Homework 1

SID:1155245699

1. Introduction

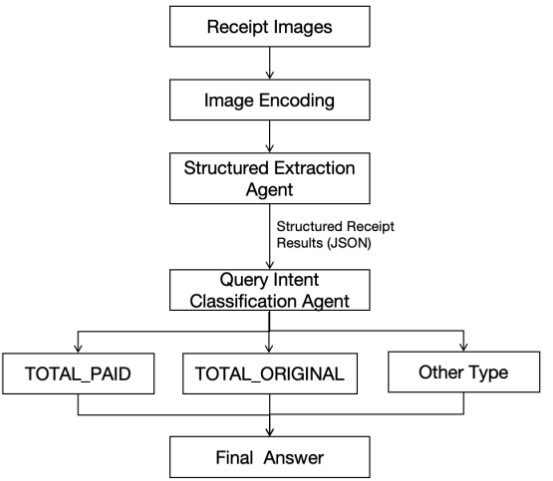
This project implements a multimodal receipt understanding and query answering system based on Large Language Models (LLMs).

The system is designed to process supermarket receipt images, extract structured financial information, understand user queries, and compute accurate numerical answers.

The project is implemented Python and LangChain, with Google Gemini accessed through API calls. Based on start code, my main work focus on designing the reasoning pipeline, structured extraction, query intent classification, routing logic, and evaluation workflow.

2. Pipeline

The system follows a modular pipeline design shown as the figure:



2.1 Structured Extraction Agent

This is to extract reliable numerical information from receipt images.

I designed a structured extraction agent using an LLM prompt that enforces explicit rules and a strict JSON output format. The agent extracts:

- total_paid: the final amount actually paid by the customer
- discount: the total discount computed by summing all negative values
- total_original: the original total before discount, computed as total_paid + discount

```
{'total_paid': 394.7, 'discount': 85.5, 'total_original': 480.2}
{'total_paid': 316.1, 'discount': 76.1, 'total_original': 392.2}
{'total_paid': 140.8, 'discount': 19.3, 'total_original': 160.1}
{'total_paid': 514.0, 'discount': 76.8, 'total_original': 590.8}
{'total_paid': 102.3, 'discount': 5.4, 'total_original': 107.7}
{'total_paid': 190.8, 'discount': 30.4, 'total_original': 221.2}
{'total_paid': 315.6, 'discount': 80.4, 'total_original': 396.0}
```

2.2. Receipt Parsing (`def parse_receipts(image_paths)`)

To process multiple receipt images, I implemented a receipt parsing pipeline that iterates over all images and applies the structured extraction agent to each one. The pipeline performs the following steps:

1. Convert each receipt image into a Base64–encoded data URL
2. Invoke the LLM extraction agent
3. Clean and normalize the LLM output
4. Parse the output into Python dictionaries

The result is a list of structured receipt objects that can be directly used for downstream computation.

2.3. Query Intent Classification

To support flexible natural language queries, I implemented a query intent classification agent using another LLM call.

The classifier determines which type of calculation the user is requesting.

The supported intents are:

- TOTAL_PAID: queries asking how much money was actually spent
- TOTAL_ORIGINAL: queries asking how much would have been paid without discounts

The classifier outputs only valid JSON and does not perform any numerical computation. This clean separation ensures that language understanding and numerical reasoning remain decoupled.

2.4. Query Routing and Numerical Computation

After the intent is classified, the system routes the query to the appropriate computation logic. All arithmetic operationsare performed in Python rather than inside the LLM. For example:

- TOTAL_PAID queries return the sum of total_paid across all receipts
- TOTAL_ORIGINAL queries return the sum of total_original across all receipts
- Other type query will reject to answer

2.5 Evaluation and Testing

The system is evaluated using a provided testing module. All test cases pass successfully.