CS 751: Introduction to Digital Libraries - Assignment 3

Jessica McConnell

April 3, 2015

1 Q1

For this question we had to set up boilerpipe and run our 10,000 body files we downloaded from assignment 1 through it to remove all html templates.

I used the boilerpipe application suggested in the Assignment 3 powerpoint by Christian Kohlschtter. [2]

1.1 Picking Files

As a reminder from assignment 1, I only downloaded the final unique URIs after all redirects from the original 10000 unique twitter URIs. This did not give me 10,000 final unique pages. Instead, it was 7405 final unique pages. I also was unable to download 477 of those pages using wget for assignment 1. In my testing of boilerpipe I learned that the boilerpipe method did not accept files that were .htm files so I threw those out. I only ran the 6810 files that ended in .html through boilerpipe.

I used the below command to get the .html uris.

```
for line in ‘ls -d */‘; do
    item=$(ls $line);
    if [[ -z "$item" ]]; then
        echo $line "empty";
    elif [[ "$item" == *.html ]]; then
        echo $line$item" Good";
    else echo $line$item" Bad";
    fi;
done > files
```

1.2 Running Files

I wrote a java program to perform my boilerpipe tasks called ‘A3.java’. It takes all the file names for the uris in the ‘GoodFiles’ file and runs the downloaded html for the uri through boilerpipe using the ArtifactExtract function. I compiled and ran my java file with the following 2 commands:

```
javac -d . -cp .:./boilerpipe-1.2.0/boilerpipe-1.2.0.jar: -
./boilerpipe-1.2.0/nekohtml-1.9.13.jar:./boilerpipe-1.2.0/ -
xerces-2.4.0.jar:./boilerpipe-1.2.0/* A3.java

java -cp .:./boilerpipe-1.2.0/boilerpipe-1.2.0.jar: -
./boilerpipe-1.2.0/nekohtml-1.9.13.jar:./boilerpipe -
-1.2.0/xerces-2.4.0.jar:./boilerpipe-1.2.0/* A3
```

I received exceptions on four files. Three of the files had a non-english character in the name and the java application could not find it. I was able to find the fourth file in the directory; however, I could not cat the file directly. There must be something wrong with the file for the fourth exception received. The exceptions are below.

```
Exception in thread "main" java.io.FileNotFoundException: -
../bodies/328/mushroom-networks-adds-voip-armor   -
%84   -its-truffle-broadband-bonding %84   -network-appliance.html
(No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:146)
        at java.io.FileInputStream.<init>(FileInputStream.java:101)
        at java.io.FileReader.<init>(FileReader.java:58)
        at A3.main(A3.java:36)


Exception in thread "main" java.io.FileNotFoundException: -
../bodies/5805/saddle-   %80%94-markens-charles.html -
(No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:146)
        at java.io.FileInputStream.<init>(FileInputStream.java:101)
        at java.io.FileReader.<init>(FileReader.java:58)
        at A3.main(A3.java:36)

Exception in thread "main" java.io.FileNotFoundException: -
../bodies/6262/what-do-you-mean-there %80%99s-child-slave -
-labor-my-chocolate.html (No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:146)
        at java.io.FileInputStream.<init>(FileInputStream.java:101)
        at java.io.FileReader.<init>(FileReader.java:58)
        at A3.main(A3.java:36)

Exception in thread "main" java.io.FileNotFoundException: -
m./bodies/999/index.html?utm_source=twitterfeed&utm_medium= -
twitter.html (No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:146)
        at java.io.FileInputStream.<init>(FileInputStream.java:101)
        at java.io.FileReader.<init>(FileReader.java:58)
        at A3.main(A3.java:36)

jmcconne@sirius:~/cs751/a3$ ls ../bodies/999/
index.html?utm_source=twitterfeed&utm_medium=twitter.html
jmcconne@sirius:~/cs751/a3$ cat ../bodies/999/index.html? -
    utm_source=twitterfeed&utm_medium=twitter.html
cat: ../bodies/999/index.html?utm_source=twitterfeed: -
    No such file or directory
```

1.3 Unsuccessful Documents After Boilerpipe

After running boilerpipe, I had 329 files that returned with no size. Doing a random sampling of the empty files they appear to consist of blogs that only contain photos, iTunes pages, enlarged pictures and sites that used a smaller URI that translated to a longer URI. For example `smarturl.it` links.

A few of the links it was unsuccessful for:
Pictures only:
`http://rcobanus.dailyfunnypics.me/young-boy-returns-home-with-a-new-porsche-this\`
`-is-priceless`
`http://newzcard.com/card/mBN5VP/singers-usher-l-and-rihanna-arrive-at-the-2010\`
`-american-music-awards-held-at-nokia-theatre-l-a-live(mBN5VP)?r=UsherPics`

Shortened URLs:
`http://smarturl.it/FourFiveSeconds`
`http://smarturl.it/NBBiTunesDLX`
`http://nblo.gs/133ljo`

For my sampling of unsuccessful pages, the original sizes for the html download varied between 2000 and 6000 bytes. The HTML web pages also varied from 50 unique words to approximately 250 unique words.

1.4 Successful Documents After Boilerpipe

For my sampling of successful web pages they included:
1. a login page for the New York Times `https://myaccount.nytimes.com/auth/login?URI=http%3A%2F%2Fwell.blogs.nytimes.com%2F2015%2F01%2F06%2Fjunk-food-in-the-new-year%2F%3Fpartner%3Drss%26emc%3Drss%26utm_content%3Dbuffera524f%26utm_medium%3Dsocial%26utm_source%3Dtwitter.com%26utm_campaign%3Dbuffer%26_r%3D5&REFUSE_COOKIE_ERROR=SHOW_ERROR`
2. a Twitter post `https://twitter.com/BernieceBryon/status/561189374608420864`
3. a website with a very common page layout `http://www.usafestival.net/?page=details&id=7117`
4. an article from the huffingtonpost `http://www.huffingtonpost.com/2015/01/30/super-bowl-trivia_n_6543044.html?utm_hp_ref=religion&ir=Religion&utm_medium=twitter&utm_source=twitterfeed`

The sizes of the web pages that were successful was much larger than the unsuccessful pages with a range from 30,000 bytes to 330,000 bytes. The unique words before boiler pipe was applied ranged from 600 to just over 9000 words. After boiler pipe was applied the unique words ranged from 20 to 8000 words. This change is indicative of websites that use common html website layouts that the boilerpipe tool can easily identify. That is easily expected just by looking at the random sample of URIs I used for my successful documents. Three out of the four documents originated from very popular websites with a defined layout that includes changing content and not just pictures.

2 Q2

For this question we were to get all unique words from the files before and after boilerpipe was applied. We had to graph the word rank vs word frequency and compare it to a Zipf distribution. We also had to apply a stop word list to the most common words.

2.1 Getting the Unique Words

I wrote two python scripts to get the unique words and count for me. They are called 'textStats.py' and 'bodyStats.py'. They essentially performed the same actions with some brief differences in how the files were found and character exceptions. To try to get the most accurate unique word assessment I removed all colons, question marks, exclamation points, commas and periods from the ends of words. I also took out all new lines and tabs. Specifically for the html documents I also removed curly brackets, single character words that were not letters, slashes and angle brackets. I used the python Counter function to get my word rank.

TEXT COMMAND:

```
python ./textStats.py ./bodyFiles/
```

HTML COMMAND:

```
python ./bodyStats.py ./GoodFiles
```

HTML		Text	
Word	Count	Word	Count
div	2278100	the	91917
a	1771169	to	50881
span	1274771	and	50490
li	910908	of	42374
script	312721	a	41382
the	286658	in	31828
td	285403	is	21936
p	254866	for	21330
to	222278	you	17413
and	189165	on	16823
ul	185602	that	16523
img	183164	this	13324
tr	174607	with	12768
meta	167120	it	12479
of	147523	or	11440
i	141436	your	10538
in	141426	are	10212
option	127241	be	10168
button	105802	as	9749
for	104511	at	9711
h3	104190	i	9430
var	99351	by	8798
b	95416	have	8094
false	90511	from	7598
br	90297	was	7574
link	88180	not	7165
type="text/javascript"	82765	an	7012
on	82258	will	6954
input	79806	we	6809
table	74401	new	5854
is	74079	if	5531
this	72885	but	5485

Continued from previous page

HTML		Text	
Word	Count	Word	Count
your	70506	more	5334
you	69909	has	5333
if	64528	can	5179
with	61316	all	5076
target="_blank"	60132	they	4815
new	57480	about	4431
h2	55091	shipping	4409
null	52895	one	4398
by	50978	their	4199
strong	48891	out	4173
label	48097	he	4082
it	47847	our	4059
type="button"	43248	get	4037
or	42528	time	4003
all	41396	other	3946
function	39852	when	3907
rel="nofollow"	39702	my	3659
that	38959	up	3649

2.2 Stop Words

I chose to use the stopwords list that is a Default English stopwords list. I found the list at Ranks NL [1]. Below are the words in the list.

Stop Words				
a	don't	in	she'll	we
about	down	into	she's	we'd
above	during	is	should	we'll
after	each	isn't	shouldn't	we're
again	few	it	so	we've
against	for	it's	some	were
all	from	its	such	weren't
am	further	itself	than	what
an	had	let's	that	what's
and	hadn't	me	that's	when
any	has	more	the	when's
are	hasn't	most	their	where
aren't	have	mustn't	theirs	where's
as	haven't	my	them	which

Continued from previous page

Stop Words				
at	having	myself	themselves	while
be	he	no	then	who
because	he'd	nor	there	who's
been	he'll	not	there's	whom
before	he's	of	these	why
being	her	off	they	why's
below	here	on	they'd	with
between	here's	once	they'll	won't
both	hers	only	they're	would
but	herself	or	they've	wouldn't
by	him	other	this	you
can't	himself	ought	those	you'd
cannot	his	our	through	you'll
could	how	ours ourselves	to	you're
couldn't	how's	out	too	you've
did	i	over	under	your
didn't	i'd	own	until	yours
do	i'll	same	up	yourself
does	i'm	shan't	very	yourselves
doesn't	i've	she	was	
doing	if	she'd	wasn't	

I created a python script named 'wordCompare.py' to perform the comparisons between the two lists. I used the commands below to get my results.

HTML:

```
python wordCompare.py htmlFilesWords stoplist
```

Text:

```
python wordCompare.py textFilesWords stoplist
```

From the Text list there were 43 words in the stop list. In the HTML list there were only 20 words in the stop list. The words remaining in each list can be seen in the table below.

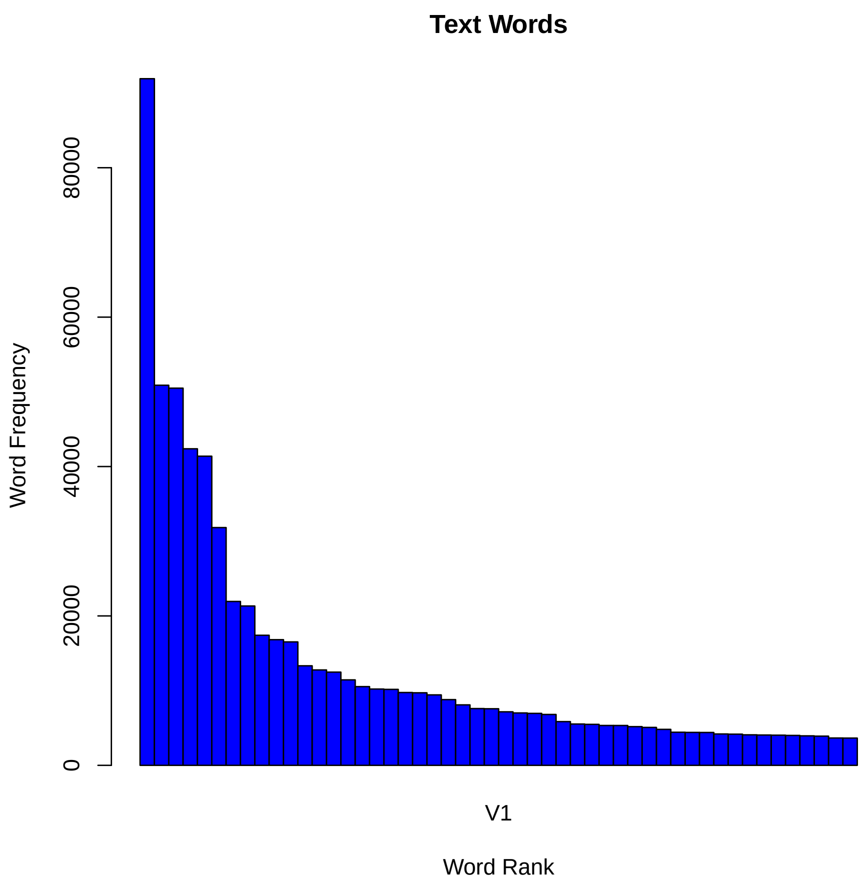
After Stop Words	
Text	HTML
will	div
new	span

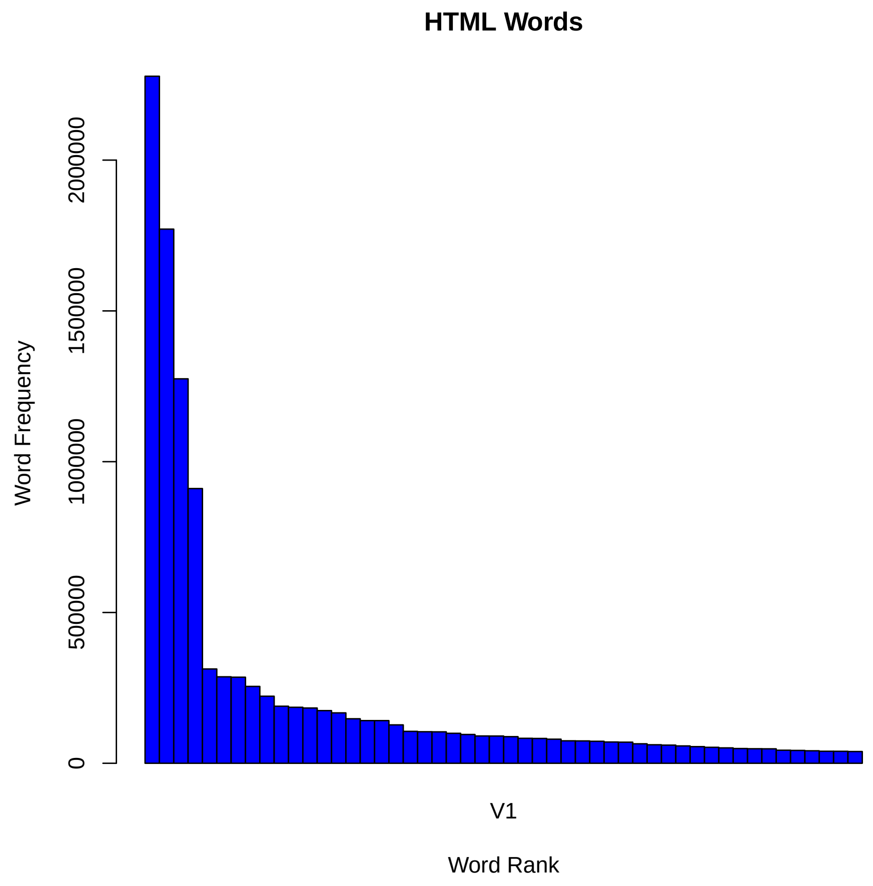
Continued from previous page

After Stop Words	
Text	HTML
can	li
shipping	script
one	td
get	p
time	ul
	img
	tr
	meta
	option
	button
	h3
	var
	b
	br
	link
	type="text/javascript"
	input
	table
	target="_blank"
	new
	h2
	null
	strong
	label
	false
	type="button"
	function
	rel="nofollow"

2.3 Graphs

Both graphs follow a zipf distribution. A zipf distribution has a small population of very popular items followed by a much larger population of less popular items. The graph looks similar to a reverse log graph.


Text Words
Word Frequency
0
20000
40000
60000
80000
V1
Word Rank



References

[1] Damian Doyle. *Stopwords*. URL: `www.ranks.nl/stopwords`.

[2] Christian Kohlschtter. *Boilerpipe*. 2011. URL: `https : / / code . google . com / p / boilerpipe/`.