Bill 5 - understandi…

```
%pyspark                                                                      FINISHED

import boto
from boto.s3.key import Key
from gzipstream import GzipStreamFile
from pyspark.sql.types import *
import warc
import ujson as json
import urlparse

watlist = sc.textFile("s3://commoncrawl/crawl-data/CC-MAIN-2017-04/wat.paths.gz")
watlist.cache()

def unpack(uri):
    conn = boto.connect_s3(anon=True, host='s3.amazonaws.com')
    bucket = conn.get_bucket('commoncrawl')
    key_ = Key(bucket, uri)
    file_ = warc.WARCFile(fileobj=GzipStreamFile(key_))
    return file_

def extract_json(id_, iterator):
    for uri in iterator:
        file = unpack(uri)
        for record in file:
            if record['Content-Type'] == 'application/json':
                try:
                    content = json.loads(record.payload.read())
                    yield content['Envelope']
                except:
                    yield None

def parse_urls(record):
    url_list = []
    try:
        page_url = record['WARC-Header-Metadata']['WARC-Target-URI']
        x = urlparse.urlparse(page_url)
        url_list += [(x.netloc, x.path)]
    except:
        pass
    try:
        links = record['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Lin
        for url in links:
            x = urlparse.urlparse(url['url'])
            url_list += [(x.netloc, x.path)]
    except:
        pass

    return url_list
```

Took 0 sec. Last updated by anonymous at September 16 2017, 11:10:57 AM.

READY

Parse URLs from JSON: Records RDD

```
%pyspark                                                          FINISHED

from __future__ import print_function

nfiles = 1
files = sc.parallelize(watlist.take(nfiles))

json_rdd = files.mapPartitionsWithIndex(extract_json)
json_rdd.cache()

print("Nr json records:", json_rdd.count())

records = json_rdd\
        .flatMap(parse_urls)\
        .filter(lambda x: x[0] is not "")\
        .groupByKey()\
        .map(lambda x: (x[0], set(x[1])))

records.cache()
json_rdd.unpersist()

record_count = records\
        .map(lambda x: (x[0], len(x[1])))\
        .sortBy(lambda x: -x[1])\
        .collect()
for x in record_count[:10]:
    print(x)
```

```
Nr json records: 162874
(u'www.facebook.com', 10872)
(u'twitter.com', 10241)
(u'www.newslocker.com', 5784)
(u'artodyssey1.blogspot.com', 5366)
(u'www.youtube.com', 5305)
(u'plus.google.com', 4337)
(u'www.socarrao.com.br', 3551)
(u'4chanarchives.cu.cc', 3249)
(u'www.price4all.ru', 3079)
(u'akulagi.com', 3034)
```

Took 4 min 21 sec. Last updated by anonymous at September 16 2017, 11:15:25 AM. (outdated)

```
%pyspark                                                          FINISHED

from __future__ import print_function

ex = records\
        .filter(lambda x: len(x[1])==10)\
        .takeSample(False,1)[0]

print("Domain:", ex[0])
print("Pages:")
for y in ex[1]:
    print(y)
```

```
Domain: www.dailypuppy.com
Pages:
/member/ecd1615731
/member/02a0e87fb7
/member/7dc7dffe6e/album/16596/photo/170032
/member/ff958e173f
/member/f509dab8e9/album/3089/photo/237585
```

```
/member/11ad5a5eb9/album/46133/photo/493622
/login.php
/dog/scooter_3898
/member/a3c18a594f/album/17164/photo/793950
/puppies/luke-the-australian-shepherd_2015-06-23
```

Took 4 sec. Last updated by anonymous at September 16 2017, 11:16:13 AM. (outdated)

We next define a string encoding of domains. READY
The idea will be to choose this so that domain structure (as contained in its URIs) can be learnt be an RNN.

```
%pyspark
```
FINISHED

```
import re
from __future__ import print_function

"""
# DEPRECATE:

def hexify(c):
    try:
        s = c.encode("utf-8").encode("hex")
    except UnicodeDecodeError:
        s = 0
    n = len(s)
    if n <= 2: return s
    a = ' '.join([s[i:i+2]+' -' for i in range(0,n,2)])
    return a[:-1]

def hexalise(str):
    return ' '.join([hexify(c) for c in str]) + ' . '

def domain_string(domain, path_set):
    out = hexalise(domain)
    for p in path_set: out += hexalise(p)
    return out
"""
```

Took 0 sec. Last updated by anonymous at September 16 2017, 11:19:25 AM. (outdated)

As the examples below show, we've chosen this encoding with the following constraints in mind: READY

- All symbols should be separated by spaces in order to parse at RNN training time.
- As well as hex symbols we include '.' to delimit different URIs.
- We include '-' as a limiter within non-Latin unicode characters. This will allow the RNN to distinguish Chinese characters, say, from sequences of Latin characters.
- Distinct domains will be delimited by '\n' at RNN training time.

```
%pyspark
```
FINISHED

```
from __future__ import print_function

ex = records\
        .filter(lambda x: len(x[1]) > 10 and len(x[1]) < 100)\
        .takeSample(False, 100)
```

```
for dom in ex:
    print("------------------------------------------")
    print("Domain:", dom[0])
    print("URIs:")
    print('\n'.join(list(dom[1])))
```

```
/smartphone-buyers-guide
/htc-and-under-armour-team-grip-fitness-tracker
/sites/androidcentral.com/files/styles/w85h55crop/public/article_images/2016/02/lg-g5-batte
ry-6_0.jpg
/
/asus-zenfone-3-zoom-hands
/best-wireless-mice-chromebooks
/chromecast-vs-chromecast-ultra-which-should-you-buy
/htc-u-ultra
/casio-wsd-f20-hands-ces-2017-android-wear-2-anywhere
/snapdragon-835-debuts-kryo-280-cpu-bluetooth-5-gigabit-lte
/honor-6x
/root
/sites/androidcentral.com/files/styles/w85h55crop/public/article_images/2017/01/k6-power-re
dmi-3s-prime-1.jpg
/search
/zenfone-ar-will-probably-be-great-once-it-works
```

```
Output exceeds 102400. Truncated.
```

Took 4 sec. Last updated by anonymous at September 16 2017, 11:19:37 AM. (outdated)

FINISHED

```
%pyspark

def domain_string(domain, path_set):
    out = domain + '\n' + '\n'.join(list(path_set)) + '\n'
    return out

ex = records.filter(lambda x: len(x[1])==10).take(10)

for dom in ex:
    print("------------------------------------------")
    print(domain_string(dom[0], dom[1]))
```

```
------------------------------------------
www.craigslist.org
/about/craigslist_is_hiring
/about/terms.of.use.en
/cal/
/images/animated-spinny.gif
/about/rss
/about/scams
/about/help/
/about/privacy.policy
/about/
------------------------------------------
americanwindsurfingtour.com
/about-3/
/pistol-river-weather-2014/
/weather-2/
/schedule-2/
```

Took 0 sec. Last updated by anonymous at September 16 2017, 11:22:07 AM.

The following count shows the motivation for encoding domains in this way. READY

We would like (for later use, when we model the string using an RNN) the alphabet of symbols in the representation to be reliably bounded. If we use the raw (unicode) string concatenation of the path URIs, then this is not the case because we get an explosion of possibilities from various languages. Here's a histogram of the symbols, together with their hex encodings:

FINISHED

```
%pyspark

from collections import Counter

char_count = records.map(lambda x: Counter('.'.join(list(x[1]))))\
                    .aggregate(Counter(),
                               lambda acc, value: acc + value,
                               lambda acc1, acc2: acc1 + acc2)
char_count = dict(char_count)

def hexify(c):
    """
    Temporary ASCII encoding for human readable hex with ' - ' as delimiter for detecting
    non-Latin unicode.
    """
    try:
        s = c.encode("utf-8").encode("hex")
    except UnicodeDecodeError:
        s = 0
    n = len(s)
    if n <= 2: return s
    a = ' - '.join([s[i:i+2] for i in range(0,n,2)])
    return a[:-1]

# examine:
print("Nr characters:", len(char_count.keys()))
for key, value in sorted(char_count.iteritems(), key=lambda (k,v): (-v,k)):
    print "%8d %4s %16s" % (value, key, hexify(key))
```

```
('Nr characters:', 2083)
 5123801    /               2f
 4146432    e               65
 3690910    a               61
 2983947    -               2d
 2879741    t               74
 2783207    i               69
 2766669    s               73
 2707176    o               6f
 2475434    .               2e
 2433279    r               72
 2270142    n               6e
 2081952    l               6c
 1763606    c               63
 1636562    d               64
 1569923    m               6d
 1536649    p               70
 1478606    0               30
```

Took 37 sec. Last updated by anonymous at September 16 2017, 11:51:44 AM.

Compare this with the distribution after hexification. The number of symbols is bounded by 256 + 2. This time it's more informative to sort by key: READY

```
%pyspark

from collections import Counter

hex_count = records\
        .map(lambda x: [h for c in list(domain_string(x[0], x[1])) for h in hexify(c).spli
        .map(lambda x: Counter(x))\
        .aggregate(Counter(),
            lambda acc, value: acc + value,
            lambda acc1, acc2: acc1 + acc2)

hex_count = dict(hex_count)

# examine:
print("Nr hex characters:", len(hex_count.keys()))
for key, value in sorted(hex_count.iteritems(), key=lambda (k,v): k):
    print "%2s %8d" % (key, value)
```

FINISHED

```
('Nr hex characters:', 202)
 -   252648
03        1
09      413
0a  1951178
0b        1
0d      414
20    25473
21     1845
22       23
24     1291
25  1122548
26     3063
27      750
28     3561
29     3541
2a     2206
2b    31840
```

Took 2 min 31 sec. Last updated by anonymous at September 16 2017, 11:57:58 AM.

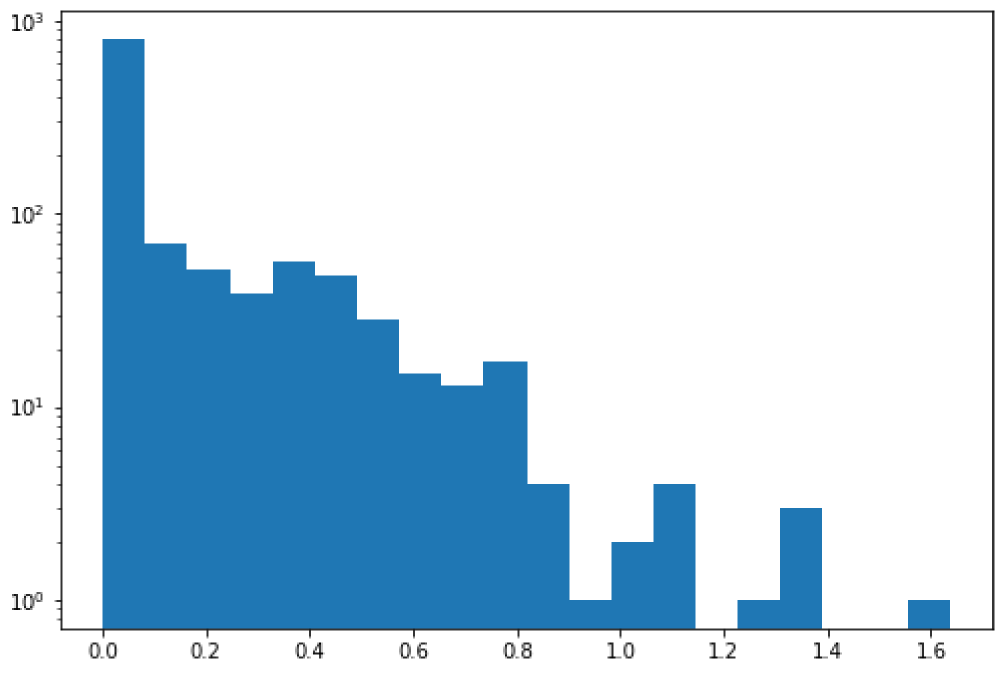
Let's use a filter on '-' to find all domains with non-Latin URIs:

READY

```
%pyspark

import matplotlib.pyplot as plt

def nonlatin_detector(dom):
    """
    Computes the excess nr bytes over nr characters in a domain string.
    """
    str = domain_string(dom[0], dom[1])
    N = len(str)
    hex = [c.encode('utf-8').encode('hex') for c in list(str)]
    return float(sum([len(h)/2 for h in hex]) - N)/N

nonlatin = records\
        .map(lambda x: (x[0], x[1], nonlatin_detector(x)))\
        .filter(lambda x: x[2] > 0)\
        .collect()
```

FINISHED

```
plt.hist([dom[2] for dom in nonlatin], bins=20)
plt.yscale("log")
plt.show()
```



Took 1 min 37 sec. Last updated by anonymous at September 16 2017, 12:19:54 PM. (outdated)

For example: READY

FINISHED

```
%pyspark

from __future__ import print_function

for dom in nonlatin:
    if dom[2] > 1.0:
        print("-------------------------------------------")
        print("%s (%g)" % (dom[0], dom[2]))
        for uri in dom[1]:
            print(uri)
```

```
-------------------------------------------
活跃 活跃  不就是活跃么? (1.375)
-------------------------------------------
替换为你的QQ空间网址 (1.38462)
-------------------------------------------
thai.tourismthailand.org (1.10448)
/สถานที่ท่องเที่ยว/ค้นหา
/ค้นหาแบบละเอียด
-------------------------------------------
スクフェスちゃねる.com (1.125)
/
-------------------------------------------
替换为你的微博网址 (1.63636)
-------------------------------------------
www.a-too.co.jp (1.13402)
/採用情報/限定社員採用/
```

```
/採用情報/2017年度新卒採用/
```

Took 0 sec. Last updated by anonymous at September 16 2017, 12:21:28 PM.

```
%pyspark

records.unpersist()
```
READY

```
PythonRDD[52] at RDD at PythonRDD.scala:48
```

READY

Save to S3

READY

The end-to-end process:

READY

```
%pyspark

nfiles = 128

files = sc.parallelize(watlist.take(nfiles))
json_rdd = files.mapPartitionsWithIndex(extract_json)
domains_rdd = json_rdd\
             .flatMap(parse_urls)\
             .filter(lambda x: x[0] is not "")\
             .groupByKey()\
             .map(lambda x: {'domain': x[0], 'path_set': set(x[1])})


# make sure the following S3 directory is deleted first:

outputURI = "s3://billsdata.net/CommonCrawl/domain_paths_from_%d_WAT_files" % nfiles
codec = "org.apache.hadoop.io.compress.GzipCodec"
domains_rdd.saveAsTextFile(outputURI, codec)
```
READY

Timings:

READY

Cluster	nr WAT files	time	output size (gzip)
16 x m4.2xlarge	128	7 min 24 sec	944.6 MiB
16 x m4.2xlarge	256	10 min 16 sec	1.7 GiB
16 x m4.2xlarge	512	19 min 31 sec	3.1 GiB
16 x m4.2xlarge	1024	40 min 43 sec	5.7 GiB

To find output size:

```
$ aws s3 ls —human-readable —summarize
s3://billsdata.net/CommonCrawl/domain_paths_from_256_WAT_files/ | grep Total
```

```
%pyspark
```
READY