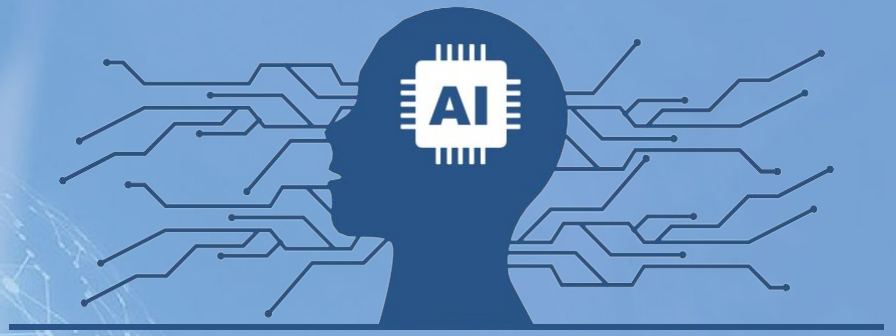
Artificial Intelligence



Homework #3

Multi-Agent Restaurant Rater

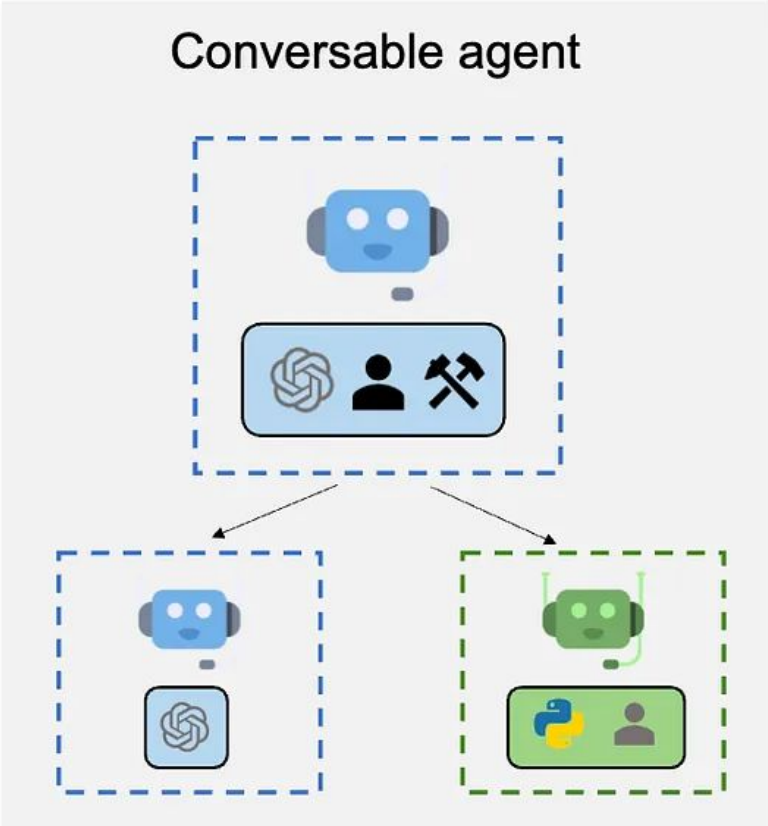
Wen-Huang Cheng (鄭文皇)

National Taiwan University

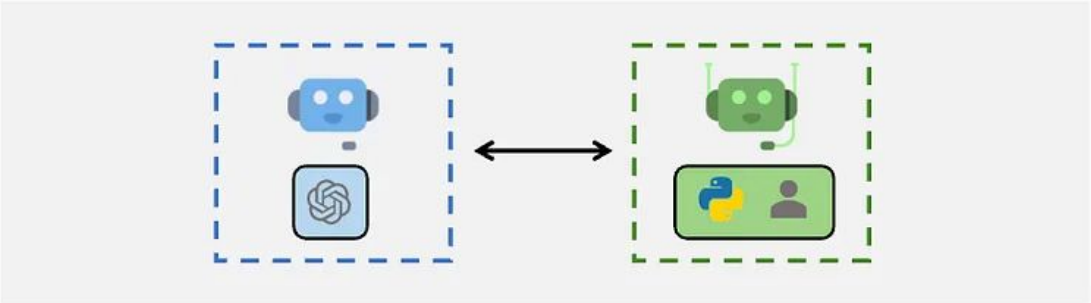
wenhuang@csie.ntu.edu.tw

Background Information: AutoGen

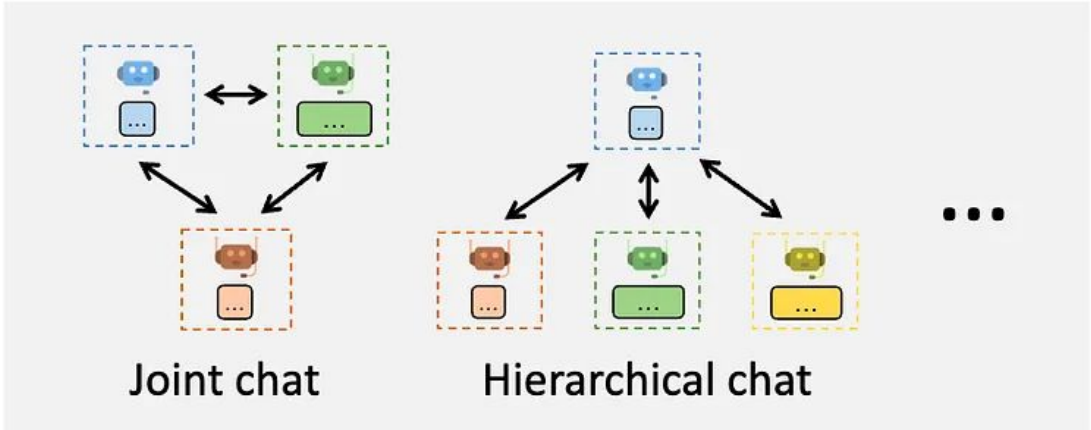
AutoGen is an AI framework that enables multiple assistants to write, compile, run code, and integrate APIs automatically by using provided documentation.



Agent Customization



Multi-Agent Conversations



Flexible Conversation Patterns

Source: https://www.microsoft.com/en-us/research/blog/autogen-enabling-next-generation-large-language-model-applications/

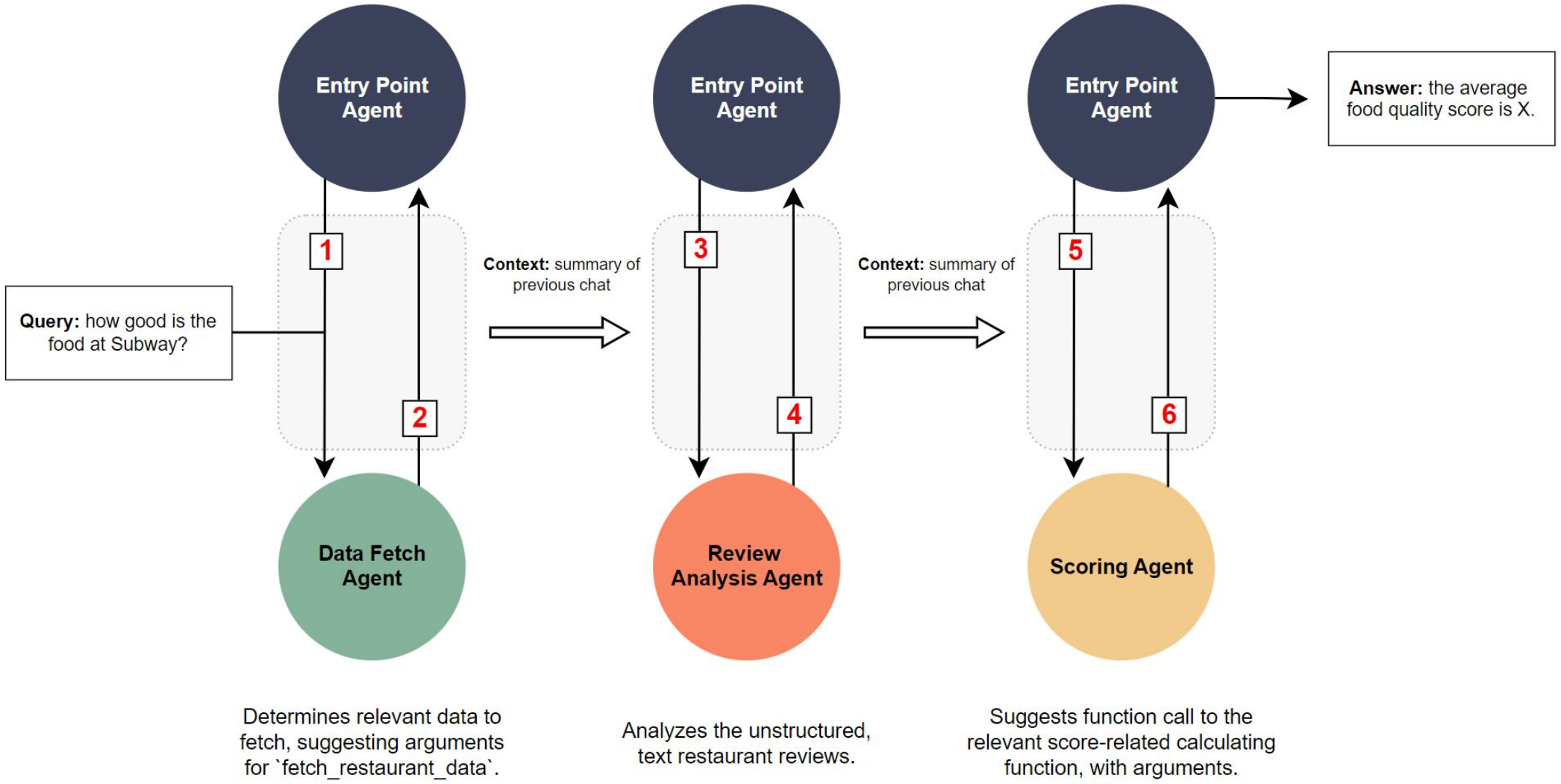
Task Description – Restaurant Rater

This lab will explore...

- **Analyzing restaurant reviews** using large language models.
- **Coordinating agent roles** in a multi-agent AutoGen system.

Source: https://embedsocial.com/blog/restaurant-reviews-examples/

Recommended Approach



Source: https://github.com/xianminx/mooc-cs294-llm-agents/blob/main/labs/lab01/Instructions.md

Stage 1: Fetching the Relevant Data

- First, analyze the query to determine which restaurant review data is needed. The data fetch agent will then suggest the function call with specific arguments.

```python
def fetch_restaurant_data(restaurant_name: str) -> dict[str, list[str]]:
    data = {}
    target = normalize(restaurant_name)
    with open(DATA_PATH, encoding="utf-8") as f:
        for line in f:
            if not line.strip(): continue
            name, review = line.split('.', 1)
            if normalize(name) == target:
                data.setdefault(name.strip(), []).append(review.strip())
    return data
```

- In both the public set and the private set, the review data follows the format below.

```
<restaurant_name>.  <review>.
```


Source: https://github.com/xianminx/mooc-cs294-llm-agents/blob/main/labs/lab01/Instructions.md

Stage 2: Analyzing Reviews

- Each review has keyword adjectives that correspond to the score that the restaurant should get for its ***food_score*** and ***customer_service_score***.
- The following rating scale (1 to 5) includes key descriptive words, along with some common synonyms and typical misspellings.

Rating scale

- Score 1/5 : awful, horrible, disgusting ...
- Score 2/5 : bad, unpleasant, offensive ...
- Score 3/5 : average, uninspiring, forgettable ...
- Score 4/5 : good, enjoyable, satisfying ...
- Score 5/5 : awesome, incredible, amazing ...

Source: https://github.com/xianminx/mooc-cs294-llm-agents/blob/main/labs/lab01/Instructions.md

Stage 3: Scoring

In the final step, a scoring agent aggregates the *food_score* and *customer_service_score* from each review, then calls *calculate_overall_score* to determine the final score.

Overall Score Formula

$$\text{Overall Score} = K \cdot \sum_{i=1}^{N} \sqrt{f_i^2 \cdot s_i}$$

Symbol	Description
f_i	Food quality score for the i^{th} entry (range: 1–5)
s_i	Service quality score for the i^{th} entry (range: 1–5)
N	Number of rating samples (i.e., number of (f_i, s_i) pairs)
K	Scaling constant: $K = \frac{10}{N \cdot \sqrt{125}}$
$\sum$	Summation over all entries $i = 1$ to N

```python
def calculate_overall_score(restaurant_name: str, food_scores: List[int], customer_service_scores: List[int]) -> dict[str, str]:
    """Geometric-mean rating rounded to 3 dp."""
    n = len(food_scores)
    if n == 0 or n != len(customer_service_scores):
        raise ValueError("food_scores and customer_service_scores must be non-empty and same length")
    total = sum(((f**2 * s)**0.5) * (1 / (n * (125**0.5))) * 10 for f, s in zip(food_scores, customer_service_scores))
    return {restaurant_name: f"{total:.3f}"}
```


Source: https://github.com/xianminx/mooc-cs294-llm-agents/blob/main/labs/lab01/Instructions.md

Setup & Guideline

In this lab, you will use the GPT-4o-mini model. Please generate your own OpenAI API key and ensure that it remains private—do not share or expose your key. You can follow the official guide here to create your key: https://platform.openai.com/docs/quickstart.

- To keep your key secure and easily accessible, store it as an environment variable named OPENAI_API_KEY.

```
export OPENAI_API_KEY=sk-proj-xxxxx
```

- Using the GPT-4o-mini model, we expect the cost of this lab to be < $1.

Setup & Guideline

Provided Files

- **main.py** – Contains the baseline implementation. Modify this file to implement your solution.
- **test.py** – Public test script used to verify your implementation. Do not modify this file.
- **requirements.txt** – Lists required Python packages.
- **restaurant-data.txt** – the public set of restaurant review data.

Usage

- To test your implementation, run the following command in the terminal:

```
python3 test.py ./restaurant-data.txt
```

Notes

- Your solution in main.py must be fully compatible with test.py. We will evaluate your code using the **additional hidden cases** and **review data.**

Grading Notes - Report (70%)

- **Screenshot of all passed cases (50%)**
  - 5 public test cases are provided in test.py.
  - Each passed case earns 10 points per test run.

```
[V] Test 0 Passed. Pyautogen 0.9.0
[V] Test 1 Passed. Expected: 3.000  Predicted: 2.982  Query: How good is the restaurant taco bell overall?
[V] Test 2 Passed. Expected: 9.350  Predicted: 9.349  Query: How good is the restaurant Chick-fil-A overall?
[V] Test 3 Passed. Expected: 8.060  Predicted: 8.060  Query: What is the overall score for Starbucks?
[V] Test 4 Passed. Expected: 9.540  Predicted: 9.539  Query: What is the overall score for In-n-Out
[V] Test 5 Passed. Expected: 3.650  Predicted: 3.653  Query: What is the overall score for McDonald's?
---------- 5/5 Tests Passed. MAE: 0.0046 ----------
```

- **Methodology & Discussion (20%)**
  - Design Structure / Prompt Design Diagram (5%)
  - Discussion of Success and Failure Cases (5%)
  - Optimizations and Improvements Made (10%)
- Please complete the report within **3 pages**; additional pages will not be graded.

Grading Notes – Performance Ranking (30%)

- **Performance Ranking (30%)**
  - Your code will be evaluated with hidden cases and additional data using the same query method as in `test.py`. If it fails to run, no performance score will be given.
  - We will not test invalid queries, such as a restaurant that is not in the dataset.
  - Scores will be assigned linearly based on the **MAE** (Mean Absolute Error) ranking.

- **Mean Absolute Error (MAE)**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- n = the number of data points
- y_i = the actual value for the i-th data point
- $\hat{y}_i$ = the predicted value for the i-th data point
- $|y_i - \hat{y}_i|$ = the absolute error for each prediction

Grading Notes

- These issues may result in a **5-point deduction** and potentially make the submission untestable.
  - Ensure that you are using the **GPT-4o-mini model.**
    - TA will check and uniformly use gpt-4o-mini during testing.
  - Confirm that the pyautogen package version is exactly **0.9.0.**

- Submissions may receive **a score of 0** if any of the following rules are violated:
  - Do not modify the test.py file under any circumstances.
  - Avoid hardcoding mappings or designing logic solely to pass public test cases.

Submission Guideline

- Deadline : 2025/05/23 (Fri.) 23:59
- Zip all files as hw3_<student_id>.zip
- Submit a zipped file (hw3_<student-id>.zip) on NTU COOL containing:
- hw3_<student-id>.zip

```
└── hw3_<student-id>/          #  (Should contain this folder, not separate files)
    ├── hw3_<student-id>.pdf   # Your report (Within 3 pages)
    ├── main_<student-id>.py   # Main Python script
    ├── requirements.txt       # List of any additional packages used
    └── README                 # Instructions or project description
```

Reference

- **Berkeley CS294: Large Language Model Agents (Fall 2024)**

  University of California, Berkeley. Retrieved from https://rdi.berkeley.edu/llm-agents/f24

- **Microsoft AutoGen: Multi-Agent Framework for Generative AI**

  Microsoft. Retrieved from https://github.com/microsoft/generative-ai-for-beginners/tree/main/17-ai-agents

- **Hugging Face Agents Course**

  Hugging Face. Retrieved from https://github.com/huggingface/agents-course


Any
Question
ai.ta.2025.spring@gmail.com