**Solution Approach:**

**Code 1**: URL Extraction and Article Text Retrieval

**Data Collection:**

* Read URLs from the "input.xlsx" Excel file using pd.read\_excel() function.

**Article Text Extraction:**

* Define a function extract\_article\_text(url, article\_id) to extract article text from each URL.
* Utilize the requests library to send GET requests to the URLs.
* Use BeautifulSoup (from bs4 import BeautifulSoup) to parse the HTML content of the web pages and extract article text.
* Save the extracted article text to text files in the "article\_texts" directory.
* Handle cases where article titles or content are not found.

**DataFrame Creation:**

* Iterate over each URL, extract article text, and store relevant information (article ID, title, text) in a DataFrame.

**Output Generation:**

* Save the DataFrame containing URL IDs and URLs to an Excel file named "URL.xlsx".

**Code 2:** Textual Analysis and Output Generation

**Dependencies Installation:**

* Download NLTK resources (punkt, averaged\_perceptron\_tagger, cmudict) using nltk.download() function.Install TextBlob (from textblob import TextBlob) for sentiment analysis.

**Textual Analysis:**

* Define a function calculate\_metrics(text) to compute various metrics for textual analysis.
* Tokenize the text into sentences and words using NLTK (import nltk) functions.
* Calculate polarity and subjectivity scores using TextBlob.
* Compute metrics such as average sentence length, percentage of complex words, Fog Index, etc.
* Utilize the CMU Pronouncing Dictionary (from nltk.corpus import cmudict) for estimating syllable counts.

**DataFrame Creation and Output Generation:**

* Iterate over text files stored in the "article\_texts" directory.
* Calculate metrics for each article text using the calculate\_metrics() function.
* Store the computed metrics in a DataFrame.
* Merge the DataFrame with the URL DataFrame using pd.merge().
* Reorder the columns to place the URL column after the URL ID.
* Save the final DataFrame to an Excel file named "textual\_analysis\_output.xlsx".

**How to Run the Solution:**

**Dependencies Required:**

1. Python (version 3.x)
2. pandas (pip install pandas)
3. requests (pip install requests)
4. BeautifulSoup (pip install beautifulsoup4)
5. NLTK (pip install nltk)
6. TextBlob (pip install textblob)

**Execution Steps:**

Ensure all required dependencies are installed.

Place both Python script files (code1.py and code2.py), the "input.xlsx" Excel file, and any text files in the same directory.

Open a terminal or command prompt.

Navigate to the directory containing the script files and data.

Run code1.py by executing:

python code1.py which is “text\_extracter”

After execution, run code2.py by executing:

python code2.py which is “textual\_analysis”

The output file "textual\_analysis\_output.xlsx" will be generated, containing the calculated metrics for each article text, along with the corresponding URLs.

**NOTE:** There were two links which are not working and i have specified and commented the links in my 1st code which is ” text\_extracter.py”.

When I ran the first code, I noticed that only half of the URL content was successfully extracted, while the rest remained incomplete. I took the initiative to thoroughly inspect the websites again and carefully analyze the HTML structure to identify the root cause of the issue. Subsequently, I made necessary modifications to my code to address the discrepancies and ensure comprehensive extraction of the URL content.

This assessment provided an excellent opportunity to put my Python skills to the test and demonstrated the importance of meticulous attention to detail in problem-solving. I am grateful for the opportunity and look forward to hearing from you soon. Thank you once again.