Instructions Documentation

Approach for Solution:

1. **Data Extraction**:
    - The first part of the task involves extracting articles from the URLs provided in the Input.xlsx file. The objective is to extract the article text while excluding headers, footers, and other unrelated content.
2. **Text Analysis**:
    - After extracting the article text, we need to compute various text analysis metrics such as sentiment scores (positive, negative, polarity), average sentence length, percentage of complex words, fog index, syllable per word, personal pronouns, and word count.
3. **Generating Output**:
    - The output needs to be saved in an Excel file that contains these computed metrics along with the corresponding URLs and URL IDs.

Detailed Approach

1. **Extracting Article Text (Data Extraction)**:
    - I used **requests** to fetch the HTML content from the URLs provided in the Input.xlsx.
    - To parse the HTML and extract only the article text, I used **BeautifulSoup**. This library helps to extract only the relevant sections of the page (like the article content) by selecting specific HTML tags or CSS classes.
    - The extracted article text was saved in a text file using the **URL_ID** as the filename to match the output requirements.

2. **Text Analysis**: The text analysis required extracting the following variables:
    - **Sentiment Analysis**:
        - For sentiment analysis, I used the **VADER Sentiment Analyzer**. It provides the polarity of the text (ranging from -1 for negative to +1 for positive), and while it doesn't provide subjectivity, we can still use the **polarity** score to identify sentiment.
    - **Average Sentence Length**:
        - I calculated the average sentence length by splitting the text into sentences (using a period as a delimiter) and then calculating the average number of words in each sentence.
    - **Percentage of Complex Words**:
        - I identified complex words as those with more than 2 syllables, and I used the **syllapy** library to count syllables in each word. Then, I calculated the percentage of complex words as a ratio of the total complex words to the total word count.
    - **Fog Index**:
        - The **Fog Index** gives an estimate of the readability of the text. It is calculated using the formula: Fog Index=0.4×(Average Sentence Length+Percentage of Complex Words)\text{Fog Index} = 0.4 \times \left(\text{Average Sentence Length} + \text{Percentage of Complex Words}\right)Fog Index=0.4×(Average Sentence Length+Percentage of Complex Words)
    - **Syllables per Word**:

        - I used **syllapy** to count syllables in each word and calculated the average syllables per word.
    - **Personal Pronouns**:
        - I counted the personal pronouns like "I", "we", "you", etc., by checking each word in the text.
    - **Word Count**:
        - This was calculated by simply splitting the text into words and counting them.
    - **Average Word Length**:
        - I calculated the average length of the words by summing the length of each word and dividing by the total word count.
3. **Creating the Output Excel**:
    - I created a structured output that includes the calculated metrics (positive score, negative score, polarity, subjectivity, etc.).
    - Each row in the output corresponds to one article, with columns for the URL, URL ID, and the computed analysis metrics.
    - I used **openpyxl** and **pandas** to write the results into an Excel file.

Detailed Workflow of the Code

1. **Read the Input**:
   The input data (URLs and URL IDs) is read from the provided Input.xlsx file.
2. **Extract the Articles**:
   For each URL, I used **requests** to fetch the page, and

BeautifulSoup to extract only the article content (ignoring the header/footer).

3. **Perform Text Analysis**: For each article text, I computed the following:
    - Sentiment scores .
    - Various readability metrics (average sentence length, percentage of complex words, fog index, syllables per word, etc.).
    - Other metrics (personal pronouns, word count, average word length).
4. **Store the Results**:
   The results were stored in a structured format (Excel or CSV) using **pandas**. I included:
    - The URL and URL ID.
    - Each of the calculated metrics.
5. **Export the Output**:
   The final output was saved in an Excel file with each article's data in a separate row.

Steps to Run the Script:

1. Install Dependencies:

You need to install the following Python libraries:

- pandas
- openpyxl
- textblob

- syllapy

You can install these using pip:

pip install pandas openpyxl textblob syllapy

2. Prepare Input Data:

- Place the `Input.xlsx` file containing the URLs and URL_ID column in the same directory as this script.

- Ensure that the `URL` column in the `Input.xlsx` file contains the URLs of the articles to analyze.

3. Run the Script:

- Run the `article_analysis.py` script in your Python environment.

- The script will extract text from each URL, perform analysis, and generate the output in an Excel file named `output_with_analysis.xlsx`.

4. Output:

- The output Excel file will contain clickable hyperlinks to the articles, along with analysis results such as Positive Score, FOG Index, and more.

5. Dependencies:

The script uses the following Python libraries:

- pandas: For handling data in DataFrame format.

- openpyxl: For writing results to Excel

- syllapy: For syllable counting to calculate the FOG Index.

-nltk :To work with text preprocessing (tokenization, stemming, and syllable counting)

6. Notes:

- Make sure the `URL` column in the `Input.xlsx` is correctly formatted.

- The text extraction logic in the `extract_text_from_url()` function must be implemented according to your scraping method (e.g., using BeautifulSoup or Selenium).

How to Run the Python Script

To run the Python script, follow these steps:

1. Make sure you have Python installed (preferably Python 3.6+).
2. Install the required libraries by running:

   pip install pandas openpyxl textblob syllapy

3. Save the script as article_analysis.py and the input file (Input.xlsx) in the same directory.
4. Run the script:

   python article_analysis.py

5. After the script finishes, you will find the output Excel file (output_Data_Structure.xlsx) in the same directory.