**Blackcoffer**

**Data Extraction and NLP**

**Instructions**

1. **Approach**
2. **Data Extraction**
   1. **Importing Required Libraries:** The script starts by importing the necessary libraries: `pandas` for data manipulation, `requests` for making HTTP requests to fetch web pages, and `BeautifulSoup` from `bs4` for web scraping and HTML parsing.
   2. **Defining the `extract\_article\_text` Function**: Next, a function named `extract\_article\_text` is defined. This function takes a URL as input, sends a GET request to the URL, and then extracts the article title and text from the HTML content using BeautifulSoup. If any errors occur during the extraction process, the function prints an error message and returns `None` for the title and article text.
   3. **Reading Input Data:** The script reads the input data from an Excel file named "Input.xlsx" using `pd.read\_excel` and stores it in a DataFrame named `df`. The input data contains columns for URL IDs and URLs.
   4. **Iterating Through Input Data**: The script iterates through each row of the input DataFrame `df` using `df.iterrows()`. For each row, it extracts the URL ID and URL.
   5. **Extracting Article Text**: The script calls the `extract\_article\_text` function with the URL as an argument to extract the article title and text from the webpage. If the article text is successfully extracted, it appends the URL ID, URL, title, and article text to a new row in a DataFrame named `article\_df`.
   6. **Saving Output Data:** Finally, the script saves the `article\_df` DataFrame to an Excel file named "Output\_Data\_Structure.xlsx" using `to\_excel` method, with the `index` parameter set to `False` to exclude the DataFrame index from the output Excel file.

**This approach ensures that the article title and text are extracted from each URL in the input Excel file and saved in the specified output format. If any errors occur during the extraction process, the script handles them gracefully and continues processing the remaining URLs.**

1. **Data Analysis & NLP**
   1. **Importing Required Libraries:** The script starts by importing the necessary libraries: `pandas` for data manipulation and `nltk` for natural language processing tasks. It also downloads the VADER lexicon using `nltk.download` to perform sentiment analysis and imports the required functions from `textstat`.
   2. **Reading Output Data:** The script reads the output data from the Excel file "Output\_Data\_Structure.xlsx" into a DataFrame named `output\_df`.
   3. **Defining the `compute\_variables` Function:** Next, a function named `compute\_variables` is defined. This function takes the article text as input and computes various text analysis variables such as sentiment scores, average sentence length, percentage of complex words, FOG index, etc.
   4. **Applying the `compute\_variables` Function:** The `compute\_variables` function is applied to the 'ARTICLE TEXT' column of the `output\_df` DataFrame using the `apply` method. This computes the variables for each article text and returns them as a DataFrame.
   5. **Updating the Output DataFrame:** The computed variables are added as new columns to the `output\_df` DataFrame using the `apply` method with `pd.Series`. This expands the DataFrame with columns for each computed variable.
   6. **Dropping Unnecessary Columns:** Finally, the script drops the columns 'URL\_ID', 'Title', and 'Article\_Text' from the DataFrame using the `drop` method with `axis=1` to indicate columns and `inplace=True` to modify the DataFrame in place.
   7. **Saving Output to CSV:** The modified `output\_df` DataFrame is then saved to a CSV file named "Output.csv" using the `to\_csv` method, with `index=False` to exclude the DataFrame index from the output file.

**This approach ensures that the computed text analysis variables are added to the output DataFrame and saved to a CSV file for further analysis or visualization. The code is structured in a way that makes it easy to understand and maintain.**

1. **How to run the .py file to generate output**
2. Open .py file or .ipynb file in vs code or Jupyter, GoogleColab
3. install the requirement.txt by using command **“pip install -r requirements.txt”** in vs code and by using this command **“!pip install -r requirements.txt”** you can install in Jupyter Notebook
4. Now run the file

**Note: Ensure that the `Output\_Data\_Structure.xlsx` file is in the same directory as the Python script or Jupyter Notebook.**

1. **Include all dependencies required**To fulfil the requirements you need to install requiment.txt file by using following commands
2. **Vs code: “pip install -r requirements.txt”**
3. **Jupyter Notebook:** **“!pip install -r requirements.txt”**
4. **Folder**

* Data\_Extraction.py/.ipynb
* Data\_Analysis.py/.ipynb
* Input.xlsx
* Output.Data.Structure.xlsx
* Output.csv
* Instructions