E-COMMERCE A.I CHATBOT DOCUMENTATION

-BY DALMEET SINGH

TABLE OF CONTENTS

Objective

Make a simple AI chatbot prototype that can respond to users' questions about e-commerce orders or customer service. In addition to answering frequently asked questions (FAQs), the chatbot must handle greetings and have a contemporary, user-friendly interface.

Technology

- Frontend:

  HTML file will provide the user interface for interacting with the chatbot. The added JavaScript will handle sending user messages to the Flask server and updating the chat window with responses.
- 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

- Frontend: Designed a modern chatbot interface using HTML with Javascript.
- Backend: Developed a Flask server to manage front-end POST requests. NLP was performed using spaCy and the {en_core_web_md} model to process user inquiries and identify the most pertinent response from the FAQs. Oversaw various user input formats, such as salutations, inquiries, and backup answers for unidentified queries.

Challenges

- Encountering issues due to a version mismatch between numpy and one of its dependencies, causing binary incompatibility. Also, between spacy and pydantic packages since spacy 3.4.0 does not support pydantic versions >=1.10.0.

- Integration of NLP, Understanding the model's strengths and weaknesses was necessary for configuring and integrating spaCy for NLP, particularly when it came to determining how similar user queries and FAQ questions were to one another.

- The nc command output indicating Connection refused suggests that the port 5003 is not open or that no service is listening on that port. Encountering a 404 Not Found error for the route. Verifying connecting to the port 5003 using *curl -X POST http://127.0.0.1:5003/chat -H "Content-Type: application/json" -d '{"message": "Hello"}'*.

Conclusion

In a SaaS framework, this project successfully created a basic AI chatbot prototype for an e-commerce helpdesk, showcasing important features like greeting users and responding to frequently asked questions. The chatbot was implemented as a web application with a contemporary HTML interface, and it accessed FAQs contained in a JSON file using Python, Flask, and spaCy for natural language processing. It describes future improvements for SaaS alignment, such as multi-tenancy, scalability, subscription management, and secure authentication, even though the current implementation satisfies the fundamental requirements. Through the integration of these components, the project creates a strong basis for a multi-tenant, scalable, and user-friendly SaaS-based chatbot solution that will improve e-commerce customer service.

Steps to Run E-Commerce A.I Chatbot

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 Virtual 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 Windows: `venv\Scripts\activate`

     On macOS and Linux: `source venv/bin/activate`

4. Install the dependencies:
   - Ensure you have a requirements.txt file in the project directory with some mandatory contents like:

     ```
     Flask==3.0.3
     spacy==3.7.5 & more ……
     ```

   - Install all 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:

   chatbot.py
   faqs.json

Step 4: Set Up the Frontend

7. Ensure you have the following frontend files:

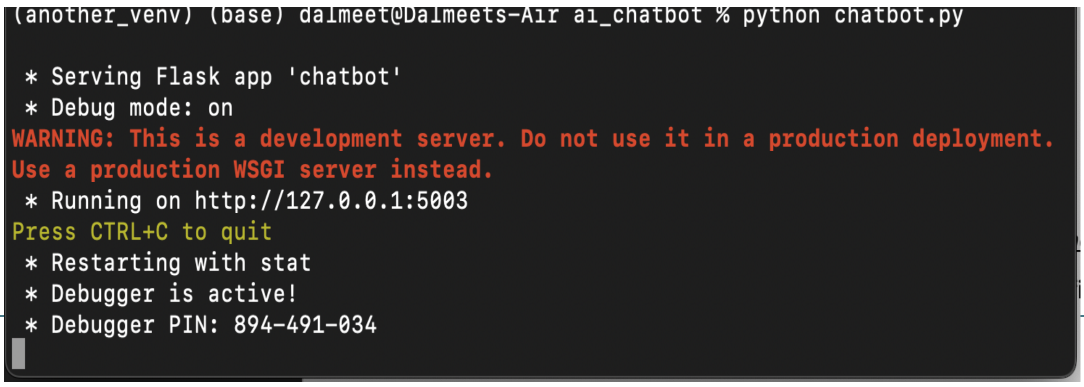
   index.html

Step 5: Run the Backend Server

8. Start the Flash server:

   python chatbot.py or *python3 chatbot.py*

   The server should start running at *http://127.0.0.1:5003/*



```
(another_venv) (base) dalmeet@Dalmeets-Air ai_chatbot % python chatbot.py

 * Serving Flask app 'chatbot'
 * 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://127.0.0.1:5003
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 894-491-034
```

Step 6: Prepare for Deployment

9. For production environment, using a production-grade WSGI server gunicorn.

   pip install gunicorn

10. Run gunicorn

   gunicorn --bind 0.0.0.0:5003 chatbot:app

Step 7: Test the A.I Chatbot

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

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