- First I ran a selenium script for data scraping in which I gave the Input xlsx file provided as input. Then I iterated through the first 2 columns and extracted the URL, then extracted the Title and webpage content and saved it in a file with the same name as that of the URL_ID provided in the xlsx file (same row).
- I iterated through all the rows until they got finished and as I was iterating I saved all the extracted text in a txt file, one for every webpage, all of them stored in a folder.
- I then wrote another script with the previously created folder by data crawling as Input for my model, for sentiment analysis and other functions mentioned in the objective.
- For the model I used the following libraries with their functions

1. `os`:
   - Standard Python library for operating system operations
   - Used for file and directory path handling

2. `re`:
   - Regular expressions library
   - Used in the code for finding personal pronouns in text patterns

3. `string`:
   - Standard Python library for string operations
   - Used to access `punctuation` for text cleaning

4. `nltk` (Natural Language Toolkit):
   - Core NLP operations
   - Used for:
     - Tokenization (breaking text into words and sentences)
     - Stopwords (common words to filter out)
     - Text processing utilities

5. `ssl`:
   - Secure Socket Layer library
   - Used to handle HTTPS certificate verification for NLTK downloads

6. `pathlib` (specifically `Path`):
   - Modern file system path handling
   - Used for iterating through text files in directories

7. `nltk.corpus`:
   - NLTK's collection of text corpora
   - Used to access stopwords dictionary

8. `nltk.tokenize`:
   - NLTK's tokenization utilities
   - Provides `word_tokenize` (splits text into words)

- Provides `sent_tokenize` (splits text into sentences)

9. `openpyxl`:
   - Excel file handling library
   - Used to create and write to Excel (.xlsx) files
   - Specifically uses `Workbook` class for creating Excel workbooks

10. `vaderSentiment`:
    - Sentiment analysis library
    - `SentimentIntensityAnalyzer` class used for:
      - Calculating positive scores
      - Calculating negative scores
      - Calculating compound (polarity) scores
      - Analyzing text sentiment

Methodology Used:

Here's a comprehensive explanation of the text analysis methodology:

Text Analysis Process Overview:
Each article undergoes multiple stages of analysis, starting with cleaning and preprocessing, followed by various measurements and calculations. Here's the detailed breakdown:

1. Text Cleaning
- Standardizes text by converting to lowercase
- Removes stopwords (common words like "the," "is," "at") that don't add meaning
- Breaks text into individual words and sentences
- This cleaning process helps focus on meaningful content

2. Readability Analysis
- Counts syllables by identifying vowel groups in words
- Identifies complex words (words with more than 2 syllables)
- Calculates average sentence length
- Determines the Fog Index (measures reading difficulty using sentence length and complex words)
- Measures average word length
- Helps assess how easy or difficult the text is to read

3. Sentiment Analysis
- Uses VADER (Valence Aware Dictionary for Sentiment Reasoning)
- Measures positive and negative sentiment in text
- Calculates overall polarity (how positive or negative the text is)
- Provides emotional context of the content

4. Content Analysis
- Counts total words and sentences
- Tracks complex word usage
- Identifies personal pronouns (I, we, my, ours, us)
- Calculates average words per sentence

5. Data Processing
- Reads all text files from a specified folder
- Organizes results into structured data

6. Output Generation
  - Creates an Excel spreadsheet with the following format
  - Each row represents one article
  - Columns contain different metrics (sentiment scores, word counts, readability measures)
  - Results are saved in an organized, accessible format

This comprehensive analysis provides insights into both technical aspects (readability, complexity) and emotional aspects (sentiment, personal connection) of the text, making it valuable for content analysis and understanding.

The results for each article are saved in an Excel file, with each row representing one article and columns showing different metrics.

How to run the .py file to generate output

- Firstly run the Datamaking.py with the input as input.xlsx file and the path of a folder to save the text files which will be generated. Make sure you have Selinium and Chromium Webdriver installed for your systems, with the path given in the code.
- Then in the DataAnalysis_VADER.py, give the folder generated by the previous script as input and the Output.xlsx is the output excel file to store the output afters it generated.

Dependencies Required:

Here's a comprehensive list of all dependencies required for both codes:

1. Python Packages (install via pip):

```

nltk

openpyxl

vaderSentiment

selenium

```

2. External Software:

- Brave Browser

- ChromeDriver (matching your Brave browser version)

3. NLTK Data Downloads:

- stopwords

- punkt

4. System Requirements:

- Python 3.x

- SSL certificates for Python

Detailed Installation Commands:

```bash

# Python packages

pip install nltk

pip install openpyxl

pip install vaderSentiment

pip install selenium
```

NLTK downloads (will be handled in code, but can be done manually)

python -m nltk.downloader stopwords

python -m nltk.downloader punkt

```

Additional Setup:

1. Download ChromeDriver:

   - Must match your Brave browser version

   - Place in accessible location

   - Update path in code

2. Required File Structure:

   - Input Excel file with URLs

   - Output folder for text files

   - Output location for analysis Excel file
```