Data-X Spring 2019: Homework 7

Webscraping

In this homework, you will do some exercises with web-scraping.

Name: Isha Mangal

SID: 3031911156

Fun with Webscraping & Text manipulation

1. Statistics in Presidential Debates

Your first task is to scrape Presidential Debates from the Commission of Presidential Debates website: https://www.debates.org/voter-education/debate-transcripts/ (https://www.debates.org/voter-education/debate-transcripts/)

To do this, you are not allowed to manually look up the URLs that you need, instead you have to scrape them. The root url to be scraped is the one listed above, namely: https://www.debates.org/voter-education/debate-transcripts/ (https://www.debates.org/voter-education/debate-transcripts/)

1. By using `requests` and `BeautifulSoup` find all the links / URLs on the website that links to transcriptions of **First Presidential Debates** from the years [1988, 1984, 1976, 1960]. In total you should find 4 links / URLs that fulfill this criteria. **Print the urls.**
2. When you have a list of the URLs your task is to create a Data Frame with some statistics (see example of output below):

    A. Scrape the title of each link and use that as the column name in your Data Frame.
    B. Count how long the transcript of the debate is (as in the number of characters in transcription string). Feel free to include `\` characters in your count, but remove any breakline characters, i.e. `\n`. You will get credit if your count is +/- 10% from our result.
    C. Count how many times the word **war** was used in the different debates. Note that you have to convert the text in a smart way (to not count the word **warranty** for example, but counting **war.**, **war!**, **war,** or **War** etc.
    D. Also scrape the most common used word in the debate, and write how many times it was used. Note that you have to use the same strategy as in C in order to do this.

    Print your final output result.

Tips:

In order to solve the questions above, it can be useful to work with Regular Expressions and explore methods on strings like `.strip(), .replace(), .find(), .count(), .lower()` etc. Both are very powerful tools to do string processing in Python. To count common words for example I used a `Counter` object and a Regular expression pattern for only words, see example:

```
from collections import Counter
import re

counts = Counter(re.findall(r"[\w']+", text.lower()))
```

Read more about Regular Expressions here: https://docs.python.org/3/howto/regex.html (https://docs.python.org/3/howto/regex.html)

Example output of all of the answers to Question 1.2:

	September 25, 1988: The First Bush-Dukakis Presidential Debate
Debate char length	87488
war_count	
most_common_w	
most_common_w_count	

.

In [79]:
```
import requests
import bs4 as bs
import pandas as pd
```

1:

In [80]:
```
source = requests.get("https://www.debates.org/voter-education/debate-transcript
soup = bs.BeautifulSoup(source.content, features='html.parser')

lst = []
for link in soup.find_all('a'):
    link_str = str(link)
    for date in [1988, 1984, 1976, 1960]:
        if str(date) in link_str and "First" in link_str:
            lst.append(link_str)
lst
```

Out[80]:
```
['<a href="/voter-education/debate-transcripts/september-25-1988-debate-transc
ript/" title="September 25, 1988 Debate Transcript">September 25, 1988: The Fi
rst Bush-Dukakis Presidential Debate</a>',
 '<a href="/voter-education/debate-transcripts/october-7-1984-debate-transcrip
t/" title="October 7, 1984 Debate Transcript">October 7, 1984: The First Reaga
n-Mondale Presidential Debate</a>',
 '<a href="/voter-education/debate-transcripts/september-23-1976-debate-transc
ript/" title="September 23, 1976 Debate Transcript">September 23, 1976: The Fi
rst Carter-Ford Presidential Debate</a>',
 '<a href="/voter-education/debate-transcripts/september-26-1960-debate-transc
ript/" title="September 26, 1960 Debate Transcript">September 26, 1960: The Fi
rst Kennedy-Nixon Presidential Debate</a>']
```

2:

In [81]:
```
titles = []
for link in soup.find_all('a'):
    link_str = str(link)
    for date in [1988, 1984, 1976, 1960]:
        if str(date) in link_str and "First" in link_str:
            titles.append(link.contents[0])
titles
```

Out[81]:
```
['September 25, 1988: The First Bush-Dukakis Presidential Debate',
 'October 7, 1984: The First Reagan-Mondale Presidential Debate',
 'September 23, 1976: The First Carter-Ford Presidential Debate',
 'September 26, 1960: The First Kennedy-Nixon Presidential Debate']
```

In [89]:
```
from collections import Counter
import re
import operator

lst3 = []
lst4 = []
lst5 = []
lst6 = []

for i in lst:
    wordcount = {}
    source = requests.get(i)
    soup = bs.BeautifulSoup(source.content, features='html.parser')
    x = soup.find(id='content-sm')
    text = x.text
    newtext = text.replace('\n', '')
    length = len(newtext)
    lst3.append(length)

    wartext = len(re.findall(r'[Ww][Aa][Rr][Ss]*\b', text))
    lst4.append(wartext)

    for word in text.lower().split():
        if word in wordcount:
            wordcount[word] += 1
        else:
            wordcount[word] = 1

    wo = max(wordcount.items(), key=operator.itemgetter(1))[0]
    lst6.append(wordcount[wo])
    lst5.append(wo)

print(lst3)
print(lst4)
print(lst5)
print(lst6)
```

```
~/anaconda3/lib/python3.6/site-packages/requests/api.py in get(url, params, **
kwargs)
     70
     71     kwargs.setdefault('allow_redirects', True)
---> 72     return request('get', url, params=params, **kwargs)
     73
     74

~/anaconda3/lib/python3.6/site-packages/requests/api.py in request(method, ur
l, **kwargs)
     56     # cases, and look like a memory leak in others.
     57     with sessions.Session() as session:
---> 58         return session.request(method=method, url=url, **kwargs)
     59
     60

~/anaconda3/lib/python3.6/site-packages/requests/sessions.py in request(self,
 method, url, params, data, headers, cookies, files, auth, timeout, allow_redi
rects, proxies, hooks, stream, verify, cert, json)
```

In [88]:
```
import numpy as np
table = pd.DataFrame([lst3, lst4, lst5, lst6], titles)
table.index = ["Debate char length",
            "war_count", "most_common_w",
            "most_common_w_count"]
table.set_index = ["a", "b", "c", "d"]
table.rename(index=str,
             columns={0: 'September 25, 1988: The First Bush-Dukakis Presidentia
                1: 'October 7, 1984: The First Reagan-Mondale Presidential Debat
                2: 'September 23, 1976: The First Carter-Ford Presidential Debat
                3: 'September 26, 1960: The First Kennedy-Nixon Presidential Deb
```

Out[88]:

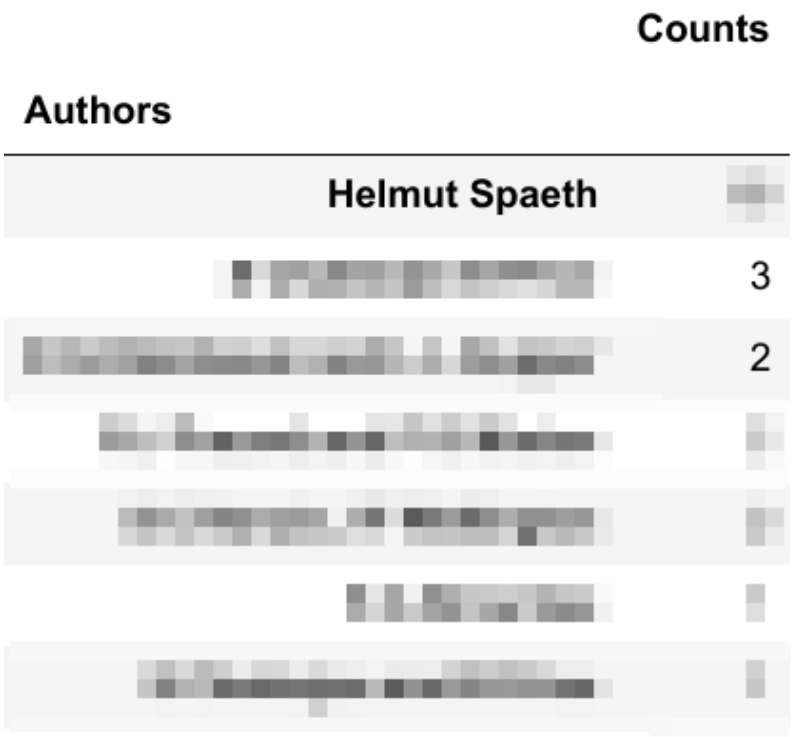
Debate char length
war_count
most_common_w
most_common_w_count

2. Download and read in specific line from many data sets

Scrape the first 27 data sets from this URL http://people.sc.fsu.edu/~jburkardt/datasets/regression/ (http://people.sc.fsu.edu/~jburkardt/datasets/regression/) (i.e.`x01.txt` - `x27.txt`). Then, save the 5th line in each data set, this should be the name of the data set author (get rid of the # symbol, the white spaces and the comma at the end).

Count how many times (with a Python function) each author is the reference for one of the 27 data sets. Showcase your results, sorted, with the most common author name first and how many times he appeared in data sets. Use a Pandas DataFrame to show your results, see example. **Print your final output result.**

Example output of the answer for Question 2:



In [84]:
```
url = "http://people.sc.fsu.edu/~jburkardt/datasets/regression/"
source = requests.get(url)
soup = bs.BeautifulSoup(source.content, features='html.parser')
```

In [85]:
```
total_data = 27
sets = []
for x in soup.find_all('a'):
    if total_data == 0:
        break
    str_x = str(x)
    splt_link = str_x.split('.txt')
    if (len(splt_link)>1):
        file = str_x.split('"')[1]
        sets.append(file)
        total_data = total_data - 1
```

In [86]:
```
print('Total # of scraped datasets:', len(sets))
author = {}
for data in sets:
    url_data = url + data
    source = requests.get(url_data)
    soup = bs.BeautifulSoup(source.content, features='html.parser')
    text = str(soup.text)
    author_split = text.split('\n')[4].split('#')[1].split(',')
    for a in author_split:
        a = a.strip()
        if a != '':
            if a in author:
                author[a] += 1
            else:
                author[a] = 1
```

```
Total # of scraped datasets: 27
```

In [87]:
```
df = pd.DataFrame({'Authors': list(author.keys()),
    'Count': list(author.values())}).set_index('Authors').sort_values(by='Count'
    ascending=False)
```

In [ ]:
```

```