**This document provides insights and guidance on the Sentimental analysis algorithm “SentimentEngine”.**

The data extraction and NLP problem is tasked with solving ways to design and implement a sentimental analysis algorithm, which analyzes textual data from web and assess its sentiments, positive, negative, or neutral, along with determining more variables such as subjectivity score, readability.

Problem approach:

The problem is solved using an extract, transform, load approach with multiprocessing for parallel processing of text files.

1. Data Extraction:

* The function created() creates a dataframe containing URL and URL\_ID from the Input.xlsx file.
* The function extractdf() is a function that leverages requests liberary and a extract\_urltext() function to fetch HTML content, then uses BeautifulSoup for efficient parsing of content. The title is extracted from <title> tag and article text is extracted using class td-pb-span8 td-main-content.
* The extractdf() function constructs filenames with unique URL\_ID and uses “with open” to write the extracted text to respective files in UTF-8 encoding.

1. Data Transformation:

* The read\_stopwords() function reads stop words from multiple files and returns a set of stop words. The clean\_stopwords() uses NLTK’s word\_tokenize function to tokenizes text and returns only those words not in stop words. The clean\_stopwords\_directory() applies clean\_stopwords() function to all text files in the directory rewriting the files.
* Create\_posneg\_dict() is a function that reads the provided files in MasterDirectiry and returns two sets, one with positive words and other with negative words.

1. Calculate variables:

* The runengine() function runs a loop for each file the directory provided and tokenizes the texts into words (word\_tokenize) and sentences (sentence\_tokenize) and calculating each variable by passing these as parameters to separate functions made incorporated with the formulae provided with the problem.
* The appendtodf() function appends a new columns with calculated variables for each row where URL ID matches to a dataframe.
* Finally the result dataframe for each process running the runengine() function appends its respective output dataframes to a list which will be concatenated to one single dataframe.

1. Data Load:

* The leadoutput() function stores the concatenated dataframe as Output.xlsx file to specified path.

The implementation of multiprocessing and error handling:

The program uses multiprocessing library to create multiple processes equal to the number of logical processors(cores) found in the executing system. The number of files to process are split to each process almost equally. The processes execute the runengine() function parallelly processing separate files list divided for each process.

1. Significant Performance Gains: Parallel processing effectively leverages multiple CPU cores, distributing workload and substantially reducing processing time, especially for large datasets.
2. Shared Data Handling: The use of a managed list to store output dataframes by each process ensures efficient communication and data collection between parallel processes.
3. Error Handling: Error handling is essential to prevent potential issues during multiprocessing, ensuring the integrity of the final output.

**How to run the .py file to generate output(engine.py):**

1. Download the SentimentEngine folder from the drive link and extract from zip file.  
   <https://drive.google.com/drive/folders/1z8DWG6xZwhDRujmelOlypP_9nXEWjME3?usp=drive_link>
2. Run the command (in terminal opened in same directory as folderpath i.e SentimentEngine)

pip install -r requirements.txt

1. Change the folder path variable in the engine.py python source file:

folderpath = r'C:\Users/suwes/SentimentEngine/'

for the running system and execute engine.py. (run: python engine.py in terminal opened in same directory as folderpath). Python should be pre-installed in this directory.

1. After the program completes its execution, the output excel file named “Output.xlsx” is generated with specified structure format in the same directory.

**Required dependencies(python liberaries):**

* multiprocessing
* os
* pandas
* requests
* bs4 and BeautifulSoup
* requests
* string
* nltk
* time