CS432 Web Science: Assignment 3

Finished on February 27, 2020

Dr. Michele C. Weigle

Hannah Holloway
hhollowa@odu.edu

Contents

Problem 1 **2**
Question . . . 2
Answer . . . 2

Problem 2 **4**
Question . . . 4
Answer . . . 5

Problem 3 **6**
Question . . . 6
Answer . . . 7

Listings

1 Command: . . . 2
2 Command: . . . 2
3 Command: . . . 2
4 Command: . . . 2
5 Command: . . . 2
6 getRawHtml.py . . . 3
7 Command: . . . 4
8 tfidf.py . . . 5

List of Figures

Problem 1

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.

Examples from the command line:

```
$ curl http://www.cnn.com/ > www.cnn.com
```

Listing 1: Command:

```
% wget -O www.cnn.com http://www.cnn.com/
```

Listing 2: Command:

```
% lynx -source http://www.cnn.com/ > www.cnn.com
```

Listing 3: Command:

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 to associate them with their respective filename, like:

```
% echo -n "http://www.cs.odu.edu/show_features.shtml?72" | md5
41d5f125d13b4bb554e6e31b6b591eeb
```

Listing 4: Command:

(md5 might be 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 for all 1000 HTML documents. python-boilerpipe will do a fair job, see http://ws-dl.blogspot.com/2017/03/2017-03-20-survey-of-5-boilerplate.html:

```
from boilerpipe.extract import Extractor
    extractor = Extractor(extractor='ArticleExtractor', html=html)
    extractor.getText()
```

Listing 5: Command:

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

Answer

My first step in approaching this problem was using curl to get each of the 1000 web page's raw HTML and to save them individually. I used the curl command that was given as a

reference in my first script. I struggled parsing through the json file so I went through Part 1 Assignment 2 again and outputted to a links.txt instead. Each raw HTML file was generated in a separate file and named based on its index (1 to 1000) inside the folder rawmfiThe file names and links were saved in the folder rawmData.

```
import commands
import os, sys
from bs4 import BeautifulSoup

import re
import sys

reload(sys)
sys.setdefaultencoding('utf8')

def clean_html(html):

    cleaned = re.sub(r"(?is)<(script|style).*?>.*?(</\1>)", "", html.strip())
    cleaned = re.sub(r"(?s)<!--(.*?)-->[\n]?", "", cleaned)
    cleaned = re.sub(r"(?s)<.*?>", " ", cleaned)
    cleaned = re.sub(r" ", " ", cleaned)
    cleaned = re.sub(r"  ", " ", cleaned)
    cleaned = re.sub(r"  ", " ", cleaned)
    cleaned = re.sub("\n\s*\n*", "\n", cleaned)

    return cleaned.strip()

def saveTextToFile(filename, text):
    try:
        outfile = open(filename, 'w')
        outfile.write(text)
        outfile.close()
    except:
        errorMessage()

def derefURL(url):

    try:
        co = 'curl -L --silent -m 20 "' + url + '"'
        data = commands.getoutput(co)
        return data
    except:
        errorMessage()
        return ''

def errorMessage():
    exc_type, exc_obj, exc_tb = sys.exc_info()
    fname = os.path.split(exc_tb.tb_frame.f_code.co_filename[1])
    print fname, str(exc_tb.tb_lineno), str(sys.exc_info())

def getUniqueURLs():

    infile = open('./links.txt', 'r')
    lines = infile.readlines()
    infile.close()

    return lines

def saveRawHTML(lines):

    for i in range(0, len(lines)):

        textOutfilename = './Text/' + str(i) + '.txt'

        url = lines[i].strip()
        html = derefURL(url)
```

```
62
        if( len(html) != 0 ):

            try:

                text = clean_html(html)
67              if( len(text) != 0 ):
                    saveTextToFile(textOutfilename, text)
                    print '\tsaved text', len(text)

72          except:
                errorMessage()

            try:
                saveTextToFile('./RawHTMLFiles/' + str(i) + '.html', html)
77              print '\tsaved html'
            except:
                errorMessage()

        print i
82
if __name__ == '__main__':

    lines = getUniqueURLs()
    saveRawHTML(lines)
```

Listing 6: getRawHtml.py

The derefURL function gets the raw HTML from the files by using curl -L to follow redirects.

Problem 2

Question

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. Rank the URIs in decreasing order by TFIDF values. For example:

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
```

Listing 7: Command:

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!

Answer

My chosen query search term was "News". To retrieve the number of time "News" appeared in the raw html documents I used grep. More specially, grep-i allowed me to search for both the capitalized and lower case term. Grep -w allowed me to match the entire word and not sub strings, even though I can't think of any sub-string off the top of my head that News would be art of, but nevertheless wanted to eliminate the case scenario. The calculate function is the meat of the program that calculates the TFIDF for all the documents that included News.

```
from getRawHtml import getUniqueURLs
from getRawHtml import errorMessage
import os
import commands
import math

def searchForTermInFiles(lines, key):

    count = 0
    for i in range(0, len(lines)):

        print i
        url = lines[i].strip()
        filename = './Text/' + str(i) + '.txt'

        if( os.path.exists(filename) == False ):
            continue

        co = 'cat ' + filename + ' | grep -i -w ' + key
        output = commands.getoutput(co)
        #output = output.strip()

        if( len(output) != 0 ):
            print '\tFound'
            print '\t', url
            count += 1

    print 'found count:', count
    print 'key:', key

def calculateTFIDFForIndex(lines, docsWithTerm, key, idfvalue):

    print 'TF-IDF for key:', key
    for docIndex in docsWithTerm:

        url = lines[docIndex].strip()
        filename = './Text/' + str(docIndex) + '.txt'

        co = 'cat ' + filename + ' | grep -icw ' + key

        try:
            output = commands.getoutput(co)
            wordCount = int(output)

            co = 'wc -w ' + filename
            totalWords = commands.getoutput(co)
            totalWords = int(totalWords.split(' ')[0].strip())

```

```
            tf = 0
            if( totalWords > 0 ):
                tf = round(wordCount/float(totalWords), 4)

            print url
            print 'wordCount:', wordCount
            print 'totalWords:', totalWords
            print 'tf:', tf
            print 'idf:', IDF
            print 'tfidf:', round(tf * IDF, 4)


        except:
            errorMessage()

        print ''



lines = getUniqueURLs()

key = 'News'

IDF = 3.4266
docsWithTerm = [5, 6, 7, 11, 13, 19, 20, 25, 26, 68]
calculateTFIDFForIndex(lines, docsWithTerm, key, IDF)
```

Listing 8: tfidf.py

TFIDF	TF	IDF	URI
0.0	0.0	3.4266	http://pr.aljazeera.com/
0.024	0.007	3.4266	http://www.aljazeera.com/
0.0103	0.003	3.4266	http://www.aljazeera.com/
0.0	0.0	3.4266	http://pr.aljazeera.com/
0.0233	0.0068	3.4266	https://www.youtube.com/
0.0164	0.0048	3.4266	https://www.youtube.com/
0.0	0.0	3.4266	http://pr.aljazeera.com/
0.0055	0.0016	3.4266	http://www.cnn.com/
0.0024	0.0007	3.4266	http://www.cnn.com/
0.0065	0.0019	3.4266	http://www.cnn.com/

Problem 3

Question

Now rank the same 10 URIs from Q2, 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$_{p}age_{r}ank.phphttp$: $//www.checkpagerank.net/$

- https://dnschecker.org/pagerank.php
- https://smallseotools.com/google-pagerank-checker/

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/

Answer

The image below shows the page rank and URI. I used the first reference given "http://pr.eyedomain.com/" to get the ranking for the 10 urls.

Normalised	PageRank	URI
0.0	0	http://pr.aljazeera.com/
0.0	0	http://www.aljazeera.com/
0.0	0	http://www.aljazeera.com/
0.0	0	http://pr.aljazeera.com/
1.0	9	https://www.youtube.com/
1.0	9	https://www.youtube.com/
0.0	0	http://pr.aljazeera.com/
1.0	9	www.cnn.com/
1.0	9	http://www.cnn.com/
1.0	9	http://www.cnn.com/

10 URIs for the term "News"

The data I collected shows that not all urls with high TFIDF have high page ranking.

References

[1] Getting Started with Twitter API: http://pr.eyedomain.com/

[2] Natural Language Toolkit: https://github.com/nltk/nltk/commit/39a303e5ddc4cdb1a0b00a3be426239b1c24c8bb