Instructions
Text Analysis Pipeline Documentation

Ramavarapu Dhanush

Contents:

APPROACH TO THE SOLUTION

APPROACH

1. Data Input:

Input data is read from an Excel file (**Input.xlsx**) containing URLs and corresponding URL IDs using the **pandas** library.

2. Web Scraping and Content Extraction:

The script fetches content from each URL using the **requests** library. **BeautifulSoup** is used to parse the HTML content and extract specific elements (e.g., article content) based on HTML attributes (classes).

3. Text Preprocessing:

Extracted text is cleaned by removing newlines and special characters. Stop words (common words that do not contribute much to the meaning of a sentence) are removed from the text using predefined stop word files.

4. Sentiment Analysis:

Sentiment scores (positive and negative) are calculated based on predefined sets of positive and negative words. Polarity and subjectivity scores are computed using the counts of positive and negative words.

5. Readability Metrics Calculation:

Readability metrics such as average sentence length, percentage of complex words, and Fog index are computed to assess the readability of the text.

6. Text Analysis Features:

Additional text analysis features are calculated, including counts of complex words, syllables, personal pronouns, and average word length.

7. Output Data Structure:

All processed data is structured into a DataFrame (**output_df**) and saved to an Excel file (**Output Data Structure.xlsx**) for further analysis.

SCRIPT COMPONENTS

1. Function Definitions:

Functions are defined for specific tasks such as content extraction (**extract_and_save_content**), stop word removal (**remove_stop_words_from_file**), sentiment analysis (**calculate_sentiment_scores**), readability calculation (**calculate_readability_metrics**), and various text analysis features (**calculate_complex_word_count**, **calculate_total_word_count**, etc.).

2. Main Execution Block (if __name__ == '__main__':):

The main block of code reads input data from **Input.xlsx**, initializes necessary resources (e.g., positive and negative word sets), processes each URL using defined functions, computes text analysis metrics, creates an output DataFrame, and saves the processed data to **Output Data Structure.xlsx**.

CONCLUSION

This Python script serves as a comprehensive solution for text analysis tasks, combining web scraping, text preprocessing, sentiment analysis, readability assessment, and feature extraction functionalities. It demonstrates proficiency in using popular Python libraries and modular coding practices for efficient text processing and analysis.

RUNNING THE TEXT ANALYSIS PIPELINE SCRIPT

STEPS TO RUN THE SCRIPT

1. Download the Script and Input Data:

Download the **text_analysis_pipeline.py** script from the provided source. Place the script in a directory where you can easily access it.

2. Prepare Input Data:

Ensure you have an input Excel file named **Input.xlsx**. This file should contain columns **URL_ID** and **URL** listing the URLs you want to analyze.

3. Prepare Stop Word Files:

Include the necessary stop word files (**StopWords_Auditor.txt**, **StopWords_Currencies.txt**, etc.) in the same directory as the script. These files are used to filter out common words during text preprocessing.

4. Positive and Negative Word Files:

Ensure you have **positive-words.txt** and **negative-words.txt** files for sentiment analysis. These files contain lists of positive and negative words used to calculate sentiment scores.

5. Open Terminal or Command Prompt:

Navigate to the directory containing the script and input files.

6. Run the Script:

Execute the script by running the following command in the terminal or command prompt:
python text_analysis_pipeline.py

7. Monitor Script Execution:

The script will start processing each URL listed in **Input.xlsx**. Progress messages will be displayed in the terminal/console as the script extracts content, performs text preprocessing, and calculates analysis metrics.

8. Verify Output:

Once the script completes execution, check for the generated output file **Output Data Structure.xlsx** in the same directory. This file contains the structured data with sentiment scores, readability metrics, and other text analysis features for each URL.

Note:

Ensure all required files (Input.xlsx, stop word files, positive-words.txt, negative-words.txt and text_analysis_pipeline.py) are correctly placed in the same directory and accessible to the script.

CONCLUSION

By following these steps, we can successfully run the **text_analysis_pipeline.py** script to analyze text content from specified URLs, perform text preprocessing, calculate sentiment and readability metrics, and generate a structured output dataset for further analysis or reporting.

LIST OF DEPENDENCIES REQUIRED

1. Python:

Install Python from the official website if not already installed.

2. Required Libraries:

Install the necessary libraries (**pandas, requests, beautifulsoup4, nltk**):
pip install pandas requests beautifulsoup4 nltk