Assignment 3

Joshua Graham

February 20, 2018

Abstract

All Scripts used in the assignment can be found in the A3 folder, if needed.

0.1 Problem 1

The code used can be found in stripall.py.

The first half of question 1 was to download the raw HTML files from the links we generated in assignment 2. I downloaded these with the command shown below.

```
for i in $(cat links.txt); do curl -I $i; done
```

I chose not to use python for this part of the assignment because I believed it would be easier to just write up a quick one-liner for it. There weren't any real issues with this part of the assignment. This was the easy part.

The second half of the assignment was more challenging. I could not get python-boilerpipe to read my downloaded .html files, so I ended up reading off of the links.txt file again. The extractor object can be used with an html file, but it can also take a link. This was an easy enough solution, although I ran into a few issues. During assignment 2, my keyword was "Olympics". Because the olympics are activley going on, a few of the sites I found have already been taken down. In the event of them being taken down, I chose to not make any sort of replacement file, but to instead leave that file missing. Another issue I ran into was that some twitter users posted direct links to images. In a few situations, this would cause python-boilerpipe to throw an exception. In the event of these execptions, I chose, again, to just not create a file for these links.

It is important to note that my files are listed as a number followed by the extension of choice, i.e. 12.txt or 12.html. If a file from the first half is generated as 12.html, the file from the second half is **also** generated with the same number. The raw html files can be found under A3/RawHTML/. The processed files can be found in .txt files in the A3/ParsedHTML/ directory.

0.2 Question 2

The first part of the question was to choose a search term and choose 10 links. Further down the question asked us to determine how many times the word shows up, as well as how many of the words appear in the document. I did all of this with a few bash commands.

A quick breakdown of these commands. The for loop iterates through every file in the directory, currently set to A3/ParsedHTML/. On each of these files, now described by $i, it tries to use grep to find all instances of Skating regardless of capital letters, all output from this stage is sent to /dev/null. In the event the command finds anything, it returns as a success, so anything after && is run. It puts all passing filenames into list1.txt, the amount of times skating shows up in Skating.txt, and the amount of total words in each file in words1.txt.

After all of this was done, I put them in a comma delimited list for easier access. Then I randomly removed all but 10 lines to match the size of the list required by problem 2. **This last part was done by hand.** Randomly selecting lines in bash is harder than I remember.

```
for i in *; do grep -i "Skating" $i >> /dev/null  && echo $i >> list1.txt && grep
    -io "Skating" $i | wc -w >> Skating.txt && wc -w $i | grep -oiE "^\S+" >>
    words1.txt; done
paste -d',' Skating.txt list1.txt words1.txt  | sort -h  > hold.txt
```

It was at this point I realized I never added the links to the documents in the list, I was just gathering them by their unique identifier, I decided to tackle this with one more set of commands. To break it down, this loop is broken into two main parts, inside the loop and outside. The outside is the more vague part to understand. I used a concept called file substitution to put the results of a command into the read command.

```
while read line; do sed "${line}q;d" ../links.txt; done < <(cat hold.txt | grep -
    oE "[0-9]+\.txt" | grep -oE "[0-9]+") > listlinks.txt
paste -d',' hold.txt listlinks.txt  > FinalList.txt
```

At the end, all the results were placed in FinalList.txt. Here are the same results, but in a more readable format. Note that this is without the links to the webpages, those will be listed further down when all we are dealing with is the scores associated with the files.

FileName	Occurance	Word Count
27.txt	3	146
62.txt	1	322
89.txt	2	189
260.txt	7	1154
400.txt	2	295
401.txt	2	284
617.txt	36	1992
725.txt	7	1348
939.txt	14	355
991.txt	15	1063

Google has around 30 trillion pages in index, found at `https://www.tennessean.com/story/money/tech/2014/05/02/jj-rosen-popular-search-engines-skim-surface/8636081/`, so I'm using this value for the total documents in the corpus. This means the IDF is found by the equation.

$$IDF = log_2(30,000,000,000,000/101,000,000) = 18.1802 \tag{1}$$

FileName	TF	IDF	TFIDF
`http://www.bbc.co.uk/news/blogs-trending-43003630?ocid=socialflow_twitter`	0.0205	18.1802	0.3727
`https://www.wired.com/story/olympic-support/`	0.0031	18.1802	0.0564
`http://www.businessinsider.com/winter-olympics-figure-skating-arena-empty-nbc-time-zone-2018-2`	0.0106	18.1802	1.9271
`https://www.cnet.com/how-to/winter-olympics-2018-when-they-start-how-to-stream-and-more/#ftag=CADf328eec`	0.0061	18.1802	0.1109
`http://www.huffingtonpost.ca/2018/02/09/olympic-mom-commercial-pg_a_23357319/?ncid=tweetlnkcahpmg00000002`	0.0068	18.1802	0.1236
`https://www.billboard.com/articles/columns/pop/8098906/winter-olympics-2018-opening-ceremony-psy-bts-skating-oasis-ed-sheeran?utm_source=twitter`	0.0070	18.1802	0.1273
`https://www.nbcnews.com/storyline/winter-olympics-2018/think-olympic-figure-skating-judges-are-biased-data-says-they-n844886`	0.0181	18.1802	0.3291
`https://www.nytimes.com/2018/02/07/watching/winter-olympics-guide-how-to-watch.html?smid=tw-nytimes&smtyp=cur`	0.0052	18.1802	0.0945
`https://www.hulu.com/nbc-winter-olympics-2018`	0.0394	18.1802	0.7163
`http://www.sheknows.com/entertainment/articles/1138091/best-olympics-ice-skating-moments-through-the-years`	0.0141	18.1802	0.2563

0.3 Question 3

This question was mostly manual labor. In fact thats all it was. I found all my results using `https://www.checkpagerank.net/index.php` . Everything was done manually, as the assignment recommended it would be.

FileName	PageRank
`http://www.bbc.co.uk/news/blogs-trending-43003630?ocid=socialflow_twitter`	0.9
`https://www.wired.com/story/olympic-support/`	0.9
`http://www.businessinsider.com/winter-olympics-figure-skating-arena-empty-nbc-time-zone-2018-2`	0.8
`https://www.cnet.com/how-to/winter-olympics-2018-when-they-start-how-to-stream-and-more/#ftag=CADf328eec`	0.8
`http://www.huffingtonpost.ca/2018/02/09/olympic-mom-commercial-pg_a_23357319/?ncid=tweetlnkcahpmg00000002`	0.7
`https://www.billboard.com/articles/columns/pop/8098906/winter-olympics-2018-opening-ceremony-psy-bts-skating-oasis-ed-sheeran?utm_source=twitter`	0.8
`https://www.nbcnews.com/storyline/winter-olympics-2018/think-olympic-figure-skating-judges-are-biased-data-says-they-n844886`	0.8
`https://www.nytimes.com/2018/02/07/watching/winter-olympics-guide-how-to-watch.html?smid=tw-nytimes&smtyp=cur`	0.9
`https://www.hulu.com/nbc-winter-olympics-2018`	0.7
`http://www.sheknows.com/entertainment/articles/1138091/best-olympics-ice-skating-moments-through-the-years`	0.7

Overall, there wasn't as much coorelation as I expected. It would probably be more obvious with more than 10 links, but I have trouble figuring out exactly what links these two together in this example. The lowest page rank websites do have the lowest TFIDF scores, but the higher ones have a large disparity. To be more specific, anything that is 0.8 or above flucuates a large amount.