Data-X Fall 2018: Homework 8

Webscraping

Authors: Alexander Fred-Ojala

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

STUDENT NAME : Weijie Yuan

SID : 3034375855

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: http://www.debates.org/index.php?page=debate-transcripts (http://www.debates.org/index.php?page=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: http://www.debates.org/index.php?page=debate-transcripts (http://www.debates.org/index.php?page=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 [2012, 2008, 2004, 2000, 1996, 1988, 1984, 1976, 1960]. In total you should find 9 links / URLs tat 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 3 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:

```python
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)](https://docs.python.org/3/howto/regex.html)

Example output of all of the answers to Question 1.2:

	October 3, 2012: The First Obama-Romney Presidential Debate	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Debate char length	94627	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
war_count	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
most_common_w	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
most_common_w_count	757	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

.

```
In [1]: # your code here
        from IPython.core.display import display, HTML
        display(HTML("<style>.container { width:90% !important; }</style>"))
        import requests
        import bs4 as bs # BeautifulSoup4 is a Python library
        import numpy as np
        import pandas as pd
```

```
In [2]: source = requests.get("http://www.debates.org/index.php?page=debate-transcripts")
        soup = bs.BeautifulSoup(source.content, features='html.parser')
```

```
In [3]: urls = [p.get("href") for p in soup.find_all('a')
                for year in  [2012, 2008, 2004, 2000, 1996, 1988, 1984, 1976, 1960]
         if str(year) in p.text and "First" in p.text]
        urls
```

```
Out[3]: ['http://www.debates.org/index.php?page=october-3-2012-debate-transcript',
         'http://www.debates.org/index.php?page=2008-debate-transcript',
         'http://www.debates.org/index.php?page=september-30-2004-debate-transcript',
         'http://www.debates.org/index.php?page=october-3-2000-transcript',
         'http://www.debates.org/index.php?page=october-6-1996-debate-transcript',
         'http://www.debates.org/index.php?page=september-25-1988-debate-transcript',
         'http://www.debates.org/index.php?page=october-7-1984-debate-transcript',
         'http://www.debates.org/index.php?page=september-23-1976-debate-transcript',
         'http://www.debates.org/index.php?page=september-26-1960-debate-transcript']
```

```
In [4]: #scrape titles
        import re
        title = [p.text for p in soup.find_all('a')
                 for year in  [2012, 2008, 2004, 2000, 1996, 1988, 1984, 1976, 1960]
         if str(year) in p.text and "First" in p.text]
        title
```

```
Out[4]: ['October 3, 2012: The First Obama-Romney Presidential Debate',
         'September 26, 2008: The First McCain-Obama Presidential Debate',
         'September 30, 2004: The First Bush-Kerry Presidential Debate',
         'October 3, 2000: The First Gore-Bush Presidential Debate',
         'October 6, 1996: The First Clinton-Dole Presidential Debate',
         '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 [5]: #scrape debate char length
        #we notice that the transcription text of 2018 repeats twice
        soup_content = []
        for url in urls:
            source_year = requests.get(url)
            soup_content.append(bs.BeautifulSoup(source_year.content, features='html.parser'))
```

In [6]:
```python
from collections import Counter
#count the debate char length
debate_char_length = []
war = []
for content in soup_content:
    str_all = ""
    for p in content.p.find_all('p'):
        str_all += p.text
    debate_char_length.append(len(str_all.replace("\n","")))
    # we count only singular 'war'
    war.append(Counter(re.findall(r"[^\w]war[^\w]+", str_all.lower())))

#we can see the matched "war" format
war
```

Out[6]:
```
[Counter({' war, ': 1, ' war ': 2}),
 Counter({' war ': 30, ' war, ': 4, ' war.': 4, ' war. ': 6}),
 Counter({' war ': 47,
          ' war, ': 6,
          ' war. ': 6,
          '-war ': 1,
          ' war.': 2,
          ' war? ': 1,
          ' war," ': 1}),
 Counter({' war ': 9, ' war. ': 2}),
 Counter({' war ': 8, ' war, ': 4, ' war. ': 1, ' war -- ': 1}),
 Counter({' war ': 5, ' war. ': 1, ' war, ': 1, ' war-': 1}),
 Counter({' war ': 2}),
 Counter({' war, ': 2, ' war.': 2, ' war ': 3}),
 Counter({' war, ': 1, ' war. ': 1, ' war ': 1})]
```

If one want to count both singular 'war' and the plural 'wars'. We can use the following regex to match 'war' and 'wars'.

```
In [7]: war_b = []
        for content in soup_content:
            str_all = ""
            for p in content.p.find_all('p'):
                str_all += p.text
            debate_char_length.append(len(str_all.replace("\n","")))
            # we count only singular 'war'
            war_b.append(Counter(re.findall(r"[^\w]wars*[^\w]+", str_all.lower())))

        #we can see the matched "war" format
        war_b
```

```
Out[7]: [Counter({' wars ': 2, ' war, ': 1, ' war ': 2}),
         Counter({' wars, ': 2,
                  ' war ': 30,
                  ' war, ': 4,
                  ' wars ': 2,
                  ' war.': 4,
                  ' war. ': 6}),
         Counter({' war ': 47,
                  ' war, ': 6,
                  ' war. ': 6,
                  '-war ': 1,
                  ' war.': 2,
                  ' war? ': 1,
                  ' war," ': 1}),
         Counter({' war ': 9, ' war. ': 2}),
         Counter({' war ': 8, ' war, ': 4, ' war. ': 1, ' wars ': 1, ' war -- ': 1}),
         Counter({' war ': 5,
                  ' war. ': 1,
                  ' war, ': 1,
                  ' wars. ': 2,
                  ' wars ': 1,
                  ' wars, ': 1,
                  ' war-': 1,
                  ' wars-': 2}),
         Counter({' war ': 2, ' wars ': 1}),
         Counter({' war, ': 2, ' war.': 2, ' war ': 3}),
         Counter({' war, ': 1, ' war. ': 1, ' war ': 1})]
```

```
In [8]: war_count = [sum(war_debate.values()) for war_debate in war]
        print("Debate char length in each debate is "+str(debate_char_length))
        print("Count times the word war was used in the different debates is " + str(war_count))
```

```
Debate char length in each debate is [94594, 182386, 82685, 91040, 93057, 87440, 86639, 80683, 60883, 9
4594, 182386, 82685, 91040, 93057, 87440, 86639, 80683, 60883]
Count times the word war was used in the different debates is [3, 44, 64, 11, 14, 8, 2, 7, 3]
```

```
In [9]: common_word = []
        common_count = []
        for content in soup_content:
            str_all = ""
            for p in content.p.find_all('p'):
                str_all += p.text
            count = Counter(re.findall(r"([\w']+)[^\w]+", str_all.lower()))
            word = max(count, key=count.get)
            common_word.append(word)
            common_count.append(count[word])
        print("The most common used words in the debate are respectively "+str(common_word))
        print("Times the most common used words were used are respectively "+str(common_count))
```

```
The most common used words in the debate are respectively ['the', 'the', 'the', 'the', 'the', 'the', 't
he', 'the', 'the']
Times the most common used words were used are respectively [757, 1470, 857, 919, 876, 804, 866, 857, 7
79]
```

So, now we combine the above process together to generate the result dataframe.

In [10]:
```
data_result = {}
for i in range(9):
    data_result[title[i]]=[debate_char_length[i],war_count[i],common_word[i],common_count[i]]
df_result = pd.DataFrame(data=data_result,
                         index = ['Debate char length', 'war_length',
                                  'most_common_w', 'most_common_w_count'])

df_result
```

Out[10]:

	October 3, 2012: The First Obama-Romney Presidential Debate	September 26, 2008: The First McCain-Obama Presidential Debate	September 30, 2004: The First Bush-Kerry Presidential Debate	October 3, 2000: The First Gore-Bush Presidential Debate	October 6, 1996: The First Clinton-Dole Presidential Debate	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
Debate char length	94594	182386	82685	91040	93057	87440	86639	80683	60883
war_length	3	44	64	11	14	8	2	7	3
most_common_w	the	the	the	the	the	the	the	the	the
most_common_w_count	757	1470	857	919	876	804	866	857	779

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 [11]:
```
# your code here
names = []
for i in range(1,28):
    if i < 10:
        source_txt = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regression/x0"+
                                  str(i)+".txt")
    else:
        source_txt = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regression/x"+
                                  str(i)+".txt")
    soup_txt = bs.BeautifulSoup(source_txt.content, features='html.parser')
    #names.extend(soup_txt.text.split('\n')[4].replace("#","").replace(' and',',').strip(',').strip().spl
    names.append(soup_txt.text.split('\n')[4].replace("#","").strip().strip(','))
names
```

Out[11]:
```
['Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'R J Freund and P D Minton',
 'D G Kleinbaum and L L Kupper',
 'Helmut Spaeth',
 'D G Kleinbaum and L L Kupper',
 'K A Brownlee',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'S Chatterjee and B Price',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'R J Freund and P D Minton',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'Helmut Spaeth',
 'S Chatterjee, B Price',
 'S Chatterjee, B Price',
 'S Chatterjee, B Price',
 'S C Narula, J F Wellington',
 'S C Narula, J F Wellington']
```

```
In [12]: Counts = list(dict(Counter(names)).values())
         Authors = list(dict(Counter(names)).keys())
         pd_result = pd.DataFrame(data={'Authors':Authors,
                                        'Counts':Counts}).set_index('Authors')
         pd_result.sort_values('Counts',ascending = False)
```

```
Out[12]:
                                  Counts
                         Authors
                   Helmut Spaeth      16
           S Chatterjee, B Price       3
        R J Freund and P D Minton      2
    D G Kleinbaum and L L Kupper       2
        S C Narula, J F Wellington     2
                     K A Brownlee      1
        S Chatterjee and B Price       1
```

Notice in index 'Authors', some .txt files have multiple authors. If one need to check the numbers of occurances of each single author name. We can use the following code.

```
In [13]: names = []
for i in range(1,28):
    if i < 10:
        source_txt = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regression/x0"+
                                  str(i)+".txt")
    else:
        source_txt = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/regression/x"+
                                  str(i)+".txt")
    soup_txt = bs.BeautifulSoup(source_txt.content, features='html.parser')
    names.extend(soup_txt.text.split('\n')[4].replace("#","").replace(' and',',').
                 strip(',').strip().split(', '))
    #names.append(soup_txt.text.split('\n')[4].replace("#","").strip().strip(','))

Counts = list(dict(Counter(names)).values())
Authors = list(dict(Counter(names)).keys())
pd_result = pd.DataFrame(data={'Authors':Authors,
                               'Counts':Counts}).set_index('Authors')
pd_result.sort_values('Counts',ascending = False)
```

Out[13]:

	Counts
Authors	
Helmut Spaeth	16
S Chatterjee	4
B Price	4
R J Freund	2
P D Minton	2
D G Kleinbaum	2
L L Kupper	2
S C Narula	2
J F Wellington	2
K A Brownlee	1