Design Documentation of ShopAssist AI (ShopAssist 2.0)

Author: Mr. SHAMEER SHEIK, ML C56 batch, IIIT-Bangalore, UpGrad

Objective

Objective is to enhance ShopAssist AI, taking it to the next level as ShopAssist 2.0. In this we have to leverage the newly introduced 'function calling' feature to create a more streamlined and efficient chatbot to gain valuable experience in conversational AI and contribute to the development of innovative AI solutions.

Here are the tasks involved:

- **Integrate Function Calling API:** Modify the existing architecture to leverage the Function Calling API's capabilities for improved performance. Scope out which layers can be removed and how the existing layers can be updated to handle the new approach.
- **Refine Conversation Flow:** Enhance the chatbot's conversation flow by leveraging function calling. Make the interaction between the user and the chatbot 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

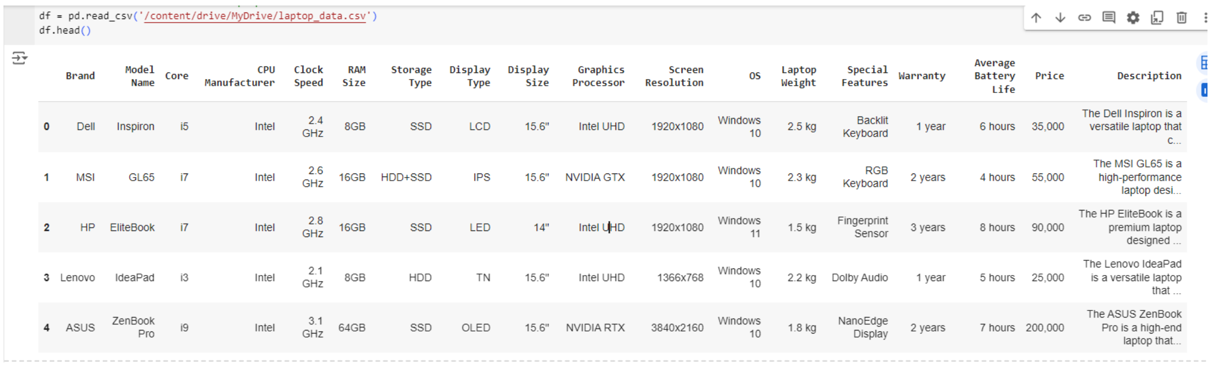
Given a dataset containing information about laptops (product names, specifications, descriptions, etc.), build a chatbot that parses the dataset and provides accurate laptop recommendations based on user requirements.

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.



```
df = pd.read_csv('/content/drive/MyDrive/laptop_data.csv')
df.head()
```

	Brand	Model Name	Core	CPU Manufacturer	Clock Speed	RAM Size	Storage Type	Display Type	Display Size	Graphics Processor	Screen Resolution	OS	Laptop Weight	Special Features	Warranty	Average Battery Life	Price	Description
0	Dell	Inspiron	i5	Intel	2.4 GHz	8GB	SSD	LCD	15.6"	Intel UHD	1920x1080	Windows 10	2.5 kg	Backlit Keyboard	1 year	6 hours	35,000	The Dell Inspiron is a versatile laptop that c...
1	MSI	GL65	i7	Intel	2.6 GHz	16GB	HDD+SSD	IPS	15.6"	NVIDIA GTX	1920x1080	Windows 10	2.3 kg	RGB Keyboard	2 years	4 hours	55,000	The MSI GL65 is a high-performance laptop desi...
2	HP	EliteBook	i7	Intel	2.8 GHz	16GB	SSD	LED	14"	Intel UHD	1920x1080	Windows 11	1.5 kg	Fingerprint Sensor	3 years	8 hours	90,000	The HP EliteBook is a premium laptop designed ...
3	Lenovo	IdeaPad	i3	Intel	2.1 GHz	8GB	HDD	TN	15.6"	Intel UHD	1366x768	Windows 10	2.2 kg	Dolby Audio	1 year	5 hours	25,000	The Lenovo IdeaPad is a versatile laptop that ...
4	ASUS	ZenBook Pro	i9	Intel	3.1 GHz	64GB	SSD	OLED	15.6"	NVIDIA RTX	3840x2160	Windows 10	1.8 kg	NanoEdge Display	2 years	7 hours	200,000	The ASUS ZenBook Pro is a high-end laptop that...

Figure: Sample laptop dataset

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

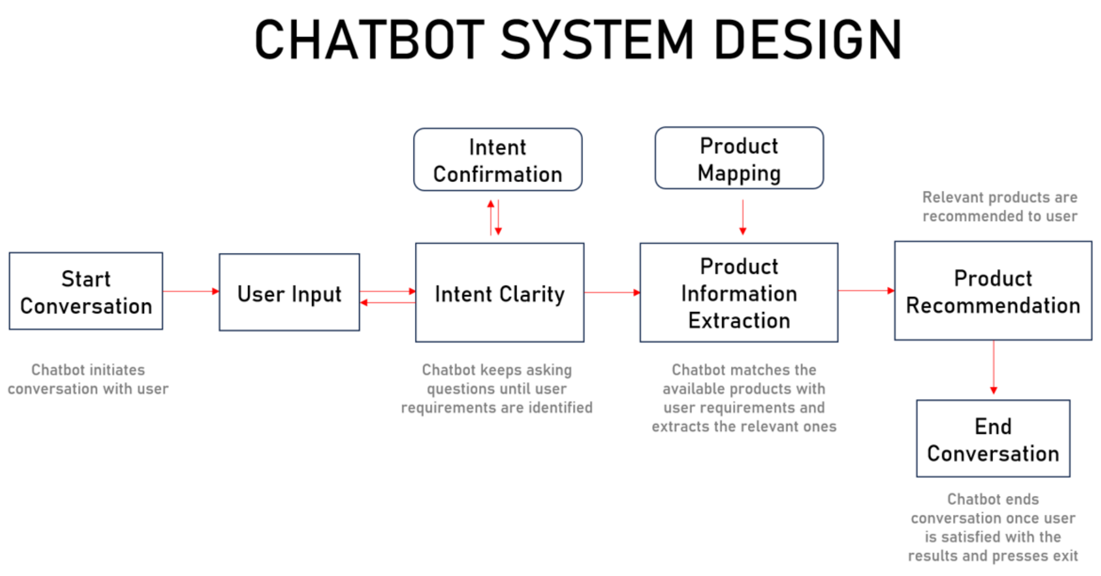
The chatbot should ask a series of questions to

1. Determine the user's requirements. For simplicity, we have used 6 features to encapsulate the user's needs. The 6 features are as follows:
2. GPU intensity
3. Display quality
4. Portability
5. Multitasking
6. Processing speed
7. Budget
8. 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

Now let's go ahead and understand the system design for the chatbot.


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 conversation once user is satisfied with the results and presses exit

1. Stage 1
    - Intent Clarity Layer
    - Intent Confirmation Layer
2. Stage 2
    - Product Mapping Layer
    - Product Information Extraction Layer
3. Stage 3
    - Product Recommendation Layer

Major functions behind the Chatbot

Let's now look at a brief overview of the major functions that form the chatbot. We'll take a deep dive later

- initialize_conversation(): This initializes the variable conversation with the system message.
- get_chat_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 GPU intensity, Display quality, Portability, Multitasking, Processing speed, 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_laptops_with_user(): This function compares the user's profile with the different laptops and come back with the top 3 recommendations.
- initialize_conv_reco(): Initializes the recommendations conversation

Implementation

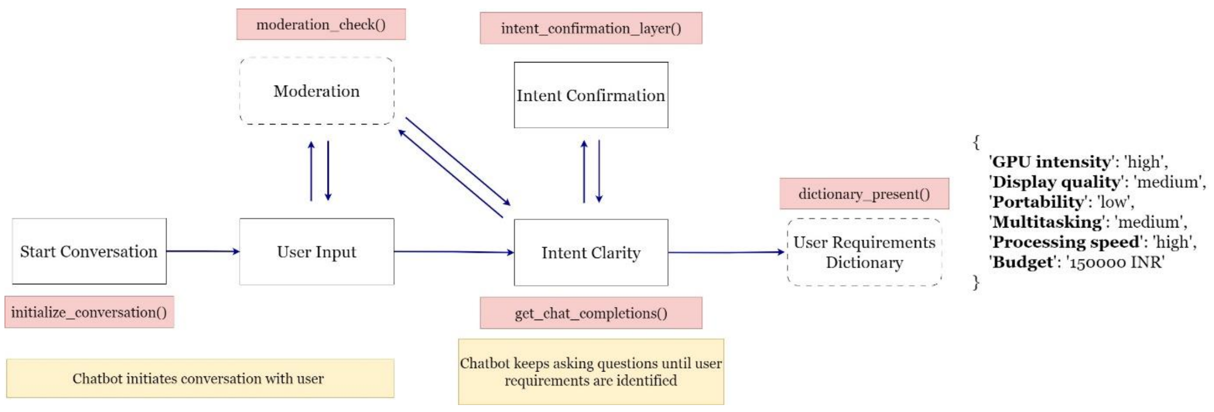
Stage 1+ Stage 2 + Stage 3

Let's start by importing the libraries that we'll require for this project. Following are the ones:

- openai
- pandas
- os, json, ast

Implementing Intent Clarity and Intent Confirmation Layers

Stage 1:


moderation_check()
Moderation
intent_confirmation_layer()
Intent Confirmation
Start Conversation
User Input
Intent Clarity
dictionary_present()
User Requirements Dictionary
initialize_conversation()
get_chat_completions()
Chatbot initiates conversation with user
Chatbot keeps asking questions until user requirements are identified
{
'GPU intensity': 'high',
'Display quality': 'medium',
'Portability': 'low',
'Multitasking': 'medium',
'Processing speed': 'high',
'Budget': '150000 INR'
}

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. 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
- 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.

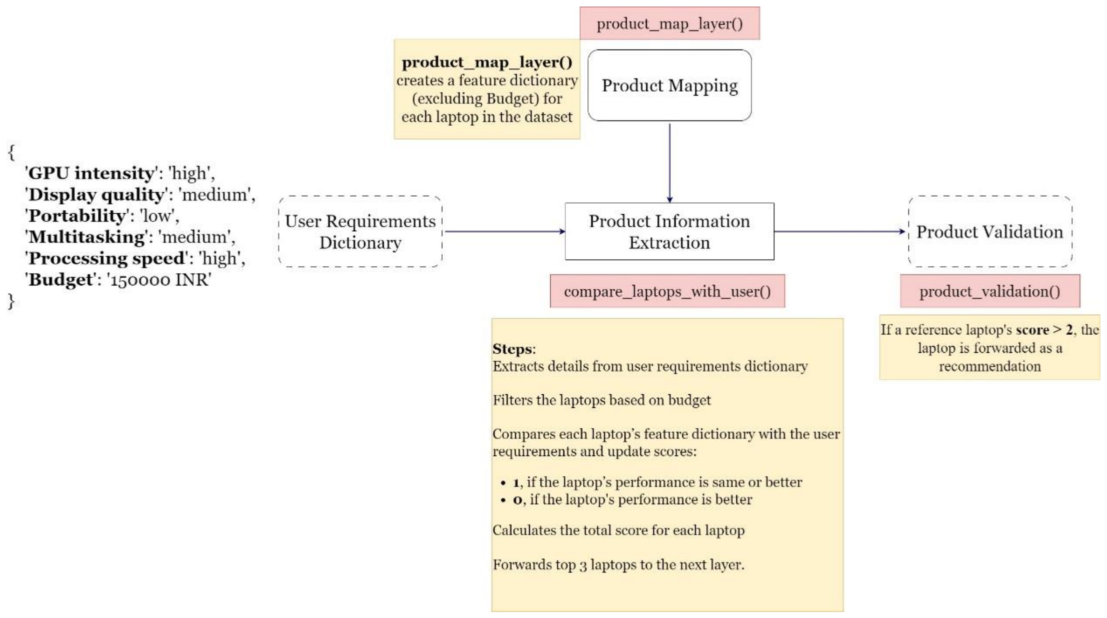
You need to understand the user's profile, which essentially means that all the features: GPU intensity, Display quality, Portability, Multitasking, Processing speed, Budget are captured or not. Let's look at the function that helps us verify that.

- 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
- GPU intensity
- Display quality
- Portability
- Multitasking
- Processing speed
- Budget
- 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 laptops using dictionary matching.

Implementing the Product Mapping and Information Extraction Layers

---

Stage 2:



```
{
  'GPU intensity': 'high',
  'Display quality': 'medium',
  'Portability': 'low',
  'Multitasking': 'medium',
  'Processing speed': 'high',
  'Budget': '150000 INR'
}
```

product_map_layer()
product_map_layer() creates a feature dictionary (excluding Budget) for each laptop in the dataset
Product Mapping
User Requirements Dictionary
Product Information Extraction
Product Validation
compare_laptops_with_user()
product_validation()
If a reference laptop's score > 2, the laptop is forwarded as a recommendation

Steps:
Extracts details from user requirements dictionary

Filters the laptops based on budget

Compares each laptop's feature dictionary with the user requirements and update scores:

- **1**, if the laptop's performance is same or better
- **0**, if the laptop's performance is better

Calculates the total score for each laptop

Forwards top 3 laptops to the next 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 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

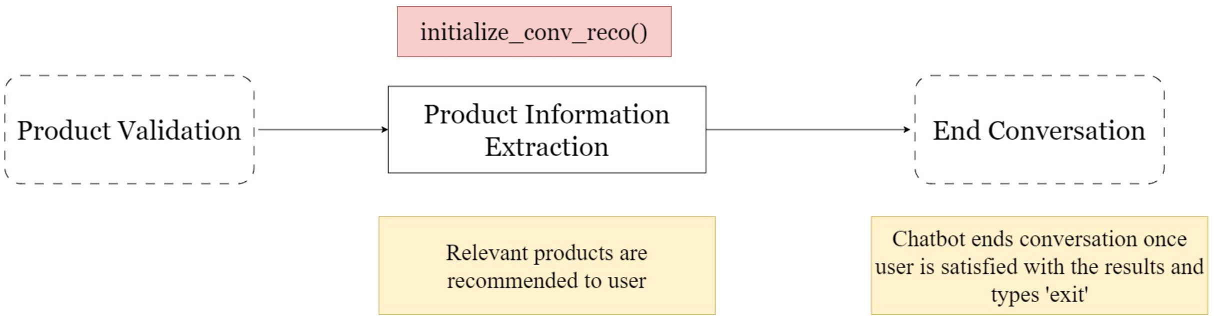
---

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

Product Recommendation Layer

Stage 3:



initialize_conv_reco()
Product Validation
Product Information Extraction
End Conversation
Relevant products are recommended to user
Chatbot ends conversation once user is satisfied with the results and types 'exit'

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. 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

Areas of Improvement and Final Comments

You can see that for gamer, the model is not able to perform well. But for the academic and business persona, it is able to perform well.

Future Scope of Work

The output format of each layer is inconsistent. You can use the function API capability of GPT to instruct the output format as per the input request.

1. The rule framework provided to classify each laptop's specification is not exhaustive. You can expand the rules to give a comprehensive context to the LLM.
2. There are misclassifications in the laptop's specifications, even after specifying clear rules for LLM. You can fine-tune an open-source LLM to make its understanding & performance better.
3. Once the products are extracted, the dialogue flow doesn't allow recalling of product extraction if there is any intent change. You can add another layer to observe any request for a change in the user intent and then use this flag to recall the product extraction based on the updated intent.
4. As an alternative & simple solution, you can use vector embeddings of each product and compare it with the user intent to find the most relevant products.
5. You can template this workflow/solution to build a chatbot for any product domain.

Note: You must add the relevant domain expertise/rules to give the LLM context understanding.