CS 532: Assignment 3

Dinesh Kumar Paladhi

Spring 2016

Contents

1 Problem 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:

% lynx -dump -force_html www.cnn.com > www.cnn.com.processed

Use another (better) tool if you know of one.  Keep both files
for each URI (i.e., raw HTML and processed).
```

1.1 Solution

1. I started the 1st question by running a sample URI with curl command syntax as given in the question. This created a new file in my directory with html content of the URI that I have mentioned.
2. So I got to know by executing the curl command I will be getting raw data file of the URI.
3. Now the next important point is naming the output files. If we name the output files with URI names,then i will be getting error at one point because it does not accept special characters.
4. This issue can be resolved by creating a hash code which is unique for each URI. I used hashlib library to convert URIs to respective hash codes.
5. Now I have named the file as Raw or processed followed by unique number for each URI followed by the hash code. I add unique number so that it would be useful for me to get URI back from the hash code in future.
6. The same process should be repeated with lynx command(given in question) to get processed file which eliminates all the html content and gives only data.
7. I have written a loop which runs 1000 times and each time it takes a URI and generates a raw file and a processed file. Therefore at the end I will be getting 1000 raw files and 1000 processed files.
8. The code for this program is shown below. Figure 1 and Figure 2 shows a sample Raw file and sample processed file. Figure 3 and Figure 4 shows list of raw and processed files created.

1.2 Code Listing

Here is the Python program for acquiring raw and processed data files for each URI.

```
import json
import commands
import hashlib

file_1 = open('my_json_data.json','r')
count = 0
for each_line in file_1.readlines():
        dumy_line=json.loads(each_line)
        url = dumy_line['url']
        hash = hashlib.md5(url.encode())
        final_hash = hash.hexdigest()
        count = count +1
        file_name_1= "Raw"+'-'+str(count) + '-' + final_hash + '.txt'
        co_1 = 'curl -s -L ' + url + ' > ./Raw_Htmldata/' + file_name_1
        commands.getoutput(co_1)
        file_name_2= "processed"+'-'+str(count) + '-' + final_hash + '.txt' #Naming a file
        co_2 = 'lynx -dump -force_html ' + url + ' > ./processed_htmldata/' + file_name_2 #
            writes files into processed_htmldata folder
        commands.getoutput(co_2)
        #print url,' ',count,' ',file_name_1
```

Listing 1: Python program for acquiring raw and processed data files for each URI

1.3 Results

Sample Raw File

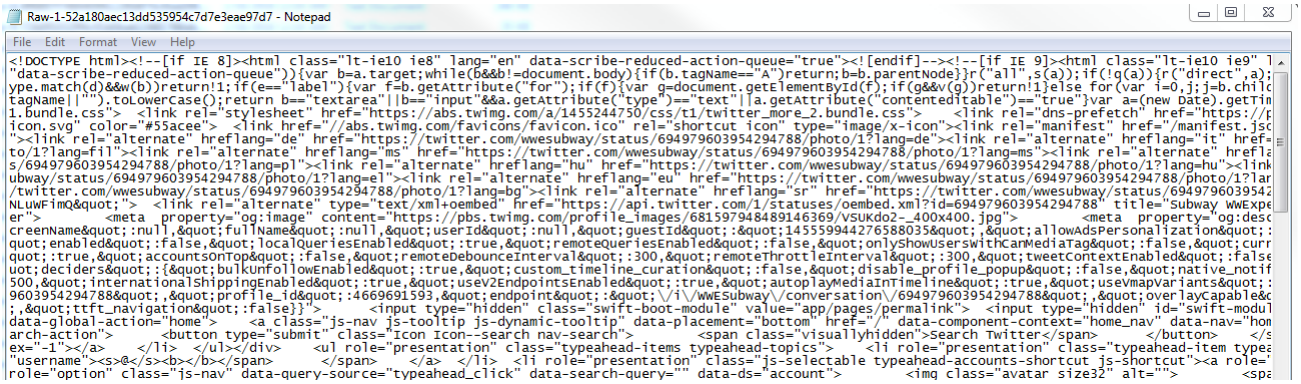

Figure 1: Sample Raw File

Sample Processed File

Figure 2: Sample Processed File

Sample Collection of Raw files

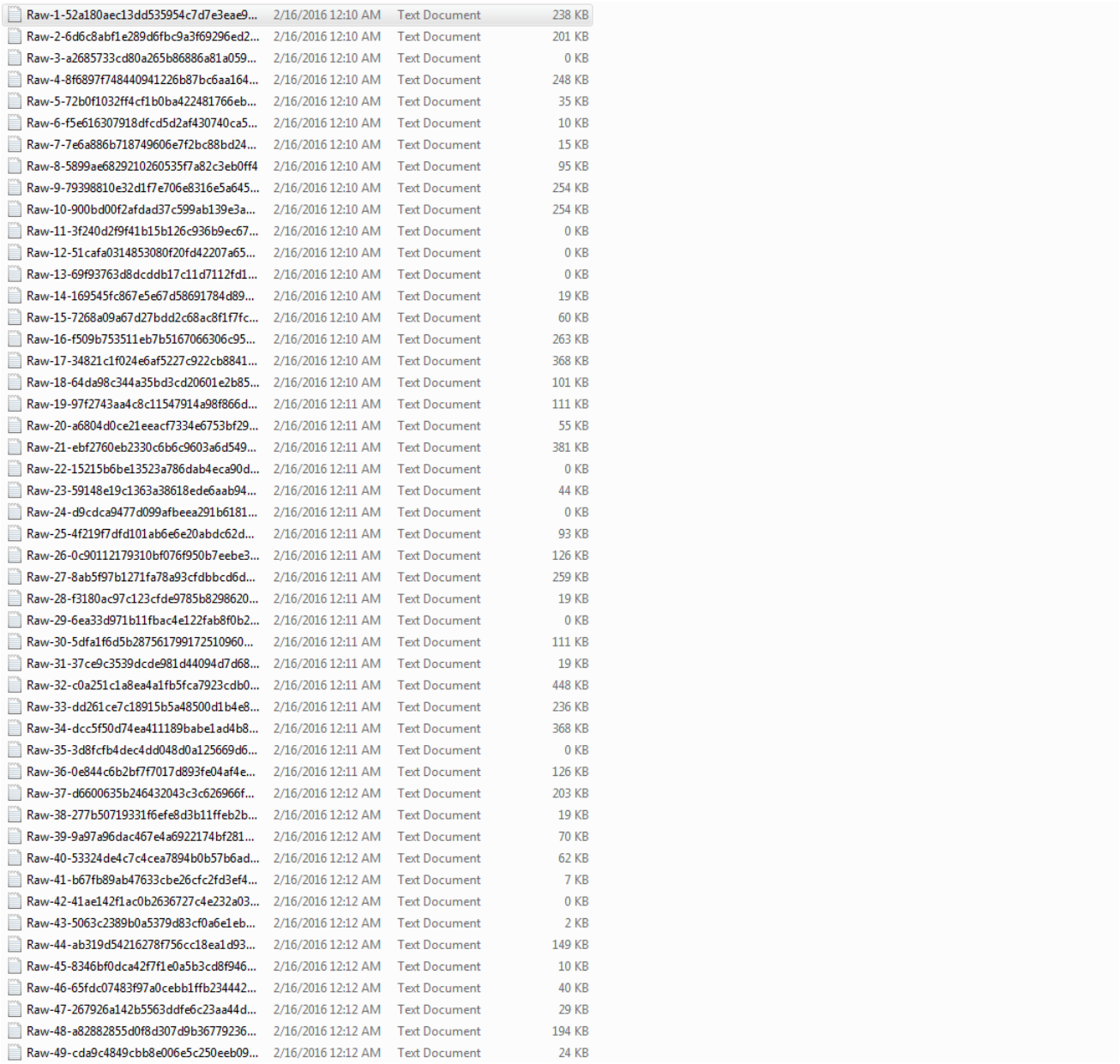

Figure 3: Sample Collection of Raw files

Sample Collection of Processed files

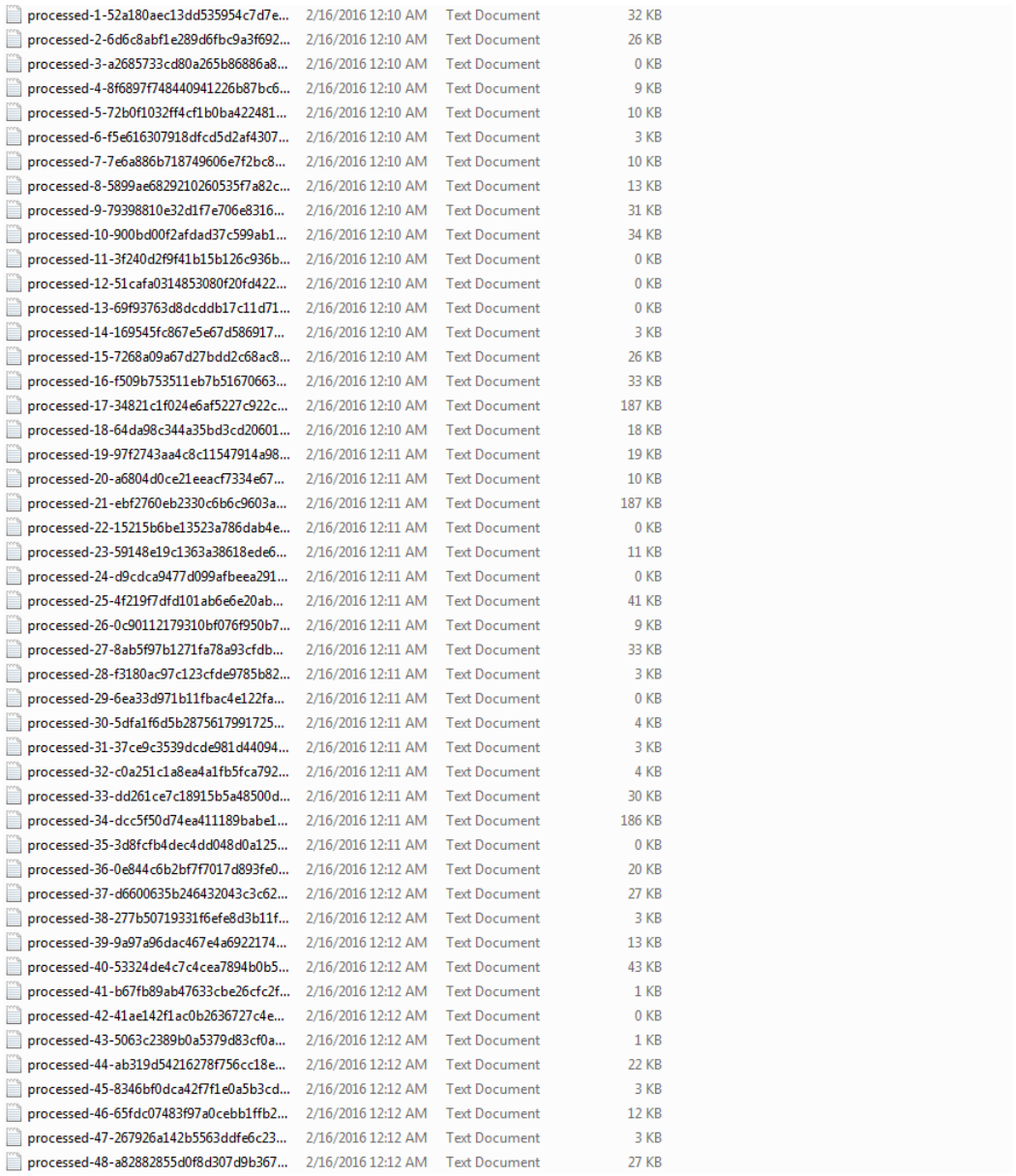

Figure 4: Sample Collection of Processed files

2 Problem 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 deonminator
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!
```

2.1 Solution

1. Aim of this question is to first get 10 URIs which contains a specific key word in it.
2. The keyword should not be a stop word like a,the,and etc and not html mark-up .I have chosen “burger” keyword.
3. So my first task was to write a program which takes 1000 URIs as input and gives all the list of URIs which have “burger” in it.
4. This can be done by using “grep -lr” command followed by “burger”. This command is executed directly in putty in the folder which has all the 1000 processed files.
5. This gave me more than 10 files and that result is showed in Figure 6. I have randomly taken 10 URIs from this list and this list is shown in Figure 7.
6. Now in order to calculate TFIDF I need to calculate TF and IDF separately and then multiply both of them.
7. IDF is common for all the 10 URIs because it is based on the corpus. I have taken “google” as my search engine and I have found the total number of document in the corpus from reference 6. rel=mementos,rel=first memento and rel=memento last from the time maps..

8. Now I queried for “burger” in google which gave 263 million results which is the number of documents with burger in it. This is shown in Figure 5.

9. So my IDF will be total documents in corpus/263 million. This is same for all the 10 URIs.

10. IDF will be growing too large as corpus grows so in order to limit it I used log base 2 function of the IDF.

11. Now TF can be calculated if we get the total number of words in each document and the number of occurrences of the word “burger” in each document.

12. I got the total number of words in each document using “wc -w” command followed by the file name of which I want the word count.

13. I got the number of times the word occurred in each document by using“grep -lr” command followed by my keyword and followed by the file name in which the keyword should be searched.

14. By dividing occurrences/word count,I got TF.

15. Now this TF is multiplied with IDF which gave me TFIDF value.

16. These TFIDF values for each URI are arranged in ascending order and these are shown in Figure 8.

17. The URI having high TFIDF value is first shown to the user when searched in a search engine.

2.2 Code Listing

Python Code for getting URIs with "burger" and TFIDF values for each URI

```
import commands
import re
import math
import json
# def find_urls(URI_num):
        # URI_number1= URI_num
        # count =0
        # for each_line in file_2.readlines():
                # count=count+1
                # print count
                # if count == URI_number1:
                        # print each_line
                        # count =0
                        # break
        # return each_line
file_1=open ('10_urls.txt','r')
file_2 = open('only urls.txt','r+') .readlines()
Tot_doc_Google= 4700000000.00
burger_in_google= 263000000.00
temp_IDF= Tot_doc_Google/burger_in_google
t_IDF= math.log(temp_IDF)/math.log(2)
IDF= round(t_IDF,4)
#print IDF
print "TFIDF",' ', "Rounded_TF",' ',"IDF",' ', "URI"
for f_name in file_1.readlines(): #To calculate TF and TFIDF
        #print f_name
        #print count
        count=0
        w_doc1= 'wc -w '+ f_name.strip()
        oc_doc1= 'grep -c ' + "burger " + f_name.strip()
        #print oc_doc1
        w_doc=commands.getoutput(w_doc1)
        want_word_count=commands.getoutput(oc_doc1)
        tot_word=w_doc.split(' ')[0]
        #print tot_word,' ',want_word_count
        TF= float(want_word_count)/float(tot_word)
        Rounded_TF= round(TF,4)
        TFIDF= round(Rounded_TF*IDF,4)
        URI_find=f_name.split('-')
        URI_num=int(URI_find[1])
        i=0
        for each in file_2:
                count =count +1
                if URI_num==count:
                        URI=each
                        print TFIDF,' ', Rounded_TF,' ',IDF,' ',URI
                        i=i+1
                        break

        # for i in a:
                # print i
                # print count
                # if count == i:
                        # print each
                        # break

        #print URI_num
        #URI_1=find_urls(URI_num)
        #print URI_1
# print TF_array
# print TFIDF_array
# print URI_array
```

Listing 2: Python Code for getting URIs with "burger" and TFIDF values for each URI

2.3 Results

Number of searches for "burger" in google

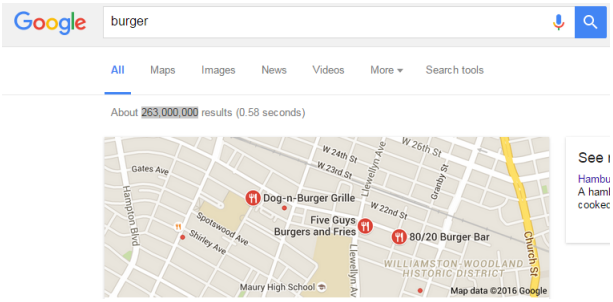

Figure 5: Number of searches for "burger" in google

Processed URI files- having "burger" keyword in it

```
processed-102-d3403063f8ebba384ca21f8614d0bd80.txt
processed-156-219f0bd3fc58c58ce84d21cf7ae07827.txt
processed-183-e3ba61a9e4c84074604480a354083001.txt
processed-209-012c0bd346f601eba30854677b1d1d96.txt
processed-284-a9d69418362909039fa5828bc69ee913.txt
processed-340-eac2bbab72a0b86a213cfa2d668723d3.txt
processed-342-87f3c109e3deb8e83a25334d756bd732.txt
processed-393-9fc057fca4226b8b203e5155168b9686.txt
processed-479-8cd74ea8fb1c84f4e98d3693f4cd4ace.txt
processed-609-e27c5af8226e59287f861b190b28ae72.txt
processed-759-e41c8c33636eb40a1fc56246cd3312d8.txt
processed-845-5224c720d259569b949f89f9bfdb5a45.txt
processed-846-8644cb3e6d47c5d068cfe2eccc62e2fa.txt
processed-849-7e29c780d4394c402c7842ddd57035d3.txt
processed-850-a3b35a81d7f9bdb9b62ea8189f429d18.txt
processed-851-b4104274a1174ac62b4f78f5d4645494.txt
processed-864-eae02d599cc73271ce8d779efa91d6ab.txt
processed-866-a36fb014fe877089b7292d043e1fd0a5.txt
processed-872-e12c98abc5643fb15caa8b814328b3db.txt
processed-886-c97c0b9c014677c50d8afb22c0e4e27b.txt
processed-897-7cb4fb9237319e0aa4030d09062cd9da.txt
processed-899-1857a08321cd0eb8b2f83663cbd0ee51.txt
processed-901-af7cdd6003e50b720abec11e7dad24e7.txt
processed-906-3ac0e5a1248a543bfcd9d1d1261e38aa.txt
processed-907-9d53e0d3174ce73c5cb8ef49cdcf363f.txt
processed-908-aff7ca85f2175022dae23a793b35f0d4.txt
processed-920-ce5ef25ce196643c430577cfbde84d35.txt
processed-942-0ffe7f8aa13096be354cdc54f1c45f9b.txt
processed-951-2ba63a1375102c33a93b3a2447729728.txt
processed-954-2e7a7645dc295ef1268c75e6c4101e12.txt
processed-962-f88162fbc92acf4463106b0cfcf082d6.txt
processed-964-296109e7fac3e1cbf6e19e632c2e443b.txt
processed-968-cce51afede0efc44bc400958fc91e090.txt
processed-969-9776da6853b90cd39cd5d5a326f32972.txt
processed-986-2800e2119587df434a92eaab941e63e9.txt
```

Figure 6: Processed URI files- having "burger" keyword in it

10 random URIs from Processed URI files- having "burger" keyword in it

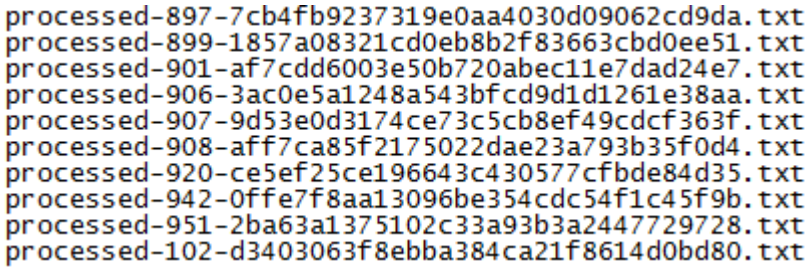

Figure 7: 10 random URIs from Processed URI files- having "burger" keyword in it

TFIDF Table

TFIDF	Rounded_TF	IDF	URI
0.0045	0.0006	7.4815	http://www.theguardian.com/us-news/2016/feb/08/mcdonalds-chef-neal-fraser-food-influencers
0.0067	0.0009	7.4815	http://www.eater.com/2016/1/28/10861108/mcdonalds-all-day-breakfast-mcgriddle-testing
0.0075	0.001	7.4815	http://www.rachaelrayshow.com/celebs/22190_rach_surprises_a_new_york_city_subway_hero/
0.0082	0.0011	7.4815	http://www.handelsblatt.com/unternehmen/handel-konsumgueter/schluss-mit-bio-mcdonalds-schmeisst-mcb-aus-dem-sortiment/12936600.html
0.0082	0.0011	7.4815	http://time.com/4206461/mcdonalds-happy-meals-books/?xid=tcoshare
0.015	0.002	7.4815	http://www.welt.de/wirtschaft/article151987009/McDonalds-scheitert-mit-seinem-Bio-Burger.html
0.0352	0.0047	7.4815	http://www.sueddeutsche.de/wirtschaft/fast-food-industrie-mcdonalds-verabschiedet-sich-schon-wieder-vom-bio-burger-1.2855377
0.0449	0.0006	7.4815	http://www.absatzwirtschaft.de/nach-durchwachsenem-markttest-in-deutschland-mcdonalds-beerdigt-den-bio-burger-74903/
0.1167	0.0156	7.4815	http://www.seriouseats.com/recipes/2015/09/the-mcwhopper-burger-king-mcdonalds.html
0.3606	0.0482	7.4815	https://twitter.com/burgerking/status/695760664028119041

Figure 8: TFIDF Table

3 Problem 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.
```

3.1 Solution

1. For this question I need to get the Page rank for each of the 10 URIs.
2. This can be found out by using one of the 3 links provided in the question.
3. I used 2nd link as it did not ask captcha to enter each time I check for a rank. This just reduced my work a little bit.
4. I manually copied each URI and pasted it in the PR estimator website and it gave me the rank.
5. First when I tried to get the rank from my URIs all my 10 URIs gave n/a output.
6. Later I read the comment in the google groups and found that it can use top-level URIs rather than my entire URI.
7. This is because deep links into a site does not have PR.
8. So I took top level of each URI and got the Page rank for each one of them. This can be seen in Figure 9.
9. Now each Page rank is normalized as asked in the question.
10. The list of Page ranks arranged in ascending order and the URIs are shown in Figure 10.
11. In the question I was asked to compare Page rank and TFIDF, so both of them are shown in a single table which can be seen in Figure 11.
12. When we compare the rankings based on TFIDF and page rank there is not relation between them.Page rank considers back link structures,so it gives the rank of the document with respect to other documents in the web,but TFIDF gives rank to the documents based on the level of similarity to the term.So If a document has high TFIDF value then it ranks it high even though if it is unpopular.

3.2 Results

Sample Rank for one URI

Pagerank Results	
Url	Pagerank
http://www.rachaelrayshow.com	6 / 10

Figure 9: Sample Rank for one URI

Page rank along with 10 URIs

Page Rank	URI
0	http://www.seriouseats.com/recipes/2015/09
0	http://time.com
0	http://www.welt.de
0.6	http://www.sueddeutsche.de/wirtschaft
0.6	http://www.handelsblatt.com/unternehmen
0.6	http://twitter.com/burgerking
0.6	http://www.absatzwirtschaft.de
0.6	http://www.eater.com
0.6	http://www.rachaelrayshow.com
0.7	http://www.theguardian.com

Figure 10: Page rank along with 10 URIs

Comparison between Pagerank and TFIDF

TFIDF	Page Rank	URI
0.1167	0	http://www.seriouseats.com/recipes/2015/09
0.0082	0	http://time.com
0.015	0	http://www.welt.de
0.0352	0.6	http://www.sueddeutsche.de/wirtschaft
0.0082	0.6	http://www.handelsblatt.com/unternehmen
0.3606	0.6	http://twitter.com/burgerking
0.0449	0.6	http://www.absatzwirtschaft.de
0.0067	0.6	http://www.eater.com
0.0075	0.6	http://www.rachaelrayshow.com
0.0045	0.7	http://www.theguardian.com

Figure 11: Comparison between Pagerank and TFIDF

4 Problem 4

```
4.  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
```

4.1 Solution

1. Here I need to calculate the kendall Tau b value and p-value.
2. I used Tau b because there are tie values in the rankings.
3. Tau b score is used to measure the association between the ranks produced by TFIDF and Page rank.
4. I have both the TFIDF and page rank values into two vectors and “cor.text” command is used in “R” giving method as “Kendall” which returns Tau b value and p-value.
5. Output :- tau = 0.7833, z= 2.8088, p-value = 0.0024
6. Here 3 URIs have 0 rank and 6 of them have 0.6 rank. So, there are a lot of ties and Kendall’s tau coefficient shows a high correlation.

4.2 Code Listing and Results

R code and results for Kendal's Tau b value and p value

```
1
2  R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
3  Copyright (C) 2014 The R Foundation for Statistical Computing
4  Platform: i386-w64-mingw32/i386 (32-bit)
5
6  R is free software and comes with ABSOLUTELY NO WARRANTY.
7  You are welcome to redistribute it under certain conditions.
8  Type 'license()' or 'licence()' for distribution details.
9
10   Natural language support but running in an English locale
11
12 R is a collaborative project with many contributors.
13 Type 'contributors()' for more information and
14 'citation()' on how to cite R or R packages in publications.
15
16 Type 'demo()' for some demos, 'help()' for on-line help, or
17 'help.start()' for an HTML browser interface to help.
18 Type 'q()' to quit R.
19
20 > Pagerank <- c(0,0,0,0.6,0.6,0.6,0.6,0.6,0.6,0.7)
21 >
22 > TFIDF <- c(0.0045,0.0067,0.0075,0.0082,0.0082,0.015,0.0352,0.0449,0.1167,0.3606)
23 >
24 > cor.test(TFIDF,Pagerank, method = "kendall", alternative = "greater")
25
26         Kendall's rank correlation tau
27
28 data:  TFIDF and Pagerank
29 z = 2.8088, p-value = 0.002486
30 alternative hypothesis: true tau is greater than 0
31 sample estimates:
32       tau
33 0.7833495
34
35 Warning message:
36 In cor.test.default(TFIDF, Pagerank, method = "kendall", alternative = "greater") :
37   Cannot compute exact p-value with ties
38 >
```

Listing 3: R code for Kendal's Tau b value

Bibliography

[1] Hashlib Secure Hashes and message digests https://docs.python.org/2/library/hashlib.html , Python Software Foundation, 2016

[2] Measures of association in R Kendall's tau-b and tau-c http://stackoverflow.com/questions/2557863/measures-of-association-in-r-kendalls-tau-b-and-tau-c, Developers from Stack Overflow, 2008

[3] Searching For A String In Multiple Files http://blamcast.net/articles/recursive-grep-find-strings-files, John, 2008

[4] getting syntax error near unexpected token ';' in python http://stackoverflow.com/questions/10900617/getting-syntax-error-near-unexpected-token-in-python, Developers from Stack Overflow, 2008

[5] math Mathematical functions https://docs.python.org/2/library/math.html, Python Software Foundation, 2016

[6] The size of the World Wide Web (The Internet) http://www.worldwidewebsize.com/, Maurice de Kunder, 2006

[7] Google PageRank Checker http://www.seocentro.com/tools/search-engines/pagerank.html, Developers at SEOCentro, 2016