Data-X Spring 2019: Homework 7

Webscraping

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

Name: Nicholas Miller

SID: 3033396225

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

.

In [187]:

```
import requests
import bs4 as bs
import pandas as pd
from collections import Counter
import re

source = requests.get(" https://www.debates.org/voter-education/debate-transcrip
ts/").content
soup = bs.BeautifulSoup(source, 'html.parser')
links = soup.find_all('a')
for i in range(len(links)):
    links[i] = "https://www.debates.org" + links[i].get('href')

relevent_links = []
for i in links:
    if 'september-25-1988' in i or 'september-26-1960' in i or 'september-23-197
6' in i or 'october-7-1984' in i:
        relevent_links.append(i)

print(relevent_links[:4],'\n')


def structure(num_debate):
    debate = bs.BeautifulSoup(requests.get(relevent_links[num_debate]).content,'
html.parser')
    title = ''

    for i in debate.find_all('strong'):
        title = debate.title.text + ': ' + i.text
    title = title.replace("CPD: ",'')
    title = title.replace(" Debate Transcript", '')

    debate = debate.body.text.replace("\n",'')
    useless,applause,debate = debate.partition(':')
    debate,applause,useless = debate.rpartition("COPYRIGHT")
    return debate, title

debate_1988 = structure(0)
debate_1984 = structure(1)
debate_1976 = structure(2)
debate_1960 = structure(3)

char_lengths = [len(debate_1988[0]), len(debate_1984[0]), len(debate_1976[0]), l
en(debate_1960[0])]
column_values = [debate_1988[1],debate_1984[1],debate_1976[1], debate_1960[1]]
index = ["Debate Character Count", "'War' Usage Count", "Most Common Word", "Mos
t Common Word Count"]
war_count_1988 = len(re.findall(r'\b[w,W]ar.?\b', debate_1988[0]))
war_count_1984 = len(re.findall(r'\b[w,W]ar.?\b', debate_1984[0]))
war_count_1976 = len(re.findall(r'\b[w,W]ar.?\b', debate_1976[0]))
war_count_1960 = len(re.findall(r'\b[w,W]ar.?\b', debate_1960[0]))
war_counts = [war_count_1988,war_count_1984,war_count_1976,war_count_1960]


most_common_words = [Counter(debate_1988[0].split()).most_common()[0], Counter(d
ebate_1984[0].split()).most_common()[0], Counter(debate_1976[0].split()).most_co
mmon()[0], Counter(debate_1960[0].split()).most_common()[0]]
most_common_word_text = [most_common_words[0][0],most_common_words[1][0], most_c
ommon_words[2][0], most_common_words[3][0]]
most_common_word_count = [most_common_words[0][1],most_common_words[1][1], most_
common_words[2][1], most_common_words[3][1]]


data = {index[0]: char_lengths,index[1]:war_counts, index[2]:most_common_word_te
xt,index[3]:most_common_word_count}
df = pd.DataFrame(data, index =column_values)
df
```

```
['https://www.debates.org/voter-education/debate-transcripts/septemb
er-25-1988-debate-transcript/', 'https://www.debates.org/voter-educa
tion/debate-transcripts/october-7-1984-debate-transcript/', 'https:/
/www.debates.org/voter-education/debate-transcripts/september-23-197
6-debate-transcript/', 'https://www.debates.org/voter-education/deba
te-transcripts/september-26-1960-debate-transcript/']
```

Out[187]:

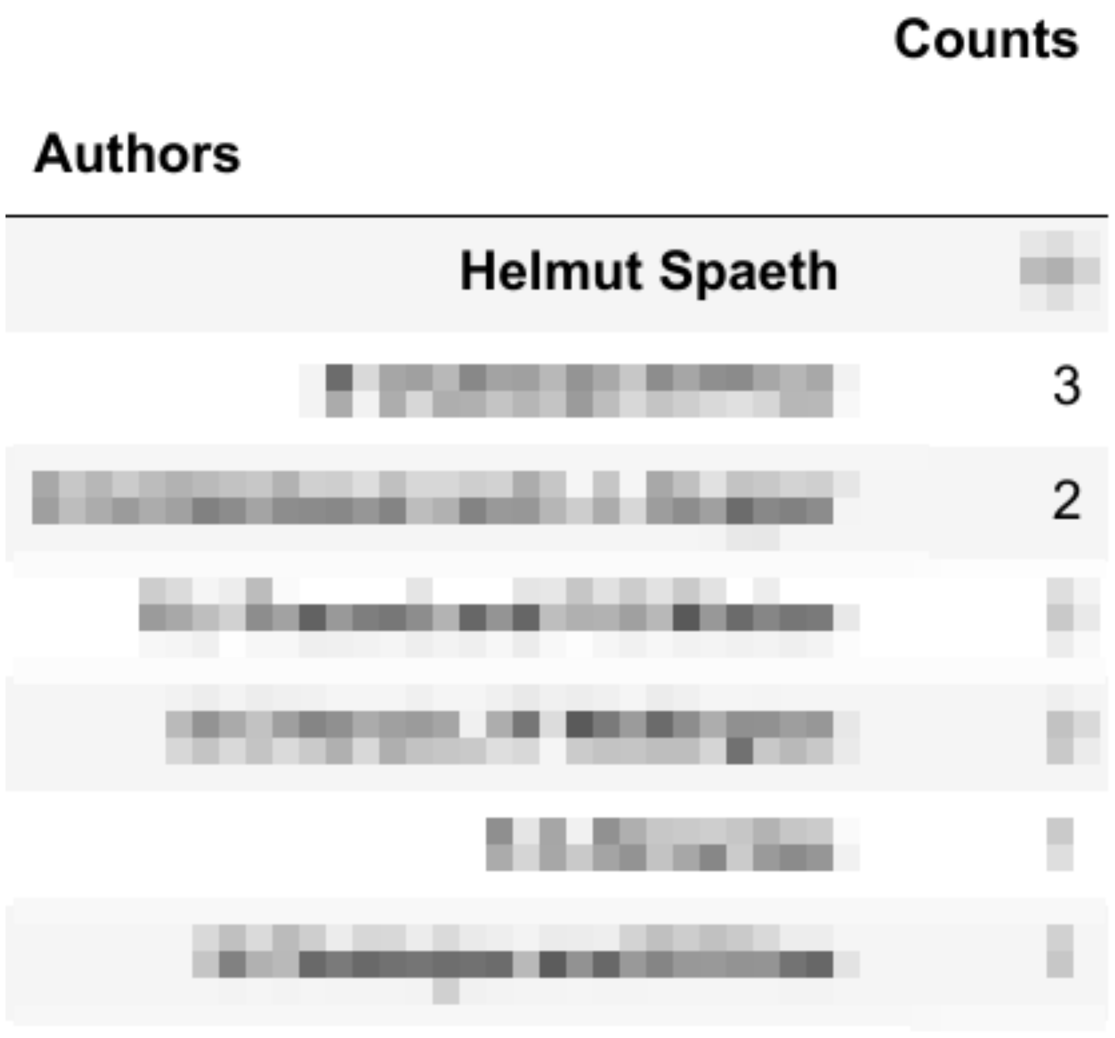
	Debate Character Count	'War' Usage Count	Most Common Word	Most Common Word Count
September 25, 1988: The First Bush-Dukakis Presidential Debate	87387	14	the	759
October 7, 1984: The First Reagan-Mondale Presidential Debate	86403	3	the	776
September 23, 1976: The First Carter-Ford Presidential Debate	80618	7	the	823
September 26, 1960: The First Kennedy-Nixon Presidential Debate	60815	3	the	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:



	Counts
Authors	
Helmut Spaeth	[redacted]
[redacted]	3
[redacted]	2
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

In [212]:

```
stuff = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regression/")
.content
soupy_stuff = bs.BeautifulSoup(stuff, 'html.parser')
data_links = soupy_stuff.find_all('a')

for i in range(len(data_links)):
    data_links[i] = "http://people.sc.fsu.edu/~jburkardt/datasets/regression/" +
data_links[i].get('href')
data_links = data_links[6:33]

def get_author(link):
    good_stuff = requests.get(link).content
    good_soupy = soupy_stuff = bs.BeautifulSoup(good_stuff, 'html.parser').text
    crappy_stuff,colon, my_stuff = good_soupy.partition(':')
    author,comma,more_crappy_stuff = my_stuff.partition(",")
    author = author.replace("\n", '')
    author = author.replace('#','')
    author = author.replace('  ', '')
    return author

authors = []
for i in data_links:
    authors.append(get_author(i))

authors_counted = Counter(authors).items()
authors = list(dict.fromkeys(authors))
counts = [i[1] for i in authors_counted]
print(counts)
author_df = pd.DataFrame({"Authors": authors, "Count": counts})
author_df
```

```
[16, 2, 2, 1, 1, 3, 2]
```

Out[212]:

	Authors	Count
0	Helmut Spaeth	16
1	R J Freund and P D Minton	2
2	D G Kleinbaum and L L Kupper	2
3	K A Brownlee	1
4	S Chatterjee and B Price	1
5	S Chatterjee	3
6	S C Narula	2

In [213]:

```
print("https://github.com/ngmiller0505/DataX/blob/master/HW7%20-%20Webscraping.ipynb")
```

```
https://github.com/ngmiller0505/DataX/blob/master/HW7%20-%20Webscrap
ing.ipynb
```

In [ ]:

```

```