Documentation for Chatbot Project

By

V. Bhuvanesh – CB.SC.U4AIE23259

M. Kaushal – CB.SC.U4AIE23145

U. Sandeep – CB.SC.U4AIE23257

Introduction

This chatbot project was developed for the CodeBot 2.0 hackathon with the aim to build a chatbot that can answer user queries related to sustainable living practices and provide users with helpful resources. The chatbot integrates external APIs to deliver information on carbon emissions, weather, and renewable energy.

Problem Statement

Many individuals are interested in adopting more sustainable lifestyles, but they struggle to find reliable information and resources to make impactful changes. This chatbot aims to provide users with easy access to information on topics like carbon emissions, weather quality, and renewable energy to guide them in making more eco-conscious decisions.

Solution Overview

The chatbot uses a combination of **Large Language Models (LLMs)** and external APIs to answer user queries and provide useful, contextdriven insights related to sustainability. Users can ask about carbon data, local weather, or renewable energy resources, and the chatbot will provide real-time responses based on external data sources.

LLM Model Used

- The project uses **Groq's Llama 3** model to generate conversational responses.
- **Why Llama 3**: This model was chosen for its ability to generate fluent, coherent responses while maintaining context through the conversation.

Data

- **Carbon Interface API**: Used to retrieve carbon emissions data.
- **OpenAQ API**: Provides current weather data and air quality information based on user-specified locations.
- **National Renewable Energy Laboratory (NREL) API**: Used to fetch renewable energy data.

Key Features

1. **Chat Interface**: The chatbot allows users to type queries related to sustainable living.
2. **LLM Integration**: The bot uses the Llama 3 model for generating natural responses.
3. **External Data Sources**:
   - **Carbon Data**: Provides information on carbon emissions.
   - **Weather Data**: Retrieves air quality and weather information from OpenAQ API.
   - **Energy Data**: Fetches renewable energy information from NREL API.

4. **Contextual Responses**: The bot retrieves relevant data and context based on the user’s latest query, thanks to its Pinecone integration.
5. **Good Humor**: tipping the model with good humor.

UI/UX

- The chatbot is implemented using **Streamlit**, which provides a clean, user-friendly interface for interactions.
- **Input**: Users can ask questions directly through a text box.
- **Output**: Responses are displayed in real-time in a chat-like format. User queries and responses are stored and displayed as a conversation history.

Technical Details

- **Tech Stack**:
  - **Front-end**: Streamlit
  - **Back-end**: Groq Llama 3, Pinecone, Python
  - **APIs**: Carbon Interface, OpenWeather, NREL
- **How it Works**:
  1. **User Input**: The user inputs a question through the chat interface.
  2. **Query Processing**: The input is processed and sent to Pinecone for embedding generation. This vector is then queried against the Pinecone index for relevant context.
  3. **External API Call**: If necessary, the chatbot uses external APIs (Carbon, Weather, or Energy) to gather additional data.

    4. **LLM Response**: The processed query, along with any retrieved context, is passed to the Groq Llama 3 model, which generates a conversational response.
    5. **Output**: The response is displayed to the user in the chat interface.
- **External APIs**:
    - **Carbon Interface**: Retrieves carbon emission estimates based on specific parameters.
    - **OpenAQ API**: Returns air quality data based on user input (city name).
    - **NREL API**: Fetches data on renewable energy initiatives and statistics.

Challenges

- **Latency in API calls**: One of the challenges was dealing with slower response times from external APIs. This affected the overall responsiveness of the chatbot, and efforts were made to minimize these delays using efficient query handling.
- **Embedding accuracy**: Ensuring that relevant context is retrieved based on user queries was a challenge and required fine-tuning of the Pinecone index and query logic.

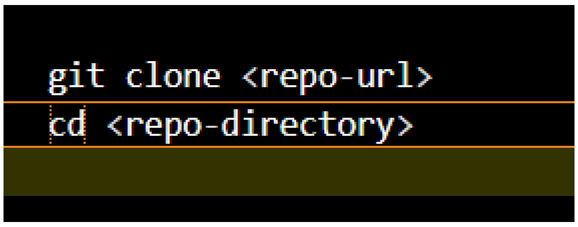
Future Improvements

1. **Expand API Integration**: Add more data sources and APIs to provide a broader range of information on sustainability (e.g., water conservation, waste management).
2. **Personalization**: Implement user-specific recommendations and tips for a sustainable lifestyle based on location, preferences, and past interactions.
3. **Enhanced User Interface**: Improve the UI with more interactivity, such as clickable suggestions or visual data representations (charts, maps).

Installation Guide

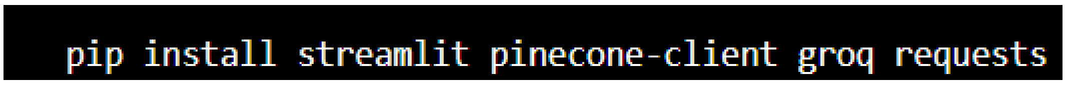
To run this chatbot on your local machine, follow these steps:

1. **Clone the Repository**:



```
git clone <repo-url>
cd <repo-directory>
```

2. **Install Dependencies**:

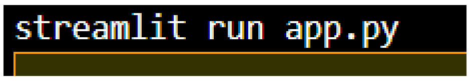


```
pip install streamlit pinecone-client groq requests
```

3. **Set Up API Keys**:

Update the API keys for **Carbon Interface**, **OpenWeather**, and **NREL** in the code.

4. **Run the Application**:



```
streamlit run app.py
```

5. **Access the Application**:

Once the Streamlit server is running, open the app in your browser to interact with the chatbot.

This documentation provides a complete overview of the project, including the problem it solves, the technologies used, and instructions for running it.