggingface-hub-0.34.1:
      Successfully uninstalled huggingface-hub-0.34.1
  Attempting uninstall: tokenizers
    Found existing installation: tokenizers 0.21.2
    Uninstalling tokenizers-0.21.2:
      Successfully uninstalled tokenizers-0.21.2
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
  Attempting uninstall: transformers
    Found existing installation: transformers 4.54.0
    Uninstalling transformers-4.54.0:
      Successfully uninstalled transformers-4.54.0
  Attempting uninstall: sentence-transformers
    Found existing installation: sentence-transformers 4.1.0
    Uninstalling sentence-transformers-4.1.0:
      Successfully uninstalled sentence-transformers-4.1.0
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14/14 [sentence-transformers]
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
diffusers 0.34.0 requires huggingface-hub>=0.27.0, but you have huggingface-hub 0.23.0 which is incompatible.
gradio 5.38.2 requires huggingface-hub>=0.28.1, but you have huggingface-hub 0.23.0 which is incompatible.
datasets 4.0.0 requires huggingface-hub>=0.24.0, but you have huggingface-hub 0.23.0 which is incompatible.
peft 0.16.0 requires huggingface_hub>=0.25.0, but you have huggingface-hub 0.23.0 which is incompatible.
Successfully installed huggingface_hub-0.23.0 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 810.5/810.5 kB 24.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 54.3 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10/10 [langchain]
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
db-dtypes 1.4.3 requires packaging>=24.2.0, but you have packaging 23.2 which is incompatible.
google-cloud-bigquery 3.35.1 requires packaging>=24.2.0, but you have packaging 23.2 which is incompatible.
diffusers 0.34.0 requires huggingface-hub>=0.27.0, but you have huggingface-hub 0.23.0 which is incompatible.
gradio 5.38.2 requires huggingface-hub>=0.28.1, but you have huggingface-hub 0.23.0 which is incompatible.
datasets 4.0.0 requires huggingface-hub>=0.24.0, but you have huggingface-hub 0.23.0 which is incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is incompatible.
xarray 2025.7.1 requires packaging>=24.1, but you have packaging 23.2 which is incompatible.
peft 0.16.0 requires huggingface_hub>=0.25.0, but you have huggingface-hub 0.23.0 which is incompatible.
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 82.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2/2 [transformers]
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
peft 0.16.0 requires huggingface_hub>=0.25.0, but you have huggingface-hub 0.23.0 which is incompatible.
```

```
# # Assignment 12: Generative AI and RAG Pipeline
# **Name**: [Your Name]
# **Program**: [Your Program Details]
# **Date**: July 29, 2025

# ## Introduction
# This notebook implements a Retrieval-Augmented Generation (RAG) pipeline to process a PDF document and generate context-a
```

```
# ## Step 2: Import Libraries
# Import libraries for document loading, chunking, embeddings, vector storage, text generation, and retries.
```

```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import FAISS
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline
import os
import requests
from tenacity import retry, stop_after_attempt, wait_exponential, retry_if_exception_type

# ## Step 3: Load and Preprocess PDF
# Load the PDF and split it into chunks. Download from Google Drive if not present.

def load_and_chunk_pdf(pdf_path="document.pdf"):
    if not os.path.exists(pdf_path):
        print("Downloading document from Google Drive...")
        try:
            # Replace with your actual Google Drive file ID
            file_id = "1s-oE5l0p8j8PDYn0nC64iufbGeh14q4c"
            url = f"https://drive.google.com/uc?export=download&id={file_id}"
            session = requests.Session()
            response = session.get(url, stream=True, timeout=30)
            response.raise_for_status()

            for key, value in response.cookies.items():
                if 'download_warning' in key:
                    url = f"https://drive.google.com/uc?export=download&confirm={value}&id={file_id}"
                    response = session.get(url, stream=True, timeout=30)
                    break

            with open(pdf_path, 'wb') as f:
                for chunk in response.iter_content(chunk_size=8192):
                    if chunk:
                        f.write(chunk)
            print(f"Downloaded document saved as {pdf_path}")
        except Exception as e:
            print(f"Failed to download document: {e}")
            return None

    try:
        loader = PyPDFLoader(pdf_path)
        docs = loader.load()
        splitter = RecursiveCharacterTextSplitter(
            chunk_size=500,
            chunk_overlap=50,
            separators=["\n\n", "\n", " ", ""]
        )
        chunks = splitter.split_documents(docs)
        print(f"Loaded {len(docs)} pages, split into {len(chunks)} chunks.")
        return chunks
    except Exception as e:
        print(f"Error processing PDF: {e}")
        return None

# ## Step 4: Create Embeddings and Vector Store
# Create embeddings with retry logic for model download.

@retry(
    stop=stop_after_attempt(3),
    wait=wait_exponential(multiplier=1, min=4, max=10),
    retry=retry_if_exception_type(Exception)
)
def create_vector_store(chunks):
    if chunks is None:
        print("Error: No chunks to process. Check the PDF loading step.")
        return None
    embeddings = HuggingFaceEmbeddings(
        model_name="sentence-transformers/all-MiniLM-L6-v2",
        model_kwargs={'device': 'cpu'}
    )
    vectorstore = FAISS.from_documents(chunks, embeddings)
    print("Vector store created successfully.")
    return vectorstore

# ## Step 5: Initialize LLM
# Load FLAN-T5 with retry logic for model download.

@retry(
    stop=stop_after_attempt(3),
    wait=wait_exponential(multiplier=1, min=4, max=10),
    retry=retry_if_exception_type(Exception)
)
```

```
)
def initialize_llm(model_name="google/flan-t5-large"):
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    pipeline_obj = pipeline(
        "text2text-generation",
        model=model,
        tokenizer=tokenizer,
        device=-1  # Use CPU
    )
    print(f"Loaded {model_name} model.")
    return pipeline_obj

# ## Step 6: Implement RAG Query Function
# Retrieve relevant chunks and generate a context-aware answer.

def query_rag(question, vectorstore, llm_pipeline, k=3):
    if vectorstore is None:
        print("Error: Vector store is not initialized.")
        return None
    relevant_docs = vectorstore.as_retriever().get_relevant_documents(question)[:k]
    context = "\n".join([doc.page_content for doc in relevant_docs])

    prompt = f"""Answer the question using only the following context. Provide a clear, concise, and complete response in f

Context:
{context}

Question: {question}

Answer:"""

    response = llm_pipeline(
        prompt,
        max_new_tokens=200,
        temperature=0.9,
        top_k=50,
        top_p=0.9,
        do_sample=True
    )
    return response[0]['generated_text']

# ## Step 7: Main Execution
# Execute the pipeline and compare RAG vs. generic answers.

if __name__ == "__main__":
    # 1. Load and chunk PDF
    chunks = load_and_chunk_pdf("document.pdf")

    # 2. Create vector store
    vectorstore = create_vector_store(chunks)

    # 3. Initialize LLM
    llm_pipeline = initialize_llm()

    # 4. Test RAG pipeline
    question = "Summarize the key points of this document in a paragraph of 100 words."
    answer = query_rag(question, vectorstore, llm_pipeline)

    if answer:
        print("\n" + "="*50)
        print("Question:", question)
        print("-"*50)
        print("Answer:", answer)
        print("="*50 + "\n")

    # 5. Compare with generic answer
    generic_prompt = f"Answer the question without any specific context:\n\nQuestion: {question}\n\nAnswer:"
    generic_answer = llm_pipeline(
        generic_prompt,
        max_new_tokens=200,
        temperature=0.9,
        top_k=50,
        top_p=0.9,
        do_sample=True
    )[0]['generated_text']
    print("\n" + "="*50)
    print("Generic Answer (No Context):")
    print("-"*50)
    print("Answer:", generic_answer)
    print("="*50 + "\n")
```

```
Loaded 1 pages, split into 5 chunks.
Vector store created successfully.
Loaded google/flan-t5-large model.

==================================================
Question: Summarize the key points of this document in a paragraph of 100 words.
--------------------------------------------------
Answer: Daniel Muthama believes that LEYSCO offers a collaborative and innovative work environment. He is excited abou
==================================================


==================================================
Generic Answer (No Context):
--------------------------------------------------
Answer: There are a number of ways to make this happen:
==================================================
```

Start coding or generate with AI.