Assignment 3

CS532-s16: Web Sciences

Spring 2016

John Berlin

Generated on February 18, 2016

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

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

Answer

Downloading the `URI's` was relatively simple thanks to using shell scripts which can be found in listing 1. The script takes no arguments as I wish to make things as fire and forget as possible. So make sure your in the working directory of all the files.

To run the script execute it as such:

```
$ chmod +x hashUrisDownload.sh
$ ./hashUrisDownload.sh
```

The process is described as such. See the comments in the listing for full details.

1. Get the working directory and make file structure
2. For each `URI` in thousand.dat downloaded the `URI-R` and gets its hash. Curl with a supplied user agent and md5sum are used to produce these results. I use awk to print only the hash produced by md5sum.
3. Append the `URI` and its hash `ulrToHash.csv`
4. Clean up after ourselves `ulrToHash.csv`. To do so I use find with -size 0c. 0c means I want to find files that are 0 bytes or empty

To strip the html from the files I first used lynx but was extremely disappointed with it. I found it to keep way too much of the javascript and style tag contents. A better method was hinted at thus I looked into it. Thanks to my favourite search engine DuckDuckGo I was directed to use this language called `Perl`.

To run the script execute it as such:

```
$ perl removeHtml.pl
```

The `Perl` script used to correctly remove the html content from the hash.html files is found in listing 2. The process in brief is as such. See the comments in the listing for full details.

1. Libraries
   (a) File::Slurp read the entire file
   (b) HTML::Restrict restrict(remove) html elements from the dom. As indicated in the parentheses I use HTML::Restrict to remove all html elements entirely. Leaving only the true text contents behind
   (c) File::Basename I want only the basename of the file xyz not xyz.html
   (d) Cwd I need to know man where are we
2. Get where we are currently and set variables to the expected directory structure
3. Create a new HTML::Restrict object with args setting extra tags to nuke and trim resulting naked dom
4. Strip the dom and create processed file

```
#!/bin/bash


#find where we are
wd=$(pwd)

#this is where we store out hash -> uri pairs
file="$wd/ulrToHash.csv"


#check for some house cleaning
if [ -f $file ]
then
    echo "File $file exists. Deleting and remaking"
    rm $file
fi


htmlDir="$wd/html"
if [ -d $htmlDir ]
then
    echo "Directory $htmlDir exists. Deleting and remaking"
    rm -rf $htmlDir
fi

processedDir="$wd/processed"
if [ -d $processedDir ]
then
    echo "Directory $processedDir exists. Deleting and remaking. We
     will use it later ;)"
    rm -rf $processedDir
fi

# make everyting
touch $file
mkdir "html"
mkdir "processed"
echo "Parsing thousand.dat"

#for each uri in the thousand
for uri in $(cat thousand.dat)
do
    # create the md5 hash of the url
    # use awk to remove the trailing -
    hash=$(echo $(md5sum <<< $uri) | awk '{print $1}')
    hashtml=$hash".html"
    $(echo "$hash,$uri" >> $file)
    $(curl -A "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:44.0)
     Gecko/20100101 Firefox/44.01" $uri > "$htmlDir/$hashtml")
    echo $uri | awk -F/ '{print $3}'
    echo $hashtml
    echo $hash
done

#clean up after ourselves
for f in $(find $htmlDir/*.html -size 0c)
do
```

```
    echo "cleaning $f because it failed to download"
    rm $f
done
```

Listing 1: Download HTML shell script

```
use strict;
use warnings;

# read an entire file
use File::Slurp;
# remove html
use HTML::Restrict;
# get base filename
use File::Basename;
#get current working directory
use Cwd;

#get directory paths
my $htmlDir = cwd();
my $pdir="$htmlDir/processed";
$htmlDir = "$htmlDir/html";

# set up html remove
# strip all enclosed tags etc and trim
my $hr = HTML::Restrict->new(
strip_enclosed_content => [ 'script', 'style', 'head' ],
trim => 1
);


# open the directory contianing the html files
opendir(DIR, $htmlDir) or die $!;
# loop through the files
while(my $file = readdir(DIR)){
    # I want only files not . or ..
    next unless (-f "$htmlDir/$file");
    print "$file\n";
    # get text of file and remoce html
    my $text = read_file( "$htmlDir/$file" );
    my $cleaned = $hr->process( $text );
    # attempt to remove none word or space characters i.e html
    quote characters
    $cleaned =~ s/[^\w\s]+//g;
    my $base = basename($file,".html");
#    print $file;
#    print "$cleaned\n";
    my $pfile="$pdir/$base.processed";
#    print "$pfile\n";
    open(my $out, ">>",$pfile) or die $!;
    print $out $cleaned;
    close($out);
}
close(DIR);
```

Listing 2: Perl script to remove all html

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 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!

Answer

The term I choose to search for was *Bernie* as I mined Twitter for `Uri's` during a democratic debate. I did so by executing the following command in the processed directory created by listing 1:

```
$ grep -wic "Bernie" *.processed | sed -r "s/([a-z0-9.]+)(:)
    ([0-9]+)/\1 \3/g" | awk '{print $2" "$1}' | sort -rn | head -n
    10 | awk '{print $2}' >> tenForTFID.txt
$ mv tenForTFID.txt ../
```

Before I begin explaining this wonderful little shell snippet, the snippet was derived from reading "Linux Command Line and Shell Scripting Bible" [1] and consulting the man pages.

1. grep: to search for Bernie in the files
   (a) w: select only lines containing matches that form whole words
   (b) i: ignore case
   (c) c: print only a count of matching lines
2. sed: replace string in a stream
   (a) r: use extended regex
   (b) expression: match md5sum separator count and replace separator with space
3. awk: swap position of output
4. sort: sort output numerically and reverse it(descending order)
5. head: take top n
6. awk: print only the md5sum
7. append to file

I executed the command with out appending the output to a file and here is the produced output:

```
d632bfc8f91eae4cf2c82eb0fd91a7fd.processed
9e1da9cae84ac0736037f14dd0970dd0.processed
404f907cde379d931becd4a90beb842a.processed
60746a6ce366dc48267249969f1cc353.processed
29e1a6b7baf3b7072cfb3c2f36262f4d.processed
e9db1fa8fe181f0b4ee4f5ae446e1a27.processed
d0bea5f71f177067314bee1bc60b2ccb.processed
3d9f071482d0909d263b37e6dc9d6b35.processed
f58549dc1ceb8afb01a6fee5daa03e48.processed
e79dcc12c17c9ac0dc5a45ac9e0f80df.processed
```

These files are only the top ten files that contain the term "Bernie" which are saved for later. Now to figuring out how to calculate TF, IDF, and $TDIDF$ for these files.

We know that TF is calculated per document as:

$$TF(Bernie, doc) = \frac{termCount(Bernie, doc)}{wordCount(doc)}$$

For the IDF calculation we must first do some searching on the internet to find out how many pages Google has and how many results are returned to use for the term *Bernie*. Google has roughly 49 billion web pages indexed according to *worldwidewebsize* [2] and a search for *Bernie* gives us $105,000,000$ results.

Using those numbers we can do some plug and chug to determine our IDF value with consideration to the effect of significant digits for logs as described in " Significant Figure Rules for Logarithm" [3].

$$IDF = \log_2\left(\frac{49,000,000,000}{105,000,000}\right) = \log_2(466.6666) = 8.866248$$

But lets just use 8.8662 as empirically you cant round up from 3 ;).

Using the IDF value we can now state the formula for the $TFIDF$ for each of our ten documents as:

$$TFIDF(Bernie, doc) = TF(Bernie, doc) * 8.8662$$

Now that we have our calculations down how about generating the numbers. Once again I felt like taking the easy mode and created a shell script to do all the work for me as seen in listing 3. Once again I direct you to view the comments in the file for further details into how it operates.

But in brief the scrip operates as such:

1. For each of the hashed uris in the *tenForTFID.txt* file
2. Get the total word count for the file
3. Get the term count again for *Bernie*
4. Calculate the TF, and TFIDF
5. Append the TF IDF TFIDF and URI to the file *tfidftable.csv*

The results dumped to the file *tfidftable.csv* can be found in table 1.

```
#!/bin/bash

#where are we
cwd=$(pwd)

#our processed dir should be near
cwd="$cwd/processed/"
echo $cwd

#output file
file="tfidftable.csv"

if [ -f $file ]
then
   echo "File $file exists. Deleting and remaking"
   rm $file
fi

touch $file

#output first line of csv file
$(echo "tfidf,tf,idf,uri" >> $file)

#for each choosen hash
for f in $(cat "tenForTFID.txt")
do
 #file location of choosen hash
 p="$cwd$f"
 #get the word count for it. Only want the count
 wdCount=$(wc -w $p| awk '{print $1}')
 # redo the termCount
 termCount=$(grep -c -i "Bernie" $p)

#extract only the hash portion of the file (hash).html
 hash=$(echo $f | sed -r "s/([a-z0-9]+)(.)([a-z]+)/\1/g")

# get the uri associated with it from the csv file in format hash.
    processed,uri
# the capture group grabs the uri in its entirity
 uri=$(grep $hash "ulrToHash.csv" | sed -r "s/[a-z0-9]+,(.+)/\1/g")

# do the tf, idf, tfidf calculation
# use scale 5 for percision and send the calulation to An arbitrary
     precision calculator language
 tf=$(echo "scale=5; $termCount / $wdCount" | bc)
 idf=8.8662
 tfidf=$(echo "scale=5; $tf * $idf" | bc)
 echo "tf=$tf idf=$idf tfidf=$tfidf uri=$uri"
# append the tf, idf, tfidf and uri to the final file
 $(echo $tf,$idf,$tfidf,$uri >> $file)
done
```

Listing 3: Find the count and TFID

TFIDF	IDF	TF	URI
0.00153	8.8061	0.01347	http://www.salon.com/2015/02/01/football_an_american_tragedy/
0.00402	8.8061	0.03540	http://theinsideleak.com/hillary-clinton-at-bernie-sanders-you-got-something-to-say-say-i
0.00126	8.8061	0.01109	http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agenda-not-at-debate
0.00314	8.8061	0.02765	http://www.salon.com/2016/01/12/emails_expose_close_ties_between_hillary_clinton_and_accused_war_criminal_henry_kissinger/
0.01282	8.8061	0.11289	http://www.weaselzippers.us/255295-debbie-wasserman-schultzs-completely-un-self-aware-tweet/
0.00358	8.8061	0.03152	http://Berniecare.org
0.00081	8.8061	0.00713	http://www.nybooks.com/daily/2016/01/30/clinton-system-donor-machine-2016-election/
0.00285	8.8061	0.02509	http://www.motherjones.com/politics/2016/02/record-number-exonerations-2015
0.00191	8.8061	0.01681	https://newrepublic.com/article/129047/bernies-army-running-congress
0.00074	8.8061	0.00651	http://therealnews.com/t2/index.php?option=com%content&task=view&id=31&Itemid=74&jumival=15580%

Table 1: 10 Hits for the term *Bernie*, With calculated TFIDF, IDF, TF, URI

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.
```

Still using the same 10 URIs gotten by Question 2, I first attempted to get a page rank for them using the full URI. I used both SEO Central and PR Checker but not CheckPageRank as my browsers would not load the anti-robot checker. Turns out using the full link does not return any scores with PR Checker listing some of the possible reasons:

1. the web page is new, and it is not indexed by Google yet,
2. the web page is indexed by Google, but it is not ranked yet,
3. the web page was indexed by Google long ago, but it is recognised as a supplemental (Supplemental Results) page,
4. the web page or the whole website is banned by Google.

This was surprising to me especially since I had link `http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agenda-not-at-debate` from October 14,2015. So I resulted to using only the top level domain name ie www.theguardian.com for the page rank as shown in table 2.

It must also be noted if the ranking returned to me by either SEO or PR was undef or n/a I substituted those values for 0/0. Since both SEO and PR gave me the same ranking for the URIs as shown in table 2 it leads me to believe these are correct. Doing the hard algebra of fractions the page rank score for the top level domains of the URIs is seen in table 3.

Table 4 shows the PageRank and TFIDF scores for the URIs. When looking at PageRank and TFIDF side by side, you can see just how different the two values are. The two highest URIs with regards to PageRank are in the middle to low end of the TFIDF ranges with PR-0.8 and TFIDFs of 0.00153 and 0.00314. What is more interesting is that they come from the same domain `http://www.salon.com` which would lead me to believe that PageRank is the better metric.

I say that because the highest TFIDF value of 0.01282 comes from the site `www.weaselzippers.us` which has a PR score of 0.5 the third lowest value. This shows that even though the term *Bernie* has a better TFIDF out of the 10 people do not like that site. Taking this a step further, on visiting the site and seeing it's banner with the tag line "Weasel Zippers Scouring the Bowels of the Internet", it is even clearer to see that the website is not your "main stream" news organization. By this I firmly declare PageRank to be the best metric as its "user" curated ranking system gives us better search results in mass.

SEO Central	**PR Checker**	**URI**
8/10	8/10	http://www.salon.com/2015/02/01/football_an_american_tragedy/
0/0	0/0	http://theinsideleak.com/hillary-clinton-at-bernie-sanders-you-got-something-to-say-say-it/
7/10	7/10	http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agenda-not-at-debate
8/10	8/10	http://www.salon.com/2016/01/12/emails_expose_close_ties_between_hillary_clinton_and_accused_war_criminal_henry_kissinger/
5/10	5/10	www.weaselzippers.us/255295-debbie-wasserman-schultzs-completely-un-self-aware-tweet/
0/0	0/0	http://Berniecare.org/
7/10	7/10	http://www.nybooks.com/daily/2016/01/30/clinton-system-donor-machine-2016-election/
7/10	7/10	http://www.motherjones.com/politics/2016/02/record-number-exonerations-2015
7/10	7/10	https://newrepublic.com/article/129047/bernies-army-running-congress
6/10	6/10	http://therealnews.com/t2/index.php?option=com%25content&task=view&id=31&Itemid=74&jumival=15580%25

Table 2: PageRank of URIs(only top level domain) containing the word *Bernie* from SEO Central, and PR Checker

PageRank	**URI**
0.8	http://www.salon.com/2015/02/01/football_an_american_tragedy/
0.8	http://www.salon.com/2016/01/12/emails_expose_close_ties_between_hillary_clinton_and_accused_war_criminal_henry_kissinger/
0.7	http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agenda-not-at-debate
0.7	http://www.nybooks.com/daily/2016/01/30/clinton-system-donor-machine-2016-election/
0.7	http://www.motherjones.com/politics/2016/02/record-number-exonerations-2015
0.7	https://newrepublic.com/article/129047/bernies-army-running-congress
0.6	http://therealnews.com/t2/index.php?option=com%25content&task=view&id=31&Itemid=74&jumival=15580%25
0.5	www.weaselzippers.us/255295-debbie-wasserman-schultzs-completely-un-self-aware-tweet/
0.0	http://theinsideleak.com/hillary-clinton-at-bernie-sanders-you-got-something-to-say-say-it/
0.0	http://Berniecare.org/

Table 3: PageRank of URIs(only top level domain) containing the word *Bernie* normalized no a scale [0 1.0] sorted largest to smallest by page rank

PageRank	TF*IDF	URI
0.8	0.00153	http://www.salon.com/2015/02/01/football_an_american_tragedy/
0.8	0.00314	http://www.salon.com/2016/01/12/emails_expose_close_ties_between_hillary_clinton_and_accused_war_criminal_henry_kissinger/
0.7	0.00126	http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agenda-not-at-debate
0.7	0.00081	http://www.nybooks.com/daily/2016/01/30/clinton-system-donor-machine-2016-election/
0.7	0.00285	http://www.motherjones.com/politics/2016/02/record-number-exonerations-2015
0.7	0.00191	https://newrepublic.com/article/129047/bernies-army-running-congress
0.6	0.00074	http://therealnews.com/t2/index.php?option=com%25content&task=view&id=31&Itemid=74&jumival=15580%25
0.5	0.01282	www.weaselzippers.us/255295-debbie-wasserman-schultzs-completely-un-self-aware-tweet
0.0	0.00402	http://theinsideleak.com/hillary-clinton-at-bernie-sanders-you-got-something-to-say-say-
0.0	0.00358	http://Berniecare.org/

Table 4: PageRank and TFIDF of URIs(only top level domain) containing the word *Bernie* normalized no a scale [0 1.0] sorted largest to smallest by page rank

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

Good Lord where to start on this on. First let me display to you some search results for Kendall's tau b figure 1. From that search I found my references.

Ok lets define the equation required to solve this problem as defined for us at wikipedia [6].

The Kendall Tau-b coefficient is defined as:

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

where:

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

Well what does all that mean? Like a good grad student I took to the internet for an explanation searching once again my favourite search engine

DuckDuckGo 1. The search turned up a great articel explaining Kendall's tau [**?**].

A concordant pair is when the rank of the second variable is greater than the rank of the first and for discordant pair is when the rank is equal to or less than the rank of the first variable[**?**]. Ok we have clue about whats going on. So n_c and n_d are total number of those pairs in our data set. Groovy so it follows that t_i and u_j are ties in the ranking that have the same rules as do n_c and n_d in regards to variable position.

What is not so groovy is the explicit warning given to *Calculating Kendall's Tau manually can be very tedious without a computer and is rarely done without a computer* . Oh boy what have I gotten myself into but there is hope when I read this line *The process to calculate Kendall's tau is quite simple using the R Environment*[**?**].

Back to the interwebs!!! Look what do we have here?? An R tutorial explicitly explaining to us how to go about calculating τ_B [5]! There is also another one explaining Kendall Rank Coefficient[4].

So using the mantra of this class *Do not do it yourself build the shoulders of greater men* I fly to R for an implementation.

I first create a file with this content:

```
PageRank, TFIDF, URI
0.8,0.00153,http://www.salon.com/2015/02/01/football_an_american_tragedy/}
0.8,0.00314,http://www.salon.com/2016/01/12/emails_expose_close_ties_between_hillary_clinton
0.7,0.00126,http://www.theguardian.com/commentisfree/2015/oct/14/womens-issues-clinton-agend
0.7,0.00081,http://www.nybooks.com/daily/2016/01/30/clinton-system-donor-machine-2016-electi
0.7,0.00285,http://www.motherjones.com/politics/2016/02/record-number-exonerations-2015
0.7,0.00191,https://newrepublic.com/article/129047/bernies-army-running-congress
0.6,0.00074,http://therealnews.com/t2/index.php?option=com%25content&task=view&id=31&Itemid=
0.5,0.01282,www.weaselzippers.us/255295-debbie-wasserman-schultzs-completely-un-self-aware-t
0.0,0.00402,http://theinsideleak.com/hillary-clinton-at-bernie-sanders-you-got-something-to-
0.0,0.00358,http://Berniecare.org/
```

And now the R script found in listing 4. This code is nicely and graciously guided by the R τ_b tutorial [5].

```
1  library(Kendall)
2  setwd(getwd())
3
4  d <- read.csv("prtfidf.csv")
5
6  # use the library Kendall to see what it says
7  k<- Kendall(d$PageRank,d$TFIDF)
8  print(k)
9
10 print(summary(k))
11
12 corr<-cor(d$PageRank,d$TFIDF  ,method="kendall", use="pairwise")
13 print(corr)
```

```
14
15 tfidf <- as.numeric(factor(d$TFIDF))
16 pagerank <- as.numeric(factor(d$PageRank))
17
18 m <- cbind(pagerank, tfidf)
19
20 print(cor(m, method="kendall", use="pairwise"))
21
22 print(cor.test(pagerank, tfidf, method="kendall"))
```

Listing 4: Calculate TauB

Now for the output:

```
> library(Kendall)
> setwd(getwd())
>
> d <- read.csv("prtfidf.csv")
>
> # use the library Kendall to see what it says
> k<- Kendall(d$PageRank,d$TFIDF)
> print(k)
tau = -0.221, 2-sided pvalue =0.45435
>
> print(summary(k))
Score =  -9 , Var(Score) = 114.3333
denominator =  40.80441
tau = -0.221, 2-sided pvalue =0.45435
NULL
>
> corr<-cor(d$PageRank,d$TFIDF ,method="kendall", use="pairwise")
> print(corr)
[1] -0.2205644
>
> tfidf <- as.numeric(factor(d$TFIDF))
> pagerank <- as.numeric(factor(d$PageRank))
>
> m <- cbind(pagerank, tfidf)
>
> print(cor(m, method="kendall", use="pairwise"))
           pagerank      tfidf
pagerank  1.0000000 -0.2205644
tfidf    -0.2205644  1.0000000
>
> print(cor.test(pagerank, tfidf, method="kendall"))

Kendall's rank correlation tau

data:  pagerank and tfidf
```

```
z = -0.8417, p-value = 0.4
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
-0.2205644

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

What does this all mean to us?

It means that the null hypothesis we do not reject the null hypothesis that variables are uncorrelated at 0.05 significance level [5].

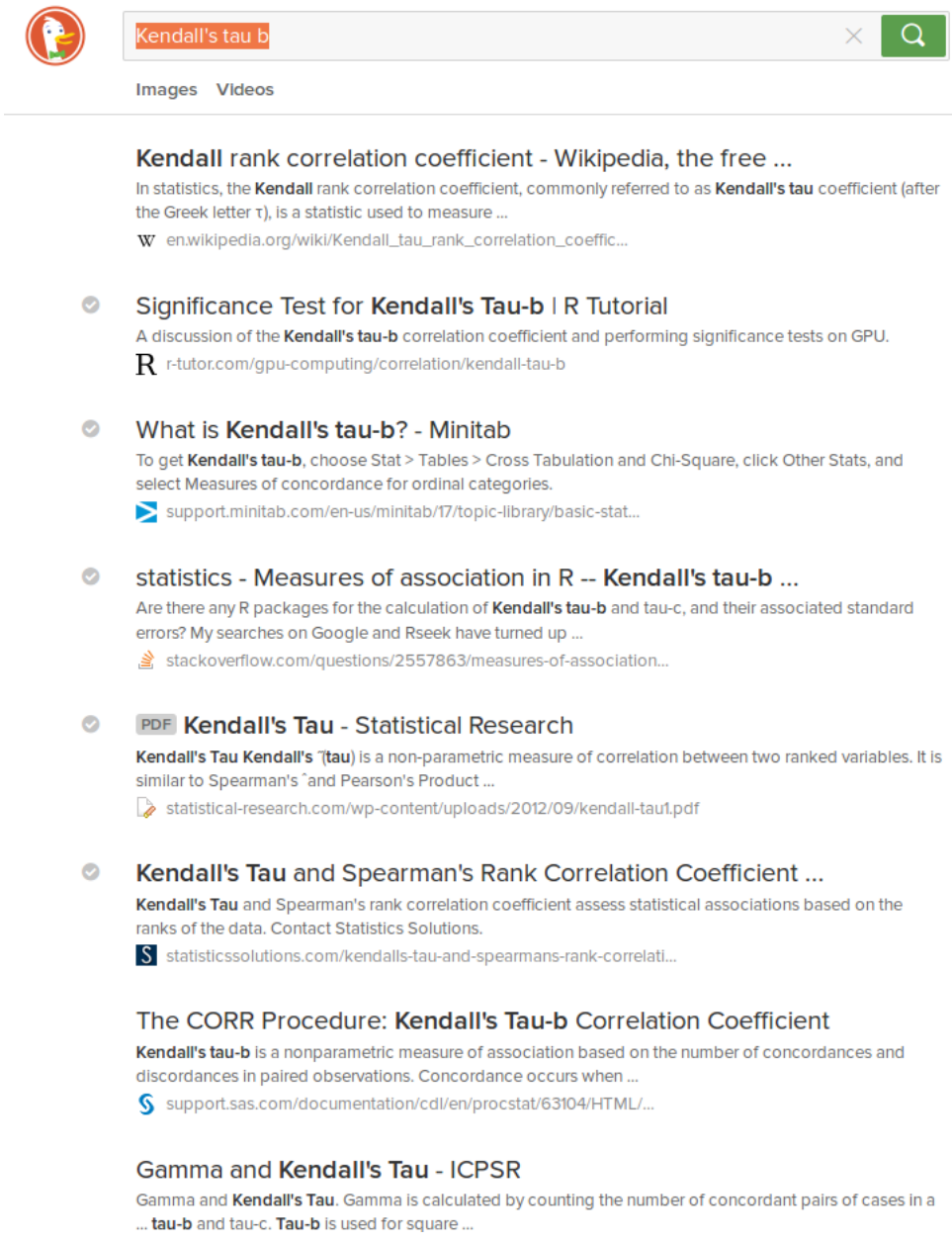

Figure 1: Kendall's tau b search

References

[1] Blum, R., and Bresnahan, C. *Linux Command Line and Shell Scripting Bible, 3rd Edition*, 3 ed. Willey, Jan. 2015.

[2] de Kunder, M. World wide web size: Daily estimated size of the world wide web. `http://www.worldwidewebsize.com`, Feb. 2016.

[3] Fossum. Significant figure rules for logarithms. `http://www.laney.edu/wp/cheli-fossum/files/2011/01/Significant-Figure-Rules-for-los.pdf`, Jan. 2011.

[4] r tutor. Kendall rank coefficient. `http://www.r-tutor.com/gpu-computing/correlation/kendall-rank-coefficient`.

[5] r tutor. Significance test for kendall's tau-b. `http://www.r-tutor.com/gpu-computing/correlation/kendall-tau-b`.

[6] Wikipedia. Kendall rank correlation coefficient — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-February-2016].