Bo Zhang
01063214

Task 1: Download the 1000 URIs from assignment #2. Use a tool to remove (most of) the HTML markup.

Algorithm:
1. Open the links file "links.txt".
2. Reset the output file index.
3. For every link, set the corresponding output file name with the index.
4. Check if the output file exists. If not, try to open the link and write the content to the output file.
5. If there is something wrong, save the link to another file "missingLinks.txt".
6. Move the output file index.
7. Open and read the html files 1 by 1.
8. Remove the HTML markup and save it to corresponding TXT output file. Use the script from `http://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text` to remove the HTML markup.

Source code:

download_HTML.py

process_HTML.py

Results:

folder: htmls

folder: htmls_processed

In the 1000 original links, there are 18 links dead and 2 mp3 links. Therefore, there are 982 files in the "htmls" folder and 980 files in the "html_processed" folder.

Note:
Sometimes there are some live URIs unable to be opened tentatively. So it is necessary to check the links in "missingLinks.txt" to find out if they are all dead links. Therefore, the script might need to be run several times until all the links in "missingLinks.txt" are dead links.

Task 2:
Choose a query term that is not a stop word 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.

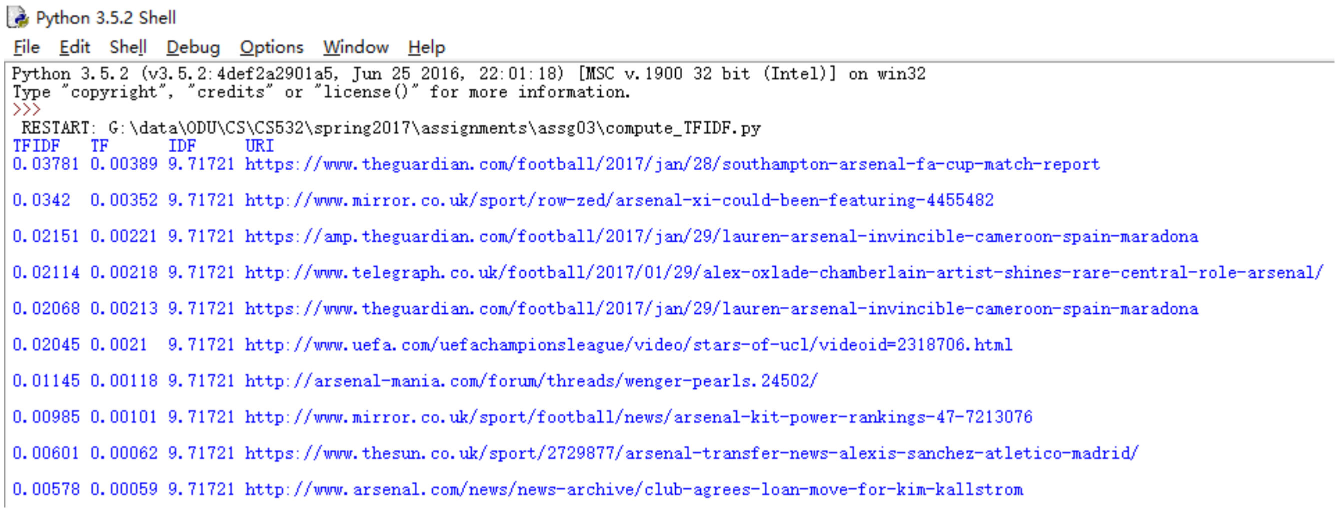
As per the example in the week 5 slides, compute TFIDF values for the term in each of the 10 documents and create a table with the TF, IDF, and TFIDF values, as well as the corresponding URIs. The URIs will be ranked in decreasing order by TFIDF values.

Algorithm:
1. Open and read the TXT files 1 by 1 until 10 suitable TXT file have been found.
2. Split the content of the TXT file into words.
3. Check the words 1 by 1 and count the number of given term.

4. Compute TF, IDF, TFIDF and add all of the content needed to the output table list.
5. Sort the output list.
6. Print the output table.

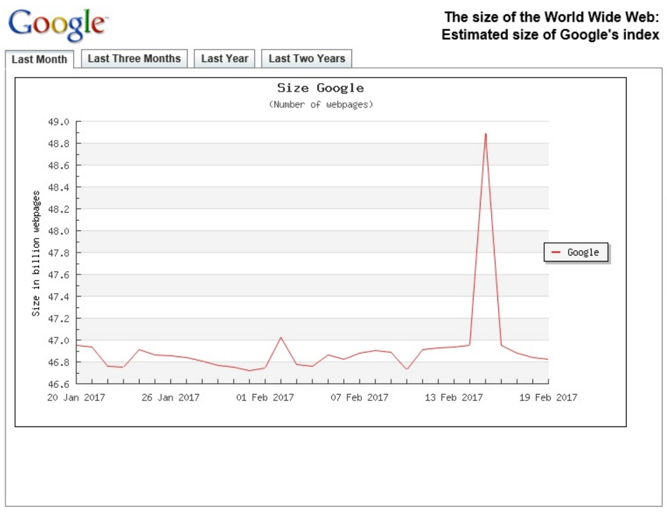
Source code: compute_TFIDF.py

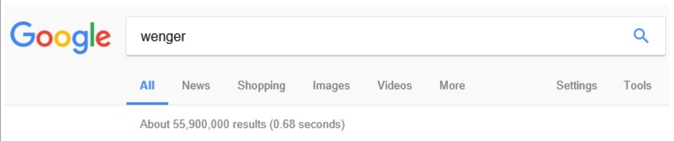
Results:



```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: G:\data\ODU\CS\CS532\spring2017\assignments\assg03\compute_TFIDF.py
TFIDF   TF      IDF     URI
0.03781 0.00389 9.71721 https://www.theguardian.com/football/2017/jan/28/southampton-arsenal-fa-cup-match-report

0.0342  0.00352 9.71721 http://www.mirror.co.uk/sport/row-zed/arsenal-xi-could-been-featuring-4455482

0.02151 0.00221 9.71721 https://amp.theguardian.com/football/2017/jan/29/lauren-arsenal-invincible-cameroon-spain-maradona

0.02114 0.00218 9.71721 http://www.telegraph.co.uk/football/2017/01/29/alex-oxlade-chamberlain-artist-shines-rare-central-role-arsenal/

0.02068 0.00213 9.71721 https://www.theguardian.com/football/2017/jan/29/lauren-arsenal-invincible-cameroon-spain-maradona

0.02045 0.0021  9.71721 http://www.uefa.com/uefachampionsleague/video/stars-of-ucl/videoid=2318706.html

0.01145 0.00118 9.71721 http://arsenal-mania.com/forum/threads/wenger-pearls.24502/

0.00985 0.00101 9.71721 http://www.mirror.co.uk/sport/football/news/arsenal-kit-power-rankings-47-7213076

0.00601 0.00062 9.71721 https://www.thesun.co.uk/sport/2729877/arsenal-transfer-news-alexis-sanchez-atletico-madrid/

0.00578 0.00059 9.71721 http://www.arsenal.com/news/news-archive/club-agrees-loan-move-for-kim-kallstrom
```

Note: The total pages in google is about 46.8 billion and the pages with term in google is about 55.9 million.





About 55,900,000 results (0.68 seconds)

Task 3: Now rank the same 10 URIs from question #2, but this time by their PageRank. Normalize the values they give you to be from 0 to 1.0. Create a table similar to Table 1. Briefly compare and contrast the rankings produced in questions 2 and 3.

Results:

```
PageRank        URI
0.8             http://www.telegraph.co.uk/football/2017/01/29/alex-oxlade-chamberlain-artist-shines-rare-central-role-arsenal/
0.7             https://www.theguardian.com/football/2017/jan/28/southampton-arsenal-fa-cup-match-report
0.7             http://www.mirror.co.uk/sport/row-zed/arsenal-xi-could-been-featuring-4455482
0.7             https://amp.theguardian.com/football/2017/jan/29/lauren-arsenal-invincible-cameroon-spain-maradona
0.7             https://www.theguardian.com/football/2017/jan/29/lauren-arsenal-invincible-cameroon-spain-maradona
0.7             http://www.uefa.com/uefachampionsleague/video/stars-of-ucl/videoid=2318706.html
0.7             http://www.mirror.co.uk/sport/football/news/arsenal-kit-power-rankings-47-7213076
0.7             https://www.thesun.co.uk/sport/2729877/arsenal-transfer-news-alexis-sanchez-atletico-madrid/
0.6             http://www.arsenal.com/news/news-archive/club-agrees-loan-move-for-kim-kallstrom
0.4             http://arsenal-mania.com/forum/threads/wenger-pearls.24502/
```

Analysis:

According to TFIDF, the difference between the 10 links is obvious. The maximum is 6.6 times of the minimum. But according to the PageRank, the difference between the 10 links is very little. Most of the links are 0.7 and the difference between the highest and lowest is only 0.8 v.s. 0.4. For these 2 links, the one with PageRank 0.8 only gets the 4th highest TFIDF and the one with PageRank 0.4 is still the 7th of TFIDF.

Task 4: Compute the Kendall Tau_b score for both lists. Report both the Tau value and the "p" value.

R code: Q4&5.R

Results:

Correlation Tau	**P-value**
0.3651484	0.1815

Analysis:

Due to the p-value, the correlation is not significant enough (just about the possibility of 82% that the correlation exists). And the coefficient itself is weak too (less than 0.5). Therefore the TFIDF and PageRank of these links have little correlation.

Task 5: Compute a ranking for the 10 URIs from Q2 using Alexa information. Compute the correlation for all pairs of combinations for TFIDF, PR, and Alexa.

R code: Q4&5.R

Results:

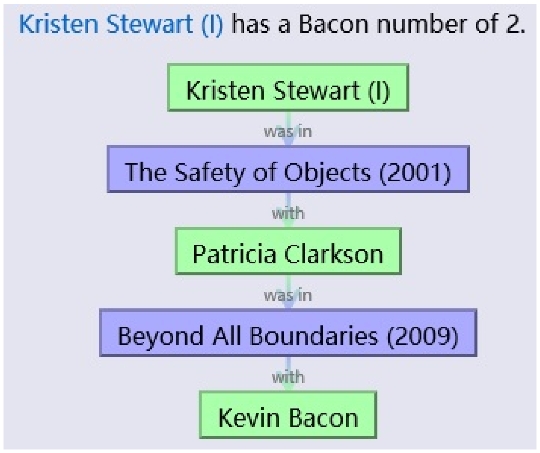
Variables	**Correlation Tau**	**P-value**
alexa & PageRank	-0.573819	0.04162
alexa & TFIDF	-0.5354632	0.03602

Analysis:

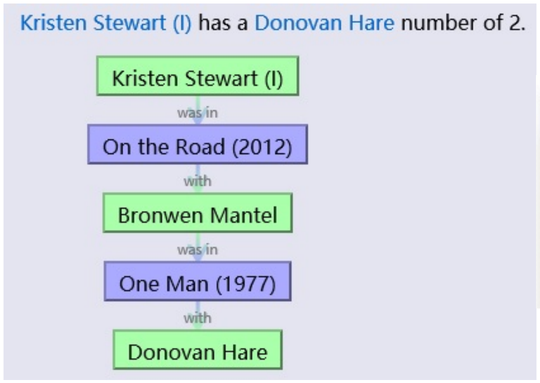
Due to the p-values, either the correlation between alexa & PageRank or alexa & TFIDF is significant enough (both of the possibilities are more than 95%). And the absolute values of the coefficients are more than 0.5. Since the coefficients are negative, either the alexa & TFIDF or alexa & PageRank of these links have negative correlation, which means the higher alexa score a link has, the lower TFIDF or PageRank it has. Because the nature of alexa score is the lower the better, it is in accordance with TFIDF and PageRank.

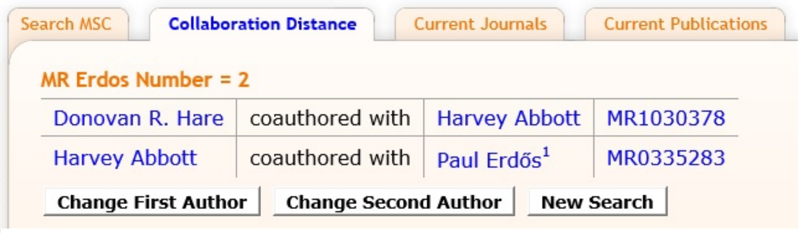
Task 6: Give an in-depth analysis, complete with examples, graphs, and all other pertinent argumentation for Kristen Stewart's (of "Twilight" fame) Erdos-Bacon number.

A person's Erdos-Bacon number is the sum of one's Erdos number and one's Bacon number. And Kristen Stewart's Bacon number is 2.


Kristen Stewart (I) has a Bacon number of 2.
Kristen Stewart (I)
was in
The Safety of Objects (2001)
with
Patricia Clarkson
was in
Beyond All Boundaries (2009)
with
Kevin Bacon

For her Erdos number, she has a Donovan Hare Number of 2 and Donovan Hare has an Erdos number of 2. Therefore, Kristen Stewart has an Erdos number of 4 and her Erdos-Bacon number = 4 + 2 = 6.





MR Erdos Number = 2

Donovan R. Hare	coauthored with	Harvey Abbott	MR1030378
Harvey Abbott	coauthored with	Paul Erdős[1]	MR0335283

Task 7: Build a simple inverted file for all the words from your 1000 URIs. Upload the entire file to github

and discuss an interesting portion of the file in your report.

Algorithm:

1. Set 2 lists for the word index(word and its number).
2. Open and read the TXT files 1 by 1.
3. Split the content of the TXT file into words.
4. Check the words 1 by 1 and compare with the words in the word index.
5. If the word is in the word index, update its number; if not, add it to the word index.
6. Merge the 2 lists into 1 single word index list.
7. Sort the word index list.
8. Output the word index list into the inverted file "inverted file.txt".

Source code: inverted_file.py

R code: Q7.R

Results:

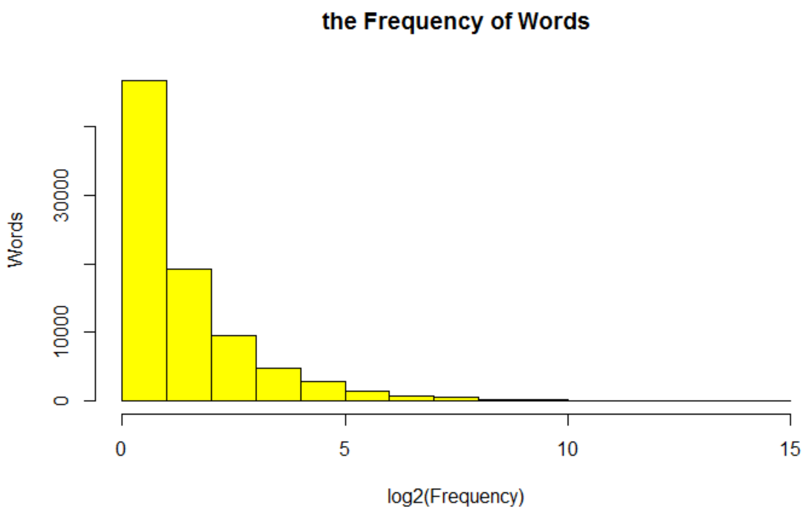
inverted file.txt

inverted file.csv

Analysis:

The top word is "the"(according to the number) and the top nonstop word is "arsenal"(because I got these links by searching "arsenal"). But it is interesting that its number is even more than some stop words such as "on", "for", "is" and "with", which may imply that these stop words are less likely to be stop words. The second place of nonstop word is taken by "news", which may imply that people tend to tweet news.

And for the frequencies of these words, most of the words occurs less than 32 times and few words occurs more than 1024 times. And the more the frequency is, the less the words are.


the Frequency of Words
Words
log2(Frequency)