Srividya Majeti

# Assignment 3

CS 532: Introduction to Web Science
Dr. Michael Nelson
Spring 2016

February 19, 2016

## Contents

# 1

# Question 1

**Download the 1000 URIs from assignment 2. “curl”, “wget”, or “lynx” are all good candidate programs to use. We want just the raw HTML, not the images, stylesheets, etc.**
**from the command line:**
**curl http://www.cnn.com/ > www.cnn.com**
**wget -O www.cnn.com http://www.cnn.com/**
**lynx -source http://www.cnn.com/ > www.cnn.com**
**“www.cnn.com” is just an example output file name, keep in mind that the shell will not like some of the characters that can occur in URIs (e.g., ). You might want to hash the URIs, like:**
**echo -n** `‘‘http://www.cs.odu.edu/show_features.shtml?72’’` **— md5**
**41d5f125d13b4bb554e6e31b6b591eeb**
**(“md5sum” on some machines; note the “-n” in echo – this removes the trailing newline.)**
**Now use a tool to remove (most) of the HTML markup. “lynx” will do a fair job:**

For solving the above problem I used Python programming language. Following are the steps I have taken to solve the given problem:

- First I got the raw data for all the 1000 unique URIs that I collected in assignment 2, using the following cURL command:
  curl <URI> > < output filename >.
- I stored the raw HTML output generated by the cURL command in separate files for each URI and named the files based on their index. This code is listed in Listing 1.1
- Then I got the processed HTML and stored the output in separate files for each URI using the command:
  lynx -dump -force_html <URI> > < output filename >.
  I named the files with the URI index followed by a hyphen and the word ‘processed’. This code is listed in Listing 1.2

**Code Listing**

```
import commands
import os

count = 0
input=open('uri.json','r')
t=open('command','w')
for line in input:
        count +=1
        if count <1001:
                filename= str(count)
                # print str(line)
                command ='curl  ' + '"' + str(line).rstrip('\
                    n') + '"'+ '> ./rawData/' + filename
                # print command
                # t.write(str(command))
                output = commands.getoutput(command)
input.close()
```

**Listing 1.1.** Python code for getting raw HTML and saving them in files

**Code Listing**

```
import commands
import os

count = 0
input=open('uri.json','r')
t=open('command','w')
for line in input:
        count +=1
        if count <1001:
                filename= str(count) + '-processed'
                # print str(line)
                command ='lynx -dump -force_html  ' + '"'+
                    str(line).rstrip('\n') + '"'+'> ./
                    processedData/' + filename
                # t.write(str(command))
                print command
                output = commands.getoutput(command)
input.close()
```

**Listing 1.2.** Python code for getting processed HTML without any images or stylesheets and saving them in files

## 2

## Question 2

**Choose a query term (e.g., "shadow") that is not a stop word (see week 5 slides) and not HTML markup from step 1 (e.g., "http") that matches at least 10 documents (hint: use "grep" on the processed files). If the term is present in more than 10 documents, choose any 10 from your list. (If you do not end up with a list of 10 URIs, you've done something wrong). As per the example in the week 5 slides, compute TFIDF values for the term in each of the 10 documents and create a table with the TF, IDF, and TFIDF values, as well as the corresponding URIs. The URIs will be ranked in decreasing order by TFIDF values. For example: Table 1. 10 Hits for the term "shadow", ranked by TFIDF.**

| TFIDF | TF | IDF | URI |
|---|---|---|---|
| 0.150 | 0.014 | 10.680 | http://foo.com/ |
| 0.044 | 0.008 | 5.510 | http://bar.com/ |

**You can use Google or Bing for the DF estimation. To count the number of words in the processed document (i.e., the denominator for TF), you can use "wc":**

**wc -w www.cnn.com.processed**

**2370 www.cnn.com.processed**

**It won't be completely accurate, but it will be probably be consistently inaccurate across all files. You can use more accurate methods if you'd like.**

**Don't forget the log base 2 for IDF, and mind your significant digits!**

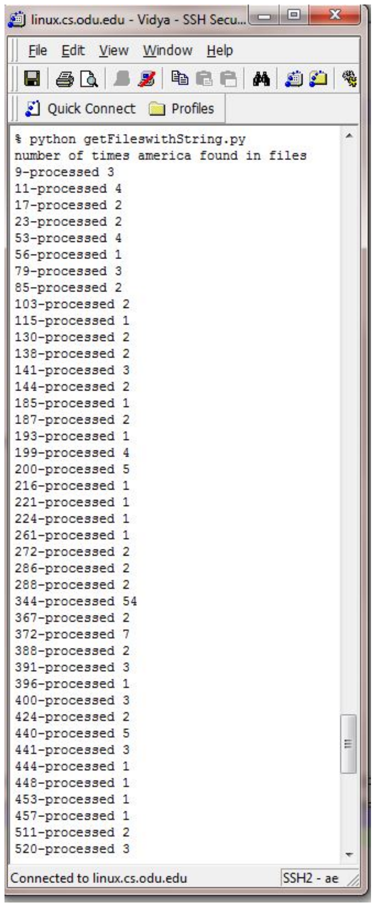
Following are the steps that I have taken to solve this problem:

- I made use of the processed data generated in question 1 and selected a query term.

- Using the following cURL command, I searched for the query term in processed files to find the frequency of the term in each URI.
  cat ./processedData/ <filename> — grep -i -c <queryTerm>
- First I looked for the word 'America' in all the processed files. I got more than 10 URIs, But the number of times the query term appeared in each URI was not satisfactory.The screenshot of the output with filename and number of times the word 'America' appeared in each file is in Figure 2.1.
- Then I changed my query term to 'food' which gave a better frequency. The screenshot for this output is in Figure 2.2
- I also counted the total number of words in each processed file using the command:
  cat ./processedData/ <filename> — wc -w.
- Then I stored the output in a JSON structure with file name, total number of words in the document and frequency of the query term in the document. This code is listed in Listing 2.1 and screenshot of this JSON structure is in Figure 2.3.
- When I searched for the term 'food' I received about 338 URIs but I selected 10 URIs among them based on the versatility of the frequency of the term in each document.
- The selected 10 URIs are as follows:
  `http://www.chilipeppermadness.com/`
  `http://tcbmag.com/Innovations/January-2016/Why-Chipotle-Is-Closing-Its-Stores-Over-The-Lunch?utm_content=27945314&utm_medium=social&utm_source=twitter`
  `http://www.CafeRio.com`
  `http://www.yelp.com/biz/chipotle-mexican-grill-gainesville-4?utm_campaign=CheckIn&utm_medium=twitter&utm_source=ashare`
  `http://www.nydailynews.com/life-style/eats/chipotle-lettuce-shortage-plagues-new-york-article-1.2520586`
  `http://fieryfork.com/video-chipotle-profit-heats-up-as-it-draws-more-diners/`
  `http://www.eater.com/2016/2/5/10922434/chipotle-e-coli-beef-australia?utm_campaign=national&utm_medium=social&utm_source=twitter`
  `http://www.havingfunsaving.com/2016/01/chipotle-cheese-dip-recipe.html`
  `http://www.ooyuz.com/geturl?aid=10248937`
  `http://smartmarkradio.com/conspiracy-theory-news/this-chipotle-conspiracy-theory-is-the-craziest-thing-weve-read-all-day-grist/`
- To calculate IDF for these URIs, I got the total number of documents in corpus which was 47 billion and documents with the term 'food' which was about 3.52 billion.
- Then I calculated the values of TF, IDF, TFIDF. The code is listed in Listing 2.2
- These values are summarized in the Table 2.1.

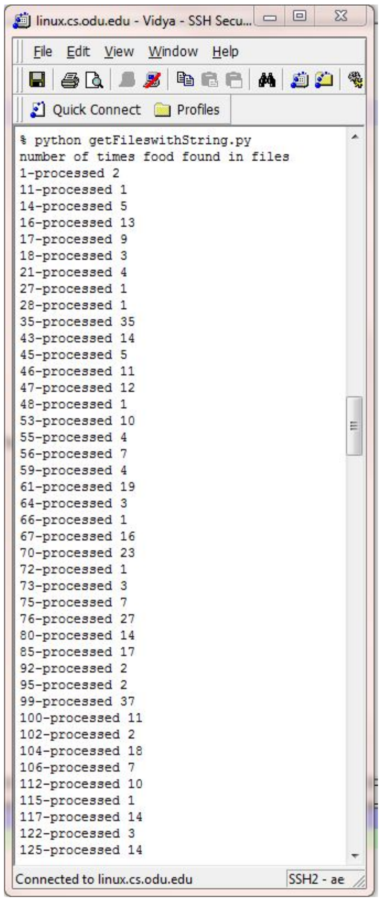
**Table 2.1.** Table with calculated TFIDF , TF , IDF and URI

| Rank | TFIDF | TF | IDF | URI |
|---|---|---|---|---|
| 1 | 0.075 | 0.02 | 3.739 | http://fieryfork.com/video-chipotle-profit-heats-up-as-it-draws-more-diners/ |
| 2 | 0.045 | 0.012 | 3.739 | http://www.chilipeppermadness.com/ |
| 3 | 0.026 | 0.007 | 3.739 | http://www.nydailynews.com/life-style/eats/chipotle-lettuce-shortage-plagues-new-york-article-1.2520586 |
| 4 | 0.022 | 0.006 | 3.739 | http://tcbmag.com/Innovations/January-2016/Why-Chipotle-Is-Closing-Its-Stores-Over-The-Lunch?... |
| 5 | 0.019 | 0.005 | 3.739 | http://www.CafeRio.com |
| 6 | 0.019 | 0.005 | 3.739 | http://www.eater.com/2016/2/5/10922434/chipotle-e-coli-beef-australia?utm_... |
| 7 | 0.019 | 0.005 | 3.739 | http://www.ooyuz.com/geturl?aid=10248937 |
| 8 | 0.011 | 0.003 | 3.739 | http://www.yelp.com/biz/chipotle-mexican-grill-gainesville-4?utm_campaign=... |
| 9 | 0.011 | 0.003 | 3.739 | http://smartmarkradio.com/conspiracy-theory-news/this-chipotle-conspiracy-theory-is-the-craziest... |
| 10 | 0.007 | 0.002 | 3.739 | http://www.havingfunsaving.com/2016/01/chipotle-cheese-dip-recipe.html |


linux.cs.odu.edu - Vidya - SSH Secu...
File Edit View Window Help
Quick Connect
Profiles
Connected to linux.cs.odu.edu
SSH2 - ae

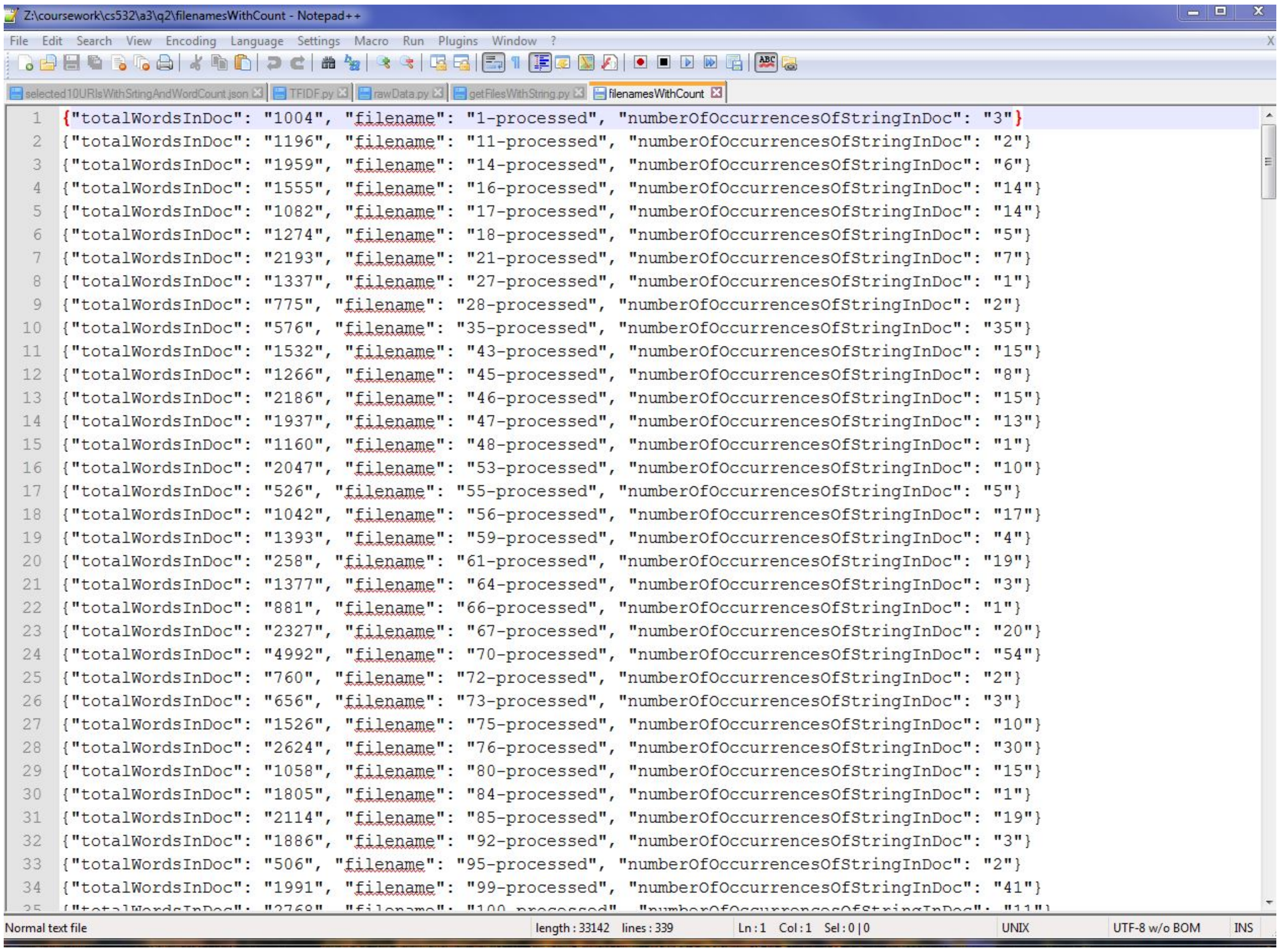
```
% python getFileswithString.py
number of times america found in files
9-processed 3
11-processed 4
17-processed 2
23-processed 2
53-processed 4
56-processed 1
79-processed 3
85-processed 2
103-processed 2
115-processed 1
130-processed 2
138-processed 2
141-processed 3
144-processed 2
185-processed 1
187-processed 2
193-processed 1
199-processed 4
200-processed 5
216-processed 1
221-processed 1
224-processed 1
261-processed 1
272-processed 2
286-processed 2
288-processed 2
344-processed 54
367-processed 2
372-processed 7
388-processed 2
391-processed 3
396-processed 1
400-processed 3
424-processed 2
440-processed 5
441-processed 3
444-processed 1
448-processed 1
453-processed 1
457-processed 1
511-processed 2
520-processed 3
```

**Fig. 2.1.** Number of times "America" appeared in processed files


linux.cs.odu.edu - Vidya - SSH Secu...
File Edit View Window Help
Quick Connect
Profiles
Connected to linux.cs.odu.edu
SSH2 - ae

```
% python getFileswithString.py
number of times food found in files
1-processed 2
11-processed 1
14-processed 5
16-processed 13
17-processed 9
18-processed 3
21-processed 4
27-processed 1
28-processed 1
35-processed 35
43-processed 14
45-processed 5
46-processed 11
47-processed 12
48-processed 1
53-processed 10
55-processed 4
56-processed 7
59-processed 4
61-processed 19
64-processed 3
66-processed 1
67-processed 16
70-processed 23
72-processed 1
73-processed 3
75-processed 7
76-processed 27
80-processed 14
85-processed 17
92-processed 2
95-processed 2
99-processed 37
100-processed 11
102-processed 2
104-processed 18
106-processed 7
112-processed 10
115-processed 1
117-processed 14
122-processed 3
125-processed 14
```

**Fig. 2.2.** Number of times "food" appeared in processed files


Z:\coursework\cs532\a3\q2\filenamesWithCount - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
selected10URIsWithSrtingAndWordCount.json
TFIDF.py
rawData.py
getFilesWithString.py
filenamesWithCount
Normal text file
length : 33142 lines : 339
Ln : 1 Col : 1 Sel : 0 | 0
UNIX
UTF-8 w/o BOM
INS

```
{"totalWordsInDoc": "1004", "filename": "1-processed", "numberOfOccurrencesOfStringInDoc": "3"}
{"totalWordsInDoc": "1196", "filename": "11-processed", "numberOfOccurrencesOfStringInDoc": "2"}
{"totalWordsInDoc": "1959", "filename": "14-processed", "numberOfOccurrencesOfStringInDoc": "6"}
{"totalWordsInDoc": "1555", "filename": "16-processed", "numberOfOccurrencesOfStringInDoc": "14"}
{"totalWordsInDoc": "1082", "filename": "17-processed", "numberOfOccurrencesOfStringInDoc": "14"}
{"totalWordsInDoc": "1274", "filename": "18-processed", "numberOfOccurrencesOfStringInDoc": "5"}
{"totalWordsInDoc": "2193", "filename": "21-processed", "numberOfOccurrencesOfStringInDoc": "7"}
{"totalWordsInDoc": "1337", "filename": "27-processed", "numberOfOccurrencesOfStringInDoc": "1"}
{"totalWordsInDoc": "775", "filename": "28-processed", "numberOfOccurrencesOfStringInDoc": "2"}
{"totalWordsInDoc": "576", "filename": "35-processed", "numberOfOccurrencesOfStringInDoc": "35"}
{"totalWordsInDoc": "1532", "filename": "43-processed", "numberOfOccurrencesOfStringInDoc": "15"}
{"totalWordsInDoc": "1266", "filename": "45-processed", "numberOfOccurrencesOfStringInDoc": "8"}
{"totalWordsInDoc": "2186", "filename": "46-processed", "numberOfOccurrencesOfStringInDoc": "15"}
{"totalWordsInDoc": "1937", "filename": "47-processed", "numberOfOccurrencesOfStringInDoc": "13"}
{"totalWordsInDoc": "1160", "filename": "48-processed", "numberOfOccurrencesOfStringInDoc": "1"}
{"totalWordsInDoc": "2047", "filename": "53-processed", "numberOfOccurrencesOfStringInDoc": "10"}
{"totalWordsInDoc": "526", "filename": "55-processed", "numberOfOccurrencesOfStringInDoc": "5"}
{"totalWordsInDoc": "1042", "filename": "56-processed", "numberOfOccurrencesOfStringInDoc": "17"}
{"totalWordsInDoc": "1393", "filename": "59-processed", "numberOfOccurrencesOfStringInDoc": "4"}
{"totalWordsInDoc": "258", "filename": "61-processed", "numberOfOccurrencesOfStringInDoc": "19"}
{"totalWordsInDoc": "1377", "filename": "64-processed", "numberOfOccurrencesOfStringInDoc": "3"}
{"totalWordsInDoc": "881", "filename": "66-processed", "numberOfOccurrencesOfStringInDoc": "1"}
{"totalWordsInDoc": "2327", "filename": "67-processed", "numberOfOccurrencesOfStringInDoc": "20"}
{"totalWordsInDoc": "4992", "filename": "70-processed", "numberOfOccurrencesOfStringInDoc": "54"}
{"totalWordsInDoc": "760", "filename": "72-processed", "numberOfOccurrencesOfStringInDoc": "2"}
{"totalWordsInDoc": "656", "filename": "73-processed", "numberOfOccurrencesOfStringInDoc": "3"}
{"totalWordsInDoc": "1526", "filename": "75-processed", "numberOfOccurrencesOfStringInDoc": "10"}
{"totalWordsInDoc": "2624", "filename": "76-processed", "numberOfOccurrencesOfStringInDoc": "30"}
{"totalWordsInDoc": "1058", "filename": "80-processed", "numberOfOccurrencesOfStringInDoc": "15"}
{"totalWordsInDoc": "1805", "filename": "84-processed", "numberOfOccurrencesOfStringInDoc": "1"}
{"totalWordsInDoc": "2114", "filename": "85-processed", "numberOfOccurrencesOfStringInDoc": "19"}
{"totalWordsInDoc": "1886", "filename": "92-processed", "numberOfOccurrencesOfStringInDoc": "3"}
{"totalWordsInDoc": "506", "filename": "95-processed", "numberOfOccurrencesOfStringInDoc": "2"}
{"totalWordsInDoc": "1991", "filename": "99-processed", "numberOfOccurrencesOfStringInDoc": "41"}
{"totalWordsInDoc": "2768", "filename": "100-processed", "numberOfOccurrencesOfStringInDoc": "11"}
```

**Fig. 2.3.** JSON structure with file name, total number of words in processed file and number of times the query term occurred in the file

**Code Listing**

```
import os
import commands
import json

f = open('filenamesWithCount','w')
queryTerm = "food"
def getFilesWithString(queryTerm):
        filenamesCountJson ={}
        iterateThroughFiles = next(os.walk('./processedData/
            '))[2]
        print 'number of times %s found in files'%queryTerm
        for filename in iterateThroughFiles:
                command = 'cat ' + './processedData/'+
                    filename + ' | grep -i -c ' + queryTerm
                numberOfOccurrencesInDoc = commands.
                    getoutput(command)
                if numberOfOccurrencesInDoc > '0':
                        filenamesCountJson['
                            numberOfOccurrencesOfStringInDoc
                            '] =numberOfOccurrencesInDoc
                        filenamesCountJson['filename'] =
                            filename
                        command = 'cat ' + './processedData/
                            '+ filename + ' | wc -w '
                        totalWords = commands.getoutput(
                            command)
                        filenamesCountJson['totalWordsInDoc'
                            ] =totalWords
                        f.write(json.dumps(
                            filenamesCountJson) + "\n")
getFilesWithString(queryTerm)
```

**Listing 2.1.** "Python code for searching a query term and counting the number of times the term appears in each processed file. Also counted the total number of words in each processed file and have written the output in a JSON structure."

**Code Listing**

```
import json
import math

totalNumberOfDocumentInCorpus =47000000000
documentWithTerm = 3520000000

def calculate():
        input  = open('selected10URIsWithSrtingAndWordCount.
            json','r')
        for line in input:
                data= json.loads(line)
                TF = round((float(data['
                    numberOfOccurrencesOfStringInDoc'])/
                    float(data['totalWordsInDoc'])),3)
                print 'TF:', TF
                IDF = round(logBase2( float(
                    totalNumberOfDocumentInCorpus) / float(
                    documentWithTerm) ),3)
                print 'IDF:',IDF
                TFIDF = round((TF*IDF),3)
                print 'TFIDF:',TFIDF

def logBase2(number):
        return math.log(number) / math.log(2)

calculate()
```

**Listing 2.2.** "Python code for calculating TF, IDF and TFIDF for 10 selected URIs"

# 3

# Question 3

**Now rank the same 10 URIs from question 2, but this time by their PageRank. Use any of the free PR estimaters on the web, such as:**
`http://www.prchecker.info/check_page_rank.php`
`http://www.seocentro.com/tools/search-engines/pagerank.html`
`http://www.checkpagerank.net/`
**If you use these tools, you'll have to do so by hand (they have anti-bot captchas), but there is only 10. Normalize the values they give you to be from 0 to 1.0. Use the same tool on all 10 (again, consistency is more important than accuracy).**
**Create a table similar to Table 1:**
**Table 2. 10 hits for the term 'shadow', ranked by PageRank.**

| **PageRank** | **URI** |
|---|---|
| **0.9** | **http://bar.com/** |
| **0.5** | **http://foo.com/** |

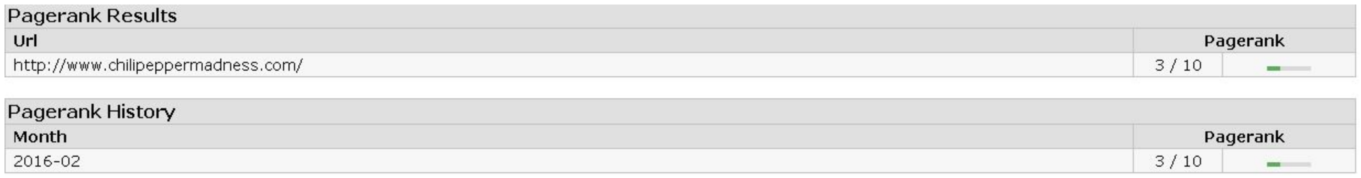
**Briefly compare and contrast the rankings produced in questions 2 and 3.**

- To find the page rank for all the 10 URIs I used a free PR estimator on the web which is located at: `http://www.seocentro.com/tools/search-engines/pagerank.html`.
- The URI and its page rank is summarized in the Table 3.1.
- While finding the page rank for each URI I took a screenshot. The screenshits are in the Figures 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 3.9, 3.10

Comparing the rankings in question 2 and 3 I observed that their is a lot of difference in rankings. The reason behind this difference is, when we rank the URIs based on TF, IDF and TFIDF, we look for the frequency of the term. But the PR estimator ranks the URIs based on the quality of the document and also the quantity of backlinks. Calculating pagerank based on frequency of the term is not a good idea because even a poor quality document with high query term frequency will be ranked high.

**Table 3.1.** PageRank for 10 selected URIs using `http://www.seocentro.com/tools/search-engines/pagerank.html`

| Rank | PageRank | URI |
|---|---|---|
| 3 | 0.7 | http://www.nydailynews.com/ |
| 8 | 0.7 | http://www.yelp.com/ |
| 6 | 0.6 | http://www.eater.com/ |
| 4 | 0.5 | http://tcbmag.com/ |
| 5 | 0.5 | http://www.CafeRio.com |
| 2 | 0.3 | http://www.chilipeppermadness.com/ |
| 10 | 0.3 | http://www.havingfunsaving.com/ |
| 9 | 0.2 | http://smartmarkradio.com/ |
| 1 | 0 | http://fieryfork.com/ |
| 7 | 0 | http://www.ooyuz.com/ |

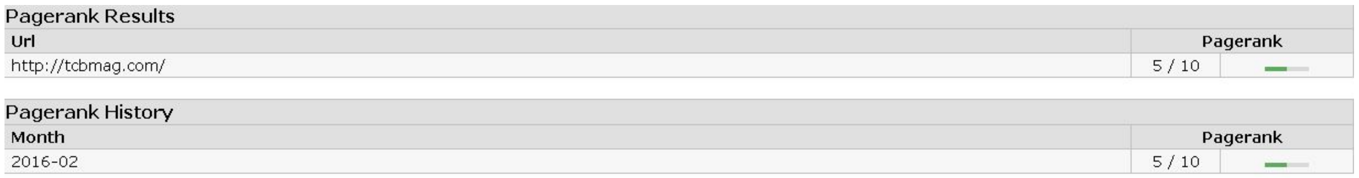

Pagerank Results

| Url | Pagerank |
|---|---|
| http://www.chilipeppermadness.com/ | 3 / 10 |

Pagerank History

| Month | Pagerank |
|---|---|
| 2016-02 | 3 / 10 |

**Fig. 3.1.** page rank for `http://www.chilipeppermadness.com/`

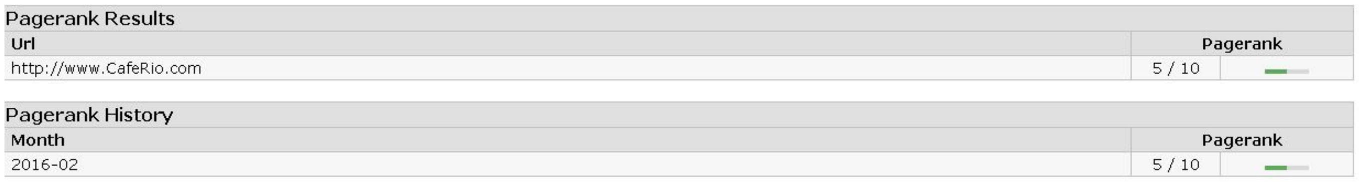

Pagerank Results

| Url | Pagerank |
|---|---|
| http://tcbmag.com/ | 5 / 10 |

Pagerank History

| Month | Pagerank |
|---|---|
| 2016-02 | 5 / 10 |

**Fig. 3.2.** page rank for `http://tcbmag.com/Innovations/January-2016/Why-Chipotle-Is-Closing-Its-Stores-Over-The-Lunch?utm_content=27945314&utm_medium=social&utm_source=twitter`



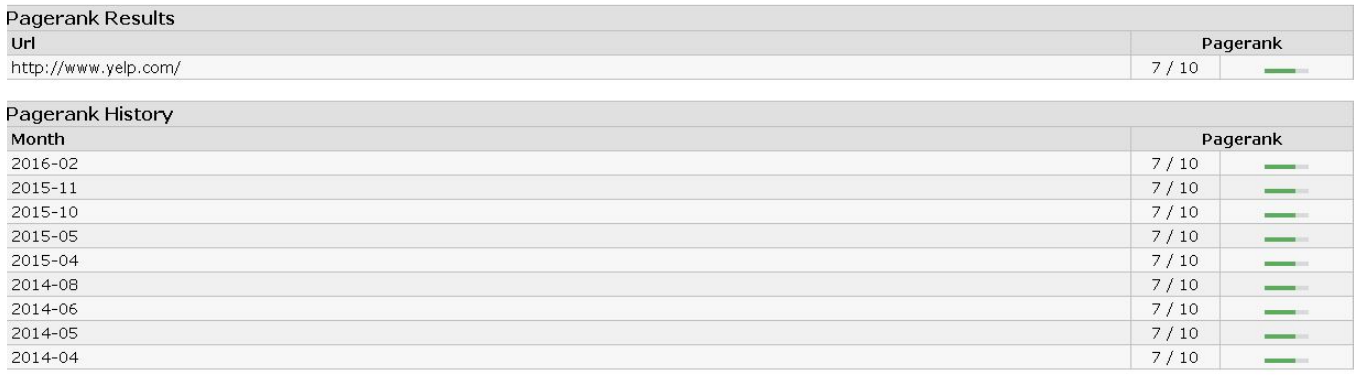
Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://www.CafeRio.com | 5 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 5 / 10 | |

**Fig. 3.3.** page rank for `http://www.CafeRio.com`



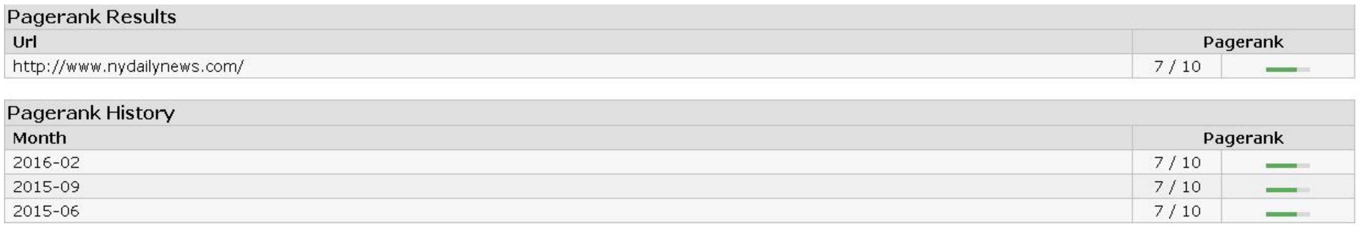
Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://www.yelp.com/ | 7 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 7 / 10 | |
| 2015-11 | 7 / 10 | |
| 2015-10 | 7 / 10 | |
| 2015-05 | 7 / 10 | |
| 2015-04 | 7 / 10 | |
| 2014-08 | 7 / 10 | |
| 2014-06 | 7 / 10 | |
| 2014-05 | 7 / 10 | |
| 2014-04 | 7 / 10 | |

**Fig. 3.4.** page rank for `http://www.yelp.com/biz/chipotle-mexican-grill-gainesville-4?utm_campaign=CheckIn&utm_medium=twitter&utm_source=ashare`



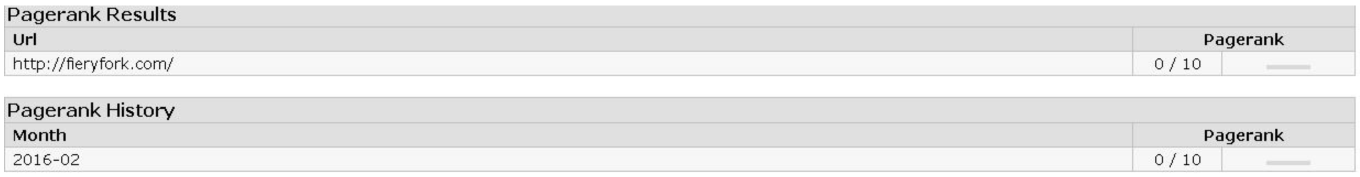
Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://www.nydailynews.com/ | 7 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 7 / 10 | |
| 2015-09 | 7 / 10 | |
| 2015-06 | 7 / 10 | |

**Fig. 3.5.** page rank for `http://www.nydailynews.com/life-style/eats/chipotle-lettuce-shortage-plagues-new-york-article-1.2520586`



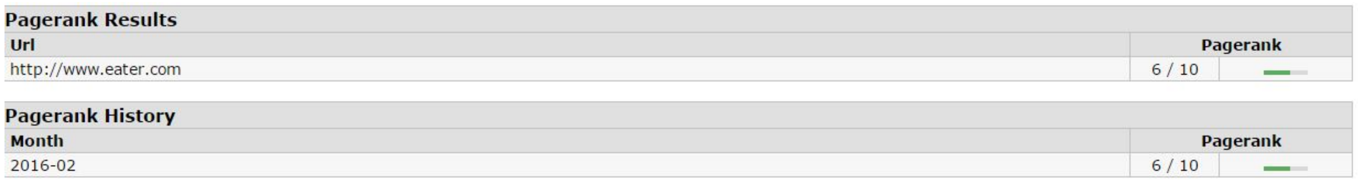
Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://fieryfork.com/ | 0 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 0 / 10 | |

**Fig. 3.6.** page rank for `http://fieryfork.com/video-chipotle-profit-heats-up-as-it-draws-more-diners/`



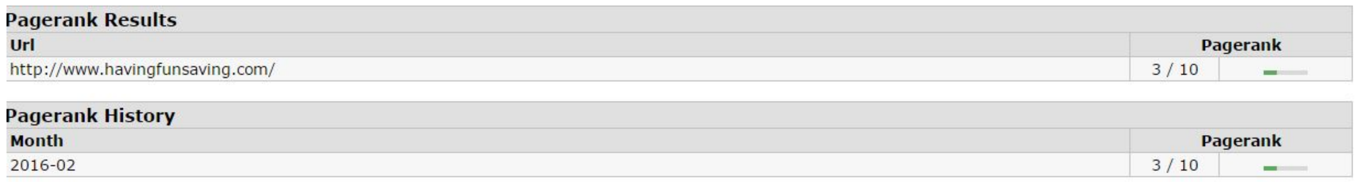
**Pagerank Results**

| Url | Pagerank | |
|---|---|---|
| http://www.eater.com | 6 / 10 | |

**Pagerank History**

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 6 / 10 | |

**Fig. 3.7.** page rank for `http://www.eater.com/2016/2/5/10922434/chipotle-e-coli-beef-australia?utm_campaign=national&utm_medium=social&utm_source=twitter`



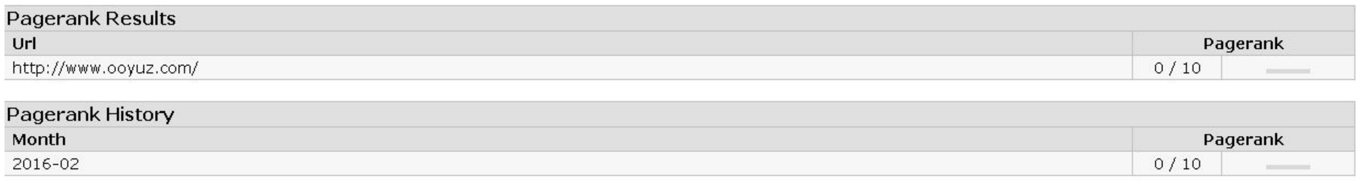
**Pagerank Results**

| Url | Pagerank | |
|---|---|---|
| http://www.havingfunsaving.com/ | 3 / 10 | |

**Pagerank History**

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 3 / 10 | |

**Fig. 3.8.** page rank for `http://www.havingfunsaving.com/2016/01/chipotle-cheese-dip-recipe.html`



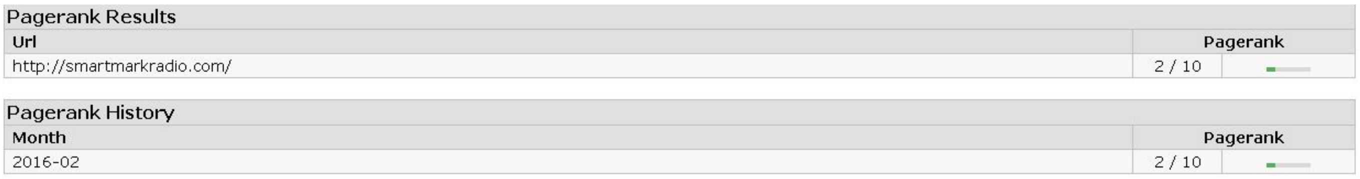
Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://www.ooyuz.com/ | 0 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 0 / 10 | |

**Fig. 3.9.** page rank for `http://www.ooyuz.com/geturl?aid=10248937`



Pagerank Results

| Url | Pagerank | |
|---|---|---|
| http://smartmarkradio.com/ | 2 / 10 | |

Pagerank History

| Month | Pagerank | |
|---|---|---|
| 2016-02 | 2 / 10 | |

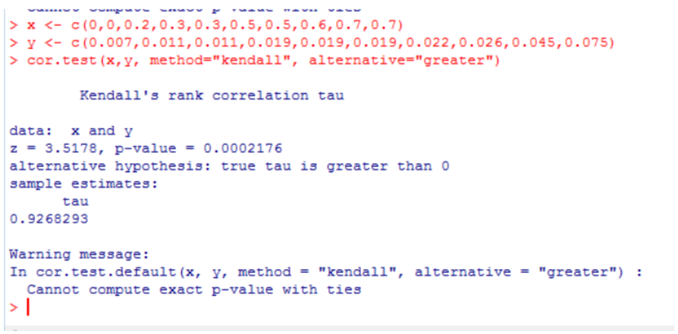
**Fig. 3.10.** page rank for `http://smartmarkradio.com/conspiracy-theory-news/this-chipotle-conspiracy-theory-is-the-craziest-thing-weve-read-all-day-grist/`

4

# Extra-Credit Question

**Compute the Kendall Tau_b score for both lists (use “b” because there will likely be tie values in the rankings). Report both the Tau value and the “p” value.**
**See:**
`http://stackoverflow.com/questions/2557863/measures-of-association-in-r-kendalls-tau-b-and-tau-c`
`http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient#Tau-b`
`http://en.wikipedia.org/wiki/Correlation_and_dependence`

- The Kendall Tau value is ‘0.926’ and p value is ‘0.000217’ for both the lists.
- Below is the R code for calculating the Tau and p values.



```
> x <- c(0,0,0.2,0.3,0.3,0.5,0.5,0.6,0.7,0.7)
> y <- c(0.007,0.011,0.011,0.019,0.019,0.019,0.022,0.026,0.045,0.075)
> cor.test(x,y, method="kendall", alternative="greater")

        Kendall's rank correlation tau

data:  x and y
z = 3.5178, p-value = 0.0002176
alternative hypothesis: true tau is greater than 0
sample estimates:
      tau
0.9268293

Warning message:
In cor.test.default(x, y, method = "kendall", alternative = "greater") :
  Cannot compute exact p-value with ties
> 
```

**Fig. 4.1.** R code for calculating Tau and p values

## References

1. On what basis PR estimator ranks a URI: http://www.seomastering.com/pagerank-prediction.php, Professional SEO authors, Igor Ivanov
2. How to find author of website: http://www.wikihow.com/Find-the-Author-Of-a-Website, Jack Herrick, 2005
3. How to calculate Tau values: http://stackoverflow.com/questions/2557863/measures-of-association-in-r-kendalls-tau-b-and-tau-c, Joel Spolsky, 2008