Introduction to Digital Libraries CS-751: Assignment #3

Due on Friday, April 3, 2015

Michael L. Nelson 4:20pm

Avinash Gosavi

Contents

Question 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.

Answer

From A1, 11191 URI with response code 200 were used for A3.
With wget data for 11,137 URI was fetched successfully.
About 67713 unique words were found. Total size of the data was about 1.38 GB for the files downloaded.

Code Listing

fetchWebpages2.py

```
import subprocess
import os
import csv
import datetime
import pymongo
from pymongo import MongoClient
import requests
import justext

client = MongoClient()
db = client.url_of_set_http

def fetchWebPage(url, fileName):
  print 'Fetching ', url
  subprocess.Popen(["wget","--output-document=" + fileName, url]) # + ".html"

sites = 'sitesq2'
if not os.path.exists(sites):
    os.makedirs(sites)

os.chdir(sites)

print datetime.datetime.now()
tweets = db.tweet_data.find({"status_code": 200}).skip(0)
for tweet in tweets:
  if not os.path.exists(str(tweet['tweet_id'])):
    fetchWebPage(str(tweet['url']), str(tweet['tweet_id']))
  else:
    print "skipped"+'-'+str(tweet['tweet_id'])

print datetime.datetime.now()
```

Listing 1: Before Boilerpipe

With JusText data for 11062 URI was fetched successfully. But, only 4312 had content. About 63365 unique words were found. Total size of the data was about 26 MB for the files downloaded.

Code Listing

fetchWebpages.py

```
import subprocess
import os
import csv
import datetime
import pymongo
from pymongo import MongoClient
import requests
import justext

client = MongoClient()
db = client.url_of_set_http

def fetchWebPage(url, fileName):
  print 'Fetching ', url
  opt = open(fileName+".txt", "w")
  response = requests.get(url)
  try:
    paragraphs = justext.justext(response.content, justext.get_stoplist("English"))
  except Exception:
    paragraphs = []
  for paragraph in paragraphs:
    if not paragraph.is_boilerplate:
      opt.write(paragraph.text.encode("utf-8"))

  opt.close()

sites = 'sitesq'
if not os.path.exists(sites):
    os.makedirs(sites)

os.chdir(sites)

print datetime.datetime.now()
tweets = db.tweet_data.find({"status_code": 200}).skip(0)
for tweet in tweets:
  if not os.path.exists(str(tweet['tweet_id'])+".txt"):
    fetchWebPage(str(tweet['url']), str(tweet['tweet_id']))
  else:
    print "skipped"+'-'+str(tweet['tweet_id'])

print datetime.datetime.now()
```

Listing 2: After Boilerpipe

Classes of pages that were not successful :-

https://twitter.com/BasedMinato_/status/564132797301129216/photo/1
http://employmentassistancesnap.com/2015/02/08/call-center-program-manager/
http://www.weather.com/weather/today/l/26705

The above pages either had lots of images instead of content or were using JavaScript to load content. With JavaScript the content is fetched later after the page is loaded by browser which Boilerpipe cannot identify because of which the files generated by Boilerpipe were empty.

Classes of pages that were successful :-

http://www.smh.com.au/federal-politics/the-pulse-live/politics-live-tony-abbott-faces-libe
html.html
http://mosaicmagazine.com/essay/2015/02/obamas-secret-iran-strategy/
http://www.rapindustry.com/news.htm?utm_source=feedburner&utm_medium=feed&utm_campaign=
Feed%3A+rapindustry%2Fhip-hop+%28Hip+Hop+Daily+news-+RAPINDUSTRY.COM%29

The above pages had lots of text content which made it easier for boiler pipe to read. All content was loaded with the document load on the browser.

Question 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.

Answer

In next few pages there are two tables showing words given by two collections with and without boilerpipe(Only showiung 45 words because with 50 words table gets distorted).
The Graph for two collections is present after the tables. It clearly shows the both the collections follow Zipf distribution. Second ranked word in both list have almost half the frequency has first ranked word. And then it has a long tail for rest of the words.

Table 1: Rank,Term and Frequency before applying Boilerpipe(JusText)

RANK	TERM	FREQUENCY
1	the	44622
2	and	25270
3	to	24455
4	of	20162
5	a	19400
6	in	15115
7	is	10787
8	for	9272
9	that	8938
10	you	7857
11	on	7063
12	with	6528
13	it	5994
14	are	5406
15	this	5315
16	i	4912
17	was	4888
18	as	4747
19	have	4523
20	be	4240
21	at	4136
22	or	4057
23	not	3931
24	by	3777
25	from	3660
26	he	3652
27	we	3373
28	your	3346
29	will	3262
30	his	3192
31	an	3041
32	has	2983
33	but	2910
34	they	2832
35	if	2719
36	their	2473
37	all	2455
38	can	2276
39	new	2254
40	who	2237
41	more	2120
42	about	2060
43	one	1986
44	so	1978
45	out	1809

Table 2: Rank,Term and Frequency From Justext File

RANK	TERM	FREQUENCY
1	the	44622
2	and	25270
3	to	24455
4	of	20162
5	a	19400
6	in	15115
7	is	10787
8	for	9272
9	that	8938
10	you	7857
11	on	7063
12	with	6528
13	it	5994
14	are	5406
15	this	5315
16	i	4912
17	was	4888
18	as	4747
19	have	4523
20	be	4240
21	at	4136
22	or	4057
23	not	3931
24	by	3777
25	from	3660
26	he	3652
27	we	3373
28	your	3346
29	will	3262
30	his	3192
31	an	3041
32	has	2983
33	but	2910
34	they	2832
35	if	2719
36	their	2473
37	all	2455
38	can	2276
39	new	2254
40	who	2237
41	more	2120
42	about	2060
43	one	1986
44	so	1978
45	out	1809

Figures

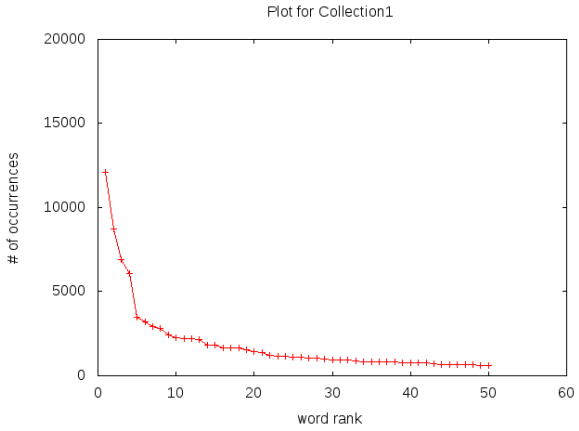

Figure 1: Before Boilerpipe

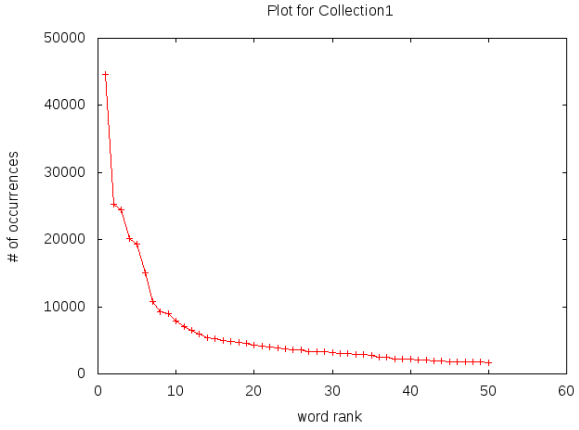

Figure 2: After Boilerpipe