Detailed Documentation for the News Fetching Script

Overview and Purpose

This script is designed to automate the process of querying and collecting news articles from the GNews API using specific keywords and date ranges. The goal is to retrieve and organise articles mentioning particular terms, saving the results in Excel files for further analysis. The use of automation simplifies data collection, making it efficient and scalable for large date ranges or extensive keyword lists. This is particularly useful for researchers, analysts, and industry professionals interested in trends or incidents related to specific topics such as farming, agriculture, or accidents.

The script includes functionalities for API interaction, data validation, error handling, and result organisation. By incorporating robust error logging and modularity, the script ensures reliability and ease of maintenance.

Script Workflow

The script follows a structured approach, divided into the following major steps:

1. **API Request Construction**:
    - The script interacts with the GNews API using the `requests` library to send GET requests. It constructs a query with multiple keywords connected by the `OR` operator, allowing for articles to match any keyword individually.
2. **Data Retrieval and Processing**:
    - Articles are retrieved in JSON format. The script parses the data to extract relevant fields like titles, descriptions, URLs, publication dates, and sources.
3. **Data Organization**:
    - The extracted data is stored in pandas DataFrames and then exported to Excel files. Each file corresponds to a specific month, organizing the results chronologically.
4. **Automation Across Timeframes**:
    - The script processes a defined date range by splitting it into monthly intervals, ensuring manageable requests and files.
5. **Error Handling and Logging**:
    - Errors during API calls or data processing are logged, and the script handles cases like empty responses gracefully.

Technical Components

Keyword-Based Query: The query construction is one of the script's core aspects. It uses logical operators (`OR`) to ensure that articles mentioning any keyword in the list are retrieved.

For example:

```
query = "Farm OR Farming OR Quad Bike OR Cattle OR Tractor OR silo
OR Farmer OR Farm Accident OR Farming Accident OR Windfarm OR
Trailer OR hay bales"
```

1. This method prevents the need for all keywords to appear simultaneously in an article. The GNews API processes these logical operators effectively to return results for any of the specified terms.

Date Formatting for API Compatibility: The GNews API requires date parameters in ISO 8601 format with UTC timezone (`Z`). The script appends time details to each date string, ensuring compatibility:

```
start_date = start_date + "T00:00:00Z"
end_date = end_date + "T23:59:59Z"
```

2. This ensures that the query covers the entire range of days for the specified month.

Monthly Data Partitioning: The script calculates monthly intervals dynamically to avoid manual adjustments for varying month lengths:

```
next_month = current_date + timedelta(days=31)
next_month = next_month.replace(day=1)
end_date_month = (next_month -
timedelta(days=1)).strftime('%Y-%m-%d')
```

3. By processing data in monthly batches, the script keeps API requests and resulting files manageable, ensuring better organisation and clarity.
4. **Error Logging and Resilience**: Error handling mechanisms are in place to ensure smooth operation even when issues arise. For example:

If the API returns an error, the script logs the HTTP status code and error message for debugging:
```
print(f"Error: {response.status_code} - {response.text}")
```

When no articles are found for a given month, the script skips file creation and logs the event:
```
print("No data to save. Skipping file creation.")
```

Saving Data to Excel: Using pandas, the script saves extracted article data into Excel files, with separate files for each month:

```
df.to_excel(filename, index=False)
```

5. This format ensures compatibility with most analytical tools and simplifies data review.

Challenges and Their Solutions

1. **Ensuring Keyword Flexibility**: Initially, the challenge was to ensure that articles matching any single keyword from a list were included. This was solved by using the `OR` operator in the query string. However, combining too many keywords could exceed API constraints, so it is advisable to test smaller subsets of keywords if needed.
2. **Handling API Rate Limits**: The GNews API may impose limits on the number of requests or articles retrieved within a specific timeframe. To address this, the script processes data in monthly intervals and limits the maximum results per request (`max: 100`). Implementing automatic retries or rate-limit handling could further enhance this feature.
3. **Dealing with Empty Responses**: In cases where no articles match the query for a given month, the script avoids creating empty Excel files and logs the occurrence. This ensures the process remains efficient without unnecessary outputs.
4. **Error-Prone API Responses**: Errors such as malformed JSON or HTTP failures were anticipated and handled with robust exception handling and logging.

Improvements for Future Versions

1. **Asynchronous Requests**: Parallelizing API requests using asynchronous programming (e.g., `asyncio`) could significantly speed up the process, especially for large date ranges.
2. **Customizable Query Inputs**: Allowing users to input their own keywords and date ranges dynamically via command-line arguments or configuration files would enhance usability.
3. **Enhanced Output Formats**: In addition to Excel, outputs could include CSV or JSON formats, providing more flexibility for integration with analytical tools.
4. **Rate Limit Awareness**: Implementing automatic backoff and retry mechanisms for API rate limits would improve robustness.
5. **Data Augmentation**: Adding metadata such as sentiment analysis or keyword frequencies could provide more insights from the retrieved articles.

Logging and Debugging Details

Logging is critical for identifying and resolving issues during script execution. The following types of logs are included:

- **Request Logs**: The full API request URL is logged for each query, enabling users to verify query parameters.
- **Error Logs**: Detailed error messages, including HTTP status codes and response texts, are printed when requests fail.
- **Progress Updates**: The script logs its progress, including the date range being processed and whether data was saved successfully.

Example log output:

```
Fetching news from 2020-01-01 to 2020-01-31
Request URL: https://gnews.io/api/v4/search?q=Farm+OR+Farming...
Error: 429 - Too Many Requests
No data to save. Skipping file creation.
```