https://github.com/anish-saha/IEOR135-SP19/tree/master/HW7

Data-X Spring 2019: Homework 7

Webscraping

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

Name:

Anish Saha

SID:

26071616

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	[redacted]
Debate char length	87488	[redacted]
war_count		[redacted]
most_common_w	[redacted]	[redacted]
most_common_w_count	[redacted]	[redacted]

.

In [1]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
from __future__ import division, print_function
from urllib.request import urlopen
import requests, re
import numpy as np
import bs4 as bs
import pandas as pd
from collections import Counter
```

In [2]:
```
# Part 1
source = requests.get("https://www.debates.org/voter-education/debate-transcripts/")
soup = bs.BeautifulSoup(source.content, features='html.parser')
links = soup.find_all('a')
df =  pd.DataFrame(columns=["Transcript Name", "URL"])
t = "https://www.debates.org"
p = re.compile("The First")
years = ['1988', '1984', '1976', '1960']
i = -1

for l in links:
    if pd.isnull(p.search(l.text)):
        p = re.compile("The First")
    else:
        i = i + 1
        df.loc[i] = [l.text, t + l.get("href")]

df = df[df['Transcript Name'].str.contains('|'.join(years))]
HTML(df.to_html())
```

Out[2]:

	Transcript Name	URL
6	September 25, 1988: The First Bush-Dukakis Pre...	https://www.debates.org/voter-education/debate...
7	October 7, 1984: The First Reagan-Mondale Pres...	https://www.debates.org/voter-education/debate...
8	September 23, 1976: The First Carter-Ford Pres...	https://www.debates.org/voter-education/debate...
9	September 26, 1960: The First Kennedy-Nixon Pr...	https://www.debates.org/voter-education/debate...

In [3]:
```
# Part 2a
df_copy = df.copy()
df_copy = df_copy.drop(labels="URL", axis=1)
transcripts = df_copy.values.T.tolist()
res = pd.DataFrame(columns=transcripts[0])
headers = pd.DataFrame({ "Detail": ["Debate Character Length",
"War Count", "Most Common Word", "Most Common Word Count"] })
res = headers.join(res)
res
```

Out[3]:

	Detail	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
0	Debate Character Length	NaN	NaN	NaN	NaN
1	War Count	NaN	NaN	NaN	NaN
2	Most Common Word	NaN	NaN	NaN	NaN
3	Most Common Word Count	NaN	NaN	NaN	NaN

In [4]:
```
# Part 2b
df_copy = df.copy()
df_copy = df_copy.drop(labels = "Transcript Name", axis = 1)
urls = df_copy.values.T.tolist()
d_len = []

for u in urls[0]:
    p = requests.get(u)
    d_len.append(len(p.text))

res.loc[0:0, 1:] = d_len
res
```

Out[4]:

	Detail	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
0	Debate Character Length	104061	103787	97345	74862
1	War Count	NaN	NaN	NaN	NaN
2	Most Common Word	NaN	NaN	NaN	NaN
3	Most Common Word Count	NaN	NaN	NaN	NaN

In [5]:
```
# Part 2c
w_count = []

for u in urls[0]:
    p = requests.get(u)
    war_sum = len(re.findall(r"(war)\W|(War)\W|(wars)\W|(Wars)
\W", p.text))
    w_count.append(war_sum)

res.loc[1:1,1:] = w_count
res
```

Out[5]:

	Detail	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
0	Debate Character Length	104061	103787	97345	74862
1	War Count	14	3	7	3
2	Most Common Word	NaN	NaN	NaN	NaN
3	Most Common Word Count	NaN	NaN	NaN	NaN

In [6]:
```
# Part 2d
most_common_word = []
most_common_count = []

for u in urls[0]:
    p = requests.get(u)
    word = Counter(p.text.split(" ")).most_common(1)[0][0]
    most_common_word.append(word)
    c = Counter(p.text.split(" ")).most_common(1)[0][1]
    most_common_count.append(c)

res.loc[2:2,1:] = most_common_word
res.loc[3:3,1:] = most_common_count
res
```

Out[6]:

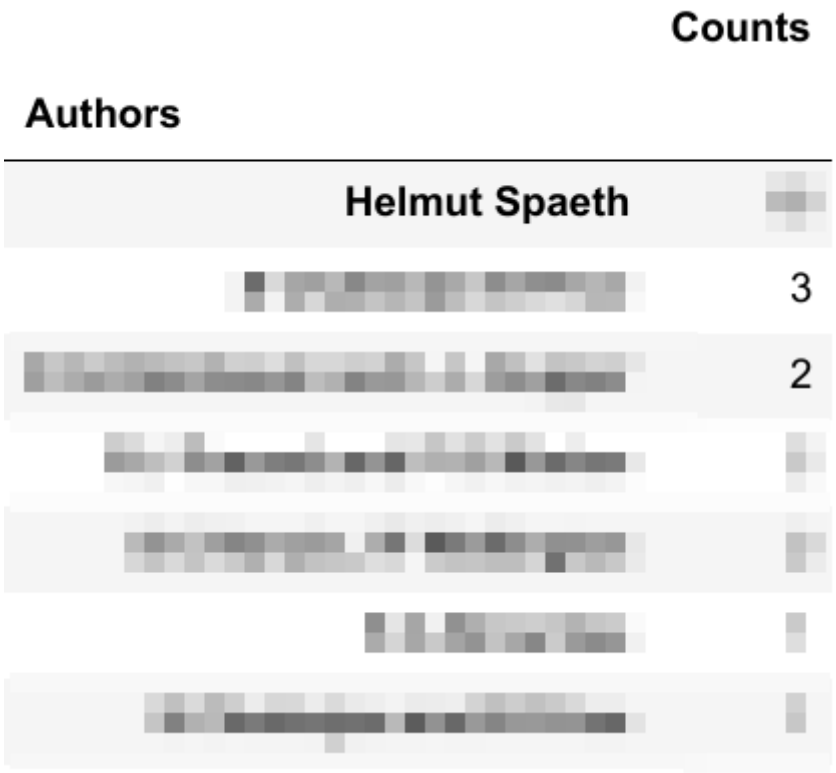
	Detail	September 25, 1988: The First Bush-Dukakis Presidential Debate	October 7, 1984: The First Reagan-Mondale Presidential Debate	September 23, 1976: The First Carter-Ford Presidential Debate	September 26, 1960: The First Kennedy-Nixon Presidential Debate
0	Debate Character Length	104061	103787	97345	74862
1	War Count	14	3	7	3
2	Most Common Word	the	the	the	the
3	Most Common Word Count	759	776	823	723

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 [7]:
```
source = requests.get('http://people.sc.fsu.edu/~jburkardt/datas
ets/regression/').content
soup = bs.BeautifulSoup(source, features='html.parser')
links = soup.find('table').find_all('a')
urls = ['http://people.sc.fsu.edu/~jburkardt/datasets/regressio
n/'+l.get('href') for l in links][6:33]
urls
```

```
Out[7]: ['http://people.sc.fsu.edu/~jburkardt/datasets/regression/x01.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x02.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x03.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x04.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x05.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x06.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x07.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x08.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x09.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x10.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x11.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x12.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x13.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x14.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x15.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x16.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x17.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x18.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x19.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x20.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x21.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x22.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x23.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x24.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x25.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x26.
        txt',
         'http://people.sc.fsu.edu/~jburkardt/datasets/regression/x27.
        txt']
```

In [8]:
```
authors = []

for u in urls:
    textfile = urlopen(u)
    file = list(textfile)
    author = file[4]
    authors.append(author)

a = pd.DataFrame(index=Counter(authors).keys())
a['counts'] = Counter(authors).values()
a = a.sort_values(by='counts', ascending=False)
a
```

Out[8]:

	counts
b'# Helmut Spaeth,\n'	16
b'# S Chatterjee, B Price,\n'	3
b'# R J Freund and P D Minton,\n'	2
b'# D G Kleinbaum and L L Kupper,\n'	2
b'# S C Narula, J F Wellington,\n'	2
b'# K A Brownlee,\n'	1
b'# S Chatterjee and B Price,\n'	1