Report Paper

Fetching and Saving Data from a Paginated API

Name :- Aanshu Maurya

Email :- anshmaurya4420@gmail.com

Github :- https://github.com/Ansh420/Assignment

Abstract :-

This report describes the process of retrieving data from a paginated API, processing the response, and saving the data to a file. The task requires processing JSON responses, including those returned as strings, and ensuring that the data is aggregated across multiple pages. The report includes the implementation details of the Python script used for this project.

Introduction :-

APIs (Application Programming Interfaces) are important tools for accessing data from web services. Many APIs use paging to process large amounts of data, providing data in chunks over multiple requests. The task is to fetch all the data from the listed API pages, process the JSON response, print the data, and save it to a file. This report details how to do this using Python.APIs (Application Programming Interfaces) are important tools for accessing data from web services. Many APIs use paging to process large amounts of data, providing data in chunks over multiple requests. The job is to fetch all the data from the listed API pages, process the JSON response, print the data, and save it to a file. This report details how to do this using Python.

Objectives :-

1. Fetch the data from the pages of the API above.

2. Identify whether the response for each response-sources pair came from any of the

sources

3. List down the sources from which the response was formed. Returns an empty

array if the response did not come from any source. The shortlisted sources will be

called citations

4. Return the citations for all objects coming from the API.

Methodology :-

The implementation is implemented in Python, using the request library to make HTTP requests, and the json library to process JSON data.

1. API Endpoint and Pagination

Objective: Identify the API endpoint and understand its pagination mechanism.

API Endpoint: The API endpoint used in this task is https://devapi.beyondchats.com/api/get_message_with_sources

Pagination: The API uses a query parameter page to paginate results. By incrementing the page number in subsequent requests, we can fetch all pages of data.

2. Fetching Data

Objective: Retrieve data from the paginated API endpoint.

HTTP Requests: Use the requests library to send GET requests to the API endpoint.

Pagination Handling: Implement a loop to handle pagination. Continue fetching data until no more data is available or the response status code indicates an error.

JSON Parsing: Parse the response content, which may be returned as a string, into JSON format using the json library.

```python
import requests
import json

def fetch_data(api_url):
    data = []
    page = 1
    while True:
        response = requests.get(f"{api_url}?page={page}")
        if response.status_code != 200:
            break
        try:
            json_data = json.loads(response.content.decode('utf-8'))
        except json.JSONDecodeError:
            break
        if not json_data:
            break
        data.extend(json_data)
        page += 1
    return data
```

3. Data Processing

Objective: Process the fetched data to ensure it is correctly formatted and aggregated.

Data Aggregation: Combine data from all pages into a single list to facilitate further processing and analysis.

4. Saving Data to a File

Objective: Store the fetched data in a file for persistence and future reference.

File Writing: Use the json.dump method to write the aggregated data to a file in JSON format. Ensure the output is human-readable by setting the indentation level.

Filename: Default to api_data.json for the output file, but allow flexibility for different filenames.

```python
def save_data_to_file(data, filename='api_data.json'):
    with open(filename, 'w') as f:
        json.dump(data, f, indent=2)
    print(f"Data saved to {filename}")
```

5. Main Function

Objective: Orchestrate the data fetching and saving process.

API URL: Define the API endpoint URL.

Data Fetching: Call the fetch_data function to retrieve data from the API.

Print Data: Output the fetched data to the console for verification.

Save Data: Call the save_data_to_file function to store the data in a file.

```python
def main():
    api_url = 'https://devapi.beyondchats.com/api/get_message_with_sources'
    data = fetch_data(api_url)

    # Print fetched data to the console
    print(json.dumps(data, indent=2))

    # Save fetched data to a file
    save_data_to_file(data)

if __name__ == '__main__':
    main()
```

Error Handling

Objective: Ensure robustness and reliability of the script.

Status Code Checking: Handle non-200 HTTP status codes gracefully by breaking the loop and avoiding further requests.

JSON Decoding: Manage JSON decoding errors by breaking the loop and avoiding incorrect data parsing.

Results:-

Upon running the script, the data from the paginated API is fetched, printed to the console, and saved to a file named **`api_data.json`**. The following example demonstrates the script's output:

Console Output


```
[
  {
    "response": "Yes, we offer online delivery services through major platforms lik
    "sources": [
      {
        "id": "71",
        "context": "Order online Thank you for your trust in us! We are available c
        "link": "https://orders.brikoven.com"
      },
      {
        "id": "75",
        "context": "Do you give franchise if the brand No, we currently don't offer
        "link": ""
      },
      {
        "id": "8",
        "context": "Breakfast Reservations For Breakfast, we recommend making reser
        "link": "https://www.brikoven.com/reservations"
      }
    ]
```

Saved File Output (`api_data.json`)


```
[
  {
    "response": "Yes, we offer online delivery services through major platforms lik
    "sources": [
      {
        "id": "71",
        "context": "Order online Thank you for your trust in us! We are available c
        "link": "https://orders.brikoven.com"
      },
      {
        "id": "75",
        "context": "Do you give franchise if the brand No, we currently don't offer
        "link": ""
      },
      {
        "id": "8",
        "context": "Breakfast Reservations For Breakfast, we recommend making reser
        "link": "https://www.brikoven.com/reservations"
      }
    ]
  }
]
```


Conclusion :-

The Python script successfully fetches data from a paginated API, processes the JSON responses, and saves the data to a file. The functions are modular and handle different parts of the task, ensuring that the data is correctly fetched, aggregated, and saved. This approach can be adapted to other similar tasks involving paginated APIs and JSON data processing.