Project Report: ShopAssist AI

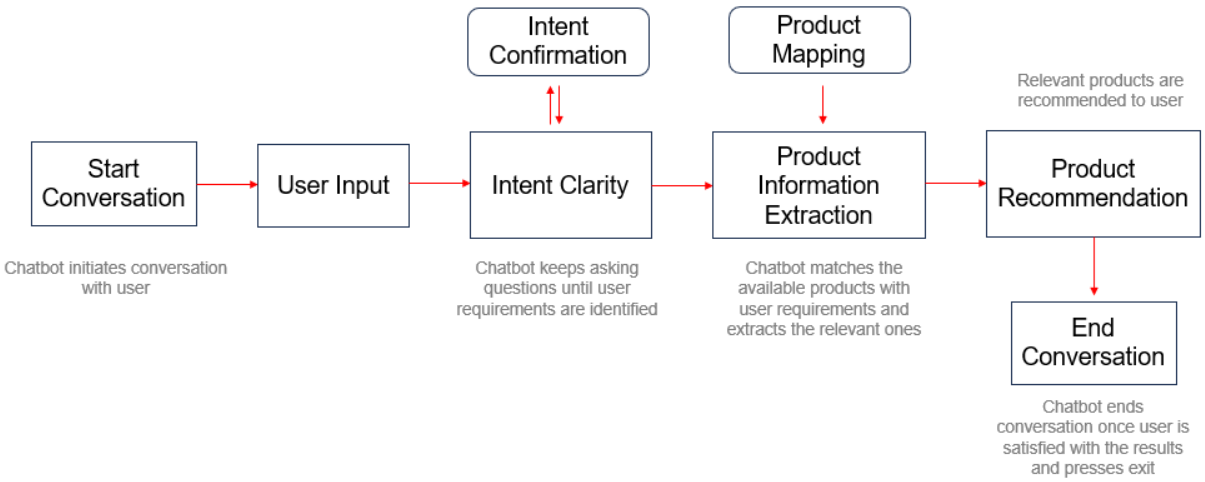
Objectives

The primary objective of ShopAssist AI is to simplify and enhance the online shopping experience by providing personalized laptop recommendations based on user requirements. The chatbot aims to:

1. Understand user preferences through natural conversation.
2. Parse a dataset containing laptop specifications and descriptions.
3. Deliver accurate and tailored recommendations efficiently.

Design

ShopAssist AI was designed with a multi-layered architecture to ensure seamless operation and user satisfaction. The chatbot's development was divided into three stages, incorporating specific layers for functionality and seamless operation:



Stage 1: Intent Clarity and Confirmation

1. **Intent Clarity Layer**: Ensures the chatbot captures the user's input accurately by identifying the key aspects of their requirements.
2. **Intent Confirmation Layer**: Validates the extracted information to confirm that the chatbot has understood the user's needs correctly. Key parameters include:
    - GPU intensity
    - Display quality
    - Portability
    - Multitasking
    - Processing speed

- Budget

Stage 2: Product Mapping and Information Extraction

1. **Product Mapping Layer**: Maps user requirements to the appropriate categories in the dataset (simplified in the updated approach).
2. **Product Information Extraction Layer**: Extracts detailed information about laptops from the dataset based on user preferences.

Stage 3: Product Recommendation

1. **Product Recommendation Layer**: Matches user requirements with the dataset to identify the top three laptops most suited to their needs. This layer also facilitates follow-up conversations for additional clarifications and refinements.

Implementation

The implementation of ShopAssist AI involved several key functions and enhancements:

1. **Core Functions**:
   - initialize_conversation(): Sets up the chatbot's initial state.
   - get_chat_completions(): Generates responses using the assistant's language model.
   - intent_confirmation_layer(): Validates the chatbot's understanding of user requirements.
   - compare_laptops_with_user(): Matches user preferences with dataset entries to identify top recommendations.
2. **Dataset Utilization**:
   - Used updated_laptop.csv with structured Python dictionaries for simplified information extraction.
3. **Testing Setup**:
   - Verified the chatbot's functionality using a Google Drive folder containing necessary files (updated_laptop.csv and OpenAI_API_Key.txt).
4. **Chat History Logging**:
   - Implemented a system to log conversations for future analysis and improvement.

Challenges

- Handling varied and unstructured data formats within the laptop dataset required additional preprocessing.
- Ensuring smooth integration of NLP and rule-based logic posed initial challenges.
- Designing a conversation flow that delivered consistent outputs across different layers.
- Optimizing performance while minimizing API call costs.

Enhancements in the Updated Design

1. **Combined Function-Calling API**:
    - Introduced a streamlined API for better integration, reducing code complexity and operational costs.
2. **Redefined Conversation Flow**:
    - Improved consistency across layers to minimize additional processing.
3. **Streamlined Input Handling**:
    - Utilized the updated dataset file updated_laptop.csv, where the last column contains Python dictionaries for easy access to laptop details.
    - Simplified architecture by excluding the Product Mapping Layer, focusing on function-calling APIs.
4. **Testing Setup**:
    - Ensured necessary files (updated_laptop.csv and OpenAI_API_Key.txt) are accessible in the Google Drive folder Shop Assist under MyDrive.
5. **Conversation History Logging**:
    - Implemented a mechanism to log chat history to a text file for future analysis and reference.

Testing and Results

Test Scenarios

1. Verifying the chatbot's ability to initiate and maintain meaningful conversations.
2. Ensuring accurate extraction of user requirements.
3. Validating recommendations against user preferences.
4. Logging chat history and ensuring proper storage.

Learnings

1. **Iterative Design Improves Efficiency**:
    - Streamlining the architecture by combining layers and focusing on a function-calling API reduced complexity and improved performance.
2. **Data Quality is Crucial**:
    - Structured datasets with embedded Python dictionaries significantly enhanced the chatbot's recommendation accuracy.
3. **Logging and Testing Enhance Robustness**:
    - Maintaining detailed chat logs proved invaluable for debugging and refining the system.
4. **User-Centric Design is Key**:
    - Emphasizing user experience through clear intent capture and validation ensures high satisfaction levels.

ShopAssist AI represents a significant step forward in personalized online shopping assistance, demonstrating the potential of combining AI with rule-based systems for practical applications.