In [1]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def extract_article(url):
    # Send a GET request to the URL
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Extract the title and text
        title = soup.title.text.strip() if soup.title else "No title found"

        # Find all paragraphs in the article and join them
        paragraphs = [p.text.strip() for p in soup.find_all('p')]
        text = '\n'.join(paragraphs)

        return title, text
    else:
        print(f"Error: Unable to fetch content from {url}")
        return None, None

# Read the Excel file with URLs
excel_file_path = 'Input.xlsx'  # Change this to your Excel file path
df = pd.read_excel(excel_file_path)

# Create empty columns for title and text
df['Title'] = ''
df['Text'] = ''

# Extract title and text for each URL
for index, row in df.iterrows():
    url = row['URL']
    title, text = extract_article(url)
    df.at[index, 'Title'] = title
    df.at[index, 'Text'] = text


# Save the updated DataFrame to a new Excel file
output_file_path = 'output.xlsx'
df.to_excel(output_file_path, index=False)

print(f"Extraction complete. Results saved to {output_file_path}")
```

```
Error: Unable to fetch content from https://insights.blackcoffer.com/how-neural-networks
-can-be-applied-in-various-areas-in-the-future/
Error: Unable to fetch content from https://insights.blackcoffer.com/covid-19-environmen
tal-impact-for-the-future/
Extraction complete. Results saved to output.xlsx
```

In [ ]:
```
pip install pandas openpyxl textblob
```

In [3]:
```
import pandas as pd
from textblob import TextBlob

def analyze_sentiment(text):
    blob = TextBlob(text)
    sentiment_polarity = blob.sentiment.polarity

    if sentiment_polarity > 0:
        sentiment = "Positive"
```

```
    elif sentiment_polarity < 0:
        sentiment = "Negative"
    else:
        sentiment = "Neutral"

    return sentiment, sentiment_polarity

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply sentiment analysis to each row in the DataFrame
df['Sentiment'], df['Sentiment Polarity'] = zip(*df['Text'].apply(analyze_sentiment).tol

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_sentiment_analysis.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_sentiment_analysis.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Sentiment analysis results saved to {output_excel_file}")
```

```
Sentiment analysis results saved to output_sentiment_analysis.xlsx
```

In [ ]:
```
import nltk
nltk.download('punkt')
```

In [4]:
```
import re
import pandas as pd
from textblob import TextBlob

def calculate_gunning_fog_index(text):
    sentences = TextBlob(text).sentences
    total_words = 0
    complex_words = 0

    for sentence in sentences:
        words = sentence.words
        total_words += len(words)

        for word in words:
            if len(re.findall(r'\b\w+\b', str(word))) >= 3:  # Check if the word has thr
                complex_words += 1

    if total_words > 0:
        fog_index = 0.4 * ((total_words / len(sentences)) + (100 * (complex_words / tota
        return fog_index
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply Gunning Fog Index calculation to each row in the DataFrame
```

```
df['Gunning Fog Index'] = df['Text'].apply(calculate_gunning_fog_index)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_gunning_fog_index.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_gunning_fog_index.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)

print(f"Gunning Fog Index results saved to {output_excel_file}")
```

```
Gunning Fog Index results saved to output_gunning_fog_index.xlsx
```

In [5]:

```
import pandas as pd
from textblob import TextBlob

def calculate_average_sentence_length(text):
    sentences = TextBlob(text).sentences
    total_sentences = len(sentences)

    if total_sentences > 0:
        total_words = sum(len(sentence.words) for sentence in sentences)
        avg_sentence_length = total_words / total_sentences
        return avg_sentence_length
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply average sentence length calculation to each row in the DataFrame
df['Average Sentence Length'] = df['Text'].apply(calculate_average_sentence_length)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_avg_sentence_length.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_avg_sentence_length.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)

print(f"Average Sentence Length results saved to {output_excel_file}")
```

```
Average Sentence Length results saved to output_avg_sentence_length.xlsx
```

In [6]:

```
import pandas as pd
from textblob import TextBlob

def calculate_subjectivity_score(text):
    blob = TextBlob(text)
    subjectivity = blob.sentiment.subjectivity
    return subjectivity

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'
```

```
# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply subjectivity score calculation to each row in the DataFrame
df['Subjectivity Score'] = df['Text'].apply(calculate_subjectivity_score)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_subjectivity_score.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_subjectivity_score.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Subjectivity Score results saved to {output_excel_file}")
```

```
Subjectivity Score results saved to output_subjectivity_score.xlsx
```

In [7]:

```
import pandas as pd
import re
from textblob import TextBlob

def calculate_percentage_of_complex_words(text):
    blob = TextBlob(text)

    # Count the total number of words and the number of complex words (three or more syl
    words = blob.words
    total_words = len(words)
    complex_words = sum(len(re.findall(r'\b\w+\b', str(word))) >= 3 for word in words)

    # Calculate the percentage of complex words
    if total_words > 0:
        percentage_complex_words = (complex_words / total_words) * 100
        return percentage_complex_words
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply percentage of complex words calculation to each row in the DataFrame
df['Percentage of Complex Words'] = df['Text'].apply(calculate_percentage_of_complex_wor

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_percentage_complex_words.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_percentage_complex_words.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Percentage of Complex Words results saved to {output_excel_file}")
```

```
Percentage of Complex Words results saved to output_percentage_complex_words.xlsx
```

In [8]:
```
import pandas as pd
from textblob import TextBlob

def calculate_average_word_count(text):
    blob = TextBlob(text)
    words = blob.words
    word_count = len(words)
    return word_count

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply average word count calculation to each row in the DataFrame
df['Average Word Count'] = df['Text'].apply(calculate_average_word_count)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_average_word_count.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_average_word_count.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)



print(f"Average Word Count results saved to {output_excel_file}")
```

```
Average Word Count results saved to output_average_word_count.xlsx
```

In [9]:
```
import pandas as pd
import re
from textblob import TextBlob

def calculate_complex_word_count(text):
    blob = TextBlob(text)

    # Count the number of complex words (three or more syllables)
    words = blob.words
    complex_words_count = sum(len(re.findall(r'\b\w+\b', str(word))) >= 3 for word in wo

    return complex_words_count

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply complex word count calculation to each row in the DataFrame
df['Complex Word Count'] = df['Text'].apply(calculate_complex_word_count)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_complex_word_count.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_complex_word_count.xlsx'
```

```
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Complex Word Count results saved to {output_excel_file}")
```

```
Complex Word Count results saved to output_complex_word_count.xlsx
```

In [10]:

```
import pandas as pd
from textblob import TextBlob

def calculate_word_count(text):
    blob = TextBlob(text)
    words = blob.words
    word_count = len(words)
    return word_count

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply word count calculation to each row in the DataFrame
df['Word Count'] = df['Text'].apply(calculate_word_count)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_word_count.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_word_count.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Word Count results saved to {output_excel_file}")
```

```
Word Count results saved to output_word_count.xlsx
```

In [11]:

```
import pandas as pd
from textblob import TextBlob

def calculate_average_word_length(text):
    blob = TextBlob(text)
    words = blob.words

    # Calculate total characters and total words
    total_characters = sum(len(word) for word in words)
    total_words = len(words)

    # Calculate average word length
    if total_words > 0:
        avg_word_length = total_characters / total_words
        return avg_word_length
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply average word length calculation to each row in the DataFrame
```

```
df['Average Word Length'] = df['Text'].apply(calculate_average_word_length)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_average_word_length.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_average_word_length.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)

print(f"Average Word Length results saved to {output_excel_file}")
```

```
Average Word Length results saved to output_average_word_length.xlsx
```

In [ ]:
```
pip install pyphen
```

In [12]:
```
import pandas as pd
from textblob import TextBlob
import pyphen

def calculate_syllables_per_word(text):
    blob = TextBlob(text)
    words = blob.words

    # Initialize Pyphen for syllable counting
    dic = pyphen.Pyphen(lang='en')

    # Calculate total syllables and total words
    total_syllables = sum(len(dic.inserted(word).split('-')) for word in words)
    total_words = len(words)

    # Calculate average syllables per word
    if total_words > 0:
        avg_syllables_per_word = total_syllables / total_words
        return avg_syllables_per_word
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply syllables per word calculation to each row in the DataFrame
df['Syllables Per Word'] = df['Text'].apply(calculate_syllables_per_word)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_syllables_per_word.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_syllables_per_word.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Syllables Per Word results saved to {output_excel_file}")
```

```
Syllables Per Word results saved to output_syllables_per_word.xlsx
```

In [13]:
```
import pandas as pd
from textblob import TextBlob

def calculate_positive_score(text):
    blob = TextBlob(text)
    sentiment_polarity = blob.sentiment.polarity

    # You can customize the threshold based on your criteria
    positive_threshold = 0.2

    if sentiment_polarity > positive_threshold:
        return sentiment_polarity
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply positive score calculation to each row in the DataFrame
df['Positive Score'] = df['Text'].apply(calculate_positive_score)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_positive_score.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file = 'output_positive_score.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)




print(f"Positive Score results saved to {output_excel_file}")
```

```
Positive Score results saved to output_positive_score.xlsx
```

In [14]:
```
import pandas as pd
from textblob import TextBlob

def calculate_negative_score(text):
    blob = TextBlob(text)
    sentiment_polarity = blob.sentiment.polarity

    # You can customize the threshold based on your criteria
    negative_threshold = -0.2

    if sentiment_polarity < negative_threshold:
        return sentiment_polarity
    else:
        return 0

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply negative score calculation to each row in the DataFrame
df['Negative Score'] = df['Text'].apply(calculate_negative_score)
```

```
# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_negative_score.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file =  'output_negative_score.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)


print(f"Negative Score results saved to {output_excel_file}")
```

```
Negative Score results saved to output_negative_score.xlsx
```

In [15]:

```
import pandas as pd
from textblob import TextBlob

def calculate_polarity_score(text):
    blob = TextBlob(text)
    sentiment_polarity = blob.sentiment.polarity
    return sentiment_polarity

# Replace 'your_file.xlsx' with the path to your Excel file
input_excel_file = 'output.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(input_excel_file)

# Apply polarity score calculation to each row in the DataFrame
df['Polarity Score'] = df['Text'].apply(calculate_polarity_score)

# Save the updated DataFrame to a new Excel file
output_excel_file = 'output_polarity_score.xlsx'
df.to_excel(output_excel_file, index=False)

# List of columns to exclude
columns_to_exclude = ['URL_ID','URL','Title','Text']

# Exclude specified columns when writing to Excel
output_excel_file =  'output_polarity_score.xlsx'
df.drop(columns=columns_to_exclude).to_excel(output_excel_file, index=False)



print(f"Polarity Score results saved to {output_excel_file}")
```

```
Polarity Score results saved to output_polarity_score.xlsx
```

In [18]:

```
import pandas as pd
import glob

# Specify the path to the folder containing Excel files
folder_path = (r'C:\Users\shita\append')

# Get a list of all Excel files in the specified folder
excel_files = glob.glob(f"{folder_path}/*.xlsx")

# Initialize an empty DataFrame to store the appended data
appended_data = pd.DataFrame()

# Iterate through each Excel file and append its data to the DataFrame
for file in excel_files:
    df = pd.read_excel(file)
    appended_data = pd.concat([appended_data, df], axis=1)
```

```
# Specify the path for the output Excel file
output_excel_file = (r'C:\Users\shita\append_n\blackCofferFinalOutput.xlsx')

# Save the appended data to a new Excel file
appended_data.to_excel(output_excel_file, index=False)

print(f"Appended data saved to {output_excel_file}")
```

```
Appended data saved to C:\Users\shita\append_n\blackCofferFinalOutput.xlsx
```

In [ ]: