CS 595: Assignment 2

Mallika Kogatam

Fall 2014

Contents

1 Problem 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:

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

Keep both files for each URI (i.e., raw HTML and processed).

If you're feeling ambitious, "boilerpipe" typically does a good
job for removing templates:

https://code.google.com/p/boilerpipe/
```

Answer

Downloading the URIs was done easily by a small shell script.The program is as follows

```
#!/bin/bash
links=`cat uniquelinks.txt`

date=`date +"%Y-%m-%d_%H.%M.%S"`

mkdir "$date"

for link in $links
do
    echo $link
    filename=`echo -n $link | md5sum | cut -f1 -d" "`
    echo "$filename $link" >> "$date/linkmap.txt"

    echo $filename
    curl -A "Mozilla/4.0" --connect-timeout 30 "$link" > "$date/$filename.raw"
done
```

Listing 1: Shell Program for downloading URIs from Assignment 2

1. The Shell program reads the file which had 1000 Unique links and gets the raw content for each link
2. Initially I was little confused with organizing the folders which have raw files and the processed files, so to make my life easy I created the name of folder with the date and time by using date command
3. In future we might need the raw files to get the plain text, some times Linux might not accept the special characters in the file name so its better to convert the file name to a non special characters format.
4. Converting the file name to a md5 hash format is the best way to deal with such problems.
5. Keeping track of the mapping between md5 hash for the respective URI is needed.
6. The respective md5 hash to a URI is stored in the file linkmap.txt.
7. I am using the CURL command to get the raw content for each URI.
8. For some URI's its taking too long to load the content so I used connect–timeout to cut down processing time to only 30 seconds. It its taking more than 30 seconds it skips that URI and checks for the next URI.
9. Raw content for each URI is stored in an file with the name as 00a8ff411899cca3c7487fa56331730a.raw as extension. All these raw files are stored in a folder with the creation date as the name of the folder.

To remove the HTML mark up from the raw content collected from the each URI the following shell Program is used.

```
#!/bin/bash

directory="/home/mkogatam/mallika/cs595/A3/q1/2014-09-27_15.51.01"
rawfiles=`ls $directory`

plainfile="/home/mkogatam/mallika/cs595/A3/q1/2014-09-27_15.51.01-processed"
mkdir $plainfile

for file in $rawfiles
do
    echo $file
    lynx -dump -force_html "$directory/$file" >> $plainfile/`basename "$file" .raw`.processed
done
```

Listing 2: Shell Program for getting the Plain Text for 1000 Raw files

1. All the files that are generated from the previous program are read and the HTML mark is removed and stored in a different file
2. I used LYNX command to get rid of the the HTML Markup.
3. I am saving the file that is generated after removing the HTML Markup as 00a8ff411899cca3c7487fa56331730a.processed
4. All the processed files are saved in a folder that matches with the raw files folder with an extension of "rocessed".

1.1 Results

linkmap.txt

1c28e2cb437bbaa45f4a799c7d4b1aaa http://greatsss.com
a34cea370b787b16d3cecd34c062362c http://ultimate-dickhead.tumblr.com/
8b72f67d454888326495f7d686ff07c2 http://tweetsta.com/index.php
80f2ca6b52a7b93f1287c0f459bf6dd7 http://itssongoku.tumblr.com
2fe1e700f6049cf1d37012bf6fd76288 https://twitter.com/zilliamson
1318dd617474c1fd04e9da41e8b21d76 http://socialistworker.co.uk
2d9d141a31fcc369f6adb699f43ad273 http://www.enzozelocchi.com
65df96316d6e1994316adc56fdd96491 http://www.youtube.com/watch?v=VAPI0ciXs6A&feature=youtu.be
4668da8f73de51e14dd24f50161b9c28 http://www.rageon.com/collections/pokemon/?rageon.com
82d1d13b747b168a47bcdb42051dc0e3 http://www.youtube.com/user/FreeFallingHQ
00844d0c3f0ebbdd518d9cf2e77efe52 http://www.youtube.com/watch?v=frNsJvV5zAs&feature=youtu.be
4b58e9674be247bc7c2768c1b320a6aa http://cespedesfamilybarbecue.com/about/
552fc11084aea84bdc99bc9ef72dac0b http://www.natethehitmaker.com/
e70539fccdbed546db13d7eeb35e844f http://ImMackenson.com
90e4b74ba16fc8f1a4b366cdb263e61c http://anhonestyear.com/
483db91dfe7a24b2967f9061d9c81f1b http://www.romeolacoste.com
fe5f1d7a5d0f14a7d96b804cd9228666 http://smarturl.it/RFLT
bc4605198469ce593cbfe1618a01e15b http://proguitarshop.com/
3bd61f876f36b90c581d9ffb8e493c6d http://ohholybutt.tumblr.com
b7e24a60c933ca433da98f351c101ede http://Instagram.com/lawsofsex
8a373d206b0dfe155604d33ee839cab0 http://www.photografyum.com
9875c3f09b696b861294860dd5ba48b7 http://www.hrtv.com
b6917d4cf55918335775fc77a8943f44 http://sensualgif.tumblr.com/
bde865b3dfc14ea228f71efa7dcaff1c http://kushoverboys.tumblr.com
a4d210430013cb5caa4295c96c127569 http://www.ew.com/ew/
05f07cbdcf7b02b5e7d90098ccec9755 http://english.alarabiya.net/
155827cff968a444d719706efa4561e9 https://twitter.com/nationofbiebs/status/457342650874474496
ea39e5ba49746ebea802221701343ba6 http://www.womenshealthmag.com

2 Problem 2

Question

```
2.  Choose a query term (e.g., "shadow") that is not a stop word
(see week 4 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 4 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/
0.085 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.

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

2.1 Solution

I choose the keyword "music" for my search. So searching for the term "music" yielded quite a few results. This was done with the following command, which also selects out the tops 10 for use in this question

```
grep -i music *.processed | awk -F: '{print $1}' | \
sort | uniq -c | sort -rn | head -n 10
```

Which returned:

```
 70 849eb65c56a594275d14344d5dbe8129.processed
 54 d56232cb8ce6fe9624ec90451435ab94.processed
 53 403420845ed7e9e3d3fcccc69b21a30f.processed
 41 ab69c40500d93f26154d61b41b584465.processed
 40 bdc1fcffc976b155b83cb489c3c50382.processed
 39 d871b4a90834a0b1369bf3bf5dae503f.processed
 38 b4a4921e1ae50ea861e2b9c0db2bf666.processed
 38 98097117c84edd657e374d07c6ea86cb.processed
 36 5d410b78a0056341523ab8e6322331a0.processed
 34 a4d210430013cb5caa4295c96c127569.processed
```

This gives us raw term frequency for these files but does not give us the word count so we can normalize it. to get the word count for the calculation I have written a small shell program which gives the word count for each file and gets the respective URI.

```
1  #!/bin/bash
2
3  links=`cat q2-1-result.txt | cut -f2 -d" "`
4
5  for link in $links
6  do
7      file="../q1/2014-09-27_15.51.01-processed/$link"
8      hash=`basename $link .processed`
9      url=`grep $hash linkmap.txt | cut -d" " -f2`
10     occur=`wc -w $file | cut -d" " -f1`
11     words=`grep -ci music $file`
12     echo "$occur $words $hash $url"
13 done
```

Listing 3: Shell Program for wordcount of each file and to get URI

This give the output as follows

```
2058 70 849eb65c56a594275d14344d5dbe8129 http://www.xxlmag.com
4680 54 d56232cb8ce6fe9624ec90451435ab94 http://www.soundclick.com/bands/
    default.cfm?bandID=1350875
1601 53 403420845ed7e9e3d3fcccc69b21a30f http://www.bet.com
320 41 ab69c40500d93f26154d61b41b584465 https://soundcloud.com/scorpios4music
766 40 bdc1fcffc976b155b83cb489c3c50382 http://www.kaskademusic.com
1878 39 d871b4a90834a0b1369bf3bf5dae503f http://runthetrap.com
789 38 b4a4921e1ae50ea861e2b9c0db2bf666 http://TrueMusicInHipHop.blogspot.com/
1293 38 98097117c84edd657e374d07c6ea86cb http://music.iamlights.com/
2071 36 5d410b78a0056341523ab8e6322331a0 http://www.famoushollywoodbay.com
3408 34 a4d210430013cb5caa4295c96c127569 http://www.ew.com/ew/
```

The Above output gives the word count for each file followed by the occurrence of the word music and the file name followed by the URI.

To generate the TDF we need to know the files that have the term "music" in it out of the total files in the corpus. so to get the number of files that have the key word i executed the following command.That gives the output as 235.

```
grep -i music *.processed | awk -F: '{print $1}' | sort | uniq -c | wc -l
```

To calculate the TF,IDF,TFDIF value i have written a small python code which give the output in a table. The program is as follows.

```
#! /usr/bin/env python

import math

def main():
    idf = math.log(
        float( 1007 ) / 235 ,
        2
    )

    #print idf

    data_file = open( "wrd_uri.txt", "r" )

    print "{:^8} {:^8} {:^8} {:^37}".format(
        "TFDIF",
        "TF",
        "IDF",
        "URI"
    )
    print "-" * 64

    for line in data_file:
        line = line.split()
        #print line[0:2] + line[3:]
        count_word    = float(line[0])
        count_keyword = float(line[1])
        url           = line[3]

        tf    = count_keyword / count_word
        tfidf = tf * idf

        print "{:>8.2f} {:>8.2f} {:>8.2f} {}".format(
            tfidf,
            tf,
            idf,
            url
        )



if __name__ == "__main__":
    main()
```

Listing 4: Python Program for calculating TF,IDF,TFDIF of each URI

1. To calculate the IDF value

$$\log_2\left(\frac{totaldocsincorpus}{docswithterm}\right) = \log_2\left(\frac{1007}{235}\right)$$

2. Here the total docs in corpus is the total number of files in which we are search for a particular keyword. so i have 1007 files in which i am intended to search for "music".

3. And the "docs with term" is the number of files which have the key word "music", in this case its 235 out of 1007 files.

4. So by the above values the IDF is calculated in the program.

5. The TF values is Count of keyword times the count of word in the files.

6. As the TF and IDF are calculated its easy to calculate TFIDF value just by multiplying them.

7. Initially I rounded the decimal value to 4 digits but later on i rounded it to two decimal so that it makes the calculations easier in the Problem 4.

2.2 Result

Resul with 4 decimal places.

TDIF	TF	IDF	URI
0.0714	0.0340	2.0993	http://www.xxlmag.com
0.0242	0.0115	2.0993	http://www.soundclick.com/bands/default.cfm?bandID=1350875
0.0695	0.0331	2.0993	http://www.bet.com
0.2690	0.1281	2.0993	https://soundcloud.com/scorpios4music
0.1096	0.0522	2.0993	http://www.kaskademusic.com
0.0436	0.0208	2.0993	http://runthetrap.com
0.1011	0.0482	2.0993	http://TrueMusicInHipHop.blogspot.com/
0.0617	0.0294	2.0993	http://music.iamlights.com/
0.0365	0.0174	2.0993	http://www.famoushollywoodbay.com
0.0209	0.0100	2.0993	http://www.ew.com/ew/

Resul with 2 decimal places.

TFDIF	TF	IDF	URI
0.07	0.03	2.10	http://www.xxlmag.com
0.02	0.01	2.10	http://www.soundclick.com/bands/default.cfm?bandID=1350875
0.07	0.03	2.10	http://www.bet.com
0.27	0.13	2.10	https://soundcloud.com/scorpios4music
0.11	0.05	2.10	http://www.kaskademusic.com
0.04	0.02	2.10	http://runthetrap.com
0.10	0.05	2.10	http://TrueMusicInHipHop.blogspot.com/
0.06	0.03	2.10	http://music.iamlights.com/
0.04	0.02	2.10	http://www.famoushollywoodbay.com
0.02	0.01	2.10	http://www.ew.com/ew/

3 Problem 3

3.1 Problem 3

```
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.
```

3.2 Solution

1. Using the URI's acquired form Question 2, I put each into one of the suggested Page Rank Calculators.
2. Unsurprisingly three websites gave different results for each URI. They seem like they have different data for each sites.We can observe this from Table 1.
3. So I picked up the values got from PR checker as the Page Ranks for each Website. And I normalized with N/A and Error with 0.0. Table 2 shows the Normalized Ranks for each Website.
4. Table 3 contains the PageRank and TFIDF values for comparison
5. For this set of URI's, doesn't seem to be a whole lot of correlation between PageRank and TFIDF.
6. The URI with the highest TFIDF of 0.27 has a PageRank of 0.0 and the page with the lowest TFIDF of 0.02 had a PageRank of 0.7 or 0.0

Check PageRank	SEO Central	PR Checker	URI
6/10	5/10	6/10	http://www.xxlmag.com
INVALID DOMAIN	undef	N/A	http://www.soundclick.com/bands/default.cfm?bandID=1350875
0/10	6/10	6/10	http://www.bet.com
INVALID DOMAIN	2/10	ERROR	https://soundcloud.com/scorpios4music
5/10	5/10	5/10	http://www.kaskademusic.com/
3/10	3/10	3/10	http://runthetrap.com
0/10	undef	N/A	http://TrueMusicInHipHop.blogspot.com
5/10	5/10	5/10	http://music.iamlights.com
0/10	undef	N/A	http://www.famoushollywoodbay.com/
INVALID DOMAIN	7/10	7/10	http://www.ew.com/ew/

Table 1: PageRank of URIs containing the word *music*, based on the CheckPageRank, SEO Central, and PR Checker Page Rank services

PAGE RANK	URI
0.6	http://www.xxlmag.com
0.0	http://www.soundclick.com/bands/default.cfm?bandID=1350875
0.6	http://www.bet.com
0.0	https://soundcloud.com/scorpios4music
0.5	http://www.kaskademusic.com/
0.3	http://runthetrap.com
0.0	http://TrueMusicInHipHop.blogspot.com
0.5	http://music.iamlights.com
0.0	http://www.famoushollywoodbay.com/
0.7	http://www.ew.com/ew/

Table 2: Normalized PageRank of URIs containing the word *music*, based on the PR Checker Page Rank services

TFIDF	PAGE RANK	URI
0.02	0.7	http://www.ew.com/ew/
0.07	0.6	http://www.xxlmag.com
0.07	0.6	http://www.bet.com
0.11	0.5	http://www.kaskademusic.com/
0.06	0.5	http://music.iamlights.com
0.04	0.3	http://runthetrap.com
0.02	0.0	http://www.soundclick.com/bands/default.cfm?bandID=1350875
0.27	0.0	https://soundcloud.com/scorpios4music
0.10	0.0	http://TrueMusicInHipHop.blogspot.com
0.04	0.0	http://www.famoushollywoodbay.com/

Table 3: PageRank of URIs containing the word *football*, based on the PR Checker Page Rank service, with *Not available* replaced with 0 and all other values normalized, sorted by decreasing PageRank

4 Problem 4

Question

```
==================================================
======Question 4 is for 3 points extra credit=======
==================================================

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
```

Answer

To calculate Kendall's τ_B, one uses the following equation[**?**]:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

where:

$$\begin{aligned}
n_0 &= n(n-1)/2 \\
n_1 &= \sum_i t_i(t_i - 1)/2 \\
n_2 &= \sum_j u_j(u_j - 1)/2 \\
n_c &= \text{Number of concordant pairs} \\
n_d &= \text{Number of discordant pairs} \\
t_i &= \text{Number of tied values in the } i^{th} \text{ group of ties for the first quantity} \\
u_j &= \text{Number of tied values in the } j^{th} \text{ group of ties for the second quantity}
\end{aligned}$$

The first step in calculating Kendall's Tau is to get the number of Concordant and Discordant pairs. I had a difficult time understanding how to go about this.

Finally after doing some research I understood that if we have n observations of data, which are tuplesx_i,$y_i$of the two variables. Take any two of the possible tuples (pairs), and if "both values go in the same direction", then they are concordant. Formally, for two observations i,j, the pairs are concordant if either $xi>xj$ and $yi>yj$, or if $xi>xj$ and $yi>yj$ other wise its discordant.

Table 4 shows the rankings listed with concordant and discordant pairs counted in columns **C** and **D** respectively, giving us $n_c = 44$ and $n_d = 46$.

This gets us the top part of the equation, but not the bottom. For that we need the number of ties for each value.

For PageRank, from Table 2 there were only ties on values 0.6 and 0.5 and 0.0, so the ties are $n_1 = \sum_i t_i(t_i - 1)/2 = t_{0.6}(t_{0.6} - 1)/2 + t_{0.5}(t_{0.5} - 1)/2 + t_{0.0}(t_{0.0} - 1)/2 = 2(2-1)/2 + 2(2-1)/2 + 4(4-1)/2 = 2(1)/2 + 2(1)/2 + 4(3)/2 = 1 + 1 + 6 = 8$. This sets $n_1 = 8$.

For TFIDF, from Table 3 there were only ties on values 0.02 and 0.07 and 0.0, so the ties are $n_1 = \sum_i u_j(u_j - 1)/2 = u_{0.6}(t_{0.6} - 1)/2 + u_{0.5}(t_{0.5} - 1)/2 + u_{0.0}(t_{0.0} - 1)/2 = 2(2-1)/2 + 2(2-1)/2 = 2(1)/2 + 2(1)/2 = 1 + 1 = 2$. This sets $n_2 = 2$.

The total number of ranked items is 10, making $n = 10$, and $n_0 = 10(10-1)/2 = 90/2 = 45$.

Therefore:

$$\tau_B = \frac{44 - 46}{\sqrt{(45-8)(45-2)}} = \frac{-2}{\sqrt{37 \times 43}} = - \approx \frac{-2}{40} = -0.05$$

To get the $p - value$ we need to calculate the Z-score, which, for Kendall Tau B, is:

$$z_B = \frac{n_c - n_d}{\sqrt{v}}$$

where

$$\begin{aligned}
v &= \frac{(v_0 - v_t - v_u)}{18 + v_1 + v_2} \\
v_0 &= n(n-1)(2n+5) \\
v_t &= \sum_i t_i(t_i - 1)(2t_i + 5) \\
v_u &= \sum_j u_j(u_j - 1)(2u_j + 5) \\
v_1 &= \frac{\sum_i t_i(t_i - 1) \sum_j u_j(u_j - 1)}{(2n(n-1))} \\
v_2 &= \frac{\sum_i t_i(t_i - 1)(t_i - 2) \sum_j u_j(u_j - 1)(u_j - 2)}{9n(n-1)(n-2)}
\end{aligned}$$

Well, we shall start from the bottom.

As noted before, for PageRank, there 2 ties on 0.6, 2 on 0.5 and 4 on 0.0, and 4 ties TF*IDF values 2 ties on 0.07, 2 on 0.02.

This makes $v_2 = 0$ due to the fact that $\sum_j u_j(u_j - 1)(u_j - 2) = 0$ because $(u_j - 2)$ becomes 0

$v_1 = \frac{((4(4-1))+(2(2-1)+(2(2-1)))+(2(2-1)+(2(2-1))}{2(10)(9)} = \frac{64}{180} = 0.355$

Seeing as we had ties in TFIDF, $v_u = 2(2-1)(2(2)+5) + 2(2-1)(2(2)+5) = 2(1)(4+5) + 2(1)(4+5) = 18 + 18 = 36$, thus $v_u = 36$.

Seeing as we had ties in PageRank, $v_t = 2(2-1)(2(2)+5) + 2(2-1)(2(2)+5) + 4(4-1)(2(4)+5) = 2(1)(4+5) + 2(1)(4+5) + 6(5)(12+5) = 2(9) + 2(9) + 12(13) = 18 + 18 + 156 = 192$, thus $v_t = 192$.

There were 10 items to compare, so $v_0 = 10(10-1)(2(10)+5) = 10(9)(20+5) = (90)(25) = 2250$, thus $v_0 = 2250$.

Now we have arrived at $v = \frac{(2250-192-36)}{18+0.355+0} = \frac{2022}{18.355} \approx 110$.

From before $n_c = 44$ and $n_d = 46$, so the Z-score is

$$z_B = \frac{44 - 46}{\sqrt{110}} \approx \frac{-2}{10.49} = -0.19$$

At this point, I realized that the Web contained a conflicting array of calculators and information on acquiring a p-value from a Z-score.I thought Using R is the best. so I execute the following command in R to get the p value.

```
> 2*pnorm(-abs(-0.19))
[1] 0.8493091
```

Which gives me

$$p = 0.849$$

Seems like there is not coorelation between PageRank and TFIDF

Page Rank Ranking	**TF*IDF Ranking**	**C**	**D**	**URI**
1	2	0	9	http://www.ew.com/ew/
2	9	6	3	http://www.xxlmag.com
3	6	6	3	http://www.bet.com
4	5	4	5	http://www.kaskademusic.com
5	10	5	4	http://music.iamlights.com/
6	1	5	4	http://runthetrap.com
7	8	5	4	http://www.soundclick.com/bands/default.cfm?bandID=1350875
8	3	9	0	https://soundcloud.com/scorpios4music
9	7	1	8	http://TrueMusicInHipHop.blogspot.com/
10	4	3	6	http://www.famoushollywoodbay.com
Totals		21	24	

Table 4: Ranking of URIs by PageRank and TF*IDF, with Concordant Pairs (**C**) and Discordant Pairs (**D**), for Kendall Tau calculations

Bibliography

[1] Calculating p values. http://www.cyclismo.org/tutorial/R/pValues.html.

[2] Check page rank of web site. http://www.prchecker.info/check_page_rank.php.

[3] Kendall tau calculation. http://math.stackexchange.com/questions/462818/kendall-tau-calculation.

[4] Kendall tau rank correlation. http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient#Tau-b.

[5] Mathematical expressions in latex. https://www.sharelatex.com/learn/Mathematical_expressions.

[6] Measure of association in r - kendall's tau-b. http://www.prchecker.info/check_page_rank.php.

[7] Page rank check. http://www.seocentro.com/tools/search-engines/pagerank.html.