Khaja Masroor Ahmed

Assignment 3

CS 751: Introduction to Digital Libraries
Dr. Michael Nelson
Spring 2015

April 4, 2015

Contents

1

Question 1

1.1 Question

- For the text you saved for the 10000 URIs from A1, Q2:
  - Use the boilerpipe software to remove the HTML templates from all HTML pages (document how many pages link from the tweets were non-HTML and had to be skipped).
  - `https://code.google.com/p/boilerpipe/`
  - WSDM 2010 paper: `http://www.l3s.de/~kohlschuetter/boilerplate/`
- For how many of the 10000 URIs was boilerpipe successful?
  - Compare the total words, unique words, and byte sizes before and after use of boilerpipe.
- For what classes of pages was it successful?
- For what classes of pages was it unsuccessful?
- Provide examples of both successful and unsuccessful removals and discuss at length.

1.2 Solution

- For this assignment I used the boilerpipe library on python.
- I installed the library on the linux machine using the command 'sudo pip install boilerpipe'.
- I saved the HTML files that I downloaded from the first assignment to my folder located at `www.cs.odu.edu/~kahmed/cs751/a3/` so I could use them as URIs to be used for the boilerpipe script.
- From the 10000 URIs I was able to download the HTML files for 9978 URIs. The files were not created for the URIs which returned a 404 response. From the remaining URIs I was able to retrieve 8476 URIs as some of them did not have any HTML data.

- From the 8476 URIs that remained after successful retrieval of HTML content, the boilerpipe was successful for 6275 URIs. Below is a comparison of the data before and after running them through the boilerpipe script.

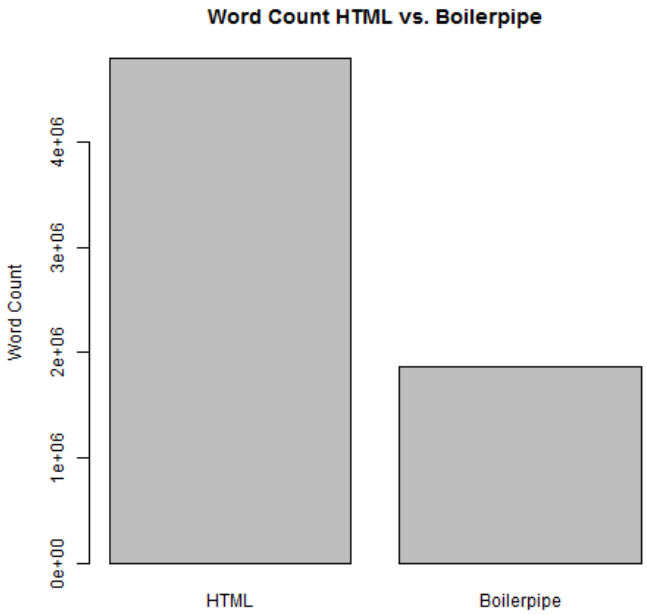

Fig. 1.1. Word Count for HTML vs. Boilerpipe

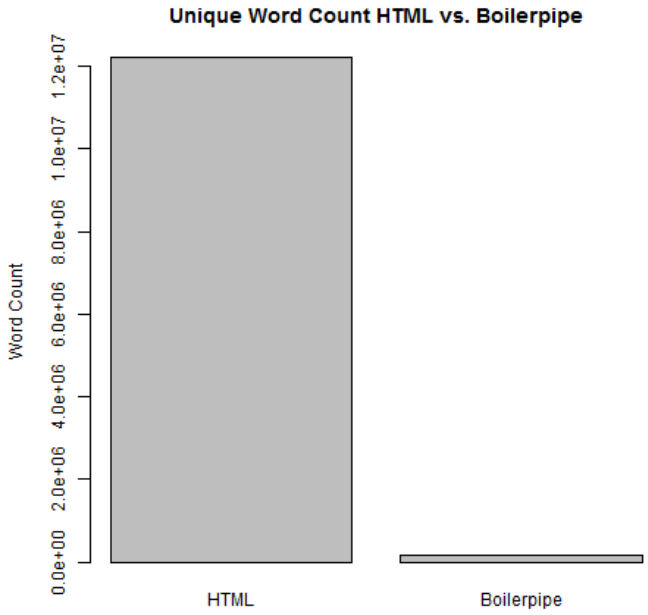

Fig. 1.2. Unique Word Count for HTML vs. Boilerpipe

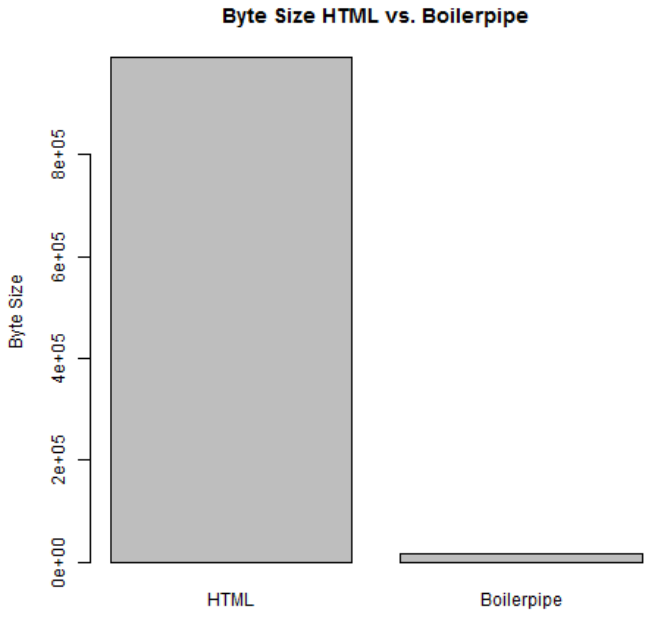

Fig. 1.3. Byte Size for HTML vs. Boilerpipe

- I used python scripts to retrieve this information for individual files and then used Microsoft Excel to get the total number of words in each case.

1.2.1 Boilerpipe Successful

- I selected a few URIs for which the boilerpipe was successful.
- A few examples of successful boilerpipe retrieval are:
  - `http://her-life-and-health.com/?a=adm`
  - `https://play.google.com/store/apps/details?id=com.maoline.kindan.droid`
- I noticed that the successful ones had HTML elements such as '¡div¿', '¡p¿', etc. with text enclosed within them.

1.2.2 Boilerpipe Unsuccessful

- I performed the same activity of selecting URIs for which boilerpipe was unsuccessful.
- A few examples of unsuccessful boilerpipe retrieval are:
  - `http://jsm084.wix.com/joy1063`
  - `http://instagram.com/p/ysl6lgFD_H/`
- Upon further investigation of these HTML pages, I came to a conclusion that the HTML pages for these URIs had only HTML script tags such as '¡script¿' within them.
- The boilerpipe is designed the ignore the data enclosed within these script tags.
- It is designed to check for data enclosed within the block elements like '¡div¿', '¡p¿', etc.
- For the example listed above, the Instagram URI ran external scripts for fetching the data to be displayed. The HTML page basically consists of scripts to be called and the necessary arguments to be passed to the script to display the necessary content which in this case was a picture and the comments for that picture from other Instagram users.
- This is applicable for the rest of the URIs for which boilerpipe was unsuccessful.

1.3 Code Listing

```
1  import json
2  from boilerpipe.extract import Extractor
3
4  f = open('status.txt', 'r+')
5  for line in f:
6          data = json.loads(line)
7          try:
8                  if data['index'] == 10001:
9                          print 'Program Executed'
10                         break
11                 finalURL=data['tweetURLData'][0]['finalURL']
12                 extractor = Extractor(extractor='
                       DefaultExtractor', url=finalURL)
13                 extracted_text = extractor.getText()
14                 link = str(data['index']) + '.txt'
15                 g = open(link, 'w')
16                 g.write(extracted_text.encode('utf-8'))
17                 g.close()
18         except:
19                 print data['index']
20                 continue
```

Listing 1.1. Python program for retrieving the boilerpipe data.

```
from bs4 import BeautifulSoup
import time

fil = open('htmlWordList.txt','w')
fileNumber = 1
wordcount = {}
while True:
        fileName = str(fileNumber) + '.html'
        try:
                f = open(fileName,'r+')
                soup = BeautifulSoup(f)
                # kill all script and style elements
                for script in soup(["script", "style"]):
                        script.extract()    # rip it out

                # get text
                text = soup.get_text()

                # break into lines and remove leading and trailing space on each
                lines = (line.strip() for line in text.splitlines())
                # break multi-headlines into a line each
                chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
                # drop blank lines
                text = '\n'.join(chunk for chunk in chunks if chunk)
                tempFile = open('tempfile.txt','w')

                tempFile.write(text.encode('utf-8'))
                tempFile.close()
                tempFileagain = open('tempfile.txt','r+')

                for word in tempFileagain.read().split():
                        word = word.translate(None, "!@#$%^&*()-=+_,.;{}[]<>?:\"`~")
                        word = word.strip()
                        word = word.lower()
                        if word not in wordcount:
                                wordcount[word] = 1
                        else:
                                wordcount[word] += 1
                tempFileagain.close()
                #print(text)
                fileNumber +=1
                if fileNumber == 10001:
                        break
```

```
44          except Exception, e:
45                  print e
46                  print fileNumber
47                  f.close()
48                  fileNumber +=1
49                  if fileNumber == 1001:
50                          break
51                  continue
52  for k,v in wordcount.items():
53          fil.write(str(k))
54          fil.write('\t')
55          fil.write(str(v))
56          fil.write('\n')
57  fil.close()
```

Listing 1.2. Python program for retrieving the unique word count from HTML.

```
import wordcount
import re
import sys

fil = open('wordCount.txt','w')
fileNumber = 1
while True:
        count = 0
        fileName = str(fileNumber) + '.txt'
        try:
                f = open(fileName,'r+')
                fil.write(fileName)
                wordcount = {}
                for word in f.read().split():
                        newword = word
                        word = word.translate(None, "!@#$
                            %^&*()-=+_,.;{}[]<>?:\"`~")
                        word = word.strip()
                        word = word.lower()
                        if word not in wordcount:
                                wordcount[word] = 1
                        else:
                                wordcount[word] += 1
                        count +=1
                fil.write('\t')
                fil.write(str(count))
                fil.write('\n')
                f.close()
                print fileNumber
                fileNumber +=1
                if fileNumber == 10001:
                        sys.exit()
        except Exception, e:
                print e
                f.close()
                fileNumber +=1
                continue
fil.close()
```

Listing 1.3. Python program for retrieving the word count for individual file.

```
import wordcount
import re
import sys

fil = open('wordList.txt','w')
fileNumber = 1
wordcount = {}
while True:
        fileName = str(fileNumber) + '.txt'
        try:
                f = open(fileName,'r+')

                for word in f.read().split():
                        word = word.translate(None, "!@#$
                            %^&*()-=+_,.;{}[]<>?:\"`~")
                        word = word.strip()
                        word = word.lower()
                        if word not in wordcount:
                                wordcount[word] = 1
                        else:
                                wordcount[word] += 1
                f.close()
                fileNumber +=1
                if fileNumber == 10001:
                        break
        except Exception, e:
                print e
                f.close()
                fileNumber +=1
                continue
for sortedValue in sorted(wordcount.values()):
        fil.write(str(sortedValue))
        fil.write('\t')
        fil.write(wordcount[sortedValue])
        fil.write('\n')
fil.close()
```

Listing 1.4. Python program for retrieving unique word list for boilerpipe successful files.

```
1 B <- c(4804616,1861950)
2 png("C:/Users/kahmed/Desktop/wordCount.png")
3 barplot(B, main="Word Count HTML vs. Boilerpipe", xlab="",
      ylab="Word Count", names.arg = c("HTML", "Boilerpipe"))
```

Listing 1.5. R program for generating the bar plot for total word count for HTML vs. Boilerpipe

```
1 B <- c(12229220,156904)
2 png("C:/Users/kahmed/Desktop/uniqueWord.png")
3 barplot(B, main="Unique Word Count HTML vs. Boilerpipe",
      xlab="", ylab="Word Count", names.arg = c("HTML", "
      Boilerpipe"))
```

Listing 1.6. R program for generating the bar plot for unique word count for HTML vs. Boilerpipe

```
1 B <- c(993280,16076.8)
2 png("C:/Users/kahmed/Desktop/byteSize.png")
3 barplot(B, main="Byte Size HTML vs. Boilerpipe", xlab="",
      ylab="Byte Size", names.arg = c("HTML", "Boilerpipe"))
```

Listing 1.7. R program for generating the bar plot for byte size of HTML vs. Boilerpipe

2

Question 2

2.1 Question

- Collection1: Extract all the unique terms and their frequency from the 10000 files*.
- Collection2: Extract all the unique terms and their frequency of the 10000 files* after running boilerpipe.
- Construct a table with the top 50 terms from each collection.
  - Find a common stop word list. How many of the 50 terms are on that stop word list?
- For both collections, construct a graph with the x-axis as word rank, and y-axis as word frequency.
  - Do either follow a Zipf distribution? Support your answer.

2.2 Solution

- I ordered the word list that I received as an output from the previous question and then got the highest frequency word list.
- I fetched the stop word list by searching for it on `www.google.com` and then compared the top 50 results with the stop word list.
- For HTML files before running boilerpipe, there were 43 common words with the stop word list.
- After running boilerpipe, there were 44 common words with the stop word list.
- Below are tables indicating the results for the high frequency word list for HTML and boilerpipe.

Table 2.1. Collection 1: Word Rank and Frequency before boilerpipe

Rank	Word	Frequency	Stop Word
1	a	484037	Yes
2	the	137978	Yes
3	to	115694	Yes
4	and	96843	Yes
5	in	73996	Yes
6	of	70699	Yes
7	for	49243	Yes
8	on	44144	Yes
9	this	38794	Yes
10	is	38490	Yes
11	with	37402	Yes
12	by	34809	Yes
13	you	33828	Yes
14	your	33537	Yes
15	that	21535	Yes
16	all	20933	Yes
17	from	19210	Yes
18	it	18456	Yes
19	at	18300	Yes
20	are	17710	Yes
21	be	17238	Yes
22	or	15885	Yes
23	as	15391	Yes
24	will	15035	Yes
25	important	14690	No
26	no	13703	Yes
27	an	12070	Yes
28	was	12054	Yes
29	more	12026	Yes
30	have	11805	Yes
31	do	11350	Yes
32	about	9915	Yes
33	out	9791	Yes
34	arabic word	9647	No
35	right	9459	No
36	we	9303	Yes
37	our	9213	Yes
38	has	9100	Yes
39	my	8601	Yes
40	l	8579	Yes
41	one	8271	No
42	only	8162	Yes
43	me	7560	Yes
44	can	7497	No
45	but	7480	Yes
46	he	7264	Yes
47	his	7163	Yes
48	when	7124	Yes
49	arabic word	6946	No
50	us	6914	No

Table 2.2. Collection 2: Word Rank and Frequency after boilerpipe

Rank	Word	Frequency	Stop Word
1	the	54339	Yes
2	to	35095	Yes
3	a	32042	Yes
4	and	30095	Yes
5	of	24486	Yes
6	in	18309	Yes
7	is	14296	Yes
8	you	12979	Yes
9	play	12961	No
10	for	11707	Yes
11	that	11592	Yes
12	this	11315	Yes
13	on	10768	Yes
14	i	9818	Yes
15	it	9112	Yes
16	now	8817	Yes
17	with	8687	Yes
18	your	8444	Yes
19	next	7133	No
20	by	6724	Yes
21	are	6502	Yes
22	as	6414	Yes
23	be	6114	Yes
24	have	5681	Yes
25	was	5613	Yes
26	or	5553	Yes
27	not	5477	Yes
28	at	4937	Yes
29	from	4894	Yes
30	we	4758	Yes
31	no	4727	Yes
32	more	4588	Yes
33	has	4064	Yes
34	but	3994	Yes
35	an	3969	Yes
36	all	3781	Yes
37	will	3780	No
38	they	3587	Yes
39	can	3585	Yes
40	if	3272	Yes
41	he	3263	Yes
42	up	3260	No
43	about	3243	Yes
44	do	3084	Yes
45	arabic word	3040	No
46	our	3010	Yes
47	when	2943	Yes
48	my	2875	Yes
49	been	2839	Yes
50	arabic word	2755	No

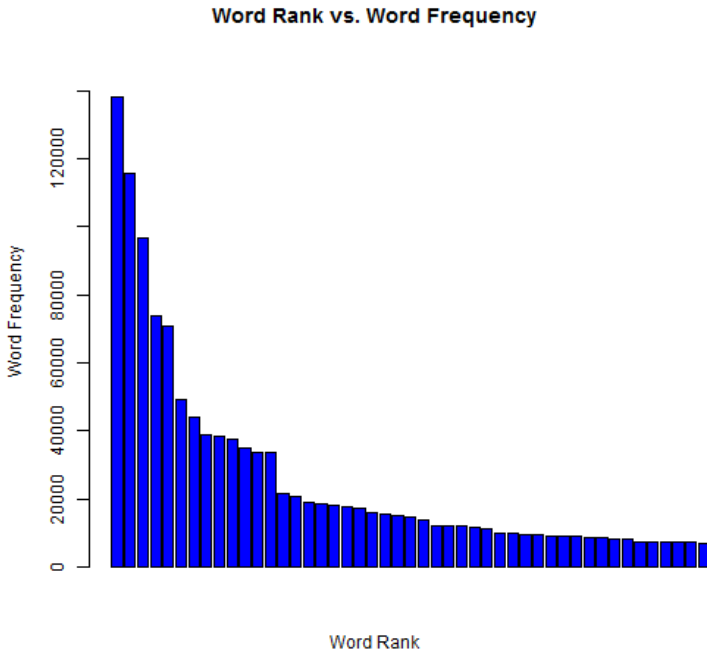

Fig. 2.1. Distribution of data for high frequency words before boilerpipe.

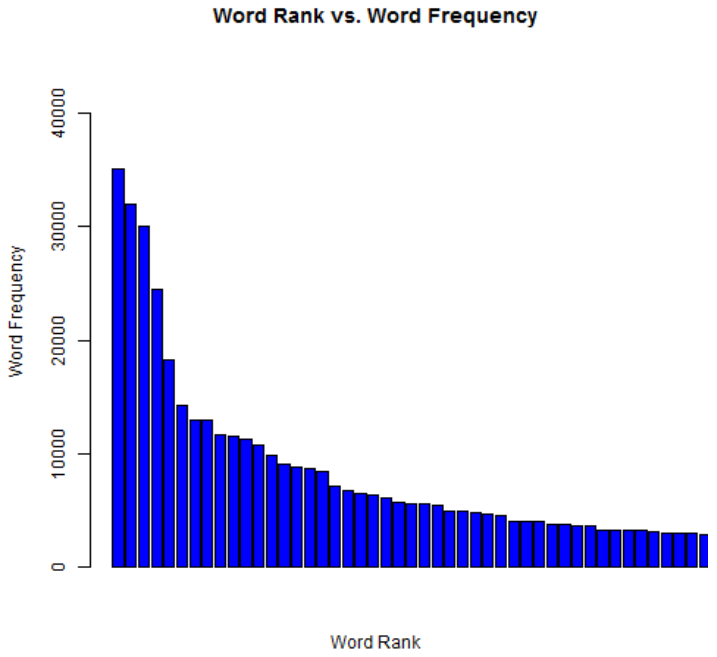

Fig. 2.2. Distribution of data for high frequency words after boilerpipe.

- The above graphs indicate that they follow the zipf distribution as the word rank is inversely proportional to its frequency.

2.3 Code Listing

```
1 dp <- read.table('c:/users/kahmed/desktop/html.txt', header=
      T)
2 datapoint <- rep(dp[,1])
3 xlimit <- c(0,50)
4 ylimit <- c(0,150000)
5 png("C:/Users/kahmed/Desktop/beforeBP.png")
6 barplot(datapoint, col='blue',xlim=xlimit,ylim=ylimit, xlab=
      'Word Rank', ylab='Word Frequency', main='Word Rank vs.
      Word Frequency')
```

Listing 2.1. R program for generating the bar plot for high frequency words before boilerpipe.

```
1 dp<- read.table('c:/users/kahmed/desktop/boilerpipe.txt',
      header=T)
2 datapoint <- rep(dp[,1])
3 xlimit <- c(0,50)
4 ylimit <- c(0,45000)
5 png("C:/Users/kahmed/Desktop/afterBP.png")
6 barplot(datapoint, col='blue',xlim=xlimit,ylim=ylimit, xlab=
      'Word Rank', ylab='Word Frequency', main='Word Rank vs.
      Word Frequency')
```

Listing 2.2. R program for generating the bar plot for high frequency words after boilerpipe.

References

1. Boilerpipe in python. https://github.com/misja/python-boilerpipe.
2. Draw bar plots in R. http://www.statmethods.net/graphs/bar.html.
3. Draw tables in LaTex. http://en.wikibooks.org/wiki/LaTeX/Tables.
4. Python sort dictionary by key. http://stackoverflow.com/questions/9001509/how-can-i-sort-a-python-dictionary-sort-by-key.
5. Stop word list. http://www.ranks.nl/stopwords.
6. Zipf's law. http://en.wikipedia.org/wiki/Zipf