# Web Science cs532-s16

## Assignment 3 Report

Dr. Michael L. Nelson

By: Huan Huang
02/18/2016

# 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).

## Answer

This problem is pretty straight forward. For each of the 1000 links from last assignment, I used curl to get the web page’s raw HTML and save them each separately. To get pass the robot blocks on some websites, I used -A “Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:44.0)Gecko/20100101 Firefox/44.0” to disguised my program as a browser. I did try to hash the links to use them as file names, but for some reason, md5sum does not work for over 100 links in my list. Therefore, I just name the files with numbers from 1 to 1000. The file names and their links are saved in a file called “fileNumWithLink.txt” which will be used in problem 2.

```
import os, sys

readfile = open('requiredurls.txt', 'r')
savefile = open('fileNumWithLink.txt', 'a')
count = 0

for line in readfile:
  count = count + 1
  #hashmd5sum = 'echo -n ' + line.strip() + ' | md5sum'
  #hashresult = os.popen(hashmd5sum).read()
  filenames = str(count) + '.txt"'
  command = 'curl -L -A "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:44.0)
    Gecko/20100101 Firefox/44.0" "' + line.strip() + '" > "'+ filenames
  os.popen(command).read()
  numberlink = "{:<10}{}".format(filenames[:-1], line)
  savefile.write(numberlink)
  #print command
  #print hashresult
```

```
 988 988.txt   http://www.forbes.com/sites/anthonydimoro/2016/02/07/2016-nfl-all-forbes-sports-money-team/#3f7ae3572
 989 989.txt   http://www.nbarevolution.com/news/bulls-jimmy-butler-salta-lall-star-game-per-infortunio/
 990 990.txt   http://basketballfanzone.org/2016/02/miami-heat-vs-charlotte-hornets-game-recap-whitesides-triple-dou
 991 991.txt   https://apps.facebook.com/doubleucasino/?adcode=56ba3e8cd8a2c
 992 992.txt   http://www.springhillhomepage.com/volunteers-recreate-alliance-volleyball-courts-in-ag-center-cms-740
 993 993.txt   http://www.amazon.com/Earnhardt-Nation-Full-Throttle-NASCARs-Family/dp/0062367714
 994 994.txt   https://twitter.com/fecrvg1/status/697085513883389952
 995 995.txt   https://twitter.com/tizzzzzzy/status/697139262018424832
 996 996.txt   http://www.azcentral.com/story/news/local/inspire/2016/02/09/special-olympics-sports-program-high-sch
 997 997.txt   https://twitter.com/dosminutos/status/697139979387072513
 998 998.txt   https://www.youtube.com/watch?v=cif7Yi7bbDM&feature=youtu.be&a
 999 999.txt   https://twitter.com/WhitworthGerald/status/696464679586107392
1000 1000.txt  http://www.nba.com/warriors/tivo-playlist-volume-vi
1001
```

Figure 1: Sample of part of the fileNumWithLink

Next step, for every file of the 1000 raw HTML files, I used lynx to strip away the HTML markups and saved the data into a new file. All of the processed files are named exactly the same from the source files.

```
import os, sys

#readfile = open('requiredurls.txt', 'r')
directory = '/mnt/hgfs/SchoolWork/WebScience/Assignments/HW3/Problem1/RawHTMLs
    /rawHTMLs/'
count = 0

for filename in os.listdir(directory):
  count = count + 1
  #hashmd5sum = 'echo -n ' + line.strip() + ' | md5sum'
  #hashresult = os.popen(hashmd5sum).read()
  command = 'lynx -dump -force_html "' + directory + filename + '" > "'+
    filename + '"'
  os.popen(command).read()
  #print command
```

```
   #[1]YouTube Video Search [2]alternate [3]alternate [4]alternate
   [5]alternate [6]Der Eisendrache Game play, Bows upgrades, Goal Round 30
   [7]Der Eisendrache Game play, Bows upgrades, Goal Round 30

   (BUTTON) Skip navigation
   (BUTTON)
   [8]Upload
   (BUTTON) Sign in
   (BUTTON) Search
   ____________________

   Loading...
   (BUTTON) Close
   (BUTTON) Yeah, keep it (BUTTON) Undo (BUTTON) Close

This video is unavailable.
```

Figure 2: Sample image of a HTML processed file

# 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!

## Answer

For this problem, I wrote a shell script to handle every thing. For every processed file, use wc -w to get the total word count of the file and assign the value into *totalwords*. Use grep -c to find the files contain the word "gamer" and count its occurrence in each matching file, then assign the value into *wordmatch*. To get TF, just use *wordmatch* divided by *totalwords*. To get IDF, I searched for the word "gamer" in Google and received 150000000 matches. Google's estimation of world wide web is roughly 47.1 billion, therefore, IDF is log(47100000000/150000000)/log(2). Lastly, TFIDF is TF multiplied by IDF. I set awk to show 6 digits before and after the dot.

```
#!/bin/bash

#google match of word gamer: 150,000,000
#google's estimation of world wide web: 47.1 billion

IDF=`awk "BEGIN {printf \"%9.2f\n\", log(47100000000/150000000)/log(2)}"`

rm data1.txt
rm data2.txt
count=0
echo 'MW TW          LINK' > data1.txt
echo 'TFIDF    TF       IDF       LINK' > data2.txt

for file in `ls /home/hhuang/Documents/CS532/WebScienceAssig/HW3/Problem1/
    ProcessedHTMLs/processedHTMLs`;
do
  totalwords=`wc -w "/home/hhuang/Documents/CS532/WebScienceAssig/HW3/Problem1
    /ProcessedHTMLs/processedHTMLs/"$file | cut -d' ' -f1`
  filewords=`grep -c "gamer" "/home/hhuang/Documents/CS532/WebScienceAssig/HW3
    /Problem1/ProcessedHTMLs/processedHTMLs/"$file`
  link=`grep -w $file "/home/hhuang/Documents/CS532/WebScienceAssig/HW3/
    Problem2/fileNumWithLink.txt" | awk '{ print $2}'`

  if [ $filewords -ne 0 ] && [ $count -lt 20 ];
    then
    count=$[$count+1]
    TF=`awk "BEGIN {printf \"%6.6f\n\", ($filewords/$totalwords)}"`
    TFIDF=`awk "BEGIN {printf \"%6.6f\n\", ($TF*$IDF)}"`
    echo $filewords $totalwords $link >> data1.txt
    echo $TFIDF $TF $IDF $link >> data2.txt

  fi

done
```

```
TFIDF    TF       IDF       LINK
0.003291 0.000397 8.29 http://store.steampowered.com/app/434500
0.004311 0.000520 8.29 https://twitter.com/supa_fresh_sikh/status/697138682420015104
0.008588 0.001036 8.29 https://www.youtube.com/watch?v=hWaB0vKHIuU&feature=youtu.be&aBlack
0.006624 0.000799 8.29 https://www.youtube.com/watch?v=l-KkCM6onMo&list=PLTYUE9O6WCrgEsQvkKwBWnbPAmZXuOOVV&index=1
0.003316 0.000400 8.29 http://gjjgames.blogspot.ca/2016/01/tabletop-game-giveaways-and-contests.html
0.005231 0.000631 8.29 http://www.polygon.com/2016/2/9/10949002/amazon-lumberyard-free-games-engine?utm_campaign=polygo
0.025939 0.003129 8.29 http://www.unravelgame.com/
0.002943 0.000355 8.29 http://www.polygon.com/2016/2/9/10951990/seafall-preview-pandemic-legacy-rob-daviau?utm_campaign=
0.007643 0.000922 8.29 http://www.themisfitsnetwork.com/2016/02/coldwood-interactive-threaded-the-damn-needle-with-thei
0.005480 0.000661 8.29 http://www.cnet.com/news/amazon-announces-its-own-game-engine-with-built-in-twitch-support/#ftag=
0.033135 0.003997 8.29 http://jp.automaton.am/articles/newsjp/fashion-media-editors-review-video-game-characters-outfit
0.002678 0.000323 8.29 http://ottawacitizen.com/sports/hockey/nhl/senatorsextra/ottawa-senators-get-phaneuf-in-a-nine-p
0.012568 0.001516 8.29 http://www.mralanc.com/category/video-game-giveaways/
0.016870 0.002035 8.29 https://www.youtube.com/watch?v=Z6BYzG3SS58&feature=youtu.be&a
0.002943 0.000355 8.29 http://www.polygon.com/2016/2/9/10951990/seafall-preview-pandemic-legacy-rob-daviau
0.004651 0.000561 8.29 http://indianexpress.com/article/entertainment/bollywood/yash-raj-films-to-back-game-on-shah-rukh
0.422011 0.050906 8.29 http://www.4gamer.net/games/332/G033200/20160209007/
0.006383 0.000770 8.29 http://www.businessinsider.com/microsoft-puts-red-dead-redemption-on-xbox-one-2016-2
0.018238 0.002200 8.29 http://musichostnetwork.com/southwest-mook-ft-helluva-deep-in-the-game/
0.016447 0.001984 8.29 http://ps3ch.com/contents/game/?gameID=615
```

Figure 3: TFIDF, TF, and IDF for 20 links that match the word "gamer"

| TFIDF | TF | IDF | URIs |
|---|---|---|---|
| 0.003291 | 0.000397 | 8.29 | http://store.steampowered.com/app/434500 |
| 0.004311 | 0.000520 | 8.29 | https://twitter.com/supa_fresh_sikh/status/697138682420015104 |
| 0.008588 | 0.001036 | 8.29 | https://www.youtube.com/watch?v=hWaB0vKHIuU&feature=youtu.be&aBlack |
| 0.003316 | 0.000400 | 8.29 | http://gjjgames.blogspot.ca/2016/01/tabletop-game-giveaways-and-contests.html |
| 0.005231 | 0.000631 | 8.29 | http://www.polygon.com/2016/2/9/10949002/amazon-lumberyard-free-games-engine?utm_campaign=polygon&utm_content=chorus&utm_medium=social&utm_source=twitter |
| 0.025939 | 0.003129 | 8.29 | http://www.unravelgame.com/ |
| 0.005480 | 0.000661 | 8.29 | http://www.cnet.com/news/amazon-announces-its-own-game-engine-with-built-in-twitch-support/#ftag=CAD590a51e |
| 0.002678 | 0.000323 | 8.29 | http://ottawacitizen.com/sports/hockey/nhl/senatorsextra/ottawa-senators-get-phaneuf-in-a-nine-player-deal |
| 0.004651 | 0.000561 | 8.29 | http://indianexpress.com/article/entertainment/bollywood/yash-raj-films-to-back-game-on-shah-rukh-khans-fan/ |
| 0.422011 | 0.050906 | 8.29 | http://www.4gamer.net/games/332/G033200/20160209007/ |

I did use the shell script to pick 10 URIs for me originally, but I encountered a problem with the list in problem 3. Therefore, I changed the script to give me 20 URIs and hand pick 10 URIs from the list.

# 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.

## Answer

To solve this problem, I used the site "http://www.seocentro.com/tools/search-engines/page rank.html", since it only requires you to fill the captcha once, It made the process a bit faster. However, I encountered the issue of every single one of my URIs have no rank in the page rank check. Therefore, I proceeded by using the hosting site of the URIs. But, because I was only able to use the hosting sites to get the ranks, it is hard to draw conclusions based on the comparison of this ranking and the original TF table in problem 2. Just looking at this ranking table, the result is what I expected. Web sites like Twitter or YouTube which are very well known and aim to server people of any interest would receive the highest ranks. Web sites that are popular amongst people of specific interest would be ranked somewhere in the middle. Then, the random blog pages or new web pages would receive very low rank or no ranking at all.

| Rank | URIs |
|---|---|
| 1.0 | https://twitter.com |
| 0.9 | https://www.youtube.com |
| 0.8 | http://www.cnet.com |
| 0.7 | http://indianexpress.com |
| 0.7 | http://ottawacitizen.com |
| 0.6 | http://www.4gamer.net |
| 0.6 | http://www.polygon.com |
| 0.6 | http://store.steampowered.com |
| 0.0 | http://gjjgames.blogspot.ca |
| 0.0 | http://www.unravelgame.com |

Table 1: A ranking of the 10 URIs from problem2