Instruction Documentation for the Python Script

Overview

This Python script is designed to process a set of URLs listed in an Excel file, fetch the contents of these URLs, perform sentiment analysis, readability analysis, and other text-based computations, and finally store the results in an output Excel file.

Approach

- Fetching Content from URLs: The script reads URLs from an input Excel file (`Input.xlsx`) and fetches their content using the `requests` and `*BeautifulSoup*` libraries. The content is extracted by targeting specific HTML tags (like headings and article content).
- Sentiment Analysis: The script creates a dictionary of positive and negative words by loading pre-defined word lists and filtering out stop words. It then calculates sentiment metrics like positive score, negative score, polarity, and subjectivity.
- Readability and Text Analysis: The script calculates various readability and text metrics such as:
    - Average sentence length
    - Percentage of complex words
    - Gunning Fog Index (a readability metric)
    - Word count, complex word count, syllable count per word
    - Personal pronoun count
    - Average word length
- Storing Results: The content from each URL is saved as a `.txt` file named after the `URL_ID` in a designated folder (`*output/*`).
- All computed metrics are compiled into a DataFrame and saved to an output Excel file (`*Output Data Structure.xlsx*`).

Dependencies

To run this script, the following Python libraries are required:

- *Pandas*
- *requests*
- *bs4 (BeautifulSoup)*
- *nltk*
- *string*
- *os*
- *re*

Ensure all these libraries are installed. If not, you can install them using `*pip*` command in the command shell:

```
>>> pip install pandas requests beautifulsoup4 nltk
```

Setup Instructions

1. Prepare the Input Files:
   - Ensure you have an Excel file named `*Input.xlsx*` in the same directory as the script. This file should contain two columns: `*URL_ID*` and `*URL*`.
   - Place the necessary stopwords and sentiment dictionaries in their respective folders:
     i. ` *StopWords/*`: Folder containing text files with stopwords.
     ii. `*MasterDictionary/*`: Folder containing `*positive-words.txt*` and `*negative-words.txt*`.
2. Organize the Output Folder: The script expects an `*output/*` folder within the working directory where it will save the content of each URL as `*.txt*` files.
3. Running the Script: Execute the script using a Python environment by running:

   >>> *python main.py*

   This will start the process, and upon completion, an output Excel file named `*Output Data Structure.xlsx*` will be generated/populated in the same directory.

How to Run the Script

1. Ensure all dependencies are installed.
2. Place the `Input.xlsx` file and the script (`*main.py*`) in the same directory.
3. Organize the `*StopWords/*` and `*MasterDictionary/*` folders as required.
4. Create an `*output/*` folder in the working directory.
5. Run the script using Python:
   >>> *python main.py*

Expected Folder Structure

```
20211030 Test Assignment
├─ MasterDictionary
│   ├─ negative-words.txt
│   └─ positive-words.txt
├─ output    //Empty at first. Populates as the program executes
├─ MasterDictionary
│   ├─ StopWords_Auditor.txt
│   ├─ StopWords_Currencies.txt
│   ├─ StopWords_DatesandNumbers.txt
│   ├─ StopWords_Generic.txt
│   ├─ StopWords_GenericLong.txt
│   ├─ StopWords_Geographic.txt
│   └─ StopWords_Names.txt
├─ Input.py
├─ main.py
└─ Output Data Structure.xlsx
```

Resulting Output

- Text Files: Each URL's content is saved as a `*.txt*` file in the `*output/*` folder with the filename corresponding to the `*URL_ID*`.
- Excel File: An Excel file named `*Output Data Structure.xlsx*` containing all computed metrics for each URL will be generated.

Final Notes

- Error Handling: The script has basic error handling for failed URL fetches. If a URL fails to load, it will return `*None*` for that URL's content.
- NLP Setup: The script uses `*nltk*` for text tokenization and stopword management. Ensure the necessary `*nltk*` datasets (like stopwords) are downloaded. The script will attempt to download them if they are not already available.