CS532S19: Assignment #3

Due on Sunday, March 03, 2019

Nwala, Alexander C

Giridharan Ganeshkumar

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, style sheets, etc. Now use a tool to remove (most) of the HTML markup for all 1000 HTML documents.Keep both files for each URI (i.e., raw HTML and processed). Upload both sets of files to your github account

1. As shown in the below code used the urllib.request to download all the html response. We read the list of unique urls from the text file and use the urllib.request.urlopen() method to get the response object.
2. The next step was to to use the read method on the response to read the raw html and store in a file.
3. The response is wrote into a separate file for each url

Listing 1: Python Script

```
import urllib.request
import json
import csv
from urllib.parse import urlparse

urlCounter = 0

for currentUrl in open("C:\ListOfUniqueUrls.txt", "r"):
    try:
        urlCounter = urlCounter + 1
        with urllib.request.urlopen(currentUrl) as res:
            currentHtml = res.read()
            parsed_uri = urlparse(currentUrl)
            fileName = "C:\\RawHtmls\\" + str(urlCounter) + "_" +
                parsed_uri.netloc + ".txt"
            output_file = open(fileName, "w")
            output_file.write(str(currentHtml))
            output_file.close()
            print("Download Completed : " + fileName)
    except:
        print('Error :', currentUrl)
```

1. The next step was to install the boilerpipe using pip commands.
2. The below code block is used to extract the text alone from the html files saved from the previous step.
3. We instantiate the object for the extractor class and call the extractor.getText method to get raw text excluding the html
4. The extracted text is wrote into a new text file for each url as done in the first part of this exercise.

Listing 2: Python Script

```
from boilerpipe.extract import Extractor
from urllib.parse import urlparse
import glob
```

Question 1 continued on next page...

```
import os

path = 'C:\\RawHtmls/*.txt'
urlCounter = 0
files=glob.glob(path)
for file in files:
    try:
        urlCounter = urlCounter + 1
        f=open(file, 'r')
        fileName = fileName = "C:\\ProcessedText\\" + os.path.basename(f.
            name)
        currentHtml = f.read()
        f.close()
        extractor = Extractor(extractor='ArticleExtractor', html=
            currentHtml)
        currentText = extractor.getText()
        output_file = open(fileName, "w")
        output_file.write(str(currentText.encode("utf-8")))
        output_file.close()
        print("Download Completed : " + fileName)
    except:
        print('Error :', urlCounter)
```

Question 2

Choose a query term shadow that is not a stop word (see week 5 slides) and not HTML markup from step 1 that matches at least 10 documents. If the term is present in more than 10 documents, choose any 10 from your list. 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

1. As in the below code block we start by searching the text files we processed in the previous step.
2. To serach we loop through all the files open it read line by line and then use re.match to look up for the text we are searching. For this purpose i choose different terms like basketball, genetics, has , cricket and music. The various results are uploaded to github under the SearchTermResults.zip file.
3. During the process we keep a hit count as well so that we can use it in for calculation at a later stage.

Listing 3: Python Script

```
import glob
import os
import csv
import re

searchString = "music"
path = 'C:\\ProcessedText/*.txt'
output_file_name = "C:\\SearcTermResults\\" + searchString +".csv"
with open(output_file_name, "w", newline='') as output_file:
    writer = csv.writer(output_file)
    files=glob.glob(path)
```

Question 2 continued on next page...

```
    for file in files:
        #try:
        f=open(file, 'r')
        fileName = os.path.basename(f.name)
        print(fileName)
        hit_count = 0
        regexSearch = "(.*)(" + searchString + ")(.*)"
        for line in f:
            if re.match(regexSearch, line):
                currenthit_count = line.count(searchString)
                hit_count = hit_count + currenthit_count
        f.close()
        currentData = [fileName,str(hit_count)]
        writer.writerow(currentData)
        #except:
        #    print('Error :', fileName)
    #print("Search Completed : " + fileName)
output_file.close()
print("Search Completed!")
```

1. We use the below code block to 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
2. The search term used for the case is basketball, we start by identifying the 10 uris and these are saved as a csv for further processing.
3. TF is Number of times term t appears in a document divide by Total number of terms in the document.
4. IDF is log base 2 of Total number of documents in which the term appears divide by Number of documents in whole.
5. idf Numerator is what we calculate first which is done by reading the basketball.csv we got from teh previous step during search. We just open the basketball.csv and add all the hits for evey uri response.
6. tfNumerator is simply the hit count which we have already calculated for each uri in the previous step
7. tfDenominator is the number words which can be calculated recursively calculated by adding up the length of array of word split fro each line in the document.
8. After identifying all the required values for each we just calculate Tf by dividing tfNumerator and tfDenominator, then calculate Tf by dividing idfNumerator and idfDenominator. Use math library to apply the log base 2.
9. The final step is to calculate the TfIdf by multiplying the above results and write it to the TFIDF.csv.

Listing 4: Python Script

```
import csv
import math

idfDenominator = 991
idfNumerator = 0
```

Question 2 continued on next page...

URL	TF	IDF	IDFLogBase2
505_www.azcentral.com.txt	0.01863000325414904	0.2875882946518668	-1.7979231381343843
435_www.northjersey.com.txt	0.00048426150121065375	0.2875882946518668	-1.7979231381343843
1006_www.gosanangelo.com.txt	0.0004005126561999359	0.2875882946518668	-1.7979231381343843
558_www.denverpost.com.txt	0.0003775579551461149	0.2875882946518668	-1.7979231381343843
256_www.indystar.com.txt	0.0002833463200396685	0.2875882946518668	-1.7979231381343843
509_www.lansingstatejournal.com.txt	0.00028143249138112993	0.2875882946518668	-1.7979231381343843
249_thetartan.org.txt	0.0002027027027027027	0.2875882946518668	-1.7979231381343843
895_leaderpost.com.txt	0.00019857029388403494	0.2875882946518668	-1.7979231381343843
186_www.thevwindependent.com.txt	0.00012893243940175349	0.2875882946518668	-1.7979231381343843
371_carver.wickedlocal.com.txt	0.00012548625925461163	0.2875882946518668	-1.7979231381343843

```
with open("C:\\SearcTermResults\\basketball.csv") as
    allSearchResultsForBasketball:
    reader_SearchResultsForBasketball = csv.reader(
        allSearchResultsForBasketball, delimiter=',')
    for searchResultsDetails in reader_SearchResultsForBasketball:
        idfNumerator = idfNumerator + int(searchResultsDetails[1])

with open("C:\\TFIDF.csv", "w", newline='') as tfIdfFile:
    writer = csv.writer(tfIdfFile)
    currentData = ['URL','TF', 'IDF','IDFLogBase2','TFIDF','TFIDFLogBase2']
    writer.writerow(currentData)
    with open("C:\\SearcTermResults\\
        BasketballSearchterm10IdentifiedForFurtherProcessing.csv") as
        idfentifiedresultFile:
        csv_reader = csv.reader(idfentifiedresultFile, delimiter=',')
        tfNumerator = 0
        tfDenominator = 0
        for row in csv_reader:
            tfNumerator = row[1]
            fileName_ResponseText = "C:\\ProcessedText\\" + row[0]
            with open(fileName_ResponseText, 'r') as f:
                for line in f:
                    words = line.split()
                    tfDenominator =  tfDenominator + len(words)
            TF = int(tfNumerator)/tfDenominator
            IDF = idfNumerator/idfDenominator
            IDFlogbase2 = math.log(IDF,2)
            TfIdf = TF * IDF
            TfIdfBasedOnlogbase2 = TF * IDFlogbase2
            currentData =[row[0],str(TF),str(IDF),str(IDFlogbase2),str(
                TfIdf),str(TfIdfBasedOnlogbase2)]
            writer.writerow(currentData)
print('Completed')
```

Question 3

Now rank the same 10 URIs from question 2, but this time by their Page Rank.

1. In order to get the page ranks used couple of sites, and the results are shown in the below table.

Question 4 continued on next page...

URL	prchecker Site	checkpagerank Site
505_www.azcentral.com.txt	6	8
435_www.northjersey.com.txt	6	7
1006_www.gosanangelo.com.txt	5	5
558_www.denverpost.com.txt	7	8
256_www.indystar.com.txt	6	7
509_www.lansingstatejournal.com.txt	6	6
249_thetartan.org.txt	6	5
895_leaderpost.com.txt	7	6
186_www.thevwindependent.com.txt	0	4
371_carver.wickedlocal.com.txt	0	6

Question 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.

1. The below script is used to to calculate kendall tau
2. To do this imported the library scipy.stats that in turn imports kendalltau
3. The library expects the data set to be a an array of ranks, we get the ranks for various options and pass it the function
4. The various options calculated are shown below.

1. The below script is used to to calculate kendall tau
2. To do this imported the library scipy.stats that in turn imports kendalltau
3. The library expects the data set to be a an array of ranks, we get the ranks for various options and pass it the function
4. The various options calculated are shown below.

Listing 5: Python Script

```
from numpy.random import rand
from numpy.random import seed
from scipy.stats import kendalltau
import csv
seed(1)

data1, data2 = [],[]
with open("C:\InputForKendallTau_b1.csv") as inputForKendallTau_b:
    reader_inputForKendallTau_b = csv.reader(inputForKendallTau_b,
        delimiter=',')
    next(reader_inputForKendallTau_b)
    for row in reader_inputForKendallTau_b:
        #print(row[1])
        data1.append(row[2])
        data2.append(row[3])

coef, p = kendalltau(data1, data2)
print('Kendall correlation coefficient: %.3f' % coef)
alpha = 0.05
```

Question 4 continued on next page...

```
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) p=%.3f' % p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

1. TFID vs CheckPageRank
   Kendall correlation coefficient: 0.200
   Samples are uncorrelated (fail to reject H0) p=0.421

Question 5

Compute a ranking for the 10 URIs from Q2 using Alexa information (see week 4 slides). Compute the correlation (as per Q4) for all pairs of combinations for TFIDF, PR, and Alexa.

1. The below script is used to calculate Alexa rank
2. To do this we post a request to http://data.alexa.com/data with the url as parameter
3. The response is read as xml and use beatifulsoup to get the Rank value from the reach node.
4. We also use the previous script to compare the ranks and we get the Kendall Tab u scores for different combinations as shown below.

Listing 6: Python Script

```
import urllib, sys, bs4, csv
with open("C:\RankAlexa.csv", "w", newline='') as csvFileRankAlexa:
    writer = csv.writer(csvFileRankAlexa)
    currentData = ['URL','AlexaPageRank']
    writer.writerow(currentData)
    with open("C:\InputForAlexa.csv") as inputForKendallTau_b:
        reader_inputForKendallTau_b = csv.reader(inputForKendallTau_b,
            delimiter=',')
        next(reader_inputForKendallTau_b)
        for row in reader_inputForKendallTau_b:
            try:
                xml = urllib.request.urlopen('http://data.alexa.com/data?
                    cli=10&dat=snbamz&url=s'+ row[0]).read()
                rank = bs4.BeautifulSoup(xml, "xml").find("REACH")['RANK']
                currentData = [row[0],rank]
                writer.writerow(currentData)
            except:
                print('Rank not found for ' + row[0])
```

1. TFID vs CheckPageRank
   Kendall correlation coefficient: 0.200
   Samples are uncorrelated (fail to reject H0) p=0.421
2. TFID vs Alexa
   Kendall correlation coefficient : 0.156
   Samples are uncorrelated (fail to reject H0) p=0.531

Question 5 continued on next page...

3. CheckPageRank vs Alexa
   Kendall correlation coefficient: 0.867
   Samples are correlated (reject H0) p=0.000