Advanced Deep Learning: Generative AI

Final Project

Title:

ReviseME - LLM Based Revision Summarization Tool

Presented By:
Ghassen Zgorni - Adham Qussay - Zeina Elgendy

Presented To:

Dr. Ammar Mohammed

ESLSCA UNIVERSITY

AI & Data Science Major

Overview

ReviseME is a comprehensive revision tool built using Streamlit. The application uses advanced scraping techniques to extract content from various sources, including PDFs and URLs. It stores the scraped content in a vector database and utilizes a generative AI (Gemini API) model to provide contextual summaries and answers to user queries. Additionally, a Jupyter Notebook replicates the core functionalities of the app for ease of testing and demonstration.

Project Structure

Main Components

- Streamlit App (app.py):

The main application interface allowing users to upload PDFs, input URLs, and ask questions about the scraped content.

Handles scraping, storing, and querying functionalities.

- PDF Scraper (pdf_scraper.py):

Contains functions for extracting text from PDF files.

- URL Scraper (url_scraper.py):

Contains functions for extracting text from web pages.

- Summarizer (summarizer.py):

Utilizes the generative AI model to generate summaries and answers based on the scraped content.

- Vector Database (vector_db.py):

Manages the creation and querying of the vector database using ChromaDB and the Gemini API for embeddings.

- Jupyter Notebook (ReviseME_Demo.ipynb):

A Jupyter Notebook that replicates the core functionalities of the app for testing and demonstration purposes.

Side Note: We have also created a .env file to securely store the gemini api key, you can use your own by changing the key in the env file

App Workflow Process:

Start: User opens the Streamlit app.

Upload PDF/Input URL: User uploads a PDF file or inputs a URL.

Content Scraping:

If PDF is uploaded, scrape_pdf_info extracts content.

If URL is entered, scrape_url_info extracts content.

Store Content: Scraped content is stored in a session-specific list of documents.

Create Vector Database:

Content is processed to create embeddings using the Gemini API.

ChromaDB creates a collection and stores these embeddings.

User Query: User inputs a query in the text input. Like ‘’summarize what the introduction says’’

Retrieve Relevant Passage: The app retrieves the most relevant passage from the vector database using the query.

Generate Response:

A prompt is created using the retrieved passage and user query.

The Gemini generative model generates a summary or answer.

Display Result: The generated summary or answer is displayed to the user.

End Session: Once the session ends, the vector database is cleared, ensuring a fresh start for the next session. We however have a small bug where we need to click scrape again before asking another question from the same document because the database gets cleared.