DATA EXTRACTION AND NLP- BY DHANYA

Double-click (or enter) to edit

accessing my drive

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install textstat
!pip install syllables
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting textstat
  Downloading textstat-0.7.3-py3-none-any.whl (105 kB)
                                                                  a 0:00:00
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 25.9 MB/s eta 0:00:00
Installing collected packages: pyphen, textstat
Successfully installed pyphen-0.14.0 textstat-0.7.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting syllables
  Downloading syllables-1.0.7-py3-none-any.whl (15 kB)
Collecting cmudict<2.0.0,>=1.0.11
  Downloading cmudict-1.0.13-py3-none-any.whl (939 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 939.3/939.3 kB 12.8 MB/s eta 0:00:00
Collecting importlib-metadata<6.0.0,>=5.1.0
  Downloading importlib_metadata-5.2.0-py3-none-any.whl (21 kB)
Requirement already satisfied: importlib-resources<6.0.0,>=5.10.1 in /usr/local/lib/python3.10/dist-packages (from cmudict<2.0.0,>=1.0.1
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata<6.0.0,>=5.1.0->syllables) (
Installing collected packages: importlib-metadata, cmudict, syllables
Successfully installed cmudict-1.0.13 importlib-metadata-5.2.0 syllables-1.0.7
```

```
import bs4
import requests
from bs4 import BeautifulSoup
import pandas as pd
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import os
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
import textstat
from nltk.corpus import stopwords
import syllables
import string
import re
```

```
nltk.download('punkt')
nltk.download('vader_lexicon')
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
# Load URLs from input file
df = pd.read_excel('/content/drive/MyDrive/Input.xlsx')
```

Data Extraction part

```
# loop through the URLs and extract the article text
for index, row in df.iterrows():
    url_id = str(row['URL_ID'])
    url = row['URL']
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    text = ""
    for para in soup.find_all('p'):
        text += para.text + "\n"
    # save the text to a file with URL_ID as its name
    with open(url_id + ".txt", "w") as f:
        f.write(text)
```

how texts look like

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
    print(contents)
```

```
!unzip -q "/content/drive/MyDrive/StopWords-20230503T100731Z-001.zip"
```

Removing all stopwords

```
# load the stop words
stop_words = []
with open('/content/StopWords/StopWords_Names.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_Geographic.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_GenericLong.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_DatesandNumbers.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_Currencies.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_Auditor.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()
with open('/content/StopWords/StopWords_Generic.txt',encoding='ISO-8859-1') as f:
    stop_words += f.read().splitlines()

# load the extracted text from the files and clean it
for url_id in df['URL_ID']:
    with open(f'{url_id}.txt', 'r') as f:
        text = f.read().replace('\n', '')
        text_without_stopwords = ' '.join([word for word in text.split() if word.lower() not in stop_words])
    with open(f'{url_id}.txt', 'w') as f:
        f.write(text_without_stopwords)
```

```
filename = "37.txt"
with open(filename, "r") as f:
    contents = f.read()
    print(contents)
```

```
 Ranking customer behaviours business strategyAlgorithmic trading multiple commodities markets, Forex, Metals, Energy, etc.Trading Bot FC
```

```
!unzip -q "/content/drive/MyDrive/MasterDictionary-20230503T104416Z-001.zip"
```

Creating a dictionary of Positive and Negative words and Extracting Derived variables

```
# load the positive and negative words
pos_words = []
with open('/content/MasterDictionary/positive-words.txt', 'r',encoding='ISO-8859-1') as f:
    for line in f:
        word = line.strip()
        if word not in stop_words:
            pos_words.append(word)

neg_words = []
with open('/content/MasterDictionary/negative-words.txt', 'r',encoding='ISO-8859-1') as f:
    for line in f:
        word = line.strip()
        if word not in stop_words:
            neg_words.append(word)

##load the extracted text from the files and clean it
text_list = []
for url_id in df['URL_ID']:
    with open(f'{url_id}.txt', 'r') as f:
        text = f.read().replace('\n', '')
        text_list.append(' '.join([word for word in text.split() if word.lower() not in stop_words]))
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
sia = SentimentIntensityAnalyzer()
positive_scores = []
negative_scores = []
polarity_scores = []
subjectivity_scores = []
for text in text_list:
    tokens = word_tokenize(text)
    pos_score = sum([1 if token in pos_words else 0 for token in tokens])
    neg_score = sum([1 if token in neg_words else 0 for token in tokens]) * -1
    polarity_score = (pos_score - neg_score) / ((pos_score + neg_score) + 0.000001)
    subjectivity_score = (pos_score + abs(neg_score)) / (len(tokens) + 0.000001)
    positive_scores.append(pos_score)
    negative_scores.append(neg_score)
    polarity_scores.append(polarity_score)
    subjectivity_scores.append(subjectivity_score)

# add the scores to the dataframe
df['POSITIVE SCORE'] = positive_scores
df['NEGATIVE SCORE'] = negative_scores
df['POLARITY SCORE'] = polarity_scores
df['SUBJECTIVITY SCORE'] = subjectivity_scores

# print the dataframe
```

```
df = df.rename(columns={'POSITIVE SCORE': 'Positive Score', 'NEGATIVE SCORE': 'Negative Score', 'POLARITY SCORE': 'Polarity Score', 'SUBJECTI
```

Analysis of Readability

```
def calculate_readability(text):
    sentences = nltk.sent_tokenize(text)
    num_sentences = len(sentences)
    num_words = len(nltk.word_tokenize(text))
    avg_sentence_len = num_words / num_sentences
    num_complex_words = len([word for word in nltk.word_tokenize(text) if textstat.syllable_count(word) >= 3])
    percent_complex_words = num_complex_words / num_words * 100
    fog_index = 0.4 * (avg_sentence_len + percent_complex_words)
    return avg_sentence_len, percent_complex_words, fog_index
```

```
# create empty lists to store the scores
avg_sentence_lengths = []
```

```
percent_complex_words = []
fog_indices = []

# loop over each text and calculate the readability scores
for text in text_list:
    avg_sentence_len, percent_complex, fog_index = calculate_readability(text)
    avg_sentence_lengths.append(avg_sentence_len)
    percent_complex_words.append(percent_complex)
    fog_indices.append(fog_index)

# add the new columns to the DataFrame
df['AVG SENTENCE LENGTH'] = avg_sentence_lengths
df['PERCENTAGE OF COMPLEX WORDS'] = percent_complex_words
df['FOG INDEX'] = fog_indices
```

Average Number of Words Per Sentence

```
# calculate average number of words per sentence
total_words = 0
total_sentences = 0
for text in text_list:
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
                                                                              tences)
    total_sentences += len(sentences)

avg_words_per_sentence = total_words / total_sentences
df['AVG NUMBER OF WORDS PER SENTENCE'] = total_words / total_sentences
```

Complex Word Count

```
nltk.download('cmudict')
d = cmudict.dict()
```

```
def nsyl(word):
    return [len(list(y for y in x if y[-1].isdigit())) for x in d.get(word.lower(), [[]])][0]

def count_complex_words(text):
    words = text.split()
    complex_word_count = 0

    for word in words:
        # remove punctuations from the word
        word = word.strip('?,.!')

        # count the number of syllables in the word
        syllable_count = nsyl(word)

        if syllable_count > 2:
            complex_word_count += 1

    return complex_word_count
```

```
 [nltk_data] Downloading package cmudict to /root/nltk_data...
 [nltk_data]   Unzipping corpora/cmudict.zip.
```

```
complex_word_counts = [count_complex_words(text) for text in text_list]
df['COMPLEX WORD COUNT'] = complex_word_counts
```

Word Count

```
def count_words(text):
    # remove punctuation from the text
    text = text.translate(str.maketrans("", "", string.punctuation))

    # tokenize the text into words
    words = word_tokenize(text)

    # remove stop words from the list of words
    words = [word.lower() for word in words if word.lower() not in stopwords.words('english')]

    # count the number of words in the list
    word_count = len(words)

    return word_count


cleaned_counts = [count_words(text) for text in text_list]
df['WORD COUNT'] = cleaned_counts
```

Syllable Count Per Word

```
def count_syllables(word):
    # remove trailing "e" except for words with only one syllable
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
    # count the number of vowels (excluding silent "e")
    num_vowels = len(re.findall(r'[aeiouy]+', word, re.IGNORECASE))

    # handle some exceptions
    if word[-2:] == "ed" or word[-2:] == "es":
        num_vowels -= 1
    if word[-3:] == "ied" or word[-3:] == "ies":
        num_vowels += 1

    # every word has at least one syllable
    return max(num_vowels, 1)


syllables_counts = [count_syllables(text) for text in text_list]
df['SYLLABLE PER WORD'] = syllables_counts
```

Personal Pronouns

```
def count_personal_pronouns(text):
    # define regex pattern to match personal pronouns
    pattern = r"\b(I|we|We|My|my|Ours|Us|ours|us)\b"

    # find all matches of the pattern in the text
    matches = re.findall(pattern, text)

    # return the count of matches
    return len(matches)


pronouns_counts = [count_personal_pronouns(text) for text in text_list]
df['PERSONAL PRONOUNS'] = pronouns_counts
```

Average Word Length

```
def calculate_average_word_length(text):
    # tokenize the text into individual words
    words = text.split()

    # remove any punctuation marks from the words
    words = [word.translate(str.maketrans('', '', string.punctuation)) for word in words]

    # calculate the total number of characters in all the words
    total_characters = sum(len(word) for word in words)

    # calculate the total number of words
    total_words = len(words)

    # divide the total number of characters by the total number of words to get the average word length
    average_word_length = total_characters / total_words

    return average_word_length


avg_counts = [calculate_average_word_length(text) for text in text_list]
df['AVG WORD LENGTH'] = avg_counts
```

```
df
```

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	PER PRO
s.blackcoffer.com/ai-in-healthc...	65	-31	2.823529	0.071217	25.433962	29.302671	21.894653	20.492587	387	1135	2989	
					25	21.769384	14.713724	20.492587	212	749	1819	
kcoffer.com/what-jobs-wil...	68	-34	3.000000	0.084788	17.955224	30.008313	19.185415	20.492587	351	981	2667	
ts.blackcoffer.com/will-machine-...	58	-21	2.135135	0.078685	13.386667	25.000000	15.354667	20.492587	240	817	1997	
blackcoffer.com/will-ai-repla...	50	-22	2.571428	0.059950	17.925373	24.063281	16.795461	20.492587	304	954	2337	
...	...	...	...	...	...	...	...	...	...	...	...	
coffer.com/blockchain-fo...	23	-27	-12.500003	0.074184	18.216216	28.189911	18.562451	20.492587	160	569	1481	
kcoffer.com/the-future-of...	38	-11	1.814815	0.058824	21.358974	25.570228	18.771681	20.492587	206	691	1744	
ickcoffer.com/big-data-anal...	29	-44	-4.866667	0.084884	14.098361	27.209302	16.523065	20.492587	239	712	1788	
ckcoffer.com/business-anal...	29	-3	1.230769	0.068522	29.187500	32.334047	24.608619	20.492587	131	393	1131	
coffer.com/challenges-an...	33	-38	-14.200003	0.095174	12.032258	26.541555	15.429525	20.492587	198	621	1630	

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
from google.colab import files

# assume your dataframe is stored in a variable named df
df.to_excel('my_data.xlsx', index=False)

# download the xlsx file
files.download('my_data.xlsx')
```

Double-click (or enter) to edit

✓ 0s completed at 3:24 AM ● ×

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

Automatic saving failed. This file was updated remotely or in another tab. Show diff