Bill 5 - understandi...

FINISHED

Constructing URI sets

What this notebook does:

Explores the URI paths under domains, introduces the idea of string signature and creates files of path sets in S3.

Took 0 sec. Last updated by anonymous at October 14 2017, 11:52:33 AM.

FINISHED

```
%pyspark

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
```

Took 50 sec. Last updated by anonymous at October 14 2017, 11:04:40 AM.

FINISHED

Parse URLs from JSON: domain records RDD

Each record is a domain name paired with set of URIs.

Took 0 sec. Last updated by anonymous at October 14 2017, 11:20:53 AM.

FINISHED

```
%pyspark

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())

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

domain_records.cache()
json_rdd.unpersist()

domain_record_count = domain_records\
        .map(lambda x: (x[0], len(x[1])))\
        .sortBy(lambda x: -x[1])\
        .collect()
for x in domain_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 8 sec. Last updated by anonymous at October 14 2017, 11:25:27 AM.

FINISHED

```
%pyspark

from __future__ import print_function

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

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

```
-----------------------------------
Domain: www.yourlifeupdated.net
Pages:
/
/android/android-wallpaper-del-lunedi-icicle-2/
/android/app-del-giorno-android-cobble-swipe-match-gratis-sul-play-store/
/android/app-del-giorno-android-noir-disponibile-al-download-gratis-sul-play-store/
/android/come-risolvere-errore-app-mediaset-premium-i-requisiti-di-sicurezza-necessari/
/android/download-lucky-patcher-v-5-8-4-per-android-apk/
/android/download-shazam-encore-5-12-1-apk-dal-play-store/
/android/trucchi-cell-connect-android-soldi-infiniti-illimitati/
/apple/app-a-pagamento-gratis-su-iphone-ipad-ios-10-senza-jailbreak-con-pp25/
/author/cashdroid/
/author/cashdroid/feed/
/author/darkleomax/
/canzoni/laura-pausini-200-note-testo-parole-e-video-ufficiale/
/category/android/
/category/apple/
```

Took 3 sec. Last updated by anonymous at October 14 2017, 11:27:27 AM.

Let's formalise these strings as signatures for the domains: FINISHED

Took 0 sec. Last updated by anonymous at October 14 2017, 11:28:39 AM.

FINISHED

```
%pyspark

def domain_string(domain, path_set):
    """
    Takes domain and concatenates with path URIs separated by newlines.
    """
    out = domain + '\n' + '\n'.join(sorted(list(path_set))) + '\n\n\n'
    return out

ex = domain_records.takeSample(False, 1000)

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

```
-----------------------------------
www.mixes.video
-----------------------------------
gaming2.sbgglobal.eu
/login/SBGGLOBAL
-----------------------------------
secure-au.imrworldwide.com
/cgi-bin/m
-----------------------------------
www.wychavon.gov.uk
/
/WDC-theme/images/common/print.png
/WDC-theme/images/logo/gov-delivery.gif
```

```
/WDC-theme/images/logo/thumbs.jpg
/WDC-theme/images/logo/wdc-logo.png
/WDC-theme/images/logo/wdc-logo2.png
/WDC-theme/images/social/email-icon.jpg

Output exceeds 102400. Truncated.
```

Took 3 sec. Last updated by anonymous at October 14 2017, 11:49:36 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 = domain_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
```

Took 23 sec. Last updated by anonymous at October 14 2017, 11:30:50 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

FINISHED

```
%pyspark

from collections import Counter

hex_count = domain_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)
```

```
('Nr hex characters:', 202)
 -   252648
03        1
09      413
0a  2287244
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 1 min 54 sec. Last updated by anonymous at October 14 2017, 11:40:18 AM.

Let's use a filter on '-' to find all domains with non-Latin URIs. (This was the reason for the temporary notation.)

FINISHED

Took 0 sec. Last updated by anonymous at October 14 2017, 11:31:38 AM.

FINISHED

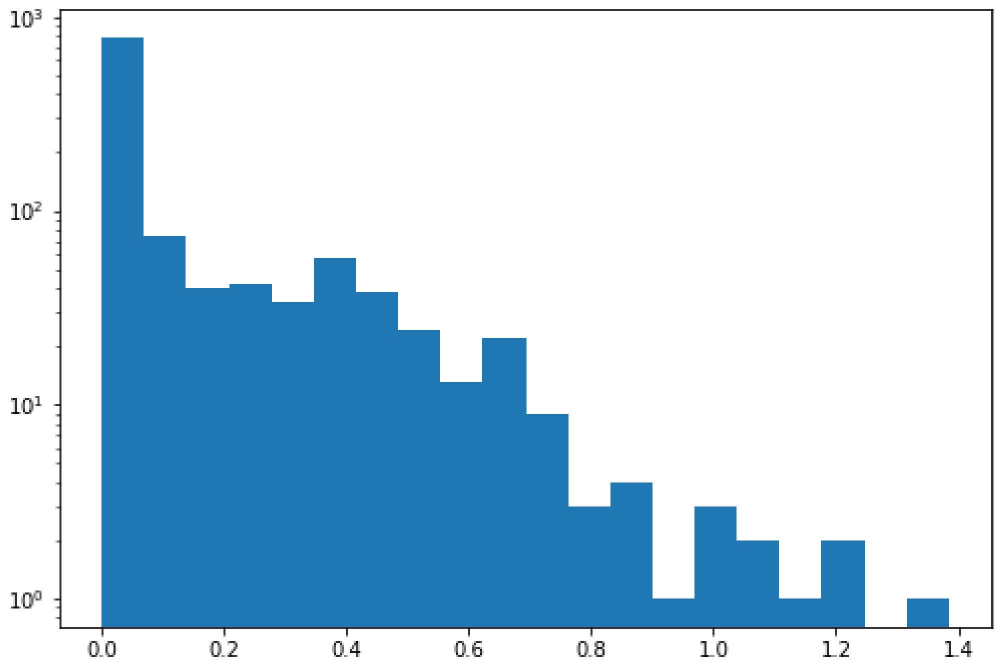
```
%pyspark

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

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



Took 1 min 8 sec. Last updated by anonymous at October 14 2017, 11:41:40 AM.

For example: READY

FINISHED

```
%pyspark

from __future__ import print_function

for dom in nonlatin:
    if dom[2] > 0.05:
        print("-------------------------------------------")
        print("%s (%g)" % (dom[0], dom[2]))
        for uri in sorted(dom[1]):
            print(uri)
```

```
-------------------------------------------
hf5.l2central.info (0.491525)
/
/Категория:Земли Иннадрила - квесты
-------------------------------------------
```

```
iau.vec.go.th (0.139066)
/%E0%B8%81%E0%B8%8E%E0%B8%9A%E0%B8%B1%E0%B8%95%E0%B8%A3.aspx
/%E0%B8%82%E0%B9%88%E0%B8%B2%E0%B8%A7%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B8%8A%E0%B8%B2%E0%B8%A
A%E0%B8%B1%E0%B8%A1%E0%B8%9E%E0%B8%B1%E0%B8%99%E0%B8%98%E0%B9%8C.aspx
/%E0%B8%94%E0%B8%B2%E0%B8%A7%E0%B8%99%E0%B9%8C%E0%B9%82%E0%B8%AB%E0%B8%A5%E0%B8%94/%E0%B8%8
1%E0%B8%B2%E0%B8%A3%E0%B8%AD%E0%B8%9A%E0%B8%A3%E0%B8%A1%E0%B8%AA%E0%B8%B1%E0%B8%A1%E0%B8%A
1%E0%B8%99%E0%B8%B2.aspx
/%E0%B8%94%E0%B8%B2%E0%B8%A7%E0%B8%99%E0%B9%8C%E0%B9%82%E0%B8%AB%E0%B8%A5%E0%B8%94/%E0%B8%8
4%E0%B8%B9%E0%B9%88%E0%B8%A1%E0%B8%B7%E0%B8%AD.aspx
/%E0%B8%94%E0%B8%B2%E0%B8%A7%E0%B8%99%E0%B9%8C%E0%B9%82%E0%B8%AB%E0%B8%A5%E0%B8%94/%E0%B8%A
B%E0%B8%99%E0%B8%B1%E0%B8%87%E0%B8%AA%E0%B8%B7%E0%B8%AD%E0%B9%80%E0%B8%A7%E0%B8%B5%E0%B8%A
2%E0%B8%99.aspx

Output exceeds 102400. Truncated.
```

Took 5 sec. Last updated by anonymous at October 14 2017, 11:44:34 AM.

READY

```
%pyspark

domain_records.unpersist()
```

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

READY

Save to S3

The end-to-end process:

READY

ERROR

```
%pyspark

nfiles = 4096

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)
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1311155247112590380.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1311155247112590380.py", line 360, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 11, in <module>
  File "/usr/lib/spark/python/pyspark/rdd.py", line 1551, in saveAsTextFile
```

```
    keyed._jrdd.map(self.ctx._jvm.BytesToString()).saveAsTextFile(path, compressionCodec)
  File "/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py", line 1131, in
 __call__
    answer = self.gateway_client.send_command(command)
  File "/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py", line 883, in s
end_command
    response = connection.send_command(command)
  File "/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py", line 1028, in
 send_command
```

Took 2 hrs 58 min 21 sec. Last updated by anonymous at October 15 2017, 10:24:57 AM.

Timings: FINISHED

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 m3.xlarge	1024	1 hrs 44 min 23 sec	5.7 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
```

Took 0 sec. Last updated by anonymous at October 14 2017, 8:05:56 PM.

```
%pyspark
```

READY