**INSTRUCTIONS**

1. **import requests**

**from bs4 import BeautifulSoup**

**import openpyxl**

**import os**

**from textblob import TextBlob**

**import textstat**

First we import the following libraries that we will be needing in our program.

* Requests
* Import BeautifulSoup from bs4 for web crawling.
* Import openpyxl library for reading the excel input file
* Import os library for communicating with the operating system and output directory.
* Import textblob library for sentiment analysis
* Import textstat library for other variables.

**Note: a)**You might need to install some of the libraries if they are not on your python editor using mange packages option from tools.

**b)** There might be some error in some of the variables for textstat library. There you will need to introduce a new function to define those variables.

1. input\_file\_path = r'C:\Users\gajen\Downloads\Input.xlsx'

workbook = openpyxl.load\_workbook(input\_file\_path)

worksheet = workbook.active

Using these statements, we read the input file and open it to provide a a handle for the data where we can work on the data.

1. output\_dir = r'C:\Users\gajen\Downloads\Extracted\_Articles'

os.makedirs(output\_dir, exist\_ok=True

output\_workbook = openpyxl.Workbook()

output\_worksheet = output\_workbook.active

Here we are communicating with output directory using os library.

1. # Add column headers to the output worksheet

output\_worksheet.append([

"URL\_ID",

"Sentiment\_Polarity",

"Sentiment\_Subjectivity",

"FOG\_Index",

"Complex\_Word\_Count",

"Word\_Count",

"Syllables\_Per\_Word",

"Personal\_Pronouns",

"Avg\_Word\_Length",

"Avg\_Sentence\_Length",

"Avg\_Words\_Per\_Sentence"

])

Here we are setting the layout for the output directory. Above are the names for the headers for the output directory.

1. for row in worksheet.iter\_rows(values\_only=True):

url\_id = row[0]

url = row[1]

Here we are iterating the rows for url\_id and url using the iter\_rows .

1. if not url.startswith("https"):

print(f"Invalid URL: {url}")

continue

Here we are setting the criteria for reading the url. If the content in a particular row in URL column does not start with https, then skip that data and move onto next using them continue.

1. response = requests.get(url)

Here we are requesting for a url using the requests library.

1. if response.status\_code == 200:

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.text, 'html.parser')

We take in the data when only the status code is 200 or we don’t take it. Then we use the BeautifulSoup library for extracting the data from the URL.

The data from the url is stored in the response variable.

Then we use the html parser for parsing the html data.

1. # Extract the text from the HTML

text = soup.get\_text()

title = soup.find('h1').text if soup.find('h1') else "Title not found"#for extracting the title of the article

content = "\n".join(p.text for p in soup.find\_all('p'))#for extracting the content of article

Here we are extracting the text, h1 and paragraphs from the urls and store them tpo external files.

1. # Perform sentiment analysis using TextBlob library

blob = TextBlob(text)

sentiment = blob.sentiment

sentiment\_polarity = sentiment.polarity

sentiment\_subjectivity = sentiment.subjectivity

# Calculate FOG index using textstat library

fog\_index = textstat.gunning\_fog(text)

# Calculate complex word count

def calculate\_complex\_word\_count(text):

complex\_word\_count = sum(1 for word in text.split() if textstat.syllable\_count(word) > 3)

return complex\_word\_count

complex\_word\_count = calculate\_complex\_word\_count(text)

# Calculate personal pronouns count

def calculate\_personal\_pronouns(text):

personal\_pronouns = ["I", "me", "my", "mine", "myself", "you", "your", "yours", "yourself", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "we", "us", "our", "ours", "ourselves", "they", "them", "their", "theirs", "themselves"]

personal\_pronoun\_count = sum(1 for word in text.split() if word.lower() inpersonal\_pronouns)

return personal\_pronoun\_count

personal\_pronouns = calculate\_personal\_pronouns(text)

# Calculate average word length

avg\_word\_length = textstat.avg\_letter\_per\_word(text)

# Calculate average sentence length. Here we are initiatinga function of our own.

def calculate\_avg\_sentence\_length(text):

sentences = text.split('.')

words = text.split()

average\_length = len(words) / len(sentences)

return average\_length

avg\_sentence\_length = calculate\_avg\_sentence\_length(text)

# Calculate average words per sentence. Here we are initiating a function of our own.

def calculate\_avg\_words\_per\_sentence(text):

sentences = text.split('.')

words = text.split()

avg\_words = len(words) / len(sentences)

return avg\_words

avg\_words\_per\_sentence = calculate\_avg\_words\_per\_sentence(text)

# Append the data to the output worksheet

output\_worksheet.append([

url\_id,

sentiment\_polarity,

sentiment\_subjectivity,

fog\_index,

complex\_word\_count,

textstat.lexicon\_count(text, removepunct=True),

textstat.syllable\_count(text),

personal\_pronouns,

avg\_word\_length,

avg\_sentence\_length,

avg\_words\_per\_sentence

])

Here we are extracting all the variables.