Documentation of ShopAssist AI

Contributor: Ayushi Pitchika, ACP Gen AI C10, UpGrad

Objective

The goal of this project is to enhance ShopAssist AI by leveraging the newly introduced 'function calling' feature. This will enable us to enhance the customer experience by creating a more streamlined and efficient chatbot, gaining valuable experience in conversational AI while contributing to the development of innovative AI solutions. The key tasks include:

1. **Integrate Function Calling API:** Update the existing architecture to incorporate the Function Calling API, improving performance. Identify which layers can be removed or updated to align with the new approach.

2. **Refine Conversation Flow:** Improve the chatbot's conversation flow using function calling to make interactions with users more natural and dynamic.

Project 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.

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.

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 top 3 laptops 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 finding the perfect laptop solution.

System Design

Dataset: We have a dataset laptop.csv where each row describes the features of a single laptop and also has a small description at the end. The chatbot that we build will leverage LLMs to parse this Description column and provide recommendations.

Here's the overall flow of conversation for the ShopAssist Chatbot:

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:

1. GPU intensity
2. Display quality
3. Portability
4. Multitasking
5. Processing speed
6. Budget

Confirm if the user's requirements have been correctly captured at the end.

After that the chatbot lists down the top 3 products that are the most relevant and engages in further conversation to help the user find the best one.

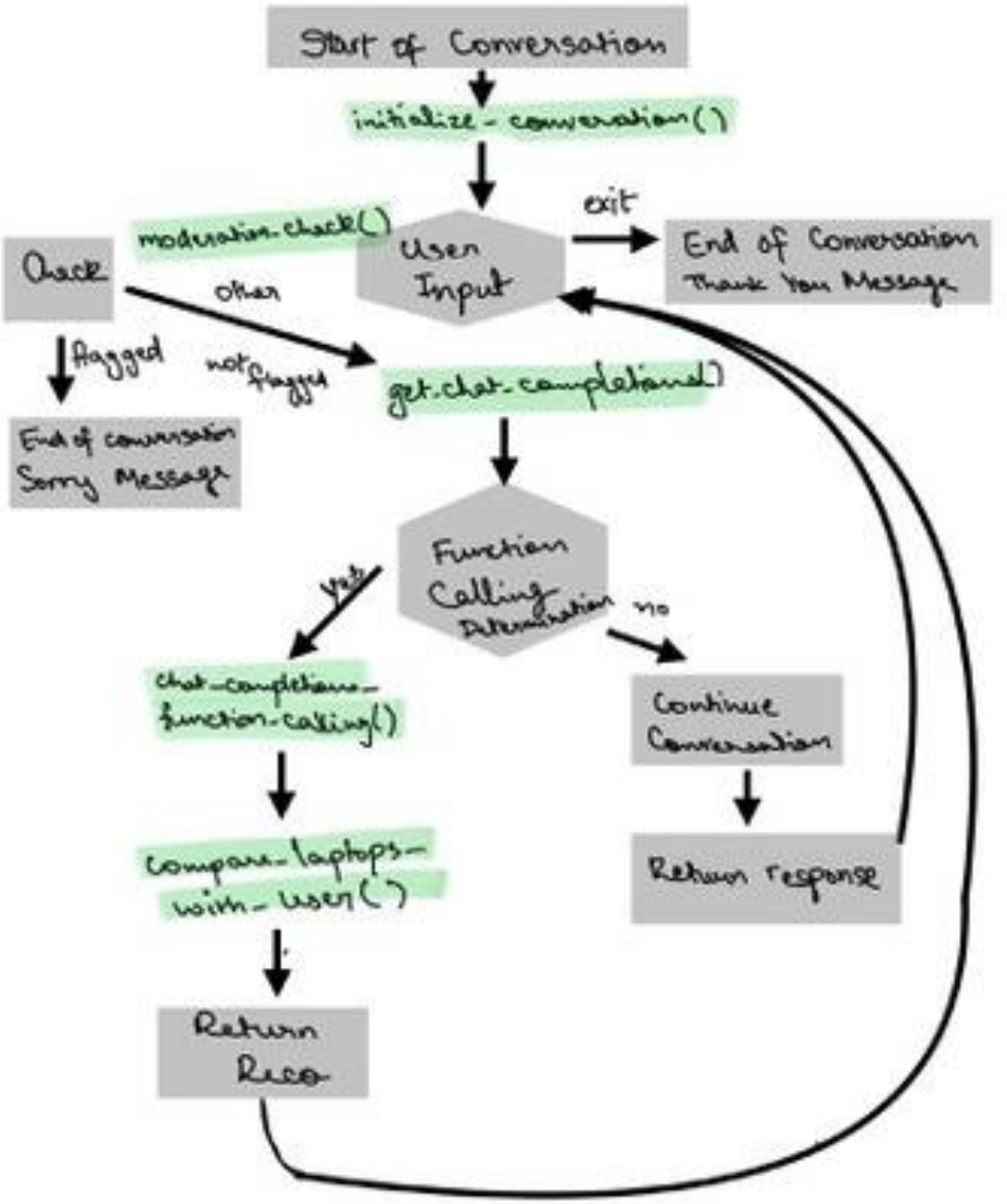
Building the Chatbot

There are three stages of the chatbot, which are as follows:

- Stage 1: Intent Clarity and Intent Confirmation
- Stage 2: Product Extraction and Product Mapping
- Stage 3: Product Recommendation

Implementation

Some layers were removed while the ShopAssist AI was adapted for Function Calling as they became redundant like the intent_confiirmation_layer, the dictionary_present layer, extract_dictionary_from_string and the initialize_conv_reco layer.



Stage 1:

Let's start with the first part of the implementation - building the intent clarity. 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. 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.

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

- get_chat_completions_function_calling(): Similar to get_chat_completions(), but includes the compare_laptops_with_user function defined as a function call. This ensures that if a function call is needed, it generates the proper JSON.

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

You need to understand the user's profile, which essentially means that all the features: GPU intensity, Display quality, Portability, Multitasking, Processing speed, Budget are captured or not.

Stage 2:

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 top 3 laptop 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 laptop descriptions. Here's a breakdown of how it works:

- Uses a prompt that assign it the role of a Laptop Specifications Classifier, whose objective is to extract key features and classify them based on laptop descriptions. Provide step-by-step instructions for extracting laptop features from description.
- Assign specific rules for each feature (e.g., GPU Intensity, Display Quality, Portability, Multitasking, Processing Speed) and associate them with the appropriate classification value (Low, Medium, or High).
- Includes Few Shot Prompting(sample conversation between the user and assistant) to demonstrate the expected result of the feature extraction and classification process.

• compare_laptops_with_user(): This function compares the user's profile with the different laptops and come back with the top recommendations. It will perform the following steps:

- It will take the user requirements dictionary as input Filter the laptops based on their price, keeping only the ones within the user's budget.
- Calculate a score for each laptop based on how well it matches the user's requirements.
- Sort the laptops based on their scores in descending order.
- Return the top 3 laptops as a JSON-formatted string.

Stage 3:

Finally, we come to the product recommendation layer. It takes the output from the compare_laptops_with_user function in the previous layer and provides the recommendations to the user. This is where Function Calling has been integrated into the code. It has the following steps.

1. Initialize the conversation for recommendation.
2. Generate the recommendations and display in a presentable format.
3. Answer questions based 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

Challenges

1. Ensuring increased accuracy in responses by prompt engineering.
2. Avoiding overwhelming the user with questions by improving the system prompt. It was a challenge to make the AI assistant ask questions about all the parameters at once.
3. Navigating the complexity of response handling as a result of function calling.

Future Scope of Work

1. Integrate real-time price and availability data from online sources like the Open Product Database to address user inquiries about purchase locations and the latest pricing information.
2. Enhance the dataset with a more comprehensive collection of laptops.
3. Conduct thorough testing, including edge case scenarios, and utilize AI to identify potential weaknesses during the testing process.
4. Develop a user-friendly interface and host the chatbot for wider accessibility.