Homework 1

陈佳欣 Chen Jiaxin 1155246854

Installing packages

```
!pip install langchain_google_genai
```

```
Requirement already satisfied: langchain_google_genai in /usr/local/lib/python3.12/dist-packages (4.2.0)
Requirement already satisfied: filetype<2.0.0,>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from langchain_
Requirement already satisfied: google-genai<2.0.0,>=1.56.0 in /usr/local/lib/python3.12/dist-packages (from langc
Requirement already satisfied: langchain-core<2.0.0,>=1.2.5 in /usr/local/lib/python3.12/dist-packages (from lang
Requirement already satisfied: pydantic<3.0.0,>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from langchain_
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /usr/local/lib/python3.12/dist-packages (from google-genai<
Requirement already satisfied: google-auth<3.0.0,>=2.47.0 in /usr/local/lib/python3.12/dist-packages (from google
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in /usr/local/lib/python3.12/dist-packages (from google-genai
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /usr/local/lib/python3.12/dist-packages (from google-ge
Requirement already satisfied: tenacity<9.2.0,>=8.2.3 in /usr/local/lib/python3.12/dist-packages (from google-gen
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from google
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /usr/local/lib/python3.12/dist-packages (from 
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from google-genai<2.0
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/dist-packages (from google-genai<2.0.0,>=1.56
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/python3.12/dist-packages (from langchai
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/python3.12/dist-packages (from langchai
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from langcha
Requirement already satisfied: pyyaml<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain-co
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/python3.12/dist-packages (from langchain
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-packages (from anyio<5.0.0,>=4.8.0->go
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from google-auth
Requirement already satisfied: cryptography>=38.0.3 in /usr/local/lib/python3.12/dist-packages (from google-auth<
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0,>=0.28.1->goo
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0,>=0.28.
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/dist-packages (from jsonpatch<2.0.0,
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmit
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-packages (from cryptography>=38.0.3->
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.12/dist-packages (from pyasn1-modul
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.12->cryptograph
```

Setup your API key

```
from google.colab import userdata
GEMINI_VERTEX_API_KEY = userdata.get('VERTEX_API_KEY')
```

Downloading receipts.zip

```
import gdown
import os
file_id = "1oe2FZd3ZTO7nrDqjCafNvxicl08oF8JF"
download_url = f"https://drive.google.com/uc?id={file_id}"
gdown.download(download_url, "receipts.zip", quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=1oe2FZd3ZTO7nrDqjCafNvxicl08oF8JF
To: /content/receipts.zip
100%|██████████| 1.61M/1.61M [00:00<00:00, 89.2MB/s]
'receipts.zip'
```

In this step, we can automatically read all the images in the folder.

```
!unzip -o receipts.zip -d receipts
image_dir = "receipts"
image_paths = [os.path.join(image_dir, f)
```

```
              for f in os.listdir(image_dir)
              if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
```

```
Archive:  receipts.zip
  inflating: receipts/receipt1.jpg
  inflating: receipts/__MACOSX/._receipt1.jpg
  inflating: receipts/receipt2.jpg
  inflating: receipts/__MACOSX/._receipt2.jpg
  inflating: receipts/receipt3.jpg
  inflating: receipts/__MACOSX/._receipt3.jpg
  inflating: receipts/receipt4.jpg
  inflating: receipts/__MACOSX/._receipt4.jpg
  inflating: receipts/receipt5.jpg
  inflating: receipts/__MACOSX/._receipt5.jpg
  inflating: receipts/receipt6.jpg
  inflating: receipts/__MACOSX/._receipt6.jpg
  inflating: receipts/receipt7.jpg
  inflating: receipts/__MACOSX/._receipt7.jpg
```

1. Helper functions

In this step,I use my teacher's code.

```
import base64
import mimetypes

# Helper function to read and encode image
def image_to_base64(img_path):
    with open(img_path, "rb") as img_file:
        return base64.b64encode(img_file.read()).decode('utf-8')

# Helper function to encode local file to Base64 Data URL
def get_image_data_url(image_path):
    # Guess the mime type (e.g., image/png, image/jpeg) based on file extension
    mime_type, _ = mimetypes.guess_type(image_path)
    if mime_type is None:
        mime_type = "image/png" # Default fallback

    encoded_string = image_to_base64(image_path)

    # Construct the Data URL
    return f"data:{mime_type};base64,{encoded_string}"
```

```
from langchain_google_genai import ChatGoogleGenerativeAI
llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash",
    api_key=userdata.get('VERTEX_API_KEY'), # Ensure this key is set in Colab secrets
    temperature=0,
    vertexai=True
)
```

Display jpg images. Alternatively, open the folder icon on the left pannel to see the images.

```
from IPython.display import HTML, display
import glob, os

image_paths = glob.glob("*.jpg")
image_paths.sort()
html_content = '<div style="display: flex; flex-wrap: wrap; gap: 20px;">'

for path in image_paths:
    b64 = image_to_base64(path)
    filename = os.path.basename(path) # Clean up path to show just the name

    # Create a vertical column for each image + text
    html_content += f'''
    <div style="display: flex; flex-direction: column; align-items: center;">
        <img src="data:image/jpeg;base64,{b64}" style="height: 300px; border: 1px solid #ddd; margin-bottom: 5px;
        <span style="font-family: monospace; font-size: 14px;">{filename}</span>
    </div>
    '''

html_content += '</div>'

display(HTML(html_content))
```

receipt1.jpg

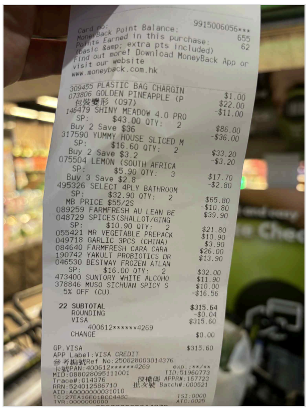
receipt2.jpg

receipt3.jpg

receipt4.jpg

receipt5.jpg

receipt6.jpg


receipt7.jpg

2.Image input to Gemini

This homework needs to solve practical problems:

•It takes as input several images of the supermarket bills, plus one user query (in the form of text).

•There are two possible queries that the users are interested to know:

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 model should have the capacity to reject irrelevant queries.

This part of the code is mainly used to determine the type of the user's input query. If it matches Question 1 or Question 2, the computation will continue. If the input query is irrelevant, it will be flagged as IRRELEVANT.(Here I take query2 as an example.)

```python
from langchain_core.prompts import ChatPromptTemplate
import re
user_query = input("Please enter your question: ")
query_router_prompt = ChatPromptTemplate.from_messages([
    ("system",
     "You classify the user query into one of the following categories:\n"
     "Q1: total money spent after discount\n"
     "Q2: total money before discount\n"
     "IRRELEVANT: anything else\n\n"
     "Return ONLY one word: Q1, Q2, or IRRELEVANT."
    ),
    ("human", "{user_query}")
])

router_chain = query_router_prompt | llm
query_type = router_chain.invoke(
    {"user_query": user_query}
).content.strip()
print(f"Detected query type: {query_type}")
```

```
Please enter your question: How much would I have had to pay without the discount?
Detected query type: Q2
```

Next, we start using the LLM to identify the total amount and the pre-discount amount from the data. The total amount is relatively easy to find in the receipts, usually indicated by words like "total." The pre-discount amount is harder to detect. Two methods are used here: one is to directly identify it from the receipt, and the other is to look for negative amounts in the receipt and add their absolute values to the payment amount to obtain the pre-discount total.

Attention:In my multiple tests, the LLM could not reliably recognize multiple images at once and tended to produce significant errors. Therefore, I chose to process each receipt individually and then sum the results to improve accuracy.

```
query1_answer = 0
query2_answer = 0
if query_type == "IRRELEVANT":
    print("I cannot answer this question.")

else:
    for i, image_path in enumerate(image_paths):
        print(f"\nprocessing {i+1}: {image_path}")
        image_url = get_image_data_url(image_path)

        per_receipt_prompt = ChatPromptTemplate.from_messages([
            ("system",
             "You are a cashier analyzing ONE receipt. Extract these exact numbers:\n"
             "1. Find the 'Total', 'Amount Payable', or 'Grand Total' AFTER all discounts.\n"
             "2. Find the total BEFORE any discounts. If not shown, add up all item prices.\n"
             "3. If you see discounts (negative numbers or 'Discount', 'Save'), note them.\n"
             "Return ONLY in this format: FINAL=XX.XX, ORIGINAL=YY.YY"
            ),
            ("human", [
                {"type": "text", "text": "What is the final total and original total on this receipt?"},
                {"type": "image_url", "image_url": {"url": image_url}}
            ])
        ])

        per_receipt_chain = per_receipt_prompt | llm
        response = per_receipt_chain.invoke({})

        text = response.content

        final_match = re.search(r'FINAL=([\d\.]+)', text, re.IGNORECASE)
        original_match = re.search(r'ORIGINAL=([\d\.]+)', text, re.IGNORECASE)

        if final_match:
            final_total = float(final_match.group(1))
            query1_answer += final_total
            print(f"after discount: {final_total}")

        if original_match:
            original_total = float(original_match.group(1))
            query2_answer += original_total
            print(f"before discount: {original_total}")

        elif final_match:
            discount_prompt = ChatPromptTemplate.from_messages([
                ("system",
                 "Find all DISCOUNT amounts on this receipt. Look for negative numbers or "
                 "words like 'Discount', 'Save', 'Reduction'. "
                 "Return the TOTAL discount as a positive number."
                ),
                ("human", [
                    {"type": "text", "text": "What is the total discount amount on this receipt?"},
                    {"type": "image_url", "image_url": {"url": image_url}}
                ])
            ])

            discount_chain = discount_prompt | llm
            discount_response = discount_chain.invoke({})

            discount_match = re.search(r'([\d\.]+)', discount_response.content)
            if discount_match:
                discount = float(discount_match.group(1))
                original_est = final_total + discount
                query2_answer += original_est

if query_type == "Q1":
    print(f"Answer (after discount): {query1_answer}")

elif query_type == "Q2":
    print(f"Answer (before discount): {query2_answer}")
```

```
processing 1: receipt1.jpg
after discount: 394.7
before discount: 480.4

processing 2: receipt2.jpg
after discount: 316.1
before discount: 392.2
```

```
processing 3: receipt3.jpg
after discount: 140.8
before discount: 160.1

processing 4: receipt4.jpg
after discount: 514.0
before discount: 590.8

processing 5: receipt5.jpg
after discount: 102.3
before discount: 107.7

processing 6: receipt6.jpg
after discount: 190.8
before discount: 221.2

processing 7: receipt7.jpg
after discount: 315.6
before discount: 396.0
Answer (before discount): 2348.3999999999996
```

3. Evaluation Code

- After running the following code blocks,the blocks does not return any error.

```
def test_query(answer, ground_truth_costs):
    # Convert string to float if necessary
    if isinstance(answer, str):
        answer = float(answer)

    # Calculate the ground truth sum once for clarity
    expected_total = sum(ground_truth_costs)

    # Check if the answer is within +/- $2 of the expected total
    assert abs(answer - expected_total) <= 2
```

Run the following code block to evaluate query 1:

> How much money did I spend in total for these bills?

```
query_1_costs = [394.7, 316.1, 140.8, 514.0, 102.3, 190.8, 315.6] # do not modify this
test_query(query1_answer, query_1_costs)
```

Run the following code block to evaluate query 2:

> How much would I have had to pay without the discount?

```
query_2_costs = [480.20, 392.20, 160.10, 590.80, 107.70, 221.20, 396.00] # do not modify this
test_query(query2_answer, query_2_costs)
```

```
sum([480.20, 392.20, 160.10, 590.80, 107.70, 221.20, 396.00])
```

```
2348.2
```