Report: Chat with Websites (Chatbot)

Author:
Murtaza Anwaar
Intern AI Developer
MakTek

Contents

Introduction

The project is an implementation of a Streamlit application that allows users to interact with a website by inputting a URL and asking questions. The application uses the OpenAI API for language model interactions, performs data retrieval from the provided URL, and processes the data to answer user queries. The primary components of the original code include setting up the Streamlit interface, loading data from a website, creating embeddings using OpenAI, and using Chroma for vector storage and retrieval.

Original Code Overview

Components:

- **Libraries Used:** Streamlit, LangChain, OpenAI, WebBaseLoader, Chroma, and related modules.
- **Functionality:**
  - Set up a simple Streamlit interface for user input.
  - Load website data using `WebBaseLoader`.
  - Split the text into chunks using `CharacterTextSplitter`.
  - Create embeddings with `OpenAIEmbeddings`.
  - Store embeddings in a Chroma vector database.
  - Retrieve answers using `RetrievalQA` and display them in the Streamlit app.

Changes Made in the Edited Code

Modifications:

- **Library Updates:**
  - Replaced OpenAI model with Google Gemini API.
  - Introduced `BeautifulSoup` for web scraping.
  - Added OCR functionality using `pytesseract` to handle CAPTCHA.
  - Implemented logging for better traceability.
- **Functionality Enhancements:**
  - Improved user input handling with sidebars in Streamlit.
  - Implemented CAPTCHA solving mechanism.
  - Modified text splitting logic to handle larger chunks of text.
  - Switched from `OpenAIEmbeddings` to `GooglePalmEmbeddings`.
  - Enhanced chat interface with persistent chat history.
  - Used session state in Streamlit to manage persistent states.
- **Modularization of Code:**
  - Refactored the original code base to be more modular by adding functions to enhance reusability of code.

Improvements Achieved

1. **Improved User Interface:**
    - Added sidebars for input fields to restrict the user from adding a prompt before adding the API key and URL, enhancing user experience and ensuring robustness of the application.
    - Implemented persistent chat history using Streamlit's session state.
2. **Logging and Error Handling:**
    - Integrated logging for better error tracking and debugging.
    - Added detailed error messages for failed web requests and CAPTCHA handling.
3. **Persistent Storage Management:**
    - Enhanced handling of vector databases using session state in Streamlit.
    - Improved text splitting for handling larger chunks of text efficiently.

Issues

- Issues regarding the persistent database; the database has to be deleted before every new session to avoid any errors.

Further Improvements

1. **Chat History:**
    - Add mechanism to continuously feed the chat history to the vector space to maintain contextual consistency.
2. **User Interface Enhancements:**
    - Improve the chat interface with more interactive elements and better styling.
    - Provide options for users to upload documents or images for analysis directly.
3. **Comprehensive Testing:**
    - Add unit and integration tests to ensure the reliability and stability of the application.
    - Conduct user testing to gather feedback and further refine the user experience.
