Assignment 3

CS 532: Introduction to Web Science

Spring 2018

Chandrasekhar Reddy Muthyala

Finished on February 21, 2018

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.

A "better" approach is to use BeautifulSoup, see:

http://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text

for some hints on how to start.  Note that none of these methods
are going to be perfect.

Keep both files for each URI (i.e., raw HTML and processed).
Upload both sets of files to your github account.
```

Answer

The problem is to extract the text from 1000 URL which we got from assignment#2 finalUrls.txt. The libraries used for extraction of text from URLs

- import urllib.request
- import hashlib
- import os, errno
- import csvn
- from boilerpipe.extract import Extractor

Initially I tried extracting a text from URLs using Beautiful soup library, after gone through many forum as well as class room discussions, there is an another library called Boilerpipe which will outperform than Beautiful soup in extracting main content of webpage. In this program I started with opening of finalUrls.txt file and then created a output folder and in which we have rawHtml and processedHtml folders to dump raw html and text of 1000 URLs respectively. All the three directories I am creating using function called createDirectories . To store raw html and processed text of webpage in new file, I hashed the URL using hashlib library to hashedUrl.html and hashedUrl.txt respectively. The function I used md5 for hashing URLs has no decrypt function, so I stored URL and corresponding hash key in hashedUrls.csv file in output directory. By using urllib.request we will get the raw html and to parse it I used boilerpipe. Raw html is stored in rawHtml folder and corresponding processed text will store in processedHtml folder.

Run on the command line

```
1 python checkHtmlContent.py
```

```
1
2 import urllib.request
3 import hashlib
4 import os, errno
5 import csv
6 from boilerpipe.extract import Extractor
7 count = 1
```

```
8
9  # creating directory  for out to store files
10 def createDirectories():
11   if not os.path.exists('output'):
12     try:
13       os.makedirs('output')
14     except OSError as e:
15       if e.errno != errno.EEXIST:
16         raise
17   if not os.path.exists('output/rawHtml'):
18     try:
19       os.makedirs('output/rawHtml')
20     except OSError as e:
21       if e.errno != errno.EEXIST:
22         raise
23   if not os.path.exists('output/processedHtml'):
24     try:
25       os.makedirs('output/processedHtml')
26     except OSError as e:
27       if e.errno != errno.EEXIST:
28         raise
29
30 with open('finalurls.txt') as fp:
31   createDirectories();
32   with open('output/hashedUrls.csv', 'w') as csvfile:
33     fieldnames = ['url', 'key']
34     writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
35     writer.writeheader()
36
37     for line in fp:
38       md5value = hashlib.md5(line.strip().encode('utf-8')).
         hexdigest()
39       writer.writerow({'url': line.strip(), 'key': md5value})
40       print(count,":",md5value)
41       count =count+1
42       try:
43         rawhtml = urllib.request.urlopen(line.strip()).read()
44         with open('output/rawHtml/%s.html' % md5value, 'w+',
           encoding='utf-8') as rawf:
45           print(rawhtml, file=rawf)
46
47         extractor = Extractor(extractor='ArticleExtractor', html
           =rawhtml)
48         htmlText=extractor.getText()
49         with open('output/processedHtml/%s.txt' % md5value, 'w
           +', encoding='utf-8') as processedf:
50           print(htmlText, file=processedf)
51           # print(htmlText)
52       except KeyboardInterrupt:
```

```
53            exit()
54          except:
55            pass
```

Listing 1: Python script for storing raw html and processed 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 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, just explain how you did it.

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

https://en.wikipedia.org/wiki/Significant_figures#Rounding_and_decimal_places

Answer

In this problem I calculated a TF-IDF, TF and IDF for 10 short listed URLs from 1000 URLs.

- TF = (total number of searched word/total number of words in a document)
- IDF = log2(total docs in corpus / docs with term)
- TF-IDF = TF IDF

First and foremost to identify 10 URLs I wrote a python code where if the search word count in a file is greater than 10 I am storing it in a tenUrls.txt. Here my search keyword is football libraries used for getting 10 URLs

- import os
- import re

Run on the command line

```
python searchWordFromFile.py>tenUrls.txt
```

```
# Import the os module, for the os.walk function
import os
import re
# Set the directory you want to start from
rootDir = 'output/processedHtml/'
filesCount = 0
for dirName, subdirList, fileList in os.walk(rootDir):
  # print('Found directory: %s' % dirName)
  for fname in fileList:
    with open(os.path.join(dirName, fname),encoding="utf8") as
    openfile:
      lines = openfile.readlines()
      counter=0
      istaken= False
      for line in lines:
        if not istaken:
          for part in line.split():
            if re.match("football", part):
              # print("line: ",line)
              counter = counter+1
            if counter>10:
```

```
21                  istaken = True
22                  filesCount = filesCount+1
23                  print(fname)
24                  # print(filesCount,": ",fname)
25                  break
26          else:
27            break
```

Listing 2: Python script for storing raw html and processed html

Libraries used for solving problem#2

- import os
- from collections import Counter
- import pandas as pd
- import numpy as np

steps to be followed :

- I am having hashed 10 Urls in tenUrls.txt files, we are having original URL and hashed URL in hashedUrls.csv file, from this file I am reading storing all the 10 URLs original name in hashedUrls data structure called dictionary.
- From tenUrls.txt reading each URL and reading the content of the URL line by line and splitting each line into word, comparing each word with search key word and keep on incrementing if search key word exist and finding total word count of each URL. Finding TF using search key word count and total number of words in a document.
- Calculating IDF=1000/345, 1000 is my total number of URLs and 345 is total number of files my search key word present
- Calculating TF-IDF =TF*IDF

Run on the command line

```
1 Python wordcount.py>problem2output.txt
```

```
import os
from collections import Counter
import pandas as pd
import numpy as np
wordVector = {}
tf = {}
idf = {}
tfidf = {}
hashedUrls = {}

df = pd.read_csv('output/hashedUrls.csv', sep=',',header=None,
skiprows=1)
with open("tenUrls.txt",'r') as file:
  lines = file.readlines()
  for line in lines:
    for index, row in df.iterrows():
      if row[1].strip()+".txt"==line.strip():
        # print(row[0])
        hashedUrls[line.strip()] = row[0]
        break

with open("tenUrls.txt",'r') as file:
  totalWordsInFile=0
  footballCount = 0
  lines = file.readlines()
  for line in lines:
    fileName = line.strip()
    # print(fileName)
    with open(os.path.join("output/processedHtml", fileName),'r
    ',encoding='utf-8') as con:
        wordcount = Counter(con.read().split())
        for item in wordcount.items():
          # print(item[0])
          totalWordsInFile = totalWordsInFile+item[1]
          if(item[0]=='football'):
            footballCount = item[1]
        wordVector={fileName:totalWordsInFile}
        tf[fileName]=(footballCount/totalWordsInFile)
# print(tf)
for key, value in tf.items():
  idf[key] = np.log2(1000/345) #total number of files my search
  key words in 1000 urls
print("TFIDF","\t","TF","\t","IDF","\t", "URL")
for key,value in tf.items():
  tfidf[key] = tf[key]*idf[key]
  print(round(tfidf[key],4),"\t",round(tf[key],4),"\t",round(idf
  [key],4),"\t",hashedUrls[key])
```

Listing 3: Python script for calculating TF-IDF, TF and IDF

TFIDF	TF	IDF	URI
0.0213	0.0138	1.5353	http://www.pressofatlanticcity.com/sports/local/highschool/football/atlantic-city-football-players-sign-to-play-division-ii-football/article_d481acc6-a0d5-5d8a-9900-230622b6e8f8.html?utm_medium=social&utm_source=twitter&utm_campaign=user-share
0.0076	0.0049	1.53532	https://www.azcentral.com/story/sports/ncaaf/2018/02/07/asu-football-herm-edwards-arizona-football-kevin-sumlin-contract/311731002/
0.0051	0.0033	1.5353	http://www.citizen-times.com/story/sports/high-school/hshuddle/2018/02/07/wnc-talent-display-during-national-signing-day/316890002/
0.0028	0.00199	1.5353	https://www.postandcourier.com/sports/citadel-football-recruiting-class-boosted-by-pair-of--star/article_bc681706-0bff-11e8-baca-7f18d5c08a83.html?utm_medium=social&utm_source=twitter&utm_campaign=user-share
0.0019	0.0013	1.5353	https://www.sportbuzzbusiness.fr/liga-sengage-football-feminin-nigeria.html?utm_source=Sociallymap&utm_medium=Sociallymap&utm_campaign=Sociallymap
0.001	0.00086	1.5353	http://www.the42.ie/liam-whelan-munich-air-disaster-busby-babes-60th-anniversary-3785714-Feb2018/?utm_source=shortlink
0.0011	0.00081	1.5353	https://www.postandcourier.com/sports/the-citadel-football-signs-new-players-awaits-at-least-one/article_bc681706-0bff-11e8-baca-7f18d5c08a83.html?utm_medium=social&utm_source=twitter&utm_campaign=user-share
0.0008	0.00034	1.5353	https://elpasoheraldpost.com/videoinfo-dimel-announces-2018-utep-football-signing-class/
0.0007	0.00088	1.5353	http://www.wbur.org/onlyagame/2018/02/02/zac-easter-cte-concussion-football
0.0007	0.00180	1.5353	http://www.nationalreview.com/article/456152/philadelphia-eagles-win-god-loves-them-us-all

Table 1: 10 URIs found containing *football*, with calculations TFIDF, TF and IDF

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://pr.eyedomain.com/
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 are only 10 to do. 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). Also note that these tools typically report on the domain rather than the page, so it's not entirely accurate.

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.

PR	TFIDF	URI
0.6	0.0213	http://www.pressofatlanticcity.com
0.8	0.0076	https://www.azcentral.com
0.7	0.0051	http://www.citizen-times.com
0.7	0.0028	https://www.postandcourier.com
0.5	0.0019	https://www.sportbuzzbusiness.fr
0.6	0.001	http://www.the42.ie
0.7	0.0011	https://www.postandcourier.com
0.5	0.0008	https://elpasoheraldpost.com
0.7	0.0007	http://www.wbur.org
0.7	0.0007	http://www.nationalreview.com

Table 2: Page Rank and TFIDF Comparison of 10 URIs with PR on a 0 to 1 scale

Answer

Using the http://www.checkpagerank.net/ website I checked the page rank of my 10 URIs obtained from question 2. This website only did domain searches on the URIs provided, making the 2 postandcourier.com URIs included the same value for each Page Rank. I also attempted prchecker.info and checkerpagerank.net but was unable to get past their captchas. The end result is shown in Table 2.

When comparing the Page Rank to the TFIDF for each of the URIs, it's apparent that Page Rank is a much higher value for each of the URIs. What was surprising to me was the difference in TFIDF but also the values of the wbur.org and nationalreview.com URIs. Those URIs tend to correlate as the top 3 highest TFIDF values as well as the top Page Rank values. The terms and the TFIDF for majority of the values actually correlates very nicely with one another. However the biggest discrepancy was the Page Rank and TFIDF forwbur.org and nationalreview.com . The Page Rank of those was 0.7 but it also had the lowest TFIDF. Those URI turned out to be a very high indexed news website, but TFIDF wasn't reliable enough to receive great values on those URIs.

References

[1] "Check last known Google PageRank." eyedomain. EyeDomain, n.d. Web. 22 Feb. 2017. http://pr.eyedomain.com/.

[2] "Inverted Index." Roseta Code Inverted index. N.p.,n.d. Web. 22 Feb. 2017. http://rosettacode.org/wiki/Inverted_index#Simple_inverted_index

[3] "Kendall rank correlation coefficient." Wikipedia - Kendall Rank Correlation coefficient. Wikipedia, n.d. Web. 22 Feb. 2017. https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient#Tau-b

[4] Atkins, Grant. "finalURIs.txt - Twitter scraped URIs." cs532-s17 Github Repository. N.p., 09 Feb. 2017. Web. 22 Feb. 2017.https://github.com/grantat/cs532-s17/blob/master/assignments/A3/src/output/finalURIs.txt.

[5] Atkins, Grant. "Inverted Index results." cs532-s17 Github Repository. N.p., 09 Feb. 2017. Web. 22 Feb. 2017.https://github.com/grantat/cs532-s17/blob/master/assignments/A3/src/output/invertedIndex.txt.

[6] Fellows, Ian. "Measures of association in R ? Kendall's tau-b and tau-c." Stackoverflow. Stackoverflow, 10 April 2010. Web. 22 Feb. 2017. http://stackoverflow.com/questions/2557863/measures-of-association-in-r-kendalls-tau-b-and-tau-c.

[7] Kunder, Maurice. "The size of the Dutch World Wide Web" worldwidewebsize. N.p., n.d. Web. 22 Feb. 2017.http://www.worldwidewebsize.com/.

[8] Stenberg, Daniel. "Curl.1 the Man Page." Curl - How To Use. N.p., n.d. Web. 24 Jan. 2017. https://curl.haxx.se/docs/manpage.html.