Introduction to Web Science/595: Assignment #3

Dr. Nelson

Francis Pruter

Saturday, 4 October, 2014

Contents

Problem 1 **3**

Problem 2 **5**

Problem 3 **10**

Problem 4 **12**

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, 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/
```

SOLUTION This was solved using 2 bash scripts, one to download the HTML and another to remove the HTML using Lynx:

Listing 1: Command: uniqueURL — ./dlURI2

```
#!/bin/bash
# gets mementos for all links
# execute: cat uniqueURI | ./dlURI

counter=1

while read line
do
    echo "$counter"

    curl $line > downloadURI2/$counter
```

Problem 1 continued on next page...

```
#    wget -O downloadURI2/$counter $line

    ((counter++))
done
```

1. **Download HTML version of each URI:** The following bash script downloaded the HTML version of all 1000 URIs.

Listing 2: processURI

```
#!/bin/bash
# gets mementos for all links
# execute: ./lynxURI

counter=1
mkdir processedURI

while [ $counter -le 1000 ]
do
    echo "$counter"

    lynx -dump -force_html downloadURI2/$counter > processedURI/$counter

    ((counter++))
done
```

2. **Remove the HTML from the downloaded websites:** This script goes through each of the downloaded websites and removes the HTML using lynx.

Problem 2

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!

SOLUTION

In order to solve this problem, I used a bash script and a python script:
Below is the queryTerm script used and it asks the user for a query word. This script checks each processed website for a match (a minimum of 10 websites must match). It then calculates the number of matches per website as well as the total number of words in each website. This will be used by the tfidfScore.py.

Additionally, it uses lynx to find the number of match that google will return.

The output (pageRank.dat) is in the github folder

Listing 3: queryTerm Script

```
#!/bin/bash
# Author: Francis Pruter
# This script will ask the used for a query word, then
#  search is of the processed websites for a match and the number
#  of matches.  If less than 10 websites, you will be prompted
#  for another query word.




notfound=true

ARRAY=()
total=0
qword=''


while $notfound; do
    count=0
    echo -n "Enter a query word: "
    read qword

    #looks for the query word in all 1000 files
    for (( i=1; i<=1000; i++ )); do
        resp=$(grep -o $qword getLynxURI/$i | wc -l)
        ((total=resp+total))

        #if there is a match, increase the foundcounter by 1
        if [ $resp -gt 0 ]; then
            ((count++))
            ARRAY+=($i)
        fi
    done
    echo "${ARRAY[@]}"
    echo 'total found: ' $count
    echo 'num occurance: ' $total

    #if there are more than 10 hit, this will exit the loop
    if [ $count -ge 10 ]; then
        notfound=false
    else #else, reset the total and arry
        total=0
        ARRAY=()
    fi
done

#uses lynx to google the number of we results for the query word
numgoogle=$(lynx -dump -force_html 'http://www.google.com/search?q='$qword | grep 'Web
    About.* results')

#removes everything but the number of results
numgoogle=$(echo $numgoogle | sed s/'Web About'// | sed s/'results.*'// | sed s/',//g
```

```
    | sed s/' '//g)


#ensure the output file exists
touch pageRank.dat

#outputs everything to be read and processed by tfidfScore.py
echo -e "Numgoogle:\t"$numgoogle"\nSizeGoogle:\t42000000000" > pageRank.dat
#size of google: http://www.worldwidewebsize.com/

echo -e "File#\t#Words\t#Q" >> pageRank.dat

counter=0
for i in "${ARRAY[@]}"
do
    #gets the number of query words in a website
    numQWords=$(cat getLynxURI/$i | grep -o $qword | wc -l)

    #counts the number of words in the website
    numWords=$(cat getLynxURI/$i | wc -w)

    echo -e  $i "\t" $numWords "\t" $numQWords >> pageRank.dat

    ((counter++))
done
```

Below the tfidfScore.py will computer the TFIDF, TF, and IDF for each website with a match and prints out 10 of the matching URI.

Listing 4: tfidfScore.py

```
# tfidfScore.py
# Author: Francis Pruter
#
# This python script processes the output of the queryTerm bash script
#  and calculated the TFIDF, TF, and IDF for each URI that was found
#

from decimal import *
from operator import itemgetter #used to sort list of lists
import math

PRECISION=3

#This function computes and returns the IDF
# IDF is log2(total docs in corpus / docs with term)
def computeIDF(f):
  numFoundPages = fin.readline()
  numFoundPages = numFoundPages.rstrip('\n').split('\t')[1]

  numGooglePages = fin.readline()
  numGooglePages = numGooglePages.rstrip('\n').split('\t')[1]
```

```
  return math.log( Decimal(numGooglePages)/Decimal(numFoundPages), 2 )

# This function computers and returns the TF
#  TF is occurrence in doc / words in doc
def computeTF(x):
  return float(x[2])/float(x[1])

#used to output the array in tfidfScore.dat
def displayTen(array):
  with open('tfidfScore.dat', 'w+') as fout:
    fout.write( "TFIDF\tTF\tIDF\tURI\n" )
    fout.write( "-----\t--\t---\t---\n" )

    ctr = 0
    for x in array:
      if (ctr < 10):
         fout.write( str(x[0]) + "\t" + str(x[1])  + "\t" + str(x[2]) + "\t" +x[3])
         ctr = ctr+1

#This function returns the URI based on the num requested
def getURI(num):
  with open('uniqueURI', 'r') as fURI:
    x = fURI.readlines()
    num = num -1
    return x[num]



#Main body of the program
with open('pageRank.dat', 'r') as fin:
  tfidfArray = []
  IDF = round(computeIDF(fin),5)



  fin.readline() #read next line of headers

  for x in fin:
    x=x.rstrip('\n').split('\t')
    TF = round(computeTF(x), PRECISION)

    TFIDF = round((TF*IDF), PRECISION)
    tfidfArray.append([ TFIDF, TF, IDF, getURI(int(x[0])) ])


  tfidfArray=sorted(tfidfArray, key= itemgetter(0), reverse=True)

  displayTen(tfidfArray)
```

Listing 5: tfidfScore.dat

```
TFIDF     TF    IDF  URI
-----     --    ---  ---
0.631     0.091     6.93184   http://timehop.com/c/dkp:228577748:4e40d
```

```
0.270     0.039     6.93184   https://photojojo.com/store/awesomeness/power-pot/
0.243     0.035     6.93184   http://hardware.slashdot.org/story/14/08/05/2033230/t-
    mobile-smartphones-outlast-competitors-identical-models?utm_source=slashdot&
    utm_medium=twitter
0.243     0.035     6.93184   http://appleinsider.com/articles/14/02/18/apple-patents-
    sensor-packed-health-monitoring-headphones-with-head-gesture-control
0.118     0.017     6.93184   http://www.technologyreview.com/news/530671/smartphone-
    movements-could-reveal-empty-parking-spots/
0.111     0.016     6.93184   http://blog.peerj.com/post/89751390423/author-interview-
    meadow-microbiome-phones
0.090     0.013     6.93184   http://pressurenet.io/
0.083     0.012     6.93184   http://www.engadget.com/2014/09/15/us-fines-would-have-
    destroyed-yahoo/
0.083     0.012     6.93184   http://www.technologyreview.com/news/527836/thermal-
    camera-turns-many-things-into-interactive-surfaces/
0.076     0.011     6.93184   http://www.electronista.com/articles/14/09/18/missing.
    language.suggests.apple.has.received.patriot.act.request/
```

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.
```

SOLUTION

After trying multiple URI for page ranks, I mostly received N/A. I ended up using the top-level domains for page ranks. Utlizating R, I was able to normalize the output.

```
> google<-c(6,5,6,6,8,6,0,8,8,5)
> normalize <- function(x){(x-min(x))/(max(x)-min(x))}
> normalize(google)
 [1] 0.750 0.625 0.750 0.750 1.000 0.750 0.000 1.000 1.000 0.625
```

```
PageRank URI
-----------

1.000  http://www.technologyreview.com/
1.000  http://www.engadget.com/
1.000  http://www.technologyreview.com/
0.750  http://timehop.com/
0.750  http://hardware.slashdot.org/
0.750  http://appleinsider.com/
```

```
0.750  http://blog.peerj.com/
0.625 https://photojojo.com/
0.625  http://www.electronista.com/
0.000  http://pressurenet.io/


Overall TFIDF and Google PageRank were quite different.  This
is mostly do to the fact TFIDF uses a query word and ranks the site.
Google PageRank ranks the overall site based on links pointing to you.
```

Problem 4

```
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
```

SOLUTION

```
I used R to solve the Tau "b" and "p" value for TFIDF and Page ranks:
Tau "B": -0.1288848
P-value = 0.6831


> googleN
 [1] 0.750 0.625 0.750 0.750 1.000 0.750 0.000 1.000 1.000 0.625
> tfidf<-c(.631,.27,.243,.243,.118,.111,.09,.083,.083,.076)
> library(stats)
> cor.test(tfidf, googleN, method="kendall", alternative="greater")

Kendall's rank correlation tau

data:  tfidf and googleN
z = -0.4765, p-value = 0.6831
alternative hypothesis: true tau is greater than 0
sample estimates:
       tau
-0.1288848

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

References

[1] http://www.worldwidewebsize.com/

[2] http://icheckrank.com/multiple-pagerank-checker.php

[3] http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient#Tau-b

[4] http://stackoverflow.com/questions/5665599/range-standardization-0-to-1-in-r