University of Science

Vietnam National University Ho Chi Minh City

ASCENDA ASSIGNMENT DOCUMENTATION

Name: Nguyen Dinh Tung (ndtungforwork@gmail.com)

TABLE OF CONTENTS

I. Main flow:

- **Step 1**: Try to crawl data from different urls and clean keys and values .

- For each json object of each url, try to normalize all keys (by **static** class **KeyNormalizer** - in **keyNormalizer.py**) and values that are not **list** or **dictionary** (by **static**.class **DataCleaner** - in **dataCleaner.py**).
  - For **KeyNormalizer**: I use some methods such as: **strip()** to remove leading and trailing whitespace from the key; **convert_to_snake_case()** to convert a key from camelCase or PascalCase to snake_case. (For example: "DestinationId" → "destination_id"); **apply_special_cases()** to replace the key with special case mappings, if applicable. For **extension** later, we just need to add into this **KEY_MAPPING** dictionary.
  - For **DataCleaner**: I use some methods such as: **strip_whitespace()** to remove leading and trailing spaces from the value. (For example: " car " → "car"); **normalize_text()** to normalize text formatting (For example: "car and toy" → "car and toy"), **remove_surrounding_quotes()** to remove surrounding quotes (For example: ""car"" → "car"); **apply_special_cases()** to replace values with None if they are in **RUBBISH_VALUES** list in config.

```python
# Key mapping
KEY_MAPPING = {
    "hotel_id": "id",
    "hotel_name": "name",
    "latitude": "lat",
    "longitude": "lng",
    "caption": "description",
    "info": "description",
    "details": "description",
    "destination": "destination_id",
    "url": "link"
}
```

```python
# Define rubbish value -> convert to None instead
RUBBISH_VALUES = ["N/A", "null", "NULL", ""]
```

- The algorithm used here is recursion (implemented in **clean_json()** method of **JsonCleaner** - in **jsonCleaner.py**).

```python
# Static class for JSON cleaning, combining key normalization and data value cleaning
class JsonCleaner:

    # Clean a json object (by normalize all keys and clean all values)
    @staticmethod
    def clean_json(json_data):
        if isinstance(json_data, dict):
            return JsonCleaner.process_dict(json_data)
        elif isinstance(json_data, list):
            return [JsonCleaner.clean_json(item) for item in json_data]
        return DataCleaner.clean_value(json_data)
```

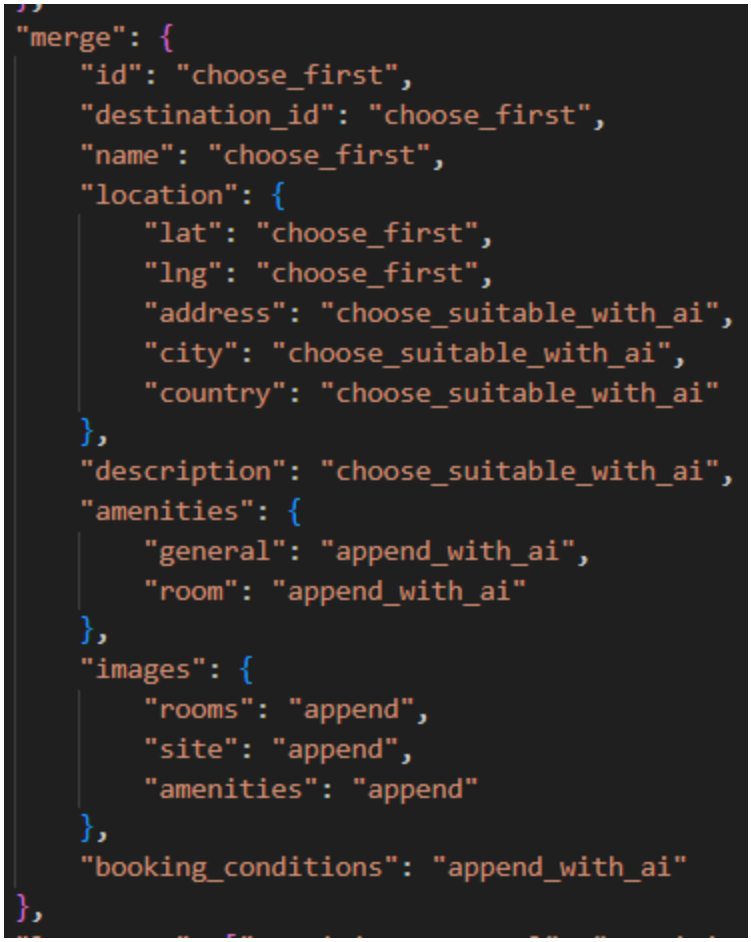
    - **Input of this step**: original json data, **output of this step**: json data (with cleaned keys and cleaned values).
- **Step 2**: Parse json data to **correct format**.
    - Use **transform()** method of **JsonToJsonConverter** class to change json data to correct format (**output format** required by this assignment) by the **selectSchema** and **recursion**.
    - Explain about **selectSchema**: for each field of correct output, we try to get element by element in the corresponding list.
        - If the **json object** has one of the keys in the list: take this value corresponding to this key.
        - If the **json object** doesn't have any keys in the list: return None

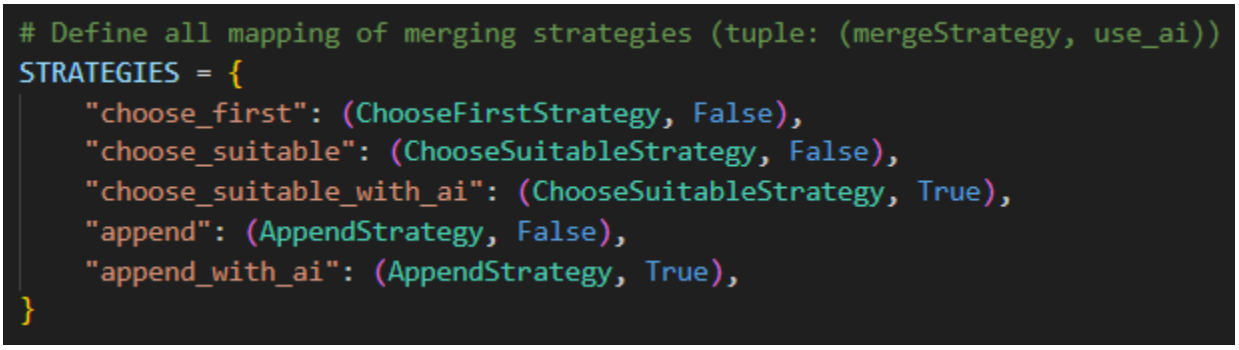
```python
# Config Schema (to convert json to correct form, merge json and lowercase some fields)
SCHEMA_CONFIG = {
    "select": {
        "id": ["id"],
        "destination_id": ["destination_id"],
        "name": ["name"],
        "location": {
            "lat": ["lat"],
            "lng": ["lng"],
            "address": ["address", "location->address"],
            "city": ["city"],
            "country": ["country", "location->country"]
        },
        "description": ["description"],
        "amenities": {
            "general": ["amenities->general", "facilities"],
            "room": ["amenities", "amenities->room"]
        },
        "images": {
            "rooms": ["images->rooms"],
            "site": ["images->site"],
            "amenities": ["images->amenities"]
        },
        "booking_conditions": ["booking_conditions"]
    },
```

```python
class JsonToJsonConverter:

    # Transform a JSON object based on the schema.
    @staticmethod
    def transform(schema, data):
        return JsonToJsonConverter.process_node(schema, data)
```

- **Step 3**: After having all json data in correct format, I try to merge a list of json (in correct format) by **merge()** method in **JsonMerger** class (in **jsonMerger.py**).
    - For each field, we use each corresponding strategy to merge, defined by **STRATEGIES** and **SCHEMA_CONFIF['merge']**.
    - We can choose to use **AI** or not. The **LLM** i use here is **awanllm** (https://www.awanllm.com/quick-start), and the logic is in **aiAPI.py** file (**choose_most_approriate()** method to choose the most appropriate single value from a list, **choose_appropriate_terms()** to choose some appropriate terms in a list), i also define **PROMPTS** in **config.py**.





  - I use **Strategy design pattern** to design all merge strategies (in mergeStrategy file). With **ChooseFirstStrategy** class to take the first non-None value (Return None if not exist non-None value), **ChooseSuitableStrategy** class to choose the suitable value among all values (can use LLM to evaluate -> if failed: take the first non-None value), **AppendStrategy** class to append values into a list, avoiding duplicates (can use AI to choose and clean some elements from this list). If we need to scale, we can add more strategy
- **Step 4**: Then I build **Supplier** class to handle each supplier url, and **SupplierHandler** class to handle app **Supplier** (both are in **supplier.py**).

    - First fetch all json data from an **url** and save into a **Supplier** instance (we can use **Supplier** class to find a hotel that matches a pair (**hotel_id, destination_id**)).
    - Then, use **SupplierHandler** to handle all supplier urls and query for list of **hotel_ids** and **destination_ids**.
    - To **scale** more urls, just need to add in **SUPPLIERS** in **config.py**.

```
# Define all Suppliers
SUPPLIERS = ['acme', 'patagonia', 'paperflies']
```

- **Step 5**: After receiving output json object, I do some lowercase with **amenities→general** and **amenities→rooms**, and also remove elements in **images→rooms**, **images→site**, i**mages→amenities** that have access denied response.

II. Analyze design and serve for data aggregation:

- I have a **Hotel** class in **objectClass.py**, we can also change from **Hotel** object to **json** object and also reverse order → easy to manage in our program.
- Scaling analysis:
    - If we want to **add/delete/modify** fields in **Hote**l, just fix in Hotel class and **SCHEMA_CONFIG['select'], SCHEMA_CONFIG['merge']** in **config.py.**
    - If we want to add more **urls**, just fix in **SUPPLIERS** in config.py.
    - If we want to add more logic of **key normalizer** or **data normalizer**, just fix it in **config.py**.
    - If we want to add more **strategies** to merge json, **strategy design pattern** help us.
- Each class, I apply **Single Responsibility Principle**, each class just does one mission.
- I apply **Open-Close Principle**, when we need to add new stuff, just add a function, don't fix old logic.
- Build a **static** class if this class does a specific task.