Assignment 2

Tarek Fouda

2016-02-18

1 Introduction

This report explains how I managed to solve Assignment 3 in Web Science class which is due 02/18/2016. It is mainly consisted of three Questions. Will show my approaches and implementation in each of the three Questions.

2 Problem 1- Extracting the raw HTML of the 1000 links

2.1 My approach

In the very first question, it was required to implement code to use lynx -source to be able to get the raw HTML for each of the URI's I got from Assignment 2.

The very first approach I wanted to implement, is to be able to loop on the urls.txt file that contains the 1000 links, and apply the following line:

```
lynx − source firstURL > www.firstURL.com
```

Keep in mind we should do the above command for all of the 1000 URL's. So I open the urls.txt file and call that line in my python code:

```
proc = subprocess.Popen(["lynx −source "+ '"'+s+'"' +"> " +
     destination + "{0}".format(i)+".txt"], stdout=subprocess.PIPE,
     shell=True)
```

where s is each URL found in the text file. The python code for getting the raw HTML is shown below:

```
import os
import subprocess
destination  = "html"
i = 1
with open('urls.txt') as fil:
        for line in fil:
                        s = line.strip('\n')
                        proc = subprocess.Popen(["lynx −source "+ '
                            "'+s+'"' +"> " + destination + "{0}".
                            format(i)+".txt"], stdout=subprocess.
                            PIPE, shell=True)
                        (out, err) = proc.communicate()
```

```
                    i = i + 1
```

Listing 1: python code to get raw HTML for all URL's

as shown above the code loops to get all URLs found in my urls.txt file, we basically perform lynx command on every URL and create html1, html2, html 3 till html1000.

The next step after getting the raw HTML is processing these documents by the following command:

```
proc = subprocess.Popen(["lynx -dump -force_html "+ text_files[i]
    +"> " + text_files[i] + ".proccesed"], stdout=subprocess.PIPE,
    shell=True)
```

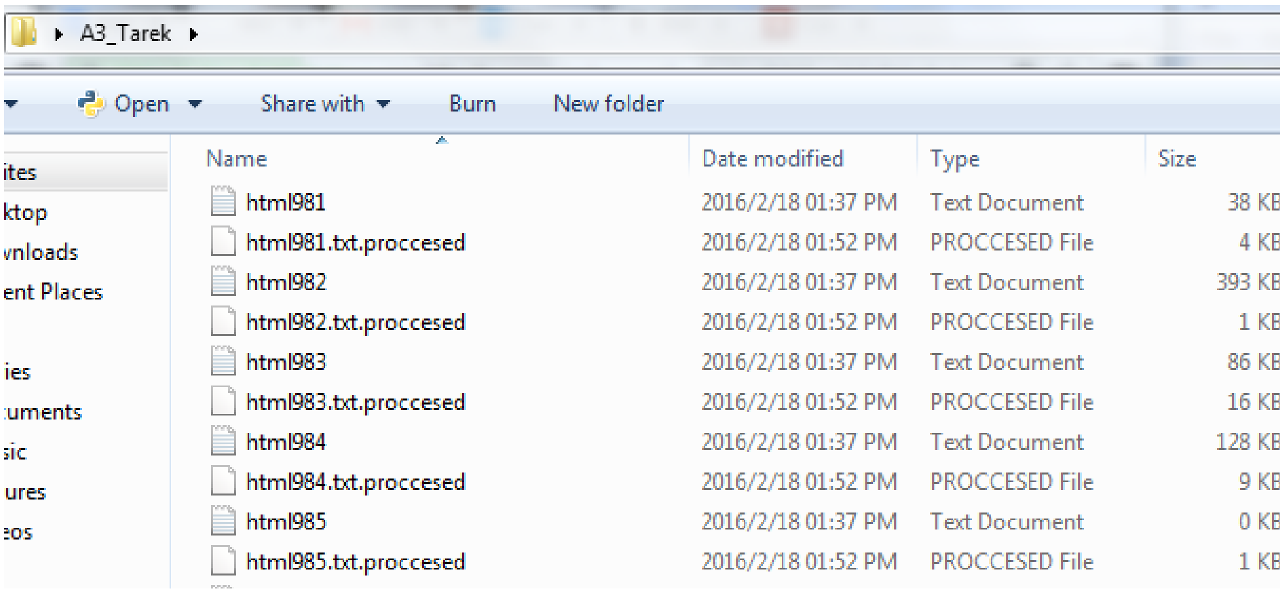
But this time I will be looping to get all the html files created in the first step because the above command is executed on a file. So I get the names of all of the HTML files created then loop and perform the above command on each one of them. The code for such a python program is shown below:

```
1  import os
2  import subprocess
3  pth = '/Users/mohamedshaaban/Desktop/Tarek_Fouda/'
4  text_files=[f for f in os.listdir(pth) if f.endswith('.txt')]
5  i= 0
6  while (i<len(text_files)-1):
7          with open(text_files[i]) as fil:
8                  proc = subprocess.Popen(["lynx -dump -force_html "+
                       text_files[i] +"> " + text_files[i] + ".
                     proccesed"], stdout=subprocess.PIPE, shell=True
                     )
9
10                 (out, err) = proc.communicate()
11
12         i=i+1
```

Listing 2: python code to create the .processed files

The challenge in this part was to get all the text files, and that was easily done in line 4, by putting the text files in a list then loop on this list to execute the command and create the processed files.

So by now I have 1000 raw html files and 1000 processed files as well.



Name	Date modified	Type	Size
html981	2016/2/18 01:37 PM	Text Document	38 KB
html981.txt.proccesed	2016/2/18 01:52 PM	PROCCESED File	4 KB
html982	2016/2/18 01:37 PM	Text Document	393 KB
html982.txt.proccesed	2016/2/18 01:52 PM	PROCCESED File	1 KB
html983	2016/2/18 01:37 PM	Text Document	86 KB
html983.txt.proccesed	2016/2/18 01:52 PM	PROCCESED File	16 KB
html984	2016/2/18 01:37 PM	Text Document	128 KB
html984.txt.proccesed	2016/2/18 01:52 PM	PROCCESED File	9 KB
html985	2016/2/18 01:37 PM	Text Document	0 KB
html985.txt.proccesed	2016/2/18 01:52 PM	PROCCESED File	1 KB

Figure 1: The files I have in my directory

3 Problem number 2 TF-IDF

The task in this question is to choose a word (in my case I chose ‘MARKET‘ to be the keyword), and find a list of the processed files that I have created in the first question in this assignment to be matching. The mission to count this word in all the thousand processed files was challenging. I tried to use gerp command but it printed out the whole lines in the Terminal and could not keep track of neither the number of the keyword repition nor the place.

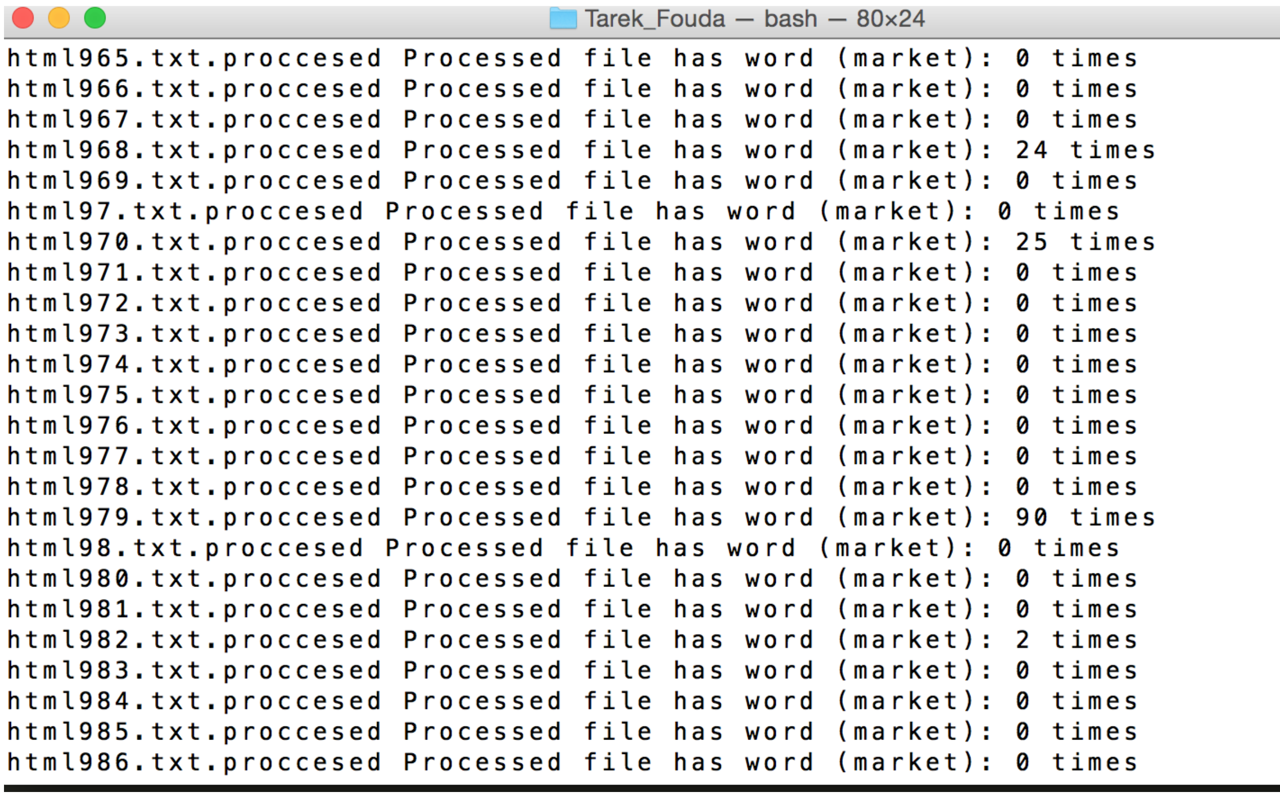
I found it more convenient and easier to use python to loop in each file and check if the Key word is there, also printing out the number of occurences of this word in each file and the number of files that this keyword occured in. The following is the program for such a task:

```python
1  import os
2  import subprocess
3  pth = '/Users/mohamedshaaban/Desktop/Tarek_Fouda/'
4  text_files=[f for f in os.listdir(pth) if f.endswith('.txt.
       proccesed')]
5  fileindex= 0
6  numberoffiles = 0
7  while (fileindex<len(text_files)):
8          #with open(text_files[i]) as fil:
9                  wordcount=0
10                 for line in open(text_files[fileindex]):
11                         if "market" in line:
12                                 wordcount = wordcount+1
13                 if(wordcount>0):
14                         numberoffiles = numberoffiles +1
15                 print text_files[fileindex] + " Processed file has
                       word (market): " + "{0}".format(wordcount) + "
                       times"
16                 fileindex=fileindex+1
17 print "Number of Total files having the word is: " + "{0}".format(
       numberoffiles)
```

Listing 3: python code to count the keyword in the processed files

Line 4 searches for all the files that ends with '.txt.processed' in my directory, as these are the files we need to implement that program on. Variable 'fileindex' is used to keep track of the file we are iterating on,'numberoffiles' variable keeps track of the number of files that has the Keyword in and finally 'wordcount' keeps track of how many times the keyword occured in every file. The following is the table that computes TFIDF values.

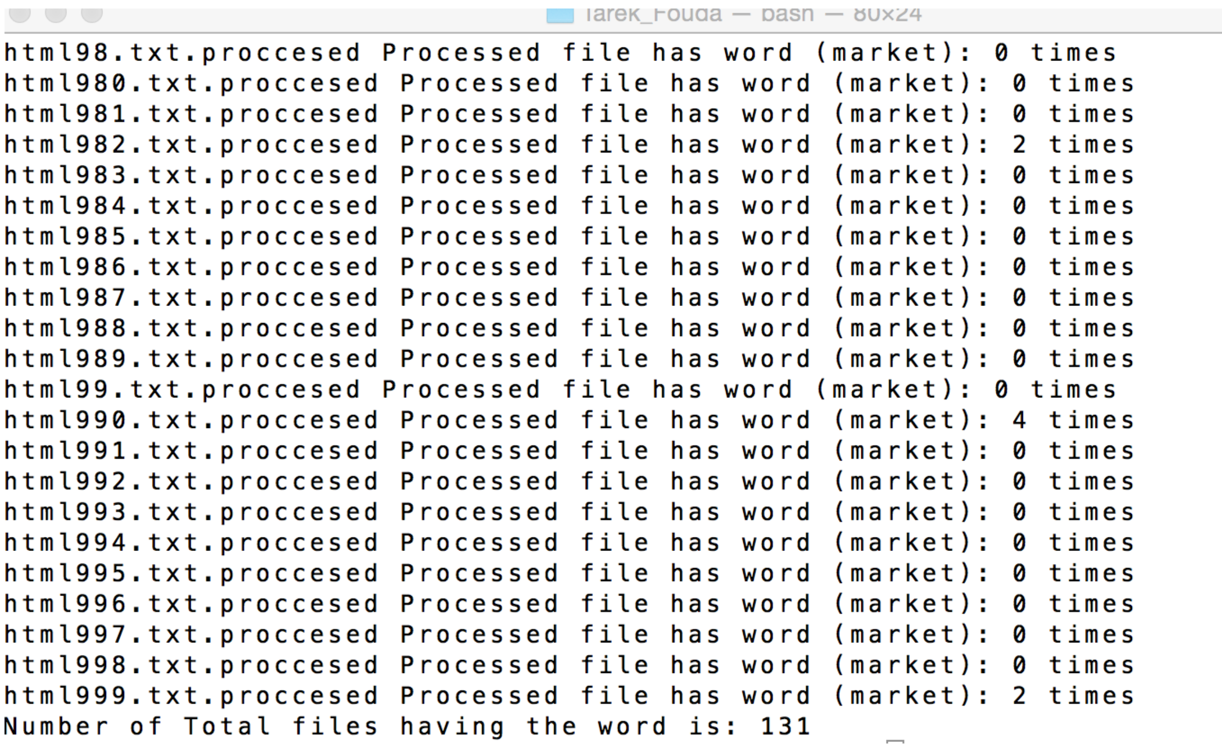
TFIDF	TF	IDF	URI
0.29	0.07	log2(50b/2.7B)= 4.2	http://bit.ly/1PFA6c1
0.12	0.03	log2(50b/2.7B)= 4.2	http://onion.com/1TTlETg
0.0748	0.18	log2(50b/2.7B)= 4.2	http://ift.tt/1L9egku
0.0707	0.017	log2(50b/2.7B)= 4.2	http://buff.ly/1NRWn4i
0.0416	0.01	log2(50b/2.7B)= 4.2	http://compassamg.com/blog/2014/11/19
0.03328	0.008	log2(50b/2.7B)= 4.2	http://bit.ly/1o3yVMi
0.020	0.005	log2(50b/2.7B)= 4.2	http://bit.ly/1VUeigI
0.012	0.003	log2(50b/2.7B)= 4.2	http://www.onlinebusinessteam.com/
0.010	0.005	log2(50b/2.7B)= 4.2	http://www.onlinebusinessteam.com/
0.002	0.0005	log2(50b/2.7B)= 4.2	http://negoogle.com//



Tarek_Fouda — bash — 80×24

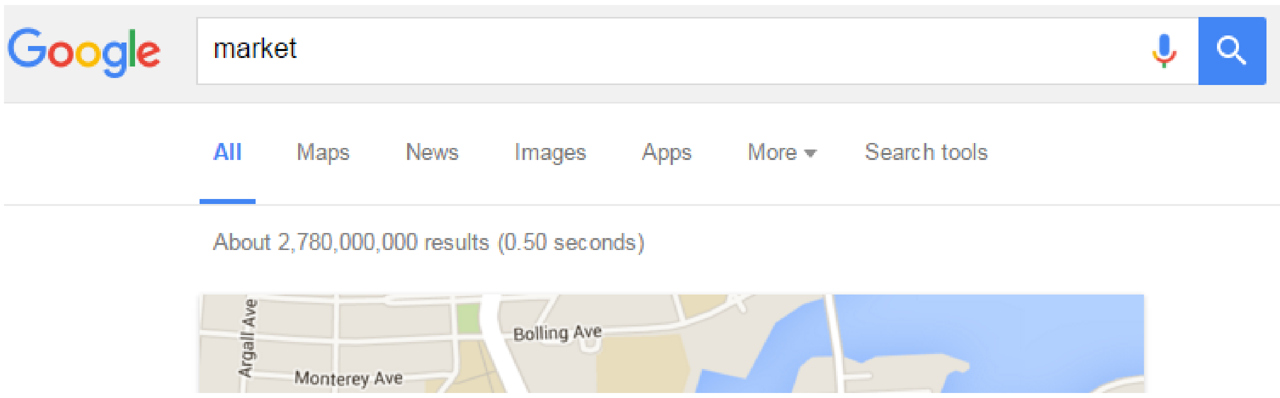
```
html965.txt.proccesed Processed file has word (market): 0 times
html966.txt.proccesed Processed file has word (market): 0 times
html967.txt.proccesed Processed file has word (market): 0 times
html968.txt.proccesed Processed file has word (market): 24 times
html969.txt.proccesed Processed file has word (market): 0 times
html97.txt.proccesed Processed file has word (market): 0 times
html970.txt.proccesed Processed file has word (market): 25 times
html971.txt.proccesed Processed file has word (market): 0 times
html972.txt.proccesed Processed file has word (market): 0 times
html973.txt.proccesed Processed file has word (market): 0 times
html974.txt.proccesed Processed file has word (market): 0 times
html975.txt.proccesed Processed file has word (market): 0 times
html976.txt.proccesed Processed file has word (market): 0 times
html977.txt.proccesed Processed file has word (market): 0 times
html978.txt.proccesed Processed file has word (market): 0 times
html979.txt.proccesed Processed file has word (market): 90 times
html98.txt.proccesed Processed file has word (market): 0 times
html980.txt.proccesed Processed file has word (market): 0 times
html981.txt.proccesed Processed file has word (market): 0 times
html982.txt.proccesed Processed file has word (market): 2 times
html983.txt.proccesed Processed file has word (market): 0 times
html984.txt.proccesed Processed file has word (market): 0 times
html985.txt.proccesed Processed file has word (market): 0 times
html986.txt.proccesed Processed file has word (market): 0 times
```

Figure 2: Sample for the links with the number of occurences of the keyword



```
html98.txt.proccesed Processed file has word (market): 0 times
html980.txt.proccesed Processed file has word (market): 0 times
html981.txt.proccesed Processed file has word (market): 0 times
html982.txt.proccesed Processed file has word (market): 2 times
html983.txt.proccesed Processed file has word (market): 0 times
html984.txt.proccesed Processed file has word (market): 0 times
html985.txt.proccesed Processed file has word (market): 0 times
html986.txt.proccesed Processed file has word (market): 0 times
html987.txt.proccesed Processed file has word (market): 0 times
html988.txt.proccesed Processed file has word (market): 0 times
html989.txt.proccesed Processed file has word (market): 0 times
html99.txt.proccesed Processed file has word (market): 0 times
html990.txt.proccesed Processed file has word (market): 4 times
html991.txt.proccesed Processed file has word (market): 0 times
html992.txt.proccesed Processed file has word (market): 0 times
html993.txt.proccesed Processed file has word (market): 0 times
html994.txt.proccesed Processed file has word (market): 0 times
html995.txt.proccesed Processed file has word (market): 0 times
html996.txt.proccesed Processed file has word (market): 0 times
html997.txt.proccesed Processed file has word (market): 0 times
html998.txt.proccesed Processed file has word (market): 0 times
html999.txt.proccesed Processed file has word (market): 2 times
Number of Total files having the word is: 131
```

Figure 3: Sample for the links with the number of occurences of the keyword



About 2,780,000,000 results (0.50 seconds)

Figure 4: IDF

TF is the keyword frequency/number of words in the document. IDF is the log2(total docs in corpus / docs with term) Finally the TFIDF is the multiplication of TF and IDF.

4 Problem number 3

Imanually entered one of the website mentioned in the assignment to calculate the page rank for each of the 10 urls I have. 7 of the URL's gave me undefined as an answer and only 3 gave me the rank.

Also if I put the URLs as a whole it will give me undefined, so I put the root url. Example: Instead of www.cs.odu.edu/1526/br? I put www.cs.odu.edu as it does not give me undefined.

The following is the table.

Page Rank	URI
0.7	http://onion.com/1TTlETg
0.6	http://buff.ly/1NRWn4i
0.2	http://compassamg.com/blog/2014
undefined	http://negoogle.com/
undefined	http://mindpluckd.com/2015
undefined	http://ift.tt/1L9egku
undefined	http://www.onlinebusinessteam.com/2015
undefined	http://bit.ly/1VUeigI
undefined	http://bit.ly/1o3yVMi
undefined	http://www.onlinebusinessteam.com/

The table in figure two is basically comparing and computing the number of occurence of word market in each of the 10 links. it calculates the frequency of this term by the rules stated in Section 2. On the other hand the page rank table shows how popular the URL is, and does not have somthing to do with the word market.