Introduction to Web Science: Assignment #3

Dr. Nelson

Alexander Nwala

Thursday, April 2, 2015

Contents

Problem 1

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.

SOLUTION 1

The solution for this problem is outlined by the following steps:

Remove boilerplate from HTML pages: Due to Listing 1, the HTML templates derived after dereferencing the URIs were removed with justtext [1]

Listing 1: Remove Boilerplate

```
#Remove boilerplate snippet
def extractText(listOfURIs):
    ...
    for i in range(0, len(listOfURIs) ):

        try:
            URI = listOfURIs[i].split(', ')[1].strip()
            print i, URI

            page = urllib2.urlopen(URI).read()
            paragraphs = justext.justext(page, justext.get_stoplist('English'))


            for paragraph in paragraphs:
                if paragraph['class'] == 'good':
                    processedText = paragraph['text']

                    processedText = processedText.encode('ascii', 'ignore')
                    outputFile.write(URI + ':\n' + processedText + '\n\n')
                else:
                    print 'class:', paragraph['class']

                    if( URI in alreadyAddedList ):
                        pass
                    else:
                        statOutputFile.write(URI + ' bad\n\n')
                        alreadyAddedList.append(URI)

                    print
```

```
30          except:
                statOutputFile.write(URI + ': ERROR\n\n')
                exc_type, exc_obj, exc_tb = sys.exc_info()
                fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
                print(fname, exc_tb.tb_lineno, sys.exc_info() )
35              print
    outputFile.close()
    statOutputFile.close()
...
```

Consider the following remarks: Given the list of 10,000 URIs, only 2,164 were unique URIs from A2 - I believe this was caused due to overlap in my search results from Twitter (in retrospect, my algorithm to retrieve URIs should not have used search).

Given my initial list of 2,164 URIs, 1,132 were skipped due to ERRORs ranging from 404s to corrupt URIs as seen by the following examples of HTTP response codes:

```
$ curl -I https://api.twitter.com/oauth/authorize?oauth_token
=8bwrg25dbghw8aszsfc0wul3q6gxswey
HTTP/1.1 403 Forbidden

$ curl -I http://click.linksynergy.com/link?id=vdg03e6l9pw&offerid=
210072.9001235091&type=2&murl=
http://www.animate-onlineshop.jp/products/detail.php%3fproduct_id%3d1235091
HTTP/1.1 400 Bad Request

$ curl -I http://p0p.pw/erh
HTTP/1.1 404 Not Found

$ curl -I http://raumu.ciao.jp/yyc/
HTTP/1.1 404 Not Found

$ curl -I http://butta.info/u1f3rd/
curl: (6) Could not resolve host: butta.info
```

There were 60 successful text extraction operations from the list of URIs.

Because of the low success rate, I began investigating a possible cause by randomly sampling 15 URIs. I consider the following results below, a justification for the low success rate.

```
5 URIs: links of Google sites with no content which required a
click to redirect
5 URIs: Japanese sites ranging from blogs to sites with no content
3 URIs: 2 Spanish news sites which required clicks to see content,
1 Facebook site
2 URIs: 1 image and one was unavailable (404)
```

Consider the following statistic collected from 60 successful text extraction operations:

```
Total words before removal: 549,788
Total words after removal: 31,135

Total unique words before removal: 51,044
Total unique words after removal: 7,819

Total size (bytes) before removal: 31,010,041
Total size (bytes) after removal: 192,437
```

The classes of pages for which boilerplate removal was successful includes: organized pages with well defined/extensive text areas such as blogs and pages with comment sections. Overall, pages in which the template was on the left and right with text in the middle were successfull. Consider the following examples (charts 1 - 3) of successful removals (green boxes) based on the format described.

```
Chart Example 1 URI:
https://www-304.ibm.com/connections/blogs/socialbusiness/
entry/you_are_sitting_on_a_volcano_that_is_ready_to_blow?lang=en_us

Chart Example 2 URI
http://www.thedailybeast.com/articles/2015/02/03/average-
soldiers-don-t-trust-their-generals-and-they-have-a-point.html

Chart Example 3 URI
http://prod.www.giants.clubs.nfl.com/news-and-blogs/article-
1/know-your-giants-te-larry-donnell-/dbaf4b75-cb7c-40f3-b326-
22a5600aebbd?utm_source=dlvr.it&utm_medium=twitter
```

Chart Example 1: Successful removal

Chart Example 2: Successful removal

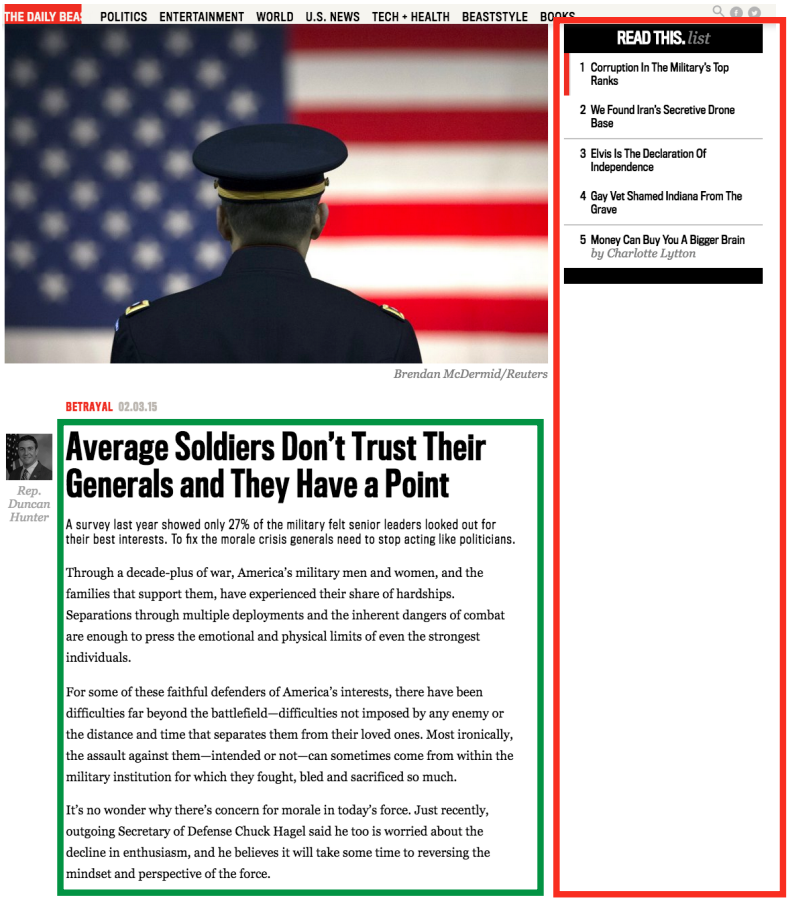


Chart Example 3: Successful removal

The classes of pages for which boilerplate removal was unsuccessful includes: Pages with disorganized format (no well defined template) in which text was spread accross template as well as pages with unconventional layouts:

```
Chart Example 4 URI:
http://www.stocktradepartner.com/

Chart Example 5 URI:
http://www.fandango.com/dawnoftheplanetoftheapes_156265/movieo
verview?wssaffid=11838&wssac=123&cjid=cj_10576763_7644471_

Chart Example 6 URI:
http://www.yesasia.com/us/you-who-came-from-the-stars-blu-ray-
vol-2-japan-version/1037741202-0-0-0-en/info.html
```

Chart Example 4: Unsuccessful removal

Chart Example 5: Unsuccessful removal

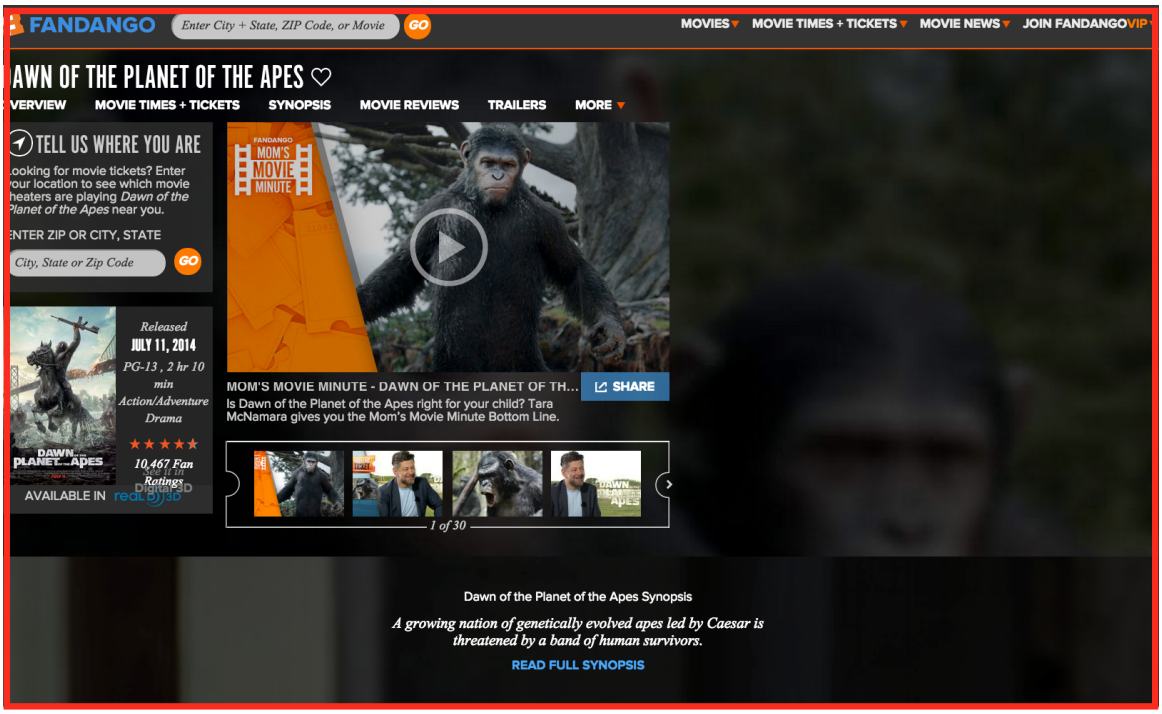


Chart Example 6: Unsuccessful removal

Problem 2

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.

SOLUTION 2

The solution for this problem is outlined by the following steps:

Extract unique terms: Due to Listing 2, the unique terms were retrieved by splitting the string of all the HTML files on the space character. Subsequently Listing 2. kept track of the term and term frequencies in a dictionary.

Listing 2: Get Unique Terms

```
# Get unique terms from HTML files and plaintext (after boilerplate removal)
def getTermsFromPlaintext():

    page = ''
    wordFrequencyDict = {}
    try:
        inputFile = open('plaintext.txt', 'r')
        page = inputFile.read()
        inputFile.close()
    except:
        exc_type, exc_obj, exc_tb = sys.exc_info()
        fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(fname, exc_tb.tb_lineno, sys.exc_info() )

    getWordFrequencyDict(page, wordFrequencyDict)
    wordFrequencyDict = sorted(wordFrequencyDict.items(), key=lambda x:x[1], reverse=
        True)

    print
    print 'begin'

    #total words
    totalWords = 0
    for tup in wordFrequencyDict:
        totalWords = tup[1] + totalWords

    print 'totalWords:', totalWords
    print 'totalWordsUnique:', len(wordFrequencyDict)
    '''
    # top 50 terms - start
```

```
	count = 0
	for tup in wordFrequencyDict:
		count = count + 1

		print tup[0].strip(), tup[1]

		if count == 50:
			break
	# top 50 terms - end
	'''

def getUniqueTermsFromHTML():

	lines = []
	try:
		inputFile = open('justURLs.txt', 'r')
		lines = inputFile.readlines()
		inputFile.close()
	except:
		exc_type, exc_obj, exc_tb = sys.exc_info()
		fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
		print(fname, exc_tb.tb_lineno, sys.exc_info() )

	wordFrequencyDict = {}
	instanceCount = 0
	for l in lines:
		print instanceCount, len(wordFrequencyDict)
		l = l.strip()
		page = ''
		try:
			page = urllib2.urlopen(l).read()
		except:
			#print 'Error: ', l
			exc_type, exc_obj, exc_tb = sys.exc_info()
			fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
			print(fname, exc_tb.tb_lineno, sys.exc_info() )

		getWordFrequencyDict(page, wordFrequencyDict)
		#print len(wordFrequencyDict)
		instanceCount = instanceCount + 1

	wordFrequencyDict = sorted(wordFrequencyDict.items(), key=lambda x:x[1], reverse=
	    True)
	print
	print 'begin'

	#total words
	totalWords = 0
	for tup in wordFrequencyDict:
		totalWords = tup[1] + totalWords

	print 'totalWords:', totalWords
	'''
```

```
    # top 50 terms - start
    count = 0
    #print wordFrequencyDict
    for tup in wordFrequencyDict:
        count = count + 1

        print tup[0].strip(), tup[1]

        if count == 50:
            break
    # top 50 terms - end
    '''
```

Construct table: Consider Table 1 and 2.

Find terms which coincide in Stopwords list: 43 terms from Table 2. (Plaintext files) coincided with a stopwords list derived from http://www.ranks.nl/stopwords. Also **16** terms from Table 1. (HTML files) coincided with the same list.

```
The file stopwords.txt contains the stopwords.
```

Plot chart: Due to Listing 3, the terms and term frequencies pre/post boilerplate removal was plotted. As seen in Chart Example 7, both charts follow the Zipfian distribution [2]: The frequency of each word is approximately inversely proportional to its rank in the frequency table - "power law." This means the most frequent word occurs approximately twice as often as the second most frequent word and so on.

Listing 3: Plot Term Frequencies

```
#!/usr/bin/env Rscript


TermTermFrequencyP <- read.table('plaintextTop50Terms.txt', header=T)

TermTermFrequencyH <- read.table('rawHTMLTop50Terms.txt', header=T)

plot(TermTermFrequencyP$Frequency, type='o', col='blue', xlab='Word Rank', ylab='Word
    Frequency', main='Distribution of terms in HTML files pre boilerplate \nremoval (
    red) and post boilerplate removal (blue)')

#plot(TermTermFrequencyH$Frequency, type='o', col='blue', xlab='Word Rank', ylab='Word
     Frequency', main='Distribution of terms in HTML files')
lines(TermTermFrequencyH$Frequency, type='o', col='red')
```

Table 1: Top 50 Terms Extracted From HTML Files

Item	Term	Frequency
1	<div	2924
2	=	2033
3	the	1944
4	<a	1819
5	to	1556
6	and	1503
7	a	1247
8	of	1094
9	{	1085
10	<a	971
11	<span	953
12		871
13	in	834
14	{	792
15	for	767
16	-	692
17	<li	682
18	at	661
19	var	579
20	point:false,	574
21	on	561
22	+	557
23		555
24	is	550
25	target="_blank"	547
26	B	529
27	position:"left"},	506
28	you	460
29	your	459
30		446
31	class="b-link"	442
32	with	428
33	I	419
34	blog-admin	401
35	target='_self'	400
36	&	392
37	2014	389
38		376
39	December	373
40	0	372
41	rel="nofollow"	368
42	<img	355
43	onclick="return	352
44	if	350
45	nofollow:false,tab_target:false,	348
46	class="b-ico"	337
47	type="hidden"	333
48	The	322
49	}	305
50	class="b-link	296

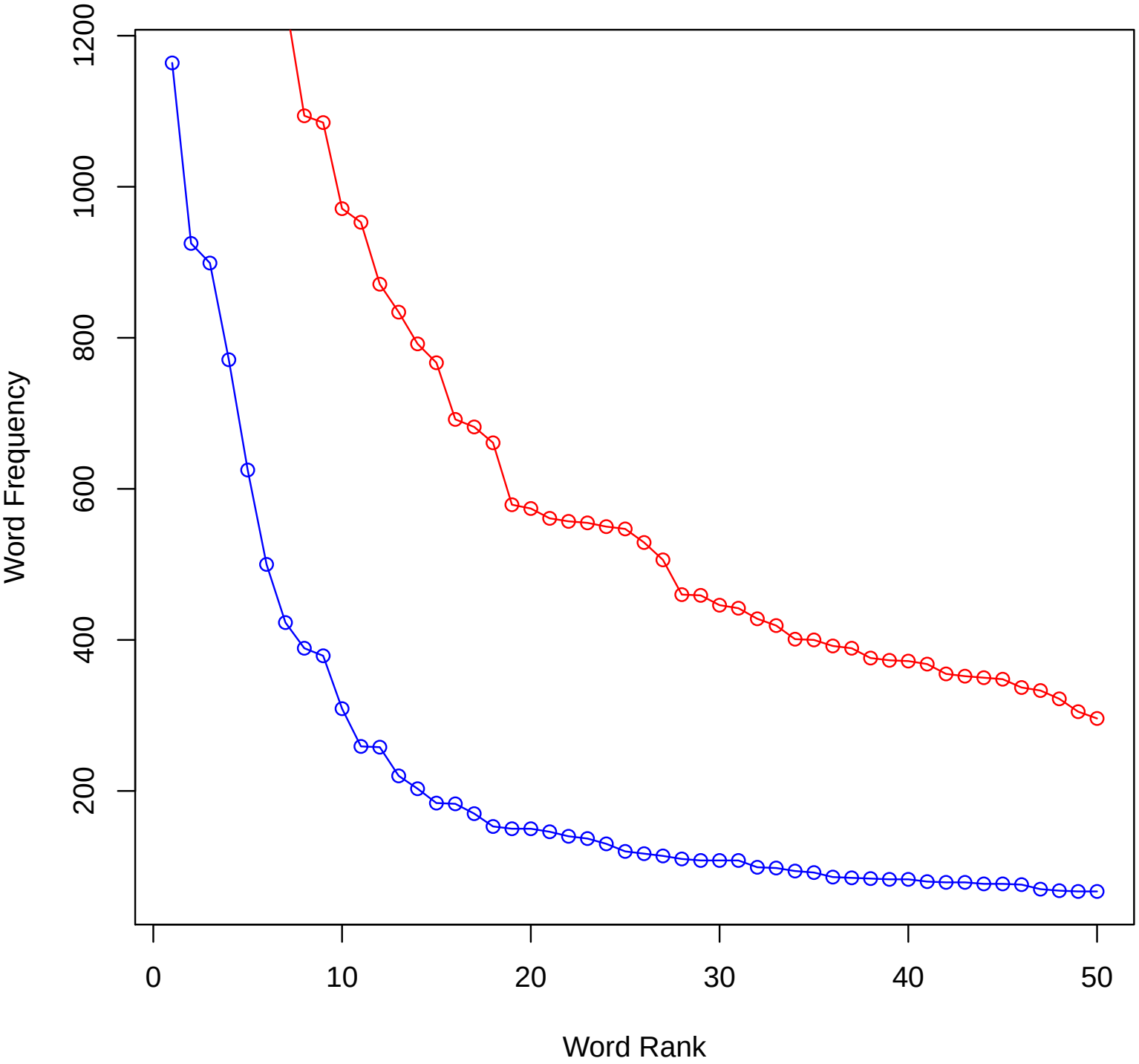
Problem 2 continued on next page...

Table 2: Top 50 Terms Extracted Post Boilerplate Removal

Item	Term	Frequency
1	the	1164
2	and	925
3	to	899
4	a	771
5	of	625
6	in	500
7	is	423
8	you	389
9	I	379
10	for	309
11	with	259
12	that	258
13	your	220
14	on	203
15	was	184
16	are	183
17	or	170
18	it	153
19	at	150
20	be	150
21	as	146
22	this	140
23	by	137
24	will	130
25	have	120
26	can	117
27	from	114
28	her	110
29	not	108
30	my	108
31	all	108
32	an	99
33	when	98
34	The	94
35	get	92
36	we	86
37	about	85
38	domain	84
39	so	83
40	our	83
41	like	80
42	he	79
43	but	79
44	up	77
45	his	77
46	just	76
47	one	70
48	who	68
49	has	67
50	if	67

Problem 2 continued on next page...

Chart Example 7: Distribution of terms in HTML files pre boilerplate removal (red) and post boilerplate removal (blue)



References

[1] jusText. https://pypi.python.org/pypi/jusText/2.0.0. Accessed: 2015-04-01.

[2] The mystery of Zipf. https://plus.maths.org/content/mystery-zipf. Accessed: 2015-04-01.