Article Extraction and Sentiment & Text Analysis Pipeline: A Comprehensive Approach for Data Extraction, Sentiment Scoring, and Textual Insights

Instructions Documentation for the Sentiment and Text Analysis Pipeline

This documentation describes how to pull article content from URLs, conduct sentiment analysis, and calculate text analysis metrics, which are then combined into a final output file. Below are the steps to run the code given, including dependencies and how to run the.py file.

Overview of the Solution

The solution is designed to extract article titles and content from URLs listed in an input Excel file (input.xlsx), perform sentiment analysis using positive and negative word dictionaries, and generate text metrics such as word count, sentence length, and syllable counts. The final output is a merged Excel file containing both sentiment analysis and text analysis results.

Steps to Approach

1. **Extract Article Content:**
    - **Extract URLs:** URLs are read from an Excel file input.xlsx.
    - **Fetch Content:** For each URL, the article content (title and text) is extracted using the `requests` library and parsed using beautifulsoup
    - **Save Article Text:** The title and content of each article are saved as .txt files in a directory called 'articles'
2. **Sentiment Analysis:**
    - **Load Dictionaries:** Load positive and negative word dictionaries from .txt files
      Tokenization: The article text is tokenized into words, and sentiment scores are computed by checking how many words belong to the positive and negative dictionaries.
    - **Sentiment Metrics:** Positive score, negative score, polarity score, and subjectivity score are calculated for each article.
3. **Text Analysis Metrics:**
    - **Tokenization:** The article text is tokenized into sentences and words.
    - **Metrics Computation:**
        - Average sentence length
        - Percentage of complex words
        - Fog index

        - Average words per sentence
        - Word count
        - Syllable count per word
        - Personal pronoun count
        - Average word length
4. **Merge Results:**
    - The results from sentiment analysis and text analysis are merged into a single DataFrame and saved into an Excel file OutputDataStructure.xlsx

Dependencies Required

To run the provided Python script, you'll need the following Python libraries installed:

1. **pandas:** For reading and writing Excel files and handling DataFrames.
2. **requests:** For making HTTP requests to fetch article content from the URLs.
3. **beautifulsoup4 (bs4):** For parsing HTML content
4. **nltk:** For natural language processing tasks
5. **openpyxl:** For working with Excel files, required by pandas to work with .xlsx files

How to Run the Python Script

1. **Prepare the Input Files:**
    - Ensure you have the following files:
        - `input.xlsx`: Excel file containing a list of URLs
        - `positive-words.txt`: A text file containing a list of positive words.
        - `negative-words.txt`: A text file containing a list of negative words.
        - The script will create a folder called "articles" to save the extracted articles as `.txt` files.
2. **Update File Paths:**
    - Modify the paths in the `main()` functions for `input.xlsx`, `positive-words.txt`, and `negative-words.txt` to match the actual location of your files.
3. **Output:**
    - The script will generate two output Excel files:
        - `sentiment_analysis_results.xlsx`: Contains the sentiment analysis results for each article.
        - `text_analysis_results.xlsx`: Contains text analysis metrics
    - A merged file (`OutputDataStructure.xlsx`) will be created, combining sentiment and text analysis results.

Key Functions in the Code

1. **extract_article_text(url):**
    - Extracts the article title and text content from a given URL.

- Cleans up the article by removing unwanted elements like headers and footers.

2. **save_article_to_file(url_id, article_title, article_text):**
   - Saves the article title and text to a .txt file in the "articles" folder.
3. **load_dictionaries(positive_dict_file, negative_dict_file):**
   - Loads the positive and negative word dictionaries from the provided `.txt` files.
4. **calculate_sentiment_scores(text, positive_words, negative_words):**
   - Calculates sentiment scores (positive, negative, polarity, and subjectivity) based on the word dictionaries.
5. **calculate_avg_sentence_length(sentences), calculate_percentage_complex_words(text), ...:**
   - Computes various text analysis metrics like average sentence length, percentage of complex words, fog index, and more.
6. **main():**
   - Main function that orchestrates reading input data, processing articles, performing sentiment and text analysis, and saving the results.

Conclusion

This pipeline automates the process of fetching articles, performing sentiment analysis, and calculating various text analysis metrics, which can be useful for analyzing large volumes of articles or reviews.