Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 3

GeorgeC. Micros UIN: 00757376

Honor Pledge
I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagTFIDF TF IDF URIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed __________________________________
October 2, 2014

George C. Micros

Written Assignment 3

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

October 2, 2014

Contents

Chapter 1
Written Assignment 3

1.1 Question 1

1.1.1 The Question

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:
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/`

1.1.2 The Answer

```bash
#! /bin/bash

rm -rf pages/
mkdir pages/

while read -r line
do
    line=$line
    hash=$(echo $line | md5sum | awk '{print $1}')
    curl -X GET $line > ./pages/$hash

done < ./data/temp
```

Listing 1: Bash script to perform GET on URLs

```bash
#! /bin/bash

rm -rf processd
mkdir processd

while read -r line
do
    line=$line
    hash=$(echo $line | md5sum | awk '{print $1}')
    curl -X GET $line | lynx -stdin -dump -force_html -nolist > ./processd/$hash.proc
done < ./data/temp
```

Listing 2: Bash script to extract text from webpage, ignoring links

```bash
#! /bin/bash

rm LUT
while read -r line
do
    line=$line
    hash=$(echo $line | md5sum | awk '{print $1 }')
    echo $hash, $line >> LUT
```

```
done < ./data/temp
```

Listing 3: Bash script that creates a lookup tables of URLs and their hashes

1.2 Question 2

1.2.1 The Question

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:

TFIDF	TF	IDF	URL
0.150	0.014	10.680	http://foo.com
0.085	0.008	10.680	http://bar.com

Table 1.1: Table 1. 10 Hits for the term “shadow”, ranked by TFIDF.

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!

1.2.2 The Answer

```
#! /bin/bash


cnt=$(grep internet ./processd/* | awk -F: '{ print $1}' | uniq -c | awk '{ print $1}' | wc -l)

term=$(grep internet ./processd/* | awk -F: '{ print $1}' | uniq -c | awk '{ print $1}'> term.txt)

dwc=$(wc -w `grep internet ./processd/* | awk -F: '{print $1}' | uniq -c |awk '{ print$2}'` | awk
    '{print $1}' | head -n -1 > dwc.txt)

urls=$(grep internet ./processd/* |awk -F: '{print $1}' |uniq -c |  awk -F/ '{print $3}' | awk -F.
    '{print $1}' | grep -f - ./LUT | awk -F, '{print $2}'>urls.txt >urls.txt)

paste term.txt dwc.txt urls.txt > 5.txt

size=40000000
idf=$(echo ";l($size/$cnt)/l(2)"| bc -l)
#echo $size $cnt $idf

#printf "TFIDF\t   TF\t\tIDF\t   URL\n\n"
rm -f table
while read -r a b c
do
    res=$(echo "$a/$b" | bc -l)
    num=$(echo "$res*$idf" | bc -l);
    #echo $res $c
    echo $num $res $idf $c >> table
done <5.txt
```

```

head -n 10 table | sort -r -k1 | xargs printf "%.8f, %.8f, %.8f, %s\n"  >tfidf


head -n 10 table | sort -r -k1 | awk '{print $4}' | awk -F/ '{print $3}' >  urlsRes
```

Listing 4: Bash script to retrieve and rank hits for the term "internet"

.

TFIDF	TF	IDF	URLS
0.07252207	0.00393701	18.42060665	`http://www.whitehouse.gov/`
0.07217702	0.00391828	18.42060665	`http://pastebin.com/raw.php?i=n1qTeikM`
0.03936027	0.00213675	18.42060665	`http://radar.oreilly.com/2014/02/oobleck-security.html`
0.03744026	0.00203252	18.42060665	`http://www.slideshare.net/timoreilly/government-for-the-people-by-the-people-in-the-21st-century`
0.02497709	0.00135593	18.42060665	`http://solidcon.com/solid2014/public/schedule/list`
0.01074715	0.00058343	18.42060665	`http://www.economist.com/news/united-states/21614225-bright-foreigners-study-america-shame-they-cant-stay-coming-and-going`
0.00637171	0.00034590	18.42060665	`https://stopthesecrecy.net/`
0.00635632	0.00034507	18.42060665	`http://www.ed.gov/connected`
0.00631708	0.00034294	18.42060665	`http://solidcon.com/solid2014/public/schedule/speakers?cmp=tw-na-confreg-home-sld14\_solid\_twitter\_posts`
0.00184953	0.00010041	18.42060665	`http://violentmetaphors.com/2014/03/25/parents-you-are-being-lied-to/`

Table 1.2: Table 2. 10 Hits for the term "internet", ranked by TFIDF.

1.3 Question 3

1.3.1 The Question

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:

.

PR	URL
0.9	http://foo.com
0.5	http://bar.com

Table 1.3: Table 2. 10 hits for the term "shadow", ranked by PageRank.

Briefly compare and contrast the rankings produced in questions 2 and 3.

1.3.2 The Answer

```bash
#! /bin/bash

rm prRes

while read -r line
do
 a="http://"
    s=$(./page.pl $a$line)

    s=$(echo "scale=1; $s/10" | bc -l)
    echo $s, $line >> prRes
done < "urlsRes"

sort -nr -k1 -t, prRes > PR.txt
```

Listing 5: Bash script to run CarbonDate and store results

```perl
#!/usr/bin/perl

$webpage = $ARGV[0];


use WWW::Google::PageRank;
my $pagerank = WWW::Google::PageRank->new;
my $score = $pagerank->get($webpage);
if($score == "")
{
        $score = 0;
}

```

```
print scalar($score), "\n";
```

Listing 6: Python script to find “Estimated Creation Date” in JSON file

PR	URL
1.0	www.whitehouse.gov
.8	www.slideshare.net
.8	www.ed.gov
.8	www.economist.com
.6	radar.oreilly.com
.6	pastebin.com
.4	violentmetaphors.com
0	stopthesecrecy.net
0	solidcon.com
0	solidcon.com

Table 1.4: Table 3. 10 Hits for the term “internet”, ranked by PageRank.

1.4 Question 4

1.4.1 The Question

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`

1.4.2 The Answer

The RHO value is 0.9189 and the p-val 6.1359e-04

```
clear; clc; close all

#A = importdata('C:\Users\Micros\Dropbox\FALL14\CS495\hw\HW3\1.txt');
#B = importdata('C:\Users\Micros\Dropbox\FALL14\CS495\hw\HW3\2.txt');

A = importdata('1.txt');
B = importdata('2.txt');

[RHO,PVAL] = corr(B,A,'type','Kendall')
```

Listing 7: MATLAB script that computes the RHO and p value of the Kendall Tau correlation

References

1. `http://www.google.com`
2. `http://jakevdp.github.io/blog/2012/10/14/scipy-sparse-graph-module-word-ladders/`
3. `http://curl.haxx.se/docs/httpscripting.html`
4. `http://www.crummy.com/software/BeautifulSoup/bs4/doc/`
5. `http://www.rmi.net/~lutz/`
6. `http://www.cs.cornell.edu/home/kleinber/networks-book/`
7. `http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/`
8. `http://www.cs.odu.edu/~mklein/cs796/lecture/`