Shopping_Assistance_2_GenAI

1. Background:

In today's digital age, online shopping has become the go-to option for many consumers. However, the overwhelming number of choices and the lack of personalized assistance can make the shopping experience daunting. To address this, we have developed ShopAssist AI, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable information delivery.

2. Problem Statement

Given a dataset containing information about laptops (product names, specifications, descriptions, etc.), build a chatbot that parses the dataset and provides accurate laptop recommendations based on user requirements.

3. Approach

- Conversation and Information Gathering: The chatbot will utilize language models to understand and generate natural responses. Through a conversational flow, it will ask relevant questions to gather information about the user's requirements.
  The chatbot should ask a series of questions to
  - Determine the user's requirements. For simplicity, we have used 6 features to encapsulate the user's needs. The 6 features are as follows:
    - GPU intensity
    - Display quality
    - Portability
    - Multitasking
    - Processing speed
    - Budget
- Information Extraction: Once the essential information is collected, rule-based functions come into play, extracting the top 3 laptops that best match the user's needs.
- Personalized Recommendation: Leveraging this extracted information, the chatbot engages in further dialogue with the user, efficiently addressing their queries and aiding them in finding the perfect laptop solution.

4. System Functionalities

- User Interface: ShopAssistAI provides a user-friendly web interface where users can interact with the conversational AI assistant.
- Conversational AI: The core of ShopAssistAI is the conversational AI powered by OpenAI's chat model. It guides the user through the process by asking relevant questions and understanding their needs.
- User Input Moderation: User input is moderated using OpenAI's moderation API to ensure a safe and secure conversation.
- User Profile Extraction: The AI assistant extracts key information from the conversation to build a user profile that reflects their laptop preferences (budget, screen size, processing power, etc.) using OpenAI's function calling mechanism to convert a user requirement string into a JSON object.

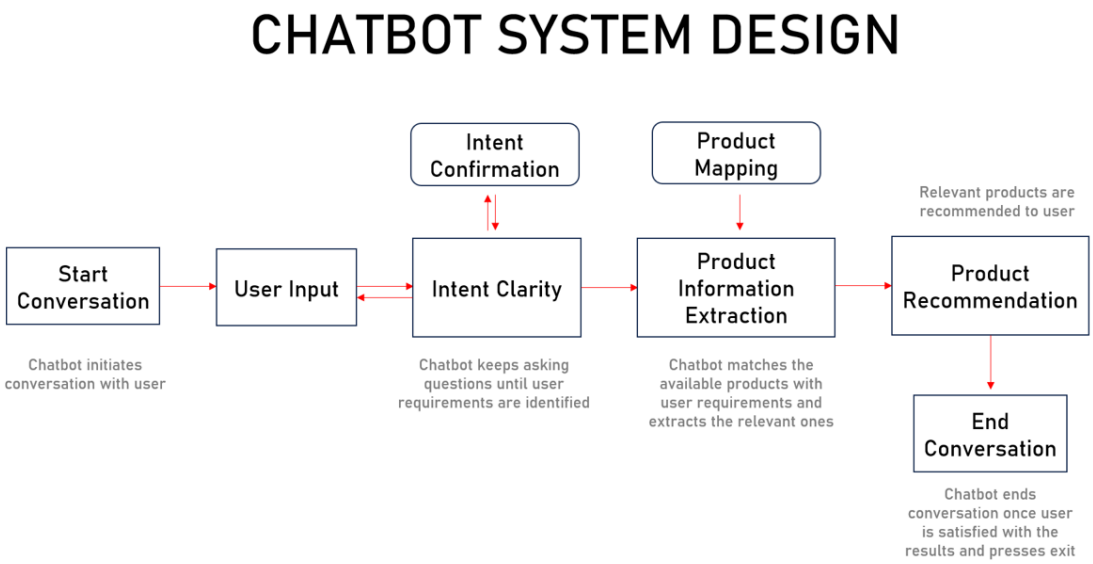
We have a dataset **`laptop_data.csv`** where each row describes the features of a single laptop and also has a small description at the end. The chatbot will leverage large language models to parse this **`Description`** column and provide recommendations.

5. System Architecture

ShopAssistAI follows a client-server architecture. Users interact with the web interface hosted on a server running the Flask application. The application interacts with OpenAI's API for conversation generation and moderation and retrieves and compares laptop data from an external database.

Stages:

System Design:



6. Implementation Details

The Flask application utilizes various functionalities:

- Routing: Maps user requests to appropriate functions based on URLs.
- Conversation Management: Handles conversation initiation, response generation through OpenAI's chat model, and conversation history maintenance.
- User Input Processing: Captures user input, performs moderation checks, and extracts user profiles from conversation history (converting user input string to JSON using OpenAI Function calling).
- Recommendation Logic: Compares user profiles with laptop data, validates recommendations, and generates recommendation text.

Major Functions:

1. Intent Clarity and Intent Confirmation Layers:
    a. initialize_conversation() : This initializes the variable conversation with the system message. Using prompt engineering and chain of thought reasoning, the function will enable the chatbot to keep asking questions until the user requirements have been captured in a dictionary. It also includes Few

Shot Prompting (sample conversation between the user and assistant) to align the model about user and assistant responses at each step.

b. **get_chat_model_completions()** : The function encapsulates the chat completions API call to Open AI

c. **get_chat_completions_func_calling()** : The function-calling API in OpenAI models allows for a structured interaction where the AI can extract and return laptop requirements in a defined format. Instead of generating unstructured text responses, the API enables the model to provide arguments for pre-defined functions related to laptop preferences. This is particularly useful for creating structured outputs, helping users find laptops that match their needs, or integrating AI responses into applications for personalized recommendations.

d. **intent_confirmation_layer()** : Evaluates if the chatbot has captured the user's profile clearly.

e. **moderation_check()** : Checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, it ends the conversation.

2. **Product Mapping and Information Extraction Layers:**
   a. **product_map_layer() :** This function is responsible for extracting key features and criteria from laptop descriptions.
   b. **compare_laptops_with_user()** : This function compares the user's profile with the different laptops and come back with the top recommendations
3. **Product Recommendation Layer:**
   a. **initialize_conv_reco()** : It takes the output from the compare_laptops_with_user function in the previous layer and provides the recommendations to the user.
4. **Dialogue Management ChatBot:** It's done in Flask to build the User Interactive Application

Prerequisites

- Python 3.7+
- OpenAI API Key(you have to add openai api key in the empty txt file (OpenAI_API_Key))

Getting Started

To get started with ShopAssist AI, follow these steps:

1. Clone the repository:

```
git clone https://github.com/Smrutirekha90/Shopping_Assistance_2_GenAI.git
cd Shopping_Assistance_2_GenAI
```

2. Lunch VS Code from Anaconda

- In VS Code go to `File` > `Open Folder...` and select the `Shopping_Assistance_2_GenAI` folder.
- Open a terminal in VS Code (``Ctrl+` `` or go to `Terminal` > `New Terminal`).

3. Enable the virtual environment:

- Enabled Virtula Env in VSCode

4. Install dependencies in Virtual Env:

```
pip install -r requirements.txt
```

5. Initialize the ChatBot:

```
flask run
```

Note: This version includes steps to create and activate the Conda environment with Python 3.11.9 0r above, ensuring users set up python environment correctly before installing dependencies and running the application.

7. Appendix - B

User output example screenshot: