Data-X Spring 2019: Homework 7

Webscraping

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

Name: Zhang Jiaheng

SID: 3034453700 ¶

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

.

In [95]:

```
# your code here
from __future__ import division, print_function
import re
from collections import Counter


from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
import requests # The requests library is an
# HTTP library for getting and posting content etc.
# if 100% it would fit the screen
import bs4 as bs # BeautifulSoup4 is a Python library
# for pulling data out of HTML and XML code.
# We can query markup languages for specific content
import numpy as np
import pandas as pd

source = requests.get("https://www.debates.org/voter-education/debate-transcripts/")

soup = bs.BeautifulSoup(source.content, features = 'html.parser')

links = soup.find_all('a')
print("First 1960 debate: " + "https://www.debates.org/voter-education/debate-transcripts/september-26-1960-debate-transcript/")
print("First 1976 debate: " + "https://www.debates.org/voter-education/debate-transcripts/september-23-1976-debate-transcript/")
print("First 1984 debate: " + "https://www.debates.org/voter-education/debate-transcripts/october-7-1984-debate-transcript/")
print("First 1988 debate: " + "https://www.debates.org/voter-education/debate-transcripts/september-25-1988-debate-transcript/")

urls = ["https://www.debates.org/voter-education/debate-transcripts/september-26-1960-debate-transcript/", "https://www.debates.org/voter-education/debate-transcripts/september-23-1976-debate-transcript/", "https://www.debates.org/voter-education/debate-transcripts/october-7-1984-debate-transcript/", "https://www.debates.org/voter-education/debate-transcripts/september-25-1988-debate-transcript/"]

titles = [""]
transcripts = []
lenOfTranscript = ["Debate char length"]
lenWar = ["war_count"]
mostOccur = ["most_common_w"]
noOccurance = ["most_common_w_count"]
for x in urls:
    source = requests.get(x)
    soup = bs.BeautifulSoup(source.content, features = 'html.parser')
    titles.append(soup.find('title').text)
    totalString = ""
    for p in soup.find_all('p'):
        totalString = totalString + " " + p.text
    transcripts.append(totalString)
    lenOfTranscript.append(len(totalString))
for x in transcripts:
    p = re.compile("\swar[?!.,\s:]",re.IGNORECASE)
    lenWar.append(len(p.findall(x)))
for x in transcripts:
    strSplit = x.split()
```

```
    strCount = Counter(strSplit)
    mostCount = strCount.most_common(1)
    mostOccur.append(mostCount[0][0])
    noOccurance.append(mostCount[0][1])

df2 = pd.DataFrame(np.array([lenOfTranscript, lenWar, mostOccur,noOccura
nce]),
                   columns=titles)
df2.set_index("")
df2
```

```
First 1960 debate: https://www.debates.org/voter-education/d
ebate-transcripts/september-26-1960-debate-transcript/
First 1976 debate: https://www.debates.org/voter-education/d
ebate-transcripts/september-23-1976-debate-transcript/
First 1984 debate: https://www.debates.org/voter-education/d
ebate-transcripts/october-7-1984-debate-transcript/
First 1988 debate: https://www.debates.org/voter-education/d
ebate-transcripts/september-25-1988-debate-transcript/
```

Out[95]:

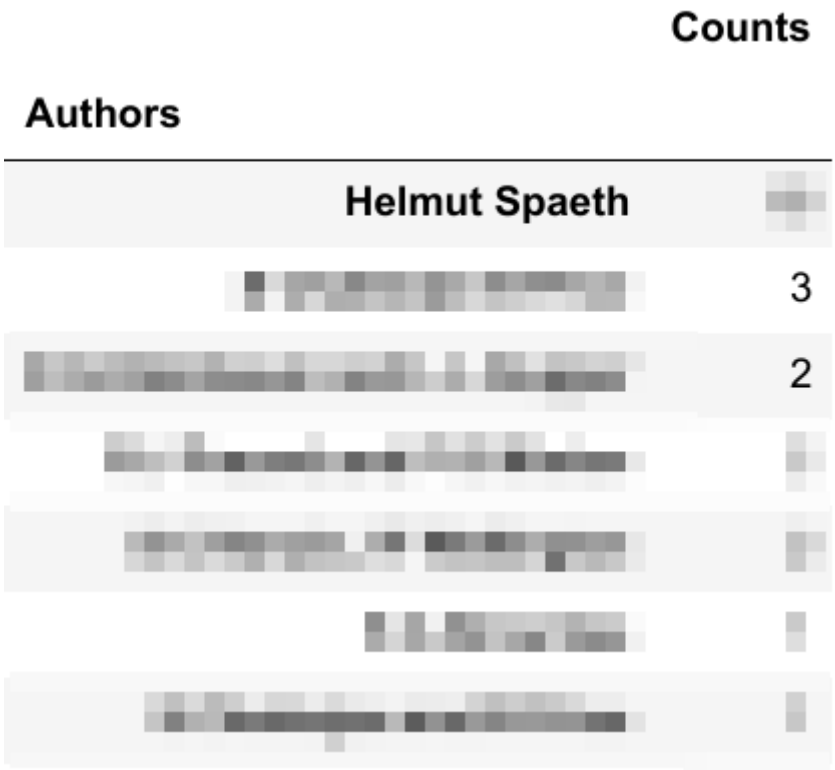
		CPD: September 26, 1960 Debate Transcript	CPD: September 23, 1976 Debate Transcript	CPD: October 7, 1984 Debate Transcript	CPD: September 25, 1988 Debate Transcript
0	Debate char length	61053	80875	86861	87770
1	war_count	3	7	2	7
2	most_common_w	the	the	the	the
3	most_common_w_count	723	823	776	759

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 [40]:

```
# your code here
import bs4 as bs
import requests
import re
from collections import Counter

source = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regr
ession/")
soup = bs.BeautifulSoup(source.content, features='html.parser', parse_on
ly=bs.SoupStrainer('a'))

links = []

for link in soup:
    links.append(link['href'])

links = links[6:33]
authors = []

df2 = pd.DataFrame()

for x in links:
    source = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/
regression/" + x)
    soup = bs.BeautifulSoup(source.content)
    author_name = ''
    i = 0
    for char in soup.get_text():
        if char == "#":
            i += 1

        if i == 5:
            author_name += char

    author_name = author_name.replace('#', "").replace(',', "").lstrip()
.rstrip()

    authors.append(author_name)

authorCount = Counter(authors)

authorList = []
authorListCount = []
for key, value in authorCount.items():
    authorList.append(key)
    authorListCount.append(value)

df2['Author'] = authorList
df2['Count'] = authorListCount

df2.sort_values(by=['Count'],ascending=False)
```

Out[40]:

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

In [ ]:

```
github link:
```