Data-X Fall 2018: Homework 8

Webscraping

Authors: Alexander Fred-Ojala

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

STUDENT NAME : Grant Pemberton ¶

SID : 3034347047

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:

```
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:

	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]

In [172]:
```
import pandas as pd
import numpy as np
import requests # The requests library is an
# HTTP library for getting and posting content etc.

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

from nltk.stem import WordNetLemmatizer


from collections import Counter
import re
```

In [173]:
```
#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 [201
2, 2008, 2004, 2000, 1996
#, 1988, 1984, 1976, 1960]. In total you should find 9 links / URLs tat
 fulfill this criteria. Print the urls.

source = requests.get("http://www.debates.org/index.php?page=debate-tran
scripts")

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

links = soup.find_all('a')

first_debates = []


for l in links:
    if ('First' in (l.text)) & ('Presidential'in (l.text)) :
        print(l.get('href'))
        first_debates.append(l.get('href'))
```

```
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-transcri
pt
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-transcri
pt
http://www.debates.org/index.php?page=october-7-1984-debate-transcript
http://www.debates.org/index.php?page=september-23-1976-debate-transcri
pt
http://www.debates.org/index.php?page=september-26-1960-debate-transcri
pt
```

In [174]:
```
#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):




#    1. Scrape the title of each link and use that as the column name in
 your Data Frame.
column_names = []
#create an array for the column names
for l in links:
    #normally I wouldn't run this forLoop twice, but the instructions sa
y to scrape the title of each link,
    #so I'm running the forLoop again. Please forgive me Asymptotic Beha
vior Gods!
    if ('First' in (l.text)) & ('Presidential'in (l.text)) :
        column_names.append(l.text)
```

In [175]:
```
#    2. 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 break
line characters, i.e. `\n`. You will get
#credit if your count is +/- 10% from our result.

transcript_lengths = []


for link in first_debates:
    transcript = requests.get(link)

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

    transcript_text = (soup.find(id='content-sm').text)

    transcript_length = int(len(transcript_text.replace("\n",'')))

    transcript_lengths.append(transcript_length)

    print (soup.find('title').text, "has a char count of: ", len(transcr
ipt_text.replace("\n",'')))
```

```
CPD: October 3, 2012 Debate Transcript has a char count of:  94627
CPD: September 26, 2008 Debate Transcript has a char count of:  182422
CPD: September 30. 2004 Debate Transcript has a char count of:  82721
CPD: October 3, 2000 Transcript has a char count of:  91066
CPD: October 6, 1996 Debate Transcript has a char count of:  93090
CPD: September 25, 1988 Debate Transcript has a char count of:  87494
CPD: October 7, 1984 Debate Transcript has a char count of:  86687
CPD: September 23, 1976 Debate Transcript has a char count of:  80737
CPD: September 26, 1960 Debate Transcript has a char count of:  60937
```

In [176]:
```
#    3. 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 exampl
e, but counting **war.**, **war!**, **war,**
#or **War** etc.

war_counts = []

for link in first_debates:
    char_count = 0
    transcript = requests.get(link)
    soup = bs.BeautifulSoup(transcript.content, features='html.parser')

    transcript_text = (soup.find(id='content-sm').text)

    transcript_text = transcript_text.replace("\n",' ')
    transcript_text = transcript_text.replace(",",' ')
    transcript_text = transcript_text.replace(".",' ')
    transcript_text = transcript_text.replace("?",' ')
    transcript_text = transcript_text.replace("-",' ')

    t_lower = transcript_text.lower().split()

    #print (t_lower)

    war_count = 0

    war_count += t_lower.count("war")

    war_counts.append(war_count)
```

In [177]:
```
#    4. Also scrape the most common used word in the debate, and write h
ow many times it was used. Note that you
#have to use the same strategy as in 3 in order to do this.

high_freq_word = []

high_freq_count = []

for link in first_debates:
    char_count = 0
    transcript = requests.get(link)
    soup = bs.BeautifulSoup(transcript.content, features='html.parser')

    transcript_text = (soup.find(id='content-sm').text)

    transcript_text = transcript_text.replace("\n",' ')
    transcript_text = transcript_text.replace(",",' ')
    transcript_text = transcript_text.replace(".",' ')
    transcript_text = transcript_text.replace("?",' ')
    transcript_text = transcript_text.replace("-",' ')

    t_lower = transcript_text.lower().split()
    highest_count = 0
    hcw = ""
    for word in t_lower:
        if t_lower.count(word) >= highest_count:
            highest_count = t_lower.count(word)
            hcw = word
    #print (hcw, highest_count)
    high_freq_word.append (hcw)
    high_freq_count.append (highest_count)
```

In [182]:

```
#    Print your final output result.

df = pd.DataFrame([transcript_lengths, war_counts, high_freq_word, high_
freq_count], columns = column_names)
#dataframe wasn't building without all of the rows that it needed


df = df.rename(index = {0:'Debate char length', 1:'war_count', 2: 'most_
common_w', 3:'most_common_w_count'})
#renaming the index for the transcript length row

df
```

Out[182]:

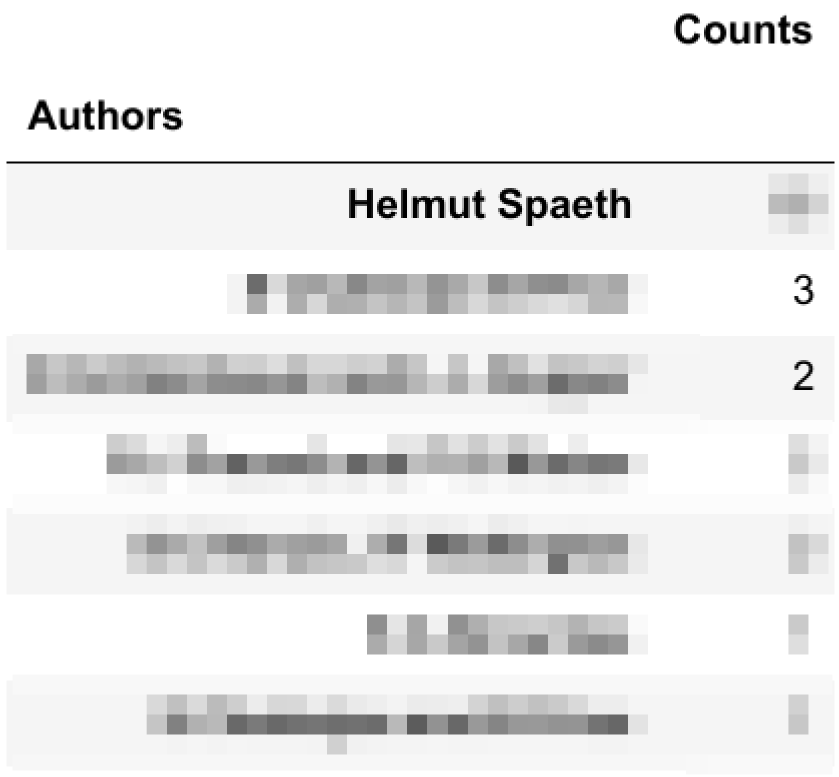
	October 3, 2012: The First Obama-Romney Presidential Debate	**September 26, 2008: The First McCain-Obama Presidential Debate**	**September 30, 2004: The First Bush-Kerry Presidential Debate**	**October 3, 2000: The First Gore-Bush Presidential Debate**	**October 6, 1996: The First Clinton-Dole Presidential Debate**
Debate char length	94627	182422	82721	91066	93090
war_count	3	44	64	11	14
most_common_w	the	the	the	the	the
most_common_w_count	757	1470	853	917	876

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 [256]:
```
# your code here

source2 = requests.get("http://people.sc.fsu.edu/~jburkardt/datasets/reg
ression/")

soup = bs.BeautifulSoup(source2.content, features='html.parser')

#print (soup)

links2 = soup.find_all('a')

author_list = []

counter = 0

for l in links2:
    if ('x' in (l.text)) and counter <27 :
        counter += 1
        data_link = "http://people.sc.fsu.edu/~jburkardt/datasets/regres
sion/" + l.get('href')
        dataset_content = requests.get(data_link)

        soup2 = bs.BeautifulSoup(dataset_content.content, features='htm
l.parser')
        #import the .txt file into soup2

        lines = soup2.text.split('\n')
        #split soup2 based on the new line character

        author = lines[4].replace('#', '')
        #get rid of the '#'

        author = author.replace('    ', '')
        #get rid of indentation

        author = author.replace(',', '')

        author_list.append(author)
#print (author_list)

reference_list = []
count_list = []
count = 0

for reference in author_list:
#    print (reference, author_list.count(reference))
    count = author_list.count(reference)
    if reference not in reference_list:
        reference_list.append(reference)
        count_list.append(count)

#print (reference_list)
#print (count_list)


aut = pd.DataFrame([reference_list, count_list])
aut = aut.transpose()

aut.columns = ['Authors', 'Counts']


aut = aut.sort_values('Counts', ascending = False)

aut



#Scrape the first 27 data sets from this URL http://people.sc.fsu.edu/~j
burkardt/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 refere
nce for one of the 27 data sets.
#Showcase your results, sorted, with the most common author name first a
nd how many times he appeared in data
#sets. Use a Pandas DataFrame to show your results, see example. Print y
our final output result.
```

Out[256]:

	Authors	Counts
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 [ ]: