TravelAssistAI

Part 1: Introduction

Project Background

In today's busy and fast paced life, everyone wants a quick escape every now and then in form of a holiday. However, the overwhelming number of choices and the lack of personalized assistance can make the holiday planning daunting. To address this, we have developed TravelAssist AI, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable information delivery.

Problem Statement

Given a dataset containing information about holiday packages (Package name, destination, No. of days, Sighseeing etc.), build a chatbot that parses the dataset and provides accurate holiday recommendations based on user requirements.

Dataset

Data is gathered form Kaggle's make my trip dataset on holiday package from the follwing link: 'https://www.kaggle.com/datasets/promptcloud/travel-listing-from-makemytrip'

Original dataset contained 30k rows, however for this problem statement this has been trimmed down to just 4 rows **to** overcome timeout issues with Open AI

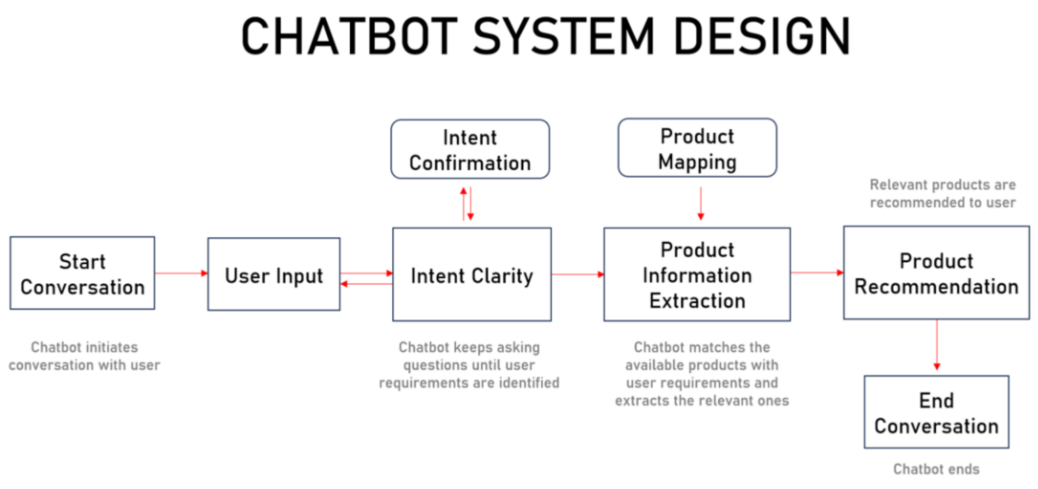
Approach:

1. **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.

2. **Information Extraction**: Once the essential information is collected, rule-based functions come into play, extracting relevant holiday packages that best matches the user's needs.

3. **Personalized Recommendation**: Leveraging this extracted information, the chatbot engages in further dialogue with the user, efficiently addressing their queries and aiding them in answering any question related to the package.

Part 2: System Design



CHATBOT SYSTEM DESIGN
Intent Confirmation
Product Mapping
Relevant products are recommended to user
Start Conversation
User Input
Intent Clarity
Product Information Extraction
Product Recommendation
Chatbot initiates conversation with user
Chatbot keeps asking questions until user requirements are identified
Chatbot matches the available products with user requirements and extracts the relevant ones
End Conversation
Chatbot ends

As shown in the image, the chatbot contains the following layers:

- Intent Clarity Layer
- Intent Confirmation Layer
- Product Mapping Layer
- Product Information Extraction Layer
- Product Recommendation Layer

Major functions behind the Chatbot

Let's now look at a brief overview of the major functions that form the chatbot.

- **initialize_conversation():** This initializes the variable conversation with the system message.

- **get_chat_model_completions():** This takes the ongoing conversation as the input and returns the response by the assistant

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

- **`intent_confirmation_layer()`:** This function takes the assistant's response and evaluates if the chatbot has captured the user's profile clearly. Specifically, this checks if the following properties for the user has been captured or not 'Destination','Package','Origin','Duration','Budget'

- **`dictionary_present()`:** This function checks if the final understanding of user's profile is returned by the chatbot as a python dictionary or not. If there is a dictionary, it extracts the information as a Python dictionary.

- **`compare_holiday_with_user()`:** This function compares the user's requirements with the different holiday packages and come back with the top recommendations.

- **`initialize_conv_reco()`:** Initializes the recommendations conversation

Part 3: Implementation

Implementing Intent Clarity & Intent Confirmation layers

Let's start with the first part of the implementation - building the `intent clarity` and `intent confirmation` layers. As mentioned earlier, this layer helps in identifying the user requirements and passing it on to the product matching layer. Here are the functions that we would be using for building these layers:

- **`initialize_conversation()`:** This initializes the variable conversation with the system message. used 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.

- `**intent_confirmation_layer**()`: This function takes the assistant's response and evaluates if the chatbot has captured the user's profile clearly. Specifically, this checks if the following properties for the user has been captured or not

  - Destination

  - Package

  - Origin

  - Duration

- Budget

Implementing dictionary present & moderation checks

- `**dictionary_present()**`**:** This function checks if the final understanding of user's profile is returned by the chatbot is a Python dictionary or not. This is important as it'll be used later on for finding the right holiday packages using dictionary matching.

- `**moderation_check()**`**:** This checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, one can add a break statement to end the conversation.

Implementing Product mapping and information extraction layer

In this section, we take in the output of the previous layers, i.e. the user requirements, which is in the format of a Python dictionary, and extract the relevant holiday recommendations based on that. Here are the functions that we will use to help us implement the information extraction and product matching layers

- `**product_map_layer()**`**:** This function is responsible for extracting key features and criteria from holiday package dataset. Here's a breakdown of how it works:

  - Uses a prompt that assign it the role of a holiday expert, whose objective is to extract key features as per requirements

  - Assign specific rules for each feature.

  - Includes Few Shot Prompting(sample conversation between the user and assistant) to demonstrate the expected result of the feature extraction.

- `**extract_dictionary_from_string()**`**:** This function takes in the output of the previous layer and extracts the user requirements dictionary

- `**compare_holiday_with_user()**`**:** This function compares the user's profile with the different holiday packages and come back with the top recommendations. It will perform the following steps:

  - It will take the user requirements dictionary as input

- Filter the holidays based on their per person budget, keeping only the ones within the user's budget.

- Match the user requirement with holiday data.

- Return the matching holidays as a JSON-formatted string.

Product Recommendation Layer

Finally, we come to the product recommendation layer. It takes the output from the `compare_holiday_with_user` function in the previous layer and provides the recommendations to the user. It has the following steps.

1. Initialize the conversation for recommendation.

2. Generate the recommendations and display in a presentable format.

3. Ask questions basis the recommendations.

Dialogue management System

Bringing everything together, we create a **`diagloue_mgmt_system()`** function that contains the logic of how the different layers would interact with each other. This will be the function that we'll call to initiate the chatbot

User Interface

Finally all the functions above are encompassed together into a UI using the Flask framework. This helps in seamless interaction between the bot and the user.

Chatbot Functionalities , Limitations & Challenges

- Chatbot can cater to irrelevant requests from the users apart from travel assistance
- It can cater to offensive/prohibited content generated from both user and assistant through moderation checks
- Data has a limitation that Origin/Start City is only New Delhi or Mumbai, hence the chatbot asks the question for origin giving only these cities as options. However it can handle the scenario if the user enters any other city than Delhi or Mumbai.
- It can handle a scenario wherein user enters the per person budget lower than the minimum value (6500 INR) in the dataset

- One limitation of the chatbot is that it looks for the exact value entered by the user in the database and returns if there are matching rows in the data. It iss unable to recommend any alternatives to keep the user engaged.
- Since the dataset contain only four rows, the positive scenario wherein chatbot recommends a holiday to a user are only 4 when the user enters correct values as present in the dataset. If we want to test one of the positive scenario, we can enter following values as a user:
  Destination: Gangtok
  Duration: 7 nights
  Package: Deluxe
  Origin: New Delhi
  Budget: 36000 INR
- When going through a positive scenario as above, once the user provides all the details and the chatbot replies “Thank you for providing all the information. Kindly wait, while I fetch the holidays”, user again needs to enter ‘ok’ in order to fetch the results.
- Chatbot can handle a scenario wherein user provides the requirements that do not match with the dataset and replies appropriately.
- One particular challenge faced was when I tried to increase the dataset size, the OpenAI timeout error was quite frequent and hence had to use only 4 rows.
- Sometimes the bot still times out when trying to fetch the results, therefore please try again should you face any such errors.