Text Analysis and Web Scraping

Project Documentation

---

1. Project Overview

This project is an scraping and text analysis tool designed to perform text analysis on web articles. The process is split into two core modules: a robust web scraper to extract clean article content from a list of URLs, and a text analysis engine to compute 13 key metrics, including sentiment, readability, and word statistics.

The final output is a consolidated Excel sheet that merges the original input data with the newly calculated analytical variables.

---

2. Solution Approach

My approach was to build a resilient and modular pipeline, ensuring each stage is efficient and handles potential failures gracefully.

Phase 1: Data Extraction (Web Scraping)

For the scraping module, I chose the combination of **requests** and **BeautifulSoup** for its speed and simplicity. The goal was to fetch the raw HTML quickly and parse it effectively without the overhead of a full browser engine like Selenium, as the target content was primarily static.

- **Content Identification:** The strategy was to identify the article title by searching for the first prominent heading tag (from <h1> down to <h6>). The article body was located by searching for common structural patterns, such as <article> tags or elements with standard class names like entry-content. This pattern-based approach makes the scraper adaptable to various site layouts.
- **Robustness:** The entire scraping process for each URL is wrapped in a try...except block. This ensures that if a single URL fails due to a timeout, a block, or a "404 Not Found" error, the script logs the failure and seamlessly moves on to the next URL without crashing. A separate "retry" script was also developed to target only the failed URLs.

Phase 2: Textual Analysis

The analysis engine is built to be a standalone module that processes the locally saved text files. This separation makes the analysis phase fast and repeatable without needing to re-scrape the websites.

- **Preprocessing:** The Natural Language Toolkit (**nltk**) is used for tokenization—breaking down the raw text into a structured list of sentences and words. A comprehensive set of stop words is then used to filter out noise, leaving a clean list of meaningful words for accurate analysis.

- **Variable Computation:** Each of the 13 required variables is calculated precisely according to the formulas defined in the project documentation. This includes sentiment scoring using the master dictionaries, readability assessment using the Gunning Fog Index formula, and various other linguistic counts. All calculations are performed on the cleaned, tokenized text to ensure data integrity.
- **Output Generation:** The popular **pandas** library handles all data structuring. The results from the analysis are compiled, merged with the original Input.xlsx data based on the URL_ID, and then exported to a clean, well-ordered Excel file that matches the specified output format.

---

3. How to Run

Follow these steps to set up the environment and execute the pipeline.

1. Set Up the Environment:

- Ensure you have Python 3.8+ installed.
- It's recommended to create a virtual environment to manage dependencies:

“””

python -m venv .venv

source .venv/bin/activate # On Windows, use `.venv\Scripts\activate`

“””

- Install all required packages using pip:

“””

pip install pandas openpyxl requests beautifulsoup4 lxml nltk

“””

2. Download NLTK Data:

- Run the following command in a Python shell **once** to download the necessary punkt tokenizer model:

“””

import nltk

nltk.download('punkt')

“””

3. File Structure:

- Place your Input.xlsx file, master_dictionary folder, and stop_words folder in the same root directory as the Python scripts.

4. Execute the Scripts:

- **First, run the scraper** to download the articles. This will create an extracted_articles folder.

“””

python scrape_articles.py

“””

- **Second, run the analysis script** to process the downloaded files and generate the final report.

“””

python text_analysis.py

“””

- The final output will be saved as Output Data Structure.xlsx.

---

4. Dependencies

- **Python 3.10**
- **Libraries:**
    - pandas
    - openpyxl
    - requests
    - beautifulsoup4
    - lxml
    - nltk
- **NLTK Data Packages:**
    - Punkt_tab