Assignment #3

Wednesday, February 20TH, 2018

RANDY DO

Contents

Problem 1

```
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 to associate
them with their respective filename, 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 for all 1000 HTML documents.
"python-boilerpipe" will do a fair job see
(http://ws-dl.blogspot.com/2017/03/2017-03-20-survey-of-5-boilerplate.html):

        from boilerpipe.extract import Extractor
        extractor = Extractor(extractor='ArticleExtractor', html=html)
        extractor.getText()

Keep both files for each URI (i.e., raw HTML and processed).
Upload both sets of files to your github account.
```

Answer:

Imports I am using

```
import requests
import concurrent.futures
import md5
from bs4 import BeautifulSoup
import pickle
```

I am using to obtain the hash of the uri

```
def convert(uri):
    return md5.new(uri).hexdigest()
```

This gets me the html from the uniquelinks.txt

```
def get_html(uri):
    print('Getting {}'.format(uri))
    response = requests.get(uri)
    return response.url, response.status_code, response.content
```

With this, while I am in the uniquelinks.txt file, I can read each line

```
if __name__ == '__main__':
    with open('uniquelinks.txt') as infile:
        uris = [uri.rstrip('\n') for uri in infile]
```

While I was trying to format, I used this website as a reference.
https://docs.python.org/3/library/concurrent.futures.html#concurrent.futures.ThreadPoolExecutor

```
with concurrent.futures.ThreadPoolExecutor(max_workers=8) as executor:
    uri_futures = [executor.submit(get_html, uri) for uri in uris]
    for future in concurrent.futures.as_completed(uri_futures):
        try:
```

This allows my 1,000 links to process a bit faster, since the last assignment, it took me about 30 minutes to finish checking and writing each file.

This part writes a new txt file in either raw and processed. For processed, it uses Beautifulsoup to read then it writes it.

```
try:
    uri, status_code, content = future.result()
except Exception as exc:
    print('{} generated an exception: {}'.format(uri, exc))
    continue
if status_code == 200:
    hashed_uri = convert(uri)
    print('Writing {} as {}'.format(uri, hashed_uri))
    try:
        with open('html/raw/' + hashed_uri + '.txt', 'w') as outfile:
            outfile.write(uri + '\n')
            outfile.write(content)
        with open('html/processed/' + hashed_uri + '.processed.txt', 'w') as outfile:
            outfile.write(uri + '\n')
            outfile.write(BeautifulSoup(content).get_text().encode('utf8'))
```

```
import requests
import concurrent.futures
import md5
from bs4 import BeautifulSoup
import pickle

def convert(uri):
    return md5.new(uri).hexdigest()

def get_html(uri):
    print('Getting {}'.format(uri))
    response = requests.get(uri)
    return response.url, response.status_code, response.content

if __name__ == '__main__':
    with open('uniquelinks.txt') as infile:
        uris = [uri.rstrip('\n') for uri in infile]

    with concurrent.futures.ThreadPoolExecutor(max_workers=8) as executor:
        uri_futures = [executor.submit(get_html, uri) for uri in uris]
        for future in concurrent.futures.as_completed(uri_futures):
            try:
                uri, status_code, content = future.result()
            except Exception as exc:
                print('{} generated an exception: {}'.format(uri, exc))
                continue
            if status_code == 200:
                hashed_uri = convert(uri)
                print('Writing {} as {}'.format(uri, hashed_uri))
                try:
                    with open('html/raw/' + hashed_uri + '.txt', 'w') as outfile:
                        outfile.write(uri + '\n')
                        outfile.write(content)
                    with open('html/processed/' + hashed_uri + '.processed.txt', 'w') as outfile:
                        outfile.write(uri + '\n')
                        outfile.write(BeautifulSoup(content).get_text().encode('utf8'))
                except Exception as e:
                    print 'Ignoring:' + uri
                    print e
            else:
                print('Not writing {}, bad status code: {}'.format(uri, status_code))
```

Problem 2:

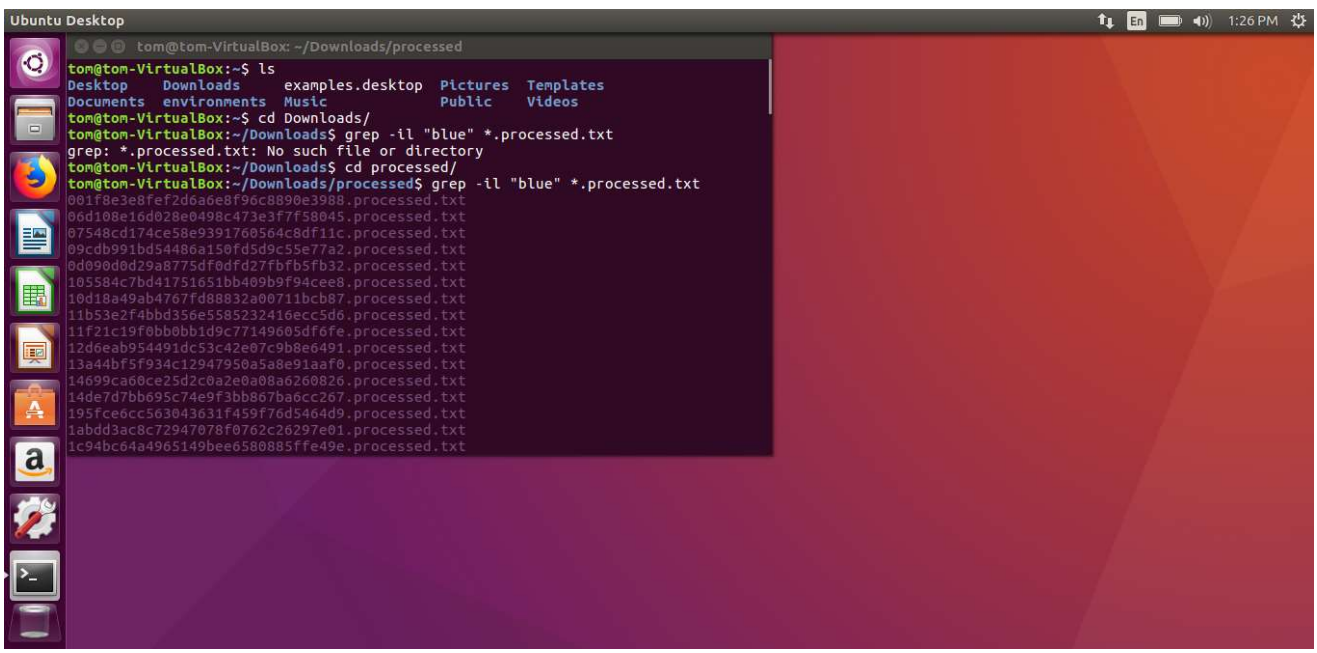
```
2.  Choose a query term (e.g., "shadow") that is not a stop word
(see week 5 slides) and not HTML markup from step 1 (e.g., "http")
that matches at least 10 documents (hint: use "grep" on the processed
files).  If the term is present in more than 10 documents, choose
any 10 from your list.  (If you 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    10.680   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, just explain how you did it.

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

https://en.wikipedia.org/wiki/Significant_figures#Rounding_and_decimal_places

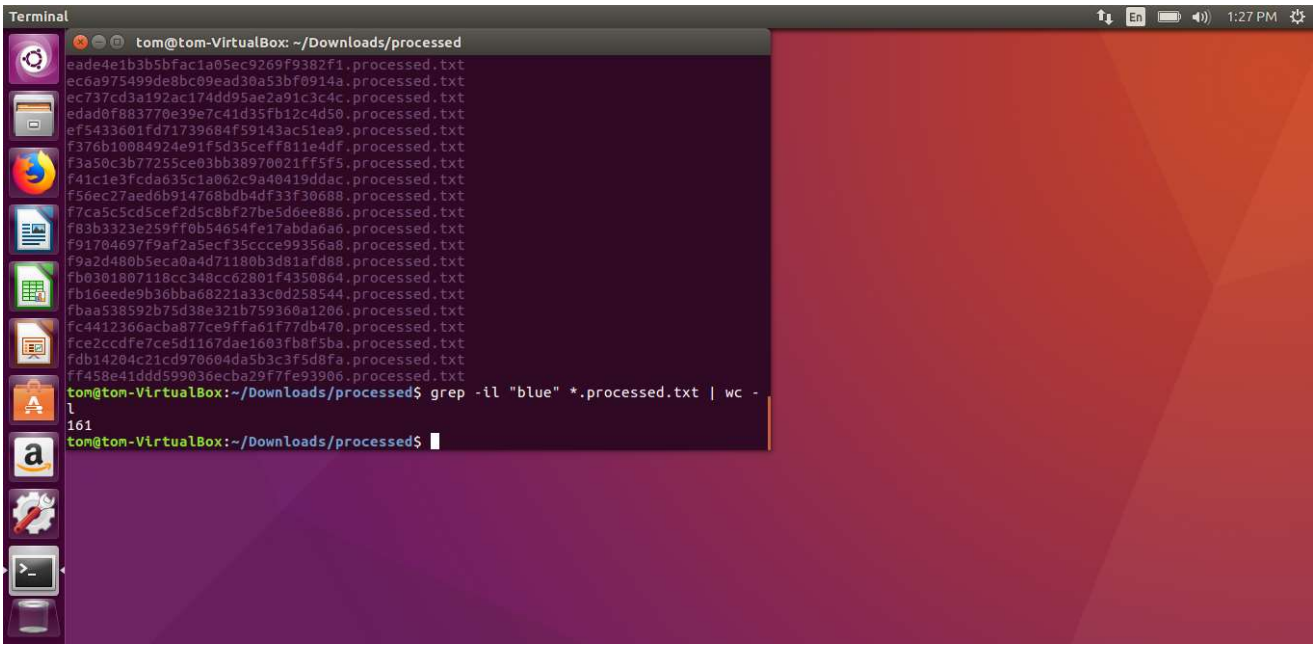
I didn't have a linux os to use the "grep" function, so instead, I used a friend's computer who had ubuntu.

For this, I used $grep .il "blue" *processed.txt



After using grep, I used

$grep -il "blue" *.processed.txt | wc -l
This will give me the number of files that contains "blue"

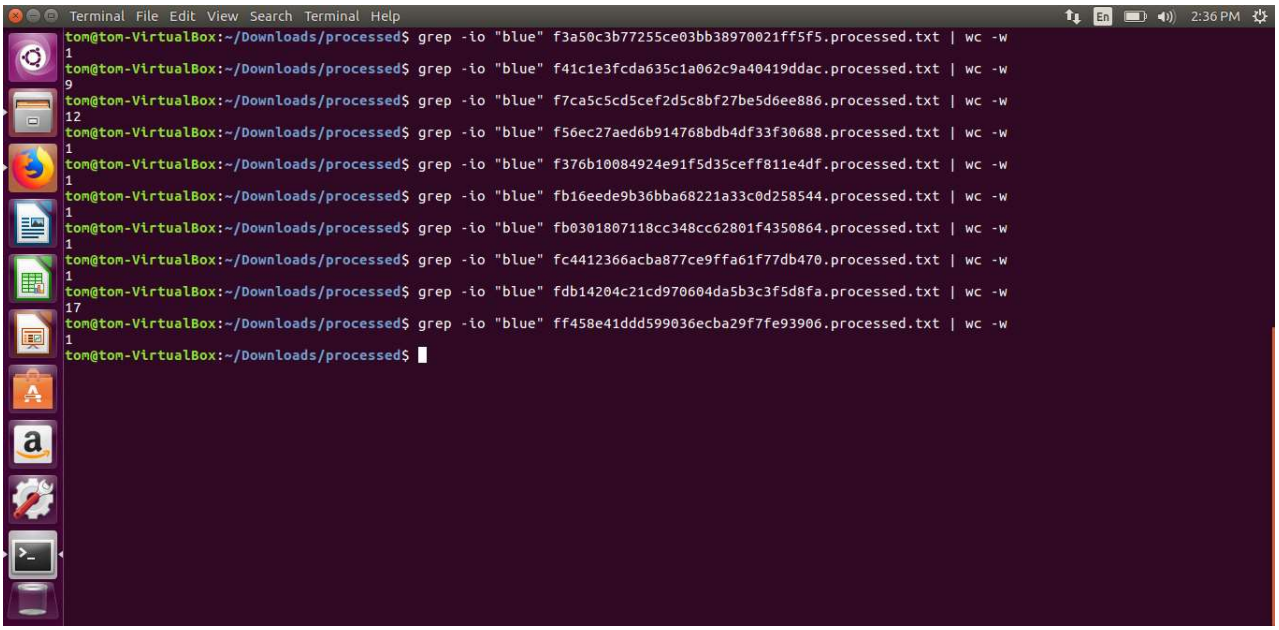


Afterwards, I chose 10 different files and put them into the folder called **keywordlinks**

Next, I want to find the “blue” word count for each files

I used

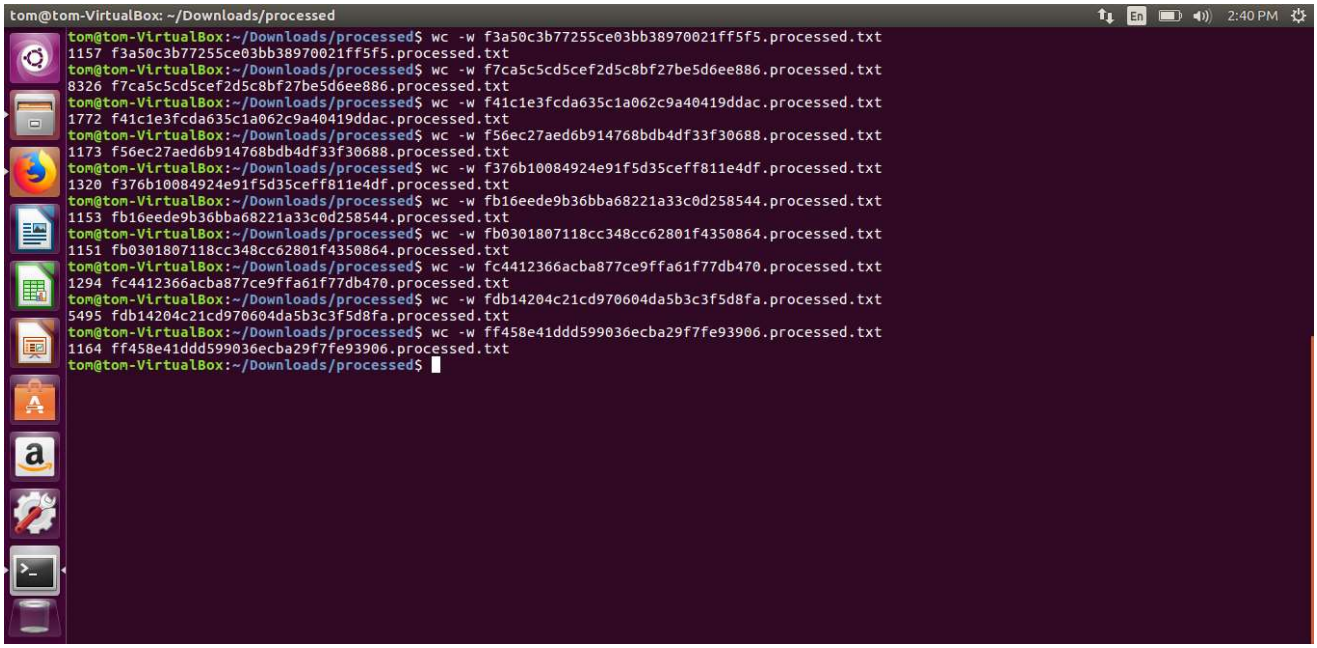
$ grep -io “blue” “DOCUMENT FILENAME” | wc -w



Next is finding the word count for each file

I used

wc – w “DOCUMENT FILENAME”



After gathering the information, I am creating a table

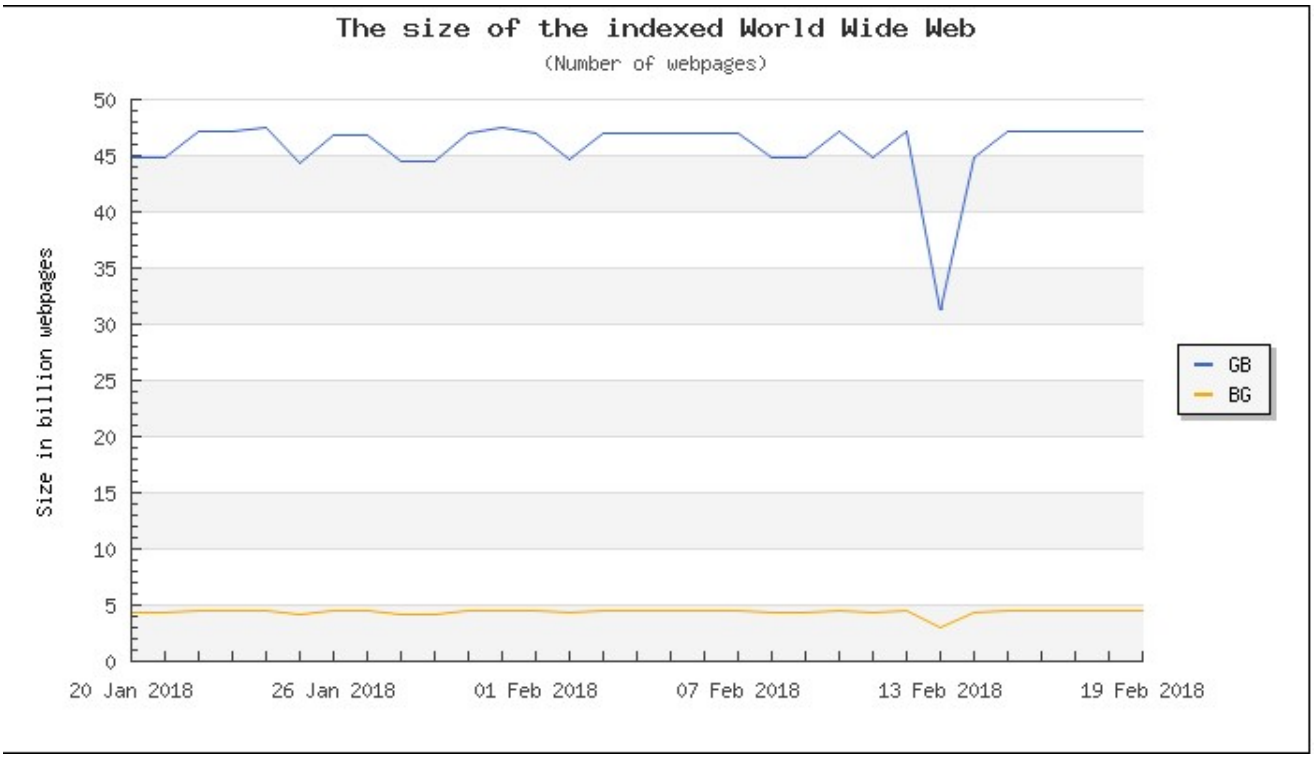
TF: # that contains the word “blue”
WC: Total word Count for a file
N-TF: The normalized TF for the document (TF/WC)
URI: The name of the URI

TF	WC	N- TF	URI
1	1157	0.00086	http://www.cnn.com/2017/02/05/entertainment/hamilton-schuyler-sisters-sisterhood-america-the-beautiful/index.html?sr=twCNN020617hamilton-schuyler-sisters-sisterhood-america-the-beautiful0730AMStoryLink&linkId=34154767
9	8326	0.00108	https://www.vevo.com/watch/taylor-swift/out-of-the-woods-the-making-of/USCJY1631657
12	1772	0.00677	https://www.facebook.com/karina.avellino/posts/10103054572229945
1	1173	0.00085	http://www.cnn.com/2017/02/03/politics/yemen-raid-trump-obama/index.html?sr=twpol020417yemen-raid-trump-obama0455PMVODtopLink&linkId=34130355
1	1320	0.00076	http://www.cnn.com/2017/02/06/us/meteor-over-midwest-caught-on-video/index.html?sr=twCNN020617N/A0455PMVODtopVideo&linkId=34174619
1	1153	0.00087	http://www.cnn.com/videos/entertainment/2017/02/01/jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn?sr=twCNN020517jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn0801AMVideoVideo&linkId=34111817
1	1151	0.00087	http://www.cnn.com/2017/02/05/politics/bernie-fashion-cnntv/index.html?sr=twCNN020617bernie-fashion-cnntv1056AMVODtopLink&linkId=34159448
1	1294	0.00077	http://www.cnn.com/2017/02/01/health/tree-man-syndrome-girl-bangladesh/index.html?sr=twCNN020617tree-man-syndrome-girl-bangladesh0630AMStoryLink&linkId=34154752
17	5495	0.00309	https://www.facebook.com/BalancedBodySystems/posts/463562493714178
1	1164	0.00086	http://www.cnn.com/2017/02/06/politics/trump-muslim-ban-travel-lawsuit/index.html?sr=twCNN020617trump-muslim-ban-travel-lawsuit0621PMVODtopLink&linkId=34178312

After inputting all of the numbers, I need to find the IDF, According to http://www.wordwidewebsize.com/ the total indexed web pages in google is roughly 47.5 (looking at the graph) When I searched “blue” in google, I got about 4.91 billion results.

IDF (‘blue’) = log2 (47.5b/4.91b)= 3.2741

From the equation to obtain TF-IDF is:
TF – IDF = (TF) x (IDF)


The size of the indexed World Wide Web
(Number of webpages)
Size in billion webpages
50
45
40
35
30
25
20
15
10
5
0
20 Jan 2018
26 Jan 2018
01 Feb 2018
07 Feb 2018
13 Feb 2018
19 Feb 2018
GB
BG

TFIDF	TF	IDF	URI
0.0028	0.00086	3.2741	http://www.cnn.com/2017/02/05/entertainment/hamilton-schuyler-sisters-sisterhood-america-the-beautiful/index.html?sr=twCNN020617hamilton-schuyler-sisters-sisterhood-america-the-beautiful0730AMStoryLink&linkId=34154767
0.0035	0.00108	3.2741	https://www.vevo.com/watch/taylor-swift/out-of-the-woods-the-making-of/USCJY1631657
0.0222	0.00677	3.2741	https://www.facebook.com/karina.avellino/posts/10103054572229945
0.0028	0.00085	3.2741	http://www.cnn.com/2017/02/03/politics/yemen-raid-trump-obama/index.html?sr=twpol020417yemen-raid-trump-obama0455PMVODtopLink&linkId=34130355
0.0025	0.00076	3.2741	http://www.cnn.com/2017/02/06/us/meteor-over-midwest-caught-on-video/index.html?sr=twCNN020617N/A0455PMVODtopVideo&linkId=34174619
0.0028	0.00087	3.2741	http://www.cnn.com/videos/entertainment/2017/02/01/jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn?sr=twCNN020517jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn0801AMVideoVideo&linkId=34111817
0.0028	0.00087	3.2741	http://www.cnn.com/2017/02/05/politics/bernie-fashion-cnntv/index.html?sr=twCNN020617bernie-fashion-cnntv1056AMVODtopLink&linkId=34159448
0.0025	0.00077	3.2741	http://www.cnn.com/2017/02/01/health/tree-man-syndrome-girl-bangladesh/index.html?sr=twCNN020617tree-man-syndrome-girl-bangladesh0630AMStoryLink&linkId=34154752
0.0101	0.00309	3.2741	https://www.facebook.com/BalancedBodySystems/posts/463562493714178
0.0028	0.00086	3.2741	http://www.cnn.com/2017/02/06/politics/trump-muslim-ban-travel-lawsuit/index.html?sr=twCNN020617trump-muslim-ban-travel-lawsuit0621PMVODtopLink&linkId=34178312

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://pr.eyedomain.com/
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 are only 10 to do.  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).  Also
note that these tools typically report on the domain rather than
the page, so it's not entirely accurate.

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.
```

Page Rank (PR)	TFIDF	TF	IDF	URI
0.9	0.0028	0.00086	3.2741	http://www.cnn.com/2017/02/05/entertainment/hamilton-schuyler-sisters-sisterhood-america-the-beautiful/index.html?sr=twCNN020617hamilton-schuyler-sisters-sisterhood-america-the-beautiful0730AMStoryLink&linkId=34154767
0.7	0.0035	0.00108	3.2741	https://www.vevo.com/watch/taylor-swift/out-of-the-woods-the-making-of/USCJY1631657
0.9	0.0222	0.00677	3.2741	https://www.facebook.com/karina.avellino/posts/10103054572229945
0.9	0.0028	0.00085	3.2741	http://www.cnn.com/2017/02/03/politics/yemen-raid-trump-obama/index.html?sr=twpol020417yemen-raid-trump-obama0455PMVODtopLink&linkId=34130355
0.9	0.0025	0.00076	3.2741	http://www.cnn.com/2017/02/06/us/meteor-over-midwest-caught-on-video/index.html?sr=twCNN020617N/A0455PMVODtopVideo&linkId=34174619
0.9	0.0028	0.00087	3.2741	http://www.cnn.com/videos/entertainment/2017/02/01/jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn?sr=twCNN020517jon-stewart-predicts-trump-executive-order-colbert-jnd-orig-vstan.cnn0801AMVideoVideo&linkId=34111817
0.9	0.0028	0.00087	3.2741	http://www.cnn.com/2017/02/05/politics/bernie-fashion-cnntv/index.html?sr=twCNN020617bernie-fashion-cnntv1056AMVODtopLink&linkId=34159448
0.9	0.0025	0.00077	3.2741	http://www.cnn.com/2017/02/01/health/tree-man-syndrome-girl-bangladesh/index.html?sr=twCNN020617tree-man-syndrome-girl-bangladesh0630AMStoryLink&linkId=34154752

Compare and Contrast:

If we compare and contrast the frequency measurement, then page rank is unrelated because the search term isn't taken as an input when calculating page rank. When using page rank, it's objective is to find pages with a higher probability of a user randomly navigating and going to that website. Basically, the higher the PR rank, the more likely the user goes to that website, when searching a something. However, this result is false, since my keyword is "blue", but the keyword blue can most likely only be found in the html/css.