Web Science: Assignment #3

Alexander Nwala

Apurva Modi

Sunday, March 03, 2019

Contents

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.

Upload both sets of files to your Github account.

SOLUTION :

1. The program requires to run the 'CURL' command over all URIs obtained during the Assignment 1, found in "1000TwitterLinks.txt".

   1. To get the raw HTML content for every URI :

```
  f = open(htmlFile, "w")
subprocess.call(curlCommand, shell=True, stdout=f)
```

   2. To get the text content for every URI :

```
extractor = Extractor(extractor='ArticleExtractor', url=link)
the_file.write(str(extractor.getText()))
```

   Text content of the page will be saved in independent files with names obtained through **md5 hashing**

All the raw HTML and the text contents can be found in the **CodeFile** directory on github submission

Listing 1: Assignmnet3_1.py

```
import subprocess
from boilerpipe.extract import Extractor
from importlib import reload
import sys
import hashlib
import traceback
import os.path
import time


reload(sys)
sys.getdefaultencoding()

linksDict = {}
linksFile = open('finally1000URIs.txt','r')
for link in linksFile:
     if(link == ''):
          pass
     else:
          try:
               link=link.strip()
               curlCommand = 'curl ' + link
               #link = link.encode('utf-8')
               hash_object = hashlib.md5(link.encode())
               print(hash_object.hexdigest() + '.html')
```

```
                htmlFile = os.path.join("sourceHTML_data",
                                hash_object.hexdigest()+ "':html'")
                textFile = os.path.join("sourceTXT_data",
                                thash_object.hexdigest()+ "':txt'")
                #file_to_open = os.path.join(data_folder, "raw_data.txt")

                #htmlFile = hash_object.hexdigest() + ':html'
                #textFile = hash_object.hexdigest() + ':txt'

                extractor = Extractor(extractor='ArticleExtractor', url=link)
                     #print (str(extractor.getText()))

                if (len(str(extractor.getText())) > 0):
                     #open(htmlFile, "w")
                     f = open(htmlFile, "w")
                     raw_html = subprocess.call(curlCommand, shell=True, stdout=f)
                     #htmlFile.write(str(extractor.getHTML()))
                     with open(textFile, 'w') as the_file:
                          the_file.write(str(extractor.getText()))
                          linksDict[textFile] = link
                          print (str(extractor.getText()))
                     #linksDict[html] = link
                else:
                     print("No text in the URi")
           except KeyboardInterrupt:
                exit()
           except:
                pass


with open('dict-hash-URL.txt', 'w') as file:
     for key,value in linksDict.items():
          file.write('%s:%s\n' % (key, value))
```

The above code will save the hashed URIs name as Text File and HTML and the URIs with hashed Text value are stored in to **"dict-hash-URL.txt'"** as a key value pair.

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.

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.

SOLUTION

The solution for this problem is outlined by the following steps:

1. The program requires to run the 'GREP' command over all the text files obtained during Problem 1.
   1. GREP will find the text **"Crypto"** on the processed files

```
  grep = "grep -Ric 'Cryto' sourceTXT_data > grepCMD_Output.txt"
subprocess.call(grep, shell=True)
```

   The output of the above step would return filenames of the processed files whose hit count is more than 0

   2. To calculate the terms and to tabulate for every URI, the code determines TF ,IDF and TFIDF values

```
Term Frequency (TF) =
             Number of occurrences of the word / Total number of words
Inverse Document Frequency (IDF) =
   log2(total URIs in corpus/total processed files with term 'goals')
TF-IDF = TF * IDF
```

Listing 2: Assignmnet3_2.py

```
#from __future__ import division
import subprocess
import math
import traceback

grep = "grep -Ric 'Bitcoin' sourceTXT_data > grepCMD_Output.txt"
subprocess.call(grep, shell=True)
count=0
num_words = 0
matchCount= round(0,4)
corpusCount=round(0,4)
docsWithTerm =round(0,4)
idf=round(0,4)
tf=round(0,4)
tfidf=round(0,4)
totalWordsInFile = round(0,4)
tfDict = {}
```

```
def countWordsInFile(fileName):
    global num_words
    with open(fileName, 'r') as f:
        for line in f:
            words = line.split()
            num_words += len(words)
    return num_words

linksInFile = open('grepCMD_Output.txt','r')
for line in linksInFile:
    line = line.replace('\n', '')

    if("':txt'" in line):
        matchCount = round(int(line[(line.rfind("':txt'")+7):]),4)
        corpusCount = corpusCount + 1
        if(matchCount >= 1):
            print(line)
            line  = line[0:line.rfind("':txt'")]
            print (line)
            totalWordsInFile = countWordsInFile(line+"':txt'")
            print ('line',line)
            print('Total Words :',totalWordsInFile)
            print ('matchCount',matchCount)
            docsWithTerm = docsWithTerm + 1
            tf = round((matchCount / totalWordsInFile),4)
            if(tf >= 0.0003):
                tfDict[line] = tf
            print('\n')
    else:
        continue
try:
    idf = round(math.log(round((corpusCount / docsWithTerm),4)) / math.log(2), 4)
    tfidf = round((tf * idf),4)
except:
    pass

data = dict()
with open('dict-hash-URL.txt','r') as raw_data:
    for item in raw_data:
        if "':txt'" in item:
            key,value = item.split("':txt':",1)
            key = key[key.rfind('/')+1:]
            data[key]=value
            #print(key)
        else:
            pass

with open('tfIdf_File.txt','w') as tfIdf_File:
    tfIdf_File.write('TFIDF    TF           IDF                 URL'+'\n')
    tfIdf_File.write('-----    --           ---                 ---'+'\n')

    for key, value in tfDict.items():

```

```
            key = key[key.rfind('/')+1:]
            #print("**************")
            #print(key)
            tfIdfValue = round((float(value) *idf),4)
75          tfValue,url = value,data[key]
            #url = data[key]
            tfIdf_File.write(str(tfIdfValue)+'    '+str(tfValue)+'    '+str(idf)+'    '+url)
```

The count of words using GREP can be seen at **"grepCMD_Output.txt"**

Listing 3: Few Extracted TF-IDF

```
TFIDF     TF          IDF                  URL
-----     --          ---                  ---

0.0006    0.0009    0.6631    https://techterrene.com/?p=16261

0.0006    0.0009    0.6631    https://blocksdecoded.com/best-crypto-youtube-channels/

0.0002    0.0003    0.6631    http://coinnews.design/
                                  positive-crypto-price-trend-moves-vechain-vet/

0.0002    0.0003    0.6631    https://blog.nomics.com/flippening/
                                  crypto-market-cap-audiobook/#more-1204
```

The TFIDF value of a URI can be found at **"tfIdf_Filet.txt"**

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://pr.eyedomain.com/
http://www.prchecker.info/check_page_rank.php
http://www.seocentro.com/tools/search-engines/pagerank.html
http://www.checkpagerank.net/

SOLUTION

The below Page Ranks are manually obtained using :

http://www.checkpagerank.net/

Listing 4: PageRanks out of 10

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

1               https://walk4cyber.blogspot.com/
2               https://blocksdecoded.com/
2               https://techterrene.com/
2               https://www.cryptoonair.com
3               https://topnews.one/
4               https://bitrss.com/
4               https://blog.nomics.com
4               https://breakermag.com
4               http://comics.trendolizer.com
5               https://www.theblockcrypto.com/
6               https://www.ccn.com/
6               https://ethereumworldnews.com
6               https://www.newsbtc.com
7               https://paper.li
7               https://cointelegraph.com
8               https://www.cnbc.com
8               https://www.forbes.com
```