E-COMMERCE AI CHATBOT PROTOTYPE

DOCUMENTATION

GURJOT SINGH AULAKH

TABLE OF CONTENTS

APPROACH 2

TECHNOLOGIES USED 2

IMPLEMENTATION STEPS 2

CHALLENGES FACED 3

CONCLUSION 3

STEP-BY-STEP GUIDE TO RUN AND TEST THE E-COMMERCE CHATBOT........ 4

Approach

The objective of this project was to create a basic AI chatbot prototype for an e-commerce domain using web technologies. The chatbot needed to handle greetings, respond to frequently asked questions (FAQs), and have a modern, user-friendly interface. Here's a breakdown of the approach taken:

Technologies Used

- Frontend:
    - HTML, CSS and JavaScript (to fetch API for making asynchronous requests).
- Backend:
    - Python (Flask framework for building the server-side application), spaCy for natural language processing (NLP) to understand user queries.
- Data Handling:
    - JSON for storing FAQs and their corresponding answers.

Implementation Steps

- Frontend:
    - Designed a modern chatbot interface using HTML and styled it with Tailwind CSS for a clean, responsive layout.
    - Implemented JavaScript to handle user interactions, send user queries to the backend, and display responses dynamically without page reloads.
- Backend:
    - Developed a Flask server to handle POST requests from the frontend.
    - Used spaCy with the `en_core_web_md` model for NLP to process user queries and find the most relevant answer from the FAQs.
    - Managed different types of user inputs including greetings, questions, and fallback responses for unrecognized queries.

Challenges Faced

- Integration of NLP:
    - Configuring and integrating spaCy for NLP required understanding the model's capabilities and limitations, especially in identifying similarity between user queries and FAQ questions.
- Frontend-Backend Communication:
    - Ensuring smooth communication between the frontend and backend, handling asynchronous requests properly to display responses in real-time.

Conclusion

This chatbot prototype successfully demonstrates integration of frontend and backend technologies to create a responsive, interactive user interface powered by natural language processing. Challenges in NLP integration, frontend-backend communication, and UX design were addressed to deliver a modern and functional e-commerce chatbot. Further enhancements could include handling more complex queries, adding multi-turn dialogues, and improving the overall user experience based on feedback and usage analytics.

Github link: https://github.com/GurjotSinghAulakh/AI-Chatbot-Prototype

Step-by-Step Guide to Run and Test the E-commerce Chatbot

The documentation can be found in the README.md file within the project files and on GitHub (Images included).

Step 1: Extract the Zip File

1. Extract the zip file:
   - Locate the zip file you received.
   - Extract the contents of the zip file to a directory of your choice.

Step 2: Set Up the Python Environment

2. Navigate to the project directory:
   - Open a terminal or command prompt.
   - Change the directory to the location where you extracted the zip file.

     `cd path/to/extracted/directory`

3. Create and activate a virtual environment:
   - Create a virtual environment:

     `python -m venv venv`

   - Activate the virtual environment:
     - On macOS and Linux:

       `source venv/bin/activate`

     - On Windows:

       `venv\Scripts\activate`

4. Install the dependencies:
   - Ensure you have a requirements.txt file in the project directory with the following content:

Flask==2.1.1

Flask-Cors==3.0.10

spacy==3.2.3

- Install the dependencies listed in the requirements.txt file:

`pip install -r requirements.txt`

5. Download the spaCy model:
   - Download the en_core_web_md model for spaCy:

`python -m spacy download en_core_web_md`

Step 3: Verify the Backend Setup

6. Ensure you have the following files in the project directory:

`app.py`

`faqs.json`

Step 4: Set Up the Frontend

7. Ensure you have the following frontend files:

`index.html`

`styles.css`

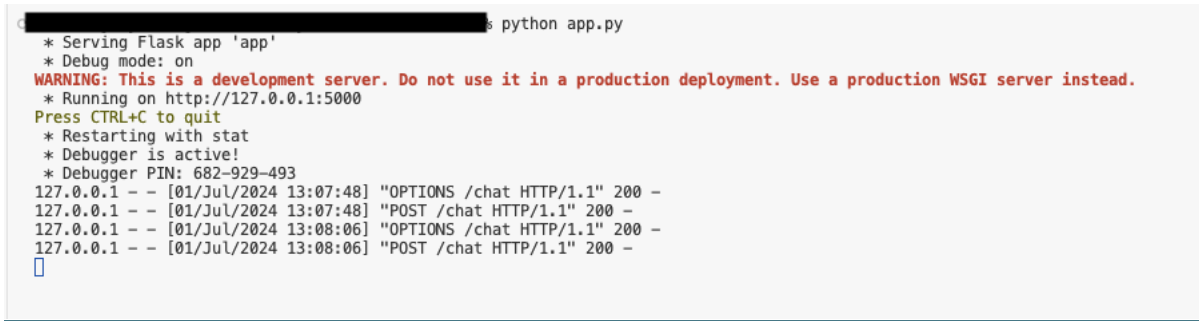
`script.js`

Step 5: Run the Backend Server

8. Start the Flash server:

`python app.py` or `python3 app.py`

The server should start running at ***http://127.0.0.1:5000***



```
[redacted] python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 682-929-493
127.0.0.1 - - [01/Jul/2024 13:07:48] "OPTIONS /chat HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2024 13:07:48] "POST /chat HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2024 13:08:06] "OPTIONS /chat HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2024 13:08:06] "POST /chat HTTP/1.1" 200 -
```

Step 6: Test the Chatbot

9. Open the `index.html` file in your web browser.

10. Interact with the chatbot:
    a. Type a message in the input field and *press Enter* or *click the send button*.
    b. The chatbot should respond to your queries based on the predefined FAQs in `faqs.json`

Contact Information for Assistance

If you encounter any issues or need further assistance, please feel free to contact:

Name: Gurjot Singh Aulakh

Email: gurjot.singh.aulakh28@gmail.com