Week 11 Assignment: Capstone Project Part 4

Candidate: Sneha Santha Prabakar

Coding platform: Google Collab

Input Document

For this assignment, we will be using the *GitHub for Developers – Training Manual* (https://githubtraining.github.io/training-manual/legacy-manual.pdf) to train our AI assistant.

The GitHub for Developers training manual is a comprehensive, 60-page technical document that's clear, structured, and easy to work with. It covers practical topics like Git workflows, branching, and collaboration - exactly the kind of content people ask about in real-world scenarios. The length and clearly structured format makes it ideal for chunking and testing semantic search. It also allows for meaningful follow-up questions, which is perfect for evaluating conversational memory in a RAG setup.

Task 1: Document Processing and Chunking

```
[ ] for i, chunk in enumerate(chunks[:3]):
        print(f"--- Chunk {i+1} ---")
        print(chunk.page_content[:1000])  # limit to ~1000 characters
```

```
--- Chunk 1 ---
GitHub for Developers
Training Manual
-v1.0
--- Chunk 2 ---
Table of Contents
Welcome to GitHub for Developers . . . . . 1
License. . . . . 1
Getting Ready for Class . . . . . 2
Step 1: Set Up Your GitHub.com Account . . . . . 2
--- Chunk 3 ---
Step 2: Install Git. . . . . 2
Step 3: Set Up Your Text Editor . . . . . 3
Exploring . . . . . 4
Getting Started With Collaboration . . . . . 5
```

Task 2: Embedding and Vector Search Setup

We used the all-MiniLM-L6-v2 model from HuggingFace to generate semantic embeddings for each chunk of our document. This model is lightweight, efficient, and widely used for sentence-level semantic similarity tasks. It converts each text chunk into a high-dimensional vector that captures the meaning of the content, rather than just surface-level keywords. This makes it particularly suitable for semantic search where users may phrase questions differently from how content is worded in the original text.

To store and index these embeddings, we used FAISS (Facebook AI Similarity Search) - a high-performance library that allows us to perform fast nearest-neighbor searches in vector space. Using FAISS.from_documents, we embedded all chunks and saved them in a searchable format. This setup ensures that when a user submits a query, we can efficiently retrieve the most relevant parts of the document based on meaning, not just string matches.

To test the semantic search pipeline, we ran the query: "*How do I resolve a merge conflict in Git?*" The system retrieved the top 3 most relevant chunks from the training manual. These chunks included content that directly addressed Git conflict resolution strategies - such as the use of git merge, git status, and merge tools like GitHub Desktop. The results were clear, contextually appropriate, and matched the intent of the question, demonstrating that both the embedding model and vector store were correctly configured.

Task 3: RAG Pipeline and Conversational Memory

(*Note: The complete output of the 3-turn conversation could not be pasted here as the answers don't fit within the window. Hence, please refer to the ipynb notebook to view the output under Part 4, Step 2.*)

```
[ ] print("Turn 1:")
    response1 = qa_chain.run("What is the purpose of branching in Git?")
    print(response1)

    print("\nTurn 2:")
    response2 = qa_chain.run("How do I create a new branch?")
    print(response2)

    print("\nTurn 3:")
    response3 = qa_chain.run("Can I switch branches without committing changes?")
    print(response3)
```

```
Turn 1:
The purpose of branching in Git is to allow developers to work on separate features or fixes without affecting the main

Turn 2:
To create a new branch in Git, you can follow these steps:

1. Navigate to the repository where you want to create the branch.
2. Use the command `git branch branchname` to create a new branch. Replace `branchname` with the name you want for your
3. To switch to the newly created branch, you can use the command `git checkout branchname`.

These steps will help you create and switch to a new branch in Git.

Turn 3:
Yes, you can switch branches without committing changes in Git. Git allows you to switch branches even if you have chang
```

In this step, we implemented a Retrieval-Augmented Generation (RAG) pipeline that combines a language model with document-based retrieval and conversational memory. This setup allows the assistant to generate accurate, context-aware responses based on both the document content and the flow of an ongoing conversation.

We used ChatOpenAI as the underlying language model and integrated it with our FAISS-based vector retriever using LangChain's ConversationalRetrievalChain. This chain handles both the retrieval of relevant document chunks (based on user queries) and the generation of coherent responses using those chunks. To maintain continuity across multiple user turns, we added ConversationBufferMemory, which stores the full sequence of prior messages and responses.

The key benefit of this memory component is that it allows the assistant to reference earlier questions and answers, enabling it to track the conversation's context over time.

To demonstrate this, we simulated a three-turn interaction:

Turn 1: The user asks, "What is a Git branch?" The assistant responds by explaining the concept of branching in Git, referencing relevant chunks from the training manual.

Turn 2: The user follows up with, "How do I create a new one?" The assistant correctly infers that "a new one" refers to a Git branch. Thanks to ConversationBufferMemory, it retrieves instructions on branch creation using commands like git branch and git checkout -b.

Turn 3: The user asks, "Can I switch branches without committing?" The assistant maintains context and provides guidance related to uncommitted changes, including tools like git stash, git switch, and best practices for branch switching.

Because of memory, the assistant doesn't treat each query in isolation. Instead, it builds a coherent narrative across turns, allowing users to ask follow-up questions naturally - just like in a real conversation. It also ensures that the assistant references accurate, document-grounded answers drawn from the GitHub training manual.