**Blackcoffer**

**Data Extraction and NLP**

**Test Assignment**

**Libraries:**

1. requests: to get HTML content from a given URL

2. beautifulsoup4: to parse HTML content and analyze it

3. pandas: for creating and handling data

4. re: to sanitize extracted text

5. nltk: used to tokenize words and sentences

**stopwords\_file = open("stopwords.txt", "r")**

**stopwords = stopwords\_file.read().splitlines()**

**stopwords = [word.lower() for word in stopwords]**

# We have listed all the words in stopwords.txt. This code takes these words and turns them into an array.

**stopwords\_set = set(stopwords)**

# Convert array into set for easier lookup

**results\_data = {**

**'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': []**

**}**

# Format for the dataframe that holds the result

**for index, row in inputxlsx.iterrows():**

# Iterate through all the rows of the Excel file. In this loop:

# Fetch the URL ID and URL from the Excel file

**url\_id = row['URL\_ID']**

**url\_link = row['URL']**

# Fetch the HTML content

**result = requests.get(url\_link)**

**result.encoding = 'utf-8'**

**result\_content = result.text**

# Store the text of the HTML source page into a variable

**if len(result\_content) == 0:**

**results\_data['URL\_ID'].append(url\_id)**

**results\_data['URL'].append(url\_link)**

**results\_data['POSITIVE SCORE'].append('Page Empty')**

**results\_data['NEGATIVE SCORE'].append('Page Empty')**

**results\_data['POLARITY SCORE'].append('Page Empty')**

**results\_data['SUBJECTIVITY SCORE'].append('Page Empty')**

**results\_data['AVG SENTENCE LENGTH'].append('Page Empty')**

**results\_data['PERCENTAGE OF COMPLEX WORDS'].append('Page Empty')**

**results\_data['FOG INDEX'].append('Page Empty')**

**results\_data['AVG NUMBER OF WORDS PER SENTENCE'].append('Page Empty')**

**results\_data['COMPLEX WORD COUNT'].append('Page Empty')**

**results\_data['WORD COUNT'].append('Page Empty')**

**results\_data['SYLLABLE PER WORD'].append('Page Empty')**

**results\_data['PERSONAL PRONOUNS'].append('Page Empty')**

**results\_data['AVG WORD LENGTH'].append('Page Empty')**

**continue**

# Parse the content using BeautifulSoup and replace nbsp characters with Unicode spaces

**soup = BeautifulSoup(result\_content, "html.parser")**

**soup.prettify(formatter=lambda s: s.replace(u'\xa0', ' '))**

# Open a file where we will write the content of the website

**with open(file\_path, "w", errors="ignore") as f:**

**for data in soup.find\_all("h1", {"class": "entry-title"}):**

**f.writelines(data.get\_text())**

**for data in soup.find\_all("div", {"class": "td-post-content tagdiv-type"}):**

**f.writelines(data.get\_text())**

# Reopen the file for text analysis

**with open(file\_path, "r", errors="ignore") as fnew:**

**extracted\_text = fnew.read()**

# Tokenize the text into words and sentences using nltk

**words = word\_tokenize(extracted\_text)**

**sentences = sent\_tokenize(extracted\_text)**

# Count the sentences

**sentences\_count = len(sentences)**

# Keep a copy of the original words and convert words to lowercase

**original\_words = words.copy()**

**words = [word.lower() for word in words]**

# Clean the words by removing stopwords

**cleaned\_words = [word for word in words if word not in stopwords\_set]**

# Remove punctuation from words and store them in a separate array

**nonPunct = re.compile('.\*[A-Za-z0-9].\*')**

**cleaned\_words\_nonPunct = [w for w in cleaned\_words if nonPunct.match(w)]**

**words\_nonPunct = [w for w in words if nonPunct.match(w)]**

**words\_count = len(words\_nonPunct)**

# Check if content is empty and return 'Page Empty' as a result

**if words\_count <= 0:**

**results\_data['URL\_ID'].append(url\_id)**

**results\_data['URL'].append(url\_link)**

**results\_data['POSITIVE SCORE'].append('Page Empty')**

**results\_data['NEGATIVE SCORE'].append('Page Empty')**

**results\_data['POLARITY SCORE'].append('Page Empty')**

**results\_data['SUBJECTIVITY SCORE'].append('Page Empty')**

**results\_data['AVG SENTENCE LENGTH'].append('Page Empty')**

**results\_data['PERCENTAGE OF COMPLEX WORDS'].append('Page Empty')**

**results\_data['FOG INDEX'].append('Page Empty')**

**results\_data['AVG NUMBER OF WORDS PER SENTENCE'].append('Page Empty')**

**results\_data['COMPLEX WORD COUNT'].append('Page Empty')**

**results\_data['WORD COUNT'].append('Page Empty')**

**results\_data['SYLLABLE PER WORD'].append('Page Empty')**

**results\_data['PERSONAL PRONOUNS'].append('Page Empty')**

**results\_data['AVG WORD LENGTH'].append('Page Empty')**

**continue**

# Store the number of cleaned words

**cleaned\_words\_count = len(cleaned\_words\_nonPunct)**

# Calculate positive and negative scores

**positive\_score = sum(1 for word in cleaned\_words\_nonPunct if word in positivewords\_set)**

**negative\_score = sum(1 for word in cleaned\_words\_nonPunct if word in negativewords\_set)**

**print(f"Positive Score: {positive\_score}")**

**print(f"Negative Score: {negative\_score}")**

# Calculate polarity score

**polarity\_score = -1**

**if positive\_score + negative\_score != 0:**

**polarity\_score = ((positive\_score - negative\_score) / (positive\_score + negative\_score)) + 0.000001**

**print(f"Polarity Score: {polarity\_score}")**

# Calculate subjectivity score

**subjectivity\_score = ((positive\_score + negative\_score) / cleaned\_words\_count) + 0.000001**

# Calculate average sentence length

**avg\_sentence\_length = words\_count / sentences\_count**

# Count complex words and syllables

**complex\_word\_count = 0**

**total\_syllable\_count = 0**

**vowels = set(['a', 'e', 'i', 'o', 'u'])**

**for word in words\_nonPunct:**

**vowel\_count = sum(1 for c in word if c in vowels)**

**if len(word) > 2 and ((word[-1] == 'd' or word[-1] == 's') and word[-2] == 'e'):**

**vowel\_count -= 1**

**total\_syllable\_count += vowel\_count**

**if vowel\_count > 2:**

**complex\_word\_count += 1**

# Calculate percentage of complex words and fog index

**percentage\_of\_complex\_words = complex\_word\_count / words\_count**

**fog\_index = 0.4 \* (avg\_sentence\_length + percentage\_of\_complex\_words)**

# Calculate syllable count per word

**syllable\_count\_per\_word = total\_syllable\_count / words\_count**

# Count personal pronouns

**personal\_pronouns = ['I', 'we', 'ours', 'my', 'us']**

**personal\_pronoun\_count = sum(1 for word in original\_words if word in personal\_pronouns)**

# Calculate total number of characters and average word length

**total\_char\_count = sum(len(word) for word in words\_nonPunct)**

**avg\_word\_length = total\_char\_count / words\_count**

# Store all the resultant values in the result dictionary

**results\_data['URL\_ID'].append(url\_id)**

**results\_data['URL'].append(url\_link)**

**results\_data['POSITIVE SCORE'].append(positive\_score)**

**results\_data['NEGATIVE SCORE'].append(negative\_score)**

**results\_data['POLARITY SCORE'].append(polarity\_score)**

**results\_data['SUBJECTIVITY SCORE'].append(subjectivity\_score)**

**results\_data['AVG SENTENCE LENGTH'].append(avg\_sentence\_length)**

**results\_data['PERCENTAGE OF COMPLEX WORDS'].append(percentage\_of\_complex\_words)**

**results\_data['FOG INDEX'].append(fog\_index)**

**results\_data['AVG NUMBER OF WORDS PER SENTENCE'].append(avg\_sentence\_length)**

**results\_data['COMPLEX WORD COUNT'].append(complex\_word\_count)**

**results\_data['WORD COUNT'].append(cleaned\_words\_count)**

**results\_data['SYLLABLE PER WORD'].append(syllable\_count\_per\_word)**

**results\_data['PERSONAL PRONOUNS'].append(personal\_pronoun\_count)**

**results\_data['AVG WORD LENGTH'].append(avg\_word\_length)**

# Load the result dictionary into a pandas dataframe to store in an Excel file

**results\_df = pd.DataFrame(results\_data)**

**results\_df.to\_excel(r'C:\S\blackcofferproj\Output.xlsx', sheet\_name='sheet1', index=False)**

Q2. How to run the .py file to generate output

In the working directory, have all the stopwords in a file called ‘stopwords.txt’, which is newline-separated. Do the same for positive words and negative words.