Approach Document

GenAI Programming Exercise: Data Extraction and Chatbot Creation from Image

1. Introduction

This project demonstrates the creation of a chatbot powered by a Large Language Model (LLM) capable of extracting structured information from a restaurant receipt image and responding to user queries. The solution involves two main components:

1. **Image-based data extraction**: Extracting structured data such as restaurant name, address, items ordered, and payment details from the receipt image.
2. **Chatbot creation**: Developing a conversational interface that answers user queries based on the extracted data while gracefully handling edge cases where data is unavailable.

2. Reasons for Selecting the Chosen LLM

For this project, I choose **Google's Gemini API** for the following reasons:

- **Image-to-text capabilities**: Gemini supports direct image-to-text extraction, making it ideal for processing the receipt image and extracting structured information.
- **Accuracy and flexibility**: Gemini provides high accuracy in understanding and parsing text from images, ensuring reliable data extraction.
- **Ease of integration**: The Gemini API is well-documented and integrates seamlessly with Python, allowing for rapid development and testing.

3. Steps Taken to Perform Image-Based Data Extraction

The process of extracting structured information from the receipt image involved the following steps:

Step 1: Loading the Image

- The receipt image (`receipt_image.jfif`) was loaded using the Python Imaging Library (PIL).
- The image was passed directly to the Gemini API for processing.

Step 2: Generating Extracted Text

- A prompt was crafted to instruct the LLM to extract specific details from the receipt, including:
  - Restaurant Name
  - Restaurant Address
  - Order Date and Time
  - Payment Method
  - Items Ordered (including quantities and prices)
  - Subtotal, Tax, and Total Amount Paid
- The Gemini API processed the image and generated a textual representation of the receipt content.

Step 3: Cleaning and Structuring the Extracted Text

- The raw extracted text was cleaned to remove unnecessary characters (e.g., bold markdown `**`), extra spaces, and duplicate lines.
- Regular expressions (regex) were used to parse the cleaned text into a structured dictionary containing all required fields.

4. Design Considerations for the Chatbot

The chatbot was designed with the following considerations:

a. Query Handling

- The chatbot processes user queries by normalizing them (removing punctuation and converting to lowercase) and matching them against predefined patterns.
- For example, queries like "What is the restaurant name?" are matched to the `restaurant_name` field in the extracted data.

b. Graceful Handling of Edge Cases

- If a query requests information not available in the extracted data (e.g., "How much was the tip?"), the chatbot responds with a polite message indicating the data is unavailable.

c. Clarity and Robustness

- The chatbot ensures clear and accurate responses by retrieving data directly from the structured dictionary.

- It also provides a breakdown of total amounts (subtotal, tax, and total paid) when queried.

d. User Interaction

- The chatbot runs in a loop, allowing users to ask multiple questions until they type "exit" to quit.

---

5. Challenges and Solutions

Challenge 1: Incomplete or Ambiguous Data

- **Solution**: Used regex patterns to ensure robust parsing of the extracted text. Missing data was handled by assigning default values (e.g., "Unknown").

Challenge 2: Handling Edge Cases

- **Solution**: Implemented logic to check for missing fields and respond appropriately (e.g., "The tip amount is not available in the receipt").

Challenge 3: Integration with the LLM API

- **Solution**: Configured the Gemini API securely using environment variables and ensured error handling for API-related issues.

---

6. Conclusion

This project successfully demonstrates the use of an LLM-powered chatbot for extracting structured information from a receipt image and answering user queries. The approach ensures accuracy, clarity, and robustness while gracefully handling edge cases. The modular design of the code and detailed documentation make the solution easy to understand and extend.

---

7. Future Enhancements

- **Support for additional LLMs**: Extend the solution to support other LLMs like ChatGPT or Claude for comparison.
- **Improved error handling**: Add more sophisticated error handling for ambiguous or incomplete data.
- **User interface**: Develop a graphical user interface (GUI) or web-based interface for better user interaction.
