Text Analysis Project

September 17, 2023

[ ]:
```
import re
import os
import nltk
import string
import requests
import openpyxl
import syllables
import pandas as pd
from textblob import TextBlob
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize, sent_tokenize
```

[ ]:
```
df=pd.read_csv(r"Output Data Structure.csv")
df.head()
```

[ ]:
```
df.columns
```

[ ]:
```
# Function to extract article text from a URL
def extract_article_text(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')
            article_title = soup.find('h1').text.strip()
            article_text = ""
            for paragraph in soup.find_all('p'):
                article_text += paragraph.text.strip() + "\n"
            return article_title, article_text
        else:
            print(f"Failed to retrieve content from {url}")
    except Exception as e:
        print(f"Error: {e}")
    return None, None

# Function to save article to text file
def save_to_text_file(url_id, article_title, article_text):
    if article_title and article_text:
        file_name = f"{url_id}.txt"
```

```
        with open(file_name, 'w', encoding='utf-8') as file:
            file.write(f"Title: {article_title}\n\n")
            file.write(article_text)
        print(f"Saved {file_name}")
```

[ ]:
```
# Download NLTK data
nltk.download('punkt')


input_excel_file = 'Input.xlsx'
wb = openpyxl.load_workbook(input_excel_file)
sheet = wb.active
```

[ ]:
```
# Assuming URL_ID is in the first column (A) of the Excel file
for row in sheet.iter_rows(min_row=2, values_only=True):
    url_id, url = row[0], row[1]
    article_title, article_text = extract_article_text(url)
    save_to_text_file(url_id, article_title, article_text)

# Load positive and negative words from files
def load_words_from_file(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        return set(file.read().splitlines())

positive_words = load_words_from_file('positive_words.txt')
negative_words = load_words_from_file('negative_words.txt')

# Load stop words from file
stop_words = set(nltk.corpus.stopwords.words('english'))
with open('combined_file.txt', 'r', encoding='utf-8') as file:
    stop_words.update(file.read().splitlines())
```

[ ]:
```
# Create a new Excel file for output
output_excel_file = 'Output.xlsx'
output_wb = openpyxl.Workbook()
output_sheet = output_wb.active

# Create headers for the output sheet
output_sheet.append(["URL_ID", "URL", "POSITIVE SCORE", "NEGATIVE SCORE",␣
 ↪"POLARITY SCORE", "SUBJECTIVITY SCORE",
                     "AVG SENTENCE LENGTH", "PERCENTAGE OF COMPLEX WORDS", "FOG␣
 ↪INDEX",
                     "AVG NUMBER OF WORDS PER SENTENCE", "COMPLEX WORD COUNT",␣
 ↪"WORD COUNT", "SYLLABLE PER WORD",
                     "PERSONAL PRONOUNS", "AVG WORD LENGTH"])
```

```
[ ]: def count_syllables(word):      # Function to calculate the number of syllables␣
      ↪in a word
        return syllables.estimate(word)


    for row in sheet.iter_rows(min_row=2, values_only=True):
        url_id, url = row[0], row[1]


        file_name = f"{url_id}.txt"    # Read the text file corresponding to the␣
      ↪URL_ID

        if os.path.exists(file_name):
            encodings_to_try = ['utf-8', 'latin-1', 'windows-1252']
            text = None

            for encoding in encodings_to_try:
                try:
                    with open(file_name, 'r', encoding=encoding) as file:
                        text = file.read()
                    break
                except UnicodeDecodeError:
                    continue

            if text is None:
                print(f"Failed to read {file_name} with all encodings. Skipping...")
                continue  # Skip processing this file

            # Tokenize the text
            words = word_tokenize(text)
            sentences = sent_tokenize(text)

            # Remove stopwords, and convert to lowercase
            clean_words = [word.lower() for word in words if word.isalpha() and␣
      ↪word.lower() not in stop_words]

            # Calculate positive and negative scores using custom word lists
            positive_score = len([word for word in clean_words if word in␣
      ↪positive_words])
            negative_score = len([word for word in clean_words if word in␣
      ↪negative_words])

            # Calculate polarity and subjectivity scores
            blob = TextBlob(text)
            polarity_score = blob.sentiment.polarity
            subjectivity_score = blob.sentiment.subjectivity
```

```
        # Calculate average sentence length
        avg_sentence_length = len(words) / len(sentences)

        # Calculate percentage of complex words
        complex_words = [word for word in clean_words if count_syllables(word)␣
↪> 2]
        percentage_complex_words = (len(complex_words) / len(clean_words)) * 100

        # Calculate Fog Index
        fog_index = 0.4 * (avg_sentence_length + percentage_complex_words)

        # Calculate average number of words per sentence
        avg_words_per_sentence = len(clean_words) / len(sentences)

        # Calculate complex word count
        complex_word_count = len(complex_words)

        # Calculate word count
        word_count = len(clean_words)

        # Calculate syllables per word
        syllables_per_word = sum(count_syllables(word) for word in clean_words)␣
↪/ len(clean_words)

        # Calculate personal pronouns count
        personal_pronouns_count = len(re.findall(r'\b(I|we|my|ours|us)\b',␣
↪text, re.IGNORECASE))

        # Calculate average word length
        avg_word_length = sum(len(word) for word in clean_words) /␣
↪len(clean_words)

        # Append the results
        output_sheet.append([url_id, url, positive_score, negative_score,␣
↪polarity_score, subjectivity_score,
                             avg_sentence_length, percentage_complex_words,␣
↪fog_index, avg_words_per_sentence,
                             complex_word_count, word_count,␣
↪syllables_per_word, personal_pronouns_count,
                             avg_word_length])

# Save the output Excel file
output_wb.save(output_excel_file)
output_wb.close()

# Close the input Excel file
```

```
wb.close()
```

[ ]: