Python Information Aggregator with Web API and Scraping

Dan Hansen Student ID 24718999

Brian Shimmer Student ID 25752306

David Pawley Student ID 92072368

Maximus Chandrasekaran Student ID 25614189

**Introduction**

This report documents the development of the Python-based Information Aggregator that combines data from web APIs and web scraping techniques. The program fetches news articles from various sources, processes the data, and presents it through a graphical user interface (GUI). The implementation adheres to object-oriented programming (OOP) principles and includes unit testing for reliability. This report outlines the challenges faced, design decisions made, and additional features implemented.

**Key Components Implemented**

1. News API Integration

The python program integrates with the **News API** to fetch current news headlines and articles as **JSON data,** using which **preferred news sources or filter articles by category and country** can be selected to be displayed. The **API key management system** ensures valid keys are used. The program is built to cycle through multiple keys if necessary to avoid rate limits.

2. Web Scraping

For sources not fully covered by the API, the web scraping method was implemented. We used **ABC Australia** as our source for those data that the API was unable to provide. ABC Australia was chosen to ensure the ethics is followed through (The articles were not protected by a paywall). Our program includes **BeautifulSoup** to scrape the additional details from the website such as article content, authors, and publication dates. This enriches the dataset beyond what the API provides.

3. Libraries and their uses

* *Tkinter*: Used to develop the UI (User Interface) of the project as GUI (Graphical User Interface). Examples include buttons, boxes etc.
* *Requests*: Used to facilitate the HTTP requests to interact with the NEWS API and to collect data from the ABC Australia website.
* *PyCountry*: Used for country and language lookups and aiding in filtering news articles by location as per the user’s preference.
* *Pandas*: Used in the manipulation of data sourced from the website and those from the API.
* *Datetime*: Used to manage the time and the dates that come along with the data while displaying.
* *JSON*: Handles the JSON data that was derived from the scraping and API.
* *Matplotlib*: Used to generate the graph that displays the redundant words from the NEWS articles.
* *Collections and Counter*: Used to analyse the trend in the articles. For example, the most frequent words. It works hand in hand with the matplotlib library.
* *String*: Used to remove punctuations within the article for better and polished results.
* *Random*: Used to avoid overloading a single key and choose a key at random.
* *Sys*: Used to handle System-specific parameters and functions along with the runtime arguments or errors.

4. Data Processing

The program combines data from **both API responses and web scraping** into a unified **pandas DataFrame**. It is used to handle data cleaning, removes duplicates, and ensures consistency in the extracted information from the respective sources. The data that was aggerated using the above-mentioned methods were cleaned using the appropriate methods that ensures data consistency.

5. Data Visualization

A **bar chart** is generated alongside the scraped and API data to visualize the most common words in article titles. This provides further insights and a better understanding into the trending topics/ news that are collected.

6. Object-Oriented Programming (OOP) Principles

The code is structured using the following OOP principles:

* **Encapsulation**: The **KeyList**class manages **32-character API keys** with the help of a method **get\_key** that returns a random key whenever needed. The **NewsAPI\_KeyList** sub class manage API keys securely. This is done by overriding the get\_key method to select the API key at random while checking it with the News API website before returning it to the program.
* **Inheritance**: As mentioned above, the NewsAPI\_KeyList class inherits the data from KeyList to extend functionality to ensure the inheritance feature of object-oriented programming.
* **Modularity**: Separate classes and methods are implemented to handle functions like API calls, scraping, and GUI components of the project.

7. Unit Testing

Critical functions such as, API key validation and data extraction were tested using this phase of the project to ensures reliability, correctness and most importantly, real world application and usage. Also, simple yet essential tasks such as fixing the inevitable bugs during JSON parsing and unstructured data during scraping the data from its source were identified and rectified during unit testing.

8. Graphical User Interface (GUI)

The GUI (Graphical User Interface) of this project allows users to:

* Select news sources, categories, and countries.
* Fetch and display articles.
* View visualizations of common title words.
* And allows the user to hide or ignore certain words from the article.

**Challenges Faced**

1. API Key Management

Handling multiple API keys and ensuring their validity was challenging. The solution involved creating a key management system that cycles through keys and checks their validity before use.

While working with the API keys, we had a daily limit of 100. Multiple keys were created using various email addresses to offset that limit.

2. Web Scraping Dynamic Content

Scraping dynamically loaded content from the news website (ABC Australia) required parsing **JSON-LD structured data** embedded in scripts. To address this, the JSON data was directly extracted and processed, eventually fixing that challenge.

3. Data Consistency

Combining data from APIs and scraping introduced inconsistencies in date formats and missing fields. For example, the date inconsistencies that were encountered during the display of news from the United States were countered to fit a consistent dd/mm/yyyy format using the **datetime** library. The program standardizes dates and handles other missing values gracefully.

4. Performance

Fetching and rendering large datasets impacted performance of the python program. Implementing caching and optimizing DataFrame operations improved responsiveness and useability.

**Design Decisions**

1. Class Structure

* KeyListand NewsAPI\_KeyList were used to manage the centralized API key.
* NewsScraper combines API calls, scraping, and GUI logic for a cohesive and interactive user experience.

2. User Interaction

* Radio buttons are used to toggle between source-based filtering and category/country-based filtering of the News articles.
* Listboxes allow multi-selection for user-flexibility and engagement.

3. Visualization

* Matplotlib was chosen for its simplicity and it’s user-friendly way to generate the bar chart and integration with Tkinter.
* Common words are displayed in the bar chart for a quick trend analysis.

**Additional Features**

1.Word Omission

The application improves keyword frequency analysis by allowing users to omit specific words from the graphical display, ensuring a clearer visualization. For instance, words like "the," "and," or "in" can be excluded using the "Omitted Words" field at the bottom of the GUI. Users can add or modify words in this field and click "Replot" to dynamically update the graph, focusing on more meaningful trends. This feature, preloaded with a default list of common stop words, ensures the display remains flexible and user-friendly.

2.User Interaction

The list boxes in the GUI offer users an intuitive way to select from all available options for Source or Category/Country. This user-friendly design makes it easy to filter and scrape news from multiple sources, countries, and categories, enhancing the overall experience and flexibility when gathering news information.

3. Dynamic Key Validation

The program validates API keys in real-time, ensuring only working keys are used. The program’s dynamic key validation ensures seamless API access by testing and cycling through a shuffled list of keys. For example, each key is validated against the News API, and invalid keys are automatically skipped. If no valid keys remain, the application exits while notifying the user through a clear warning message. This approach prevents disruptions while adhering to API rate limits.

4. Error Handling

Error handling is thoughtfully integrated throughout the code to manage common issues like network failures and malformed responses. For example, network timeouts or API errors are caught using try-except blocks, ensuring the program doesn’t crash unexpectedly. Similarly, user feedback is prioritized, with informative messages displayed when invalid inputs or data fetching errors occur. Even in critical cases, such as when no valid keys are available, the program shuts down cleanly with a helpful alert.

5. Optional Features

* Users can limit the number of articles fetched (via pageSize).
* The GUI includes clear feedback for user actions.

**Conclusion**

The Information Aggregator project successfully combines API data and web scraping to provide a comprehensive and an engaging news aggregation tool. The use of object-oriented programming principles such as inheritance, encapsulation and modularity along with unit testing, and a user-friendly GUI ensures the program’s robustness, maintainability, and accessibility.

**Ethical Considerations**

* The program respects API rate limits and terms of service.
* Web scraping is performed ethically, targeting only publicly available data from the website.
* Users are reminded to use the tool responsibly and comply with legal guidelines.

**Submission Requirements Met**

1. **Codebase**: Well-documented Python code adhering to OOP principles.
2. **README**: Instructions for running the program, API key setup, and dependencies.
3. **Report**: This document covers challenges, design decisions, and features.
4. **GUI Screenshots**: Included in the submission folder.
5. **Video Presentation**: A separate video demonstrates the tool's functionality.