

Receipt Evaluation Report

Github source: <https://github.com/SaKamian/FTEC5660>

1. Problem Background

In this task, **7 receipt images** are given and we are required to answer two evaluation queries with agentic AI method.

Query 1: How much money did I spend in total for these bills?

Query 2: How much would I have had to pay without the discount?

The core challenge is that the information (total paid, discounts) must first be **extracted from images using a multimodal LLM (Gemini)**, and then **calculated deterministically by code** to ensure evaluation stability.

2. Key Difficulties

a) **Supermarket receipts contain:**

- Mixed languages (Chinese + English)
- Item-level promotions (e.g., “Buy 2 Save \$X”)
- Percentage discounts (e.g., “5% OFF”)
- Multiple discounts per receipt
- Rounding adjustments

b) **Simple OCR or subtotal arithmetic is insufficient, because:**

- Discounts are often printed separately
- Some discounts are item-level
- Subtotal differences may include rounding or tax artifacts

Therefore, a vision-language LLM is used to directly understand and extract explicit discount lines.

3. Methodology Overview

3.1 Multimodal LLM for Receipt Understanding

Each receipt image is sent to a Gemini-based multimodal LLM using LangChain. The LLM is instructed via a structured system prompt to:

- Identify **ONLY** explicit printed discounts
- Extract negative monetary values associated with promotions
- Ignore inferred discounts from subtotal differences
- Ignore rounding adjustments
- Return **STRICT JSON** for machine parsing

This ensures robust and auditable extraction.

3.2 Prompt Design (Key Logic)

The system prompt enforces the following rules:

Only extract:

- Buy X Save \$Y
 - % OFF
 - Coupon / promotion lines
- a) Extract associated NEGATIVE amounts
 - b) Do NOT infer discounts
 - c) Do NOT double count
 - d) Ignore rounding (e.g., -\$0.01)

This prevents overcounting and ensures that only printed discount lines are used.

3.3 Discount Aggregation Logic

For each receipt:

$$\text{Receipt Discount} = \sum(\text{all extracted negative discount amounts})$$

$$\text{Pre-discount Total} = \text{Paid Amount} + |\text{Receipt Discount}|$$

This aligns with how retail systems apply promotions.

4. Solution Strategy

The solution is divided into three clear stages:

Stage 1: Multimodal Information Extraction

Use the **Google Generative AI SDK (Gemini)** to process 7 receipt images.

Prompt the model to extract, for each receipt:

total_paid

total_discount

The model is instructed to return results in a strict JSON schema.

Stage 2: Robust JSON Parsing

Because the model output may still contain extra text, we apply **defensive parsing**:

Extract the first valid JSON object from the model output using a regular expression.

Parse it safely using `json.loads()`.

This step ensures robustness and prevents `JSONDecodeError`.

Stage 3: Deterministic Computation

Once the structured data is available:

Query 1 is answered by summing all total_paid values.

Query 2 is answered by summing (total_paid + total_discount) for each receipt.

All calculations are done purely in Python to ensure correctness and reproducibility.

5. Final Output Format

The final output follows a standardized evaluation-friendly structure:

- a) Per-receipt extracted values
- b) Final numeric answers for Query 1 and Query 2

This design aligns with common benchmark and homework evaluation pipelines.