201708 common c...

FINISHED

WAT files - understanding the JSON structure

Took 0 sec. Last updated by anonymous at August 28 2017, 1:52:14 PM.

FINISHED

```
%pyspark

import boto
from boto.s3.key import Key
from gzipstream import GzipStreamFile
from pyspark.sql.types import *
import warc

import json

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

conn = boto.connect_s3(anon=True, host='s3.amazonaws.com')
bucket = conn.get_bucket('commoncrawl')

def unpack(uri):
    key_ = Key(bucket, uri)
    file_ = warc.WARCFile(fileobj=GzipStreamFile(key_))
    return file_

def mapper(id_, iterator):

    for uri in iterator:
        file = unpack(uri)
        for record in file:
            try:
                yield record['Content-Type']
            except KeyError:
                yield None

nfiles = 16
files = sc.parallelize(watlist.take(nfiles))

ct = files.mapPartitionsWithIndex(mapper)
ct.cache()
print(ct.count())
print(ct.countByValue())
ct.unpersist()
```

```
2621630
defaultdict(<type 'int'>, {'application/warc-fields': 16, 'application/json': 2621614})
PythonRDD[131] at RDD at PythonRDD.scala:48
```

Took 1 min 18 sec. Last updated by anonymous at August 28 2017, 12:28:00 PM.

Let's examine a sample of json records:

READY

```
%pyspark                                                                    FINISHED

from pprint import pprint

def json_mapper(id_, iterator):
    conn = boto.connect_s3(anon=True, host='s3.amazonaws.com')
    bucket = conn.get_bucket('commoncrawl')

    for uri in iterator:
        key_ = Key(bucket, uri)
        file_ = warc.WARCFile(fileobj=GzipStreamFile(key_))

        for record in file_:
            if record['Content-Type'] == 'application/json':
                record = json.loads(record.payload.read())

                try:
                    yield record
                except KeyError:
                    yield None

nrecords = 100
sample = files.\
        mapPartitionsWithSplit(json_mapper).\
        take(nrecords)

pprint(sample[1])
```

```
42H3S6NNFQ2MSVX7XZKYAYSCX5QBYJ',
                                                                u'Entity-Length': u'0',
                                                                u'Entity-Trailing-Slop-Byte
s': u'0',
                                                                u'Headers': {u'Accept': u't
ext/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
                                                                             u'Accept-Encod
ing': u'x-gzip, gzip, deflate',
                                                                             u'Host': u'03o
nline.com',
                                                                             u'User-Agent':
 u'CCBot/2.0 (http://commoncrawl.org/faq/)'},
                                                                u'Headers-Length': u'211',
                                                                u'Request-Message': {u'Meth
od': u'GET',
                                                                                     u'Pat
h': u'/news/3383',

                                                                                     u'Vers
```

Took 1 sec. Last updated by anonymous at August 28 2017, 12:29:33 PM.

We see that the field `['Envelope']['WARC-Header-Metadata']['WARC-Target-URI']` contains the URI of the current web page: FINISHED

Took 0 sec. Last updated by anonymous at August 28 2017, 12:31:20 PM.

```
%pyspark                                                                    FINISHED

for rec in sample:
    try:
        print(rec['Envelope']['WARC-Header-Metadata']['WARC-Target-URI'])
    except KeyError:
        print("")
```

```
http://103.2.132.3/newsdetail.asp?ID=44
http://103.2.132.3/newsdetail.asp?ID=44
http://103.2.132.3/newsdetail.asp?ID=44
http://1037theloon.com/tags/w-a-s-p/
http://1037theloon.com/tags/w-a-s-p/
http://1037theloon.com/tags/w-a-s-p/
http://1061thetwister.iheart.com/articles/entertainment-news-104651/trick-pony-reunites-fol
lowing-sevenyear-split-12059522/
http://1061thetwister.iheart.com/articles/entertainment-news-104651/trick-pony-reunites-fol
lowing-sevenyear-split-12059522/
http://1061thetwister.iheart.com/articles/entertainment-news-104651/trick-pony-reunites-fol
lowing-sevenyear-split-12059522/
http://1079thebear.iheart.com/onair/ken-dashow-32036/epic-fails-for-july-2014-12639622/
http://1079thebear.iheart.com/onair/ken-dashow-32036/epic-fails-for-july-2014-12639622/
http://1079thebear.iheart.com/onair/ken-dashow-32036/epic-fails-for-july-2014-12639622/
http://107jamz.com/events-lake-charles/icd-10-cm-coding-bootcamp/24-july-2013/
http://107jamz.com/events-lake-charles/icd-10-cm-coding-bootcamp/24-july-2013/
http://107jamz.com/events-lake-charles/icd-10-cm-coding-bootcamp/24-july-2013/
```

Took 0 sec. Last updated by anonymous at August 28 2017, 12:31:23 PM.

Later we'll want to aggregate records by web domain, and use the information in the individual page records to build features of the domains.

FINISHED

Let's build a `traverse` function to output a lists of keys of a json record together with its tree depth and boolean is-leaf indicator:

Took 0 sec. Last updated by anonymous at August 28 2017, 12:33:56 PM.

FINISHED

```
%pyspark

from __future__ import print_function

def traverse(js, depth, keys):
    if type(js) is dict:
        d = depth + 1
        for k in js.keys():
            if type(js[k]) is not dict: leaf = 1
            else: leaf = 0
            keys += [(d,leaf,k)]
            depth, keys = traverse(js[k], d, keys)
    return depth, keys

js = sample[1]
depth, keys = traverse(js, 0, [])
keys.sort()

print(len(keys), "json keys:")
for x in keys:
    print(x)
```

```
(3, 1, u'WARC-Date')
(3, 1, u'WARC-IP-Address')
(3, 1, u'WARC-Record-ID')
(3, 1, u'WARC-Target-URI')
(3, 1, u'WARC-Type')
(3, 1, u'WARC-Warcinfo-ID')
(4, 0, u'Headers')
(4, 0, u'Request-Message')
(4, 1, u'Entity-Digest')
```

```
(4, 1, u'Entity-Digest')
(4, 1, u'Entity-Length')
(4, 1, u'Entity-Trailing-Slop-Bytes')
(4, 1, u'Headers-Length')
(5, 1, u'Accept')
(5, 1, u'Accept-Encoding')
(5, 1, u'Host')
(5, 1, u'Method')
(5, 1, u'Path')
(5, 1, u'User-Agent')
```

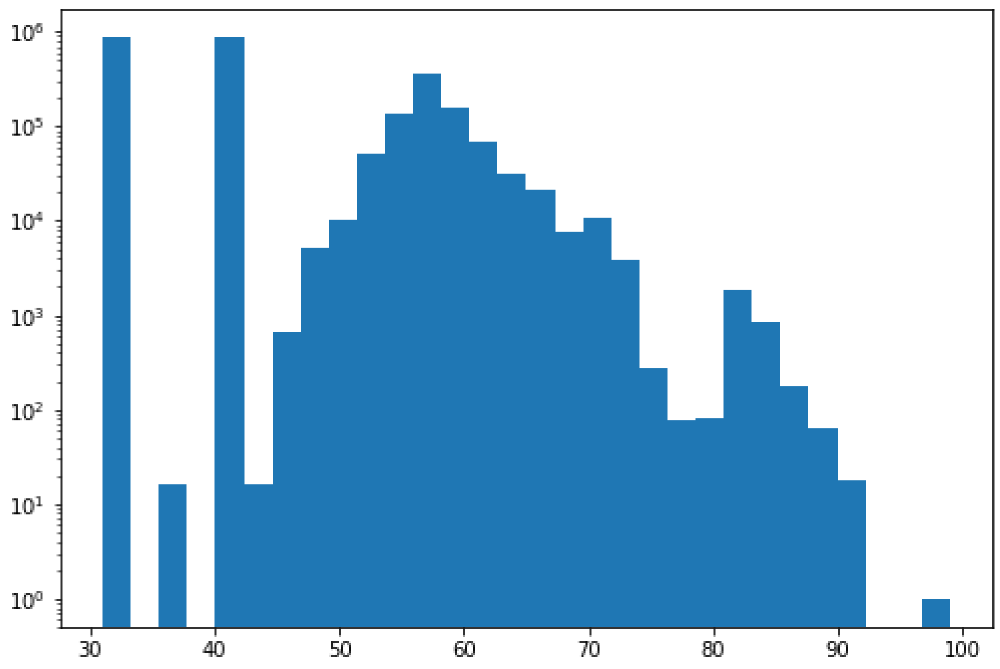
Took 0 sec. Last updated by anonymous at August 28 2017, 12:33:59 PM.

This json record has 41 keys. Let's see how that varies over all the records:

READY

FINISHED

```
%pyspark

from collections import Counter
import matplotlib.pyplot as plt

def get_json_keys(id_, iterator):

    for uri in iterator:

        file = unpack(uri)
        for record in file:
            if record['Content-Type'] == 'application/json':
                record = json.loads(record.payload.read())

                try:
                    _,k = traverse(record, 0, [])
                    """
                    yield the shape of the record:
                    """
                    yield dict(Counter([x[0:2] for x in k]))
                except KeyError:
                    yield None

json_keys = files.mapPartitionsWithIndex(get_json_keys)
json_shape = json_keys.collect()

total_shape = [sum(x.values()) for x in json_shape]

plt.hist(total_shape, bins=30)
plt.yscale('log')
plt.show()
```



Took 5 min 54 sec. Last updated by anonymous at August 28 2017, 12:40:16 PM.

Let's break down this histogram by tree depth and leaf vs node:

READY

```
%pyspark

from __future__ import print_function

maxdepth = max([k[0] for y in json_shape for k in y.keys()])

for depth in range(1, 1+maxdepth):
    print("Depth", depth)

    nodeshape = dict(Counter([x[(d,l)]  for x in json_shape for (d,l) in x.keys() if d==de
    if len(nodeshape.items()) > 0:
        print("nodes:")
        for i in nodeshape.items(): print(i)

    leafshape = dict(Counter([x[(d,l)]  for x in json_shape for (d,l) in x.keys() if d==de
    if len(leafshape.items()) > 0:
        print("leaves:")
        for i in leafshape.items(): print(i)
```

FINISHED

```
(38, 158)
(39, 1595)
(40, 129)
(41, 297)
(42, 573)
(43, 164)
(44, 13)
(45, 68)
(46, 8)
(54, 18)
(56, 1)

Depth 6
```

```
leaves:
(1, 6645)
(2, 10224)
(3, 33612)
(4, 727800)
```

Took 1 min 8 sec. Last updated by anonymous at August 28 2017, 12:41:52 PM.

In other words: FINISHED

At depth 1:
all records have two nodes

```
(1, 0, u'Container')
(1, 0, u'Envelope')
```

At depth 2:
all records have 3 nodes and 7 leaves

```
(2, 0, u'Gzip-Metadata')
(2, 0, u'Payload-Metadata')
(2, 0, u'WARC-Header-Metadata')
(2, 1, u'Actual-Content-Length')
(2, 1, u'Block-Digest')
(2, 1, u'Compressed')
(2, 1, u'Filename')
(2, 1, u'Format')
(2, 1, u'Offset')
(2, 1, u'WARC-Header-Length')
```

At depth 3:
all records a single 3 node

```
(3, 0, u'HTTP-Request-Metadata')
```

but 15, 18 or 19 leaves.

At depth 4:
2 or 3 nodes and 3,4,5 or 8 leaves.

At depth 5 and 6:

Depth 5 is where most of the leaf variance is; there's a single (optional) node

```
['Envelope']['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']
['Head']
```

with 1,2,3,4,5 leaves under it at depth 6.

Let's eyeball a record with a larger number of json keys:

Took 0 sec. Last updated by anonymous at August 28 2017, 1:40:10 PM.

FINISHED

```
%pyspark

nkeys = 80
```

```
def f(rec):
    _,k = traverse(rec, 0, [])
    return len(k)

sample = files.\
        mapPartitionsWithSplit(json_mapper).\
        filter(lambda rec: f(rec) > nkeys).\
        take(100)

rec = sample[0]
depth, keys = traverse(rec, 0, [])
keys.sort()

print(len(keys))
for x in keys:
    print(x)
```

```
(5, 1, u'X-Passed-To-BeforeDispatch')
(5, 1, u'X-Passed-To-DLL')
(5, 1, u'X-Passed-To-PostProcessResponse')
(5, 1, u'X-Returned-From')
(5, 1, u'X-Returned-From-BeforeDispatch')
(5, 1, u'X-Returned-From-DLL')
(5, 1, u'X-Returned-From-PostProcessResponse')
(5, 1, u'X-Served-By')
(5, 1, u'X-UA-Device')
(5, 1, u'X-Varnish')
(5, 1, u'X-Varnish-beresp-grace')
(5, 1, u'X-Varnish-beresp-status')
(5, 1, u'X-Varnish-beresp-ttl')
(5, 1, u'x-stale')
(6, 1, u'Link')
(6, 1, u'Metas')
(6, 1, u'Scripts')
(6, 1, u'Title')
```

Took 54 sec. Last updated by anonymous at August 28 2017, 1:41:22 PM.

Let's look, at depth 5, at the `Head` node and a leaf `Links`, which contain relevant information about outgoing links: FINISHED

Took 0 sec. Last updated by anonymous at August 28 2017, 1:44:00 PM.

FINISHED

```
%pyspark

from pprint import pprint

node = rec['Envelope']['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Hea

_,k = traverse(node, 0, [])

for x in k:
    print(x)
    print(type(node[x[2]]))
    if type(node[x[2]]) == list: print(len(node[x[2]]))

pprint(node)
```

```
ork for Denver City Electric Motor Company as an electric motor repairman. ',
              u'name': u'twitter:description'},
             {u'content': u'Gregory Young', u'property': u'og:title'},
             {u'content': u'http://www.ardmoreite.com/article/20121123/OBITUARIES/12112990
```

```
4',
             u'property': u'og:url'},
            {u'content': u'article', u'property': u'og:type'},
            {u'content': u'Gregory Young, age 52, passed away on November 20, 2012, in Ardm
ore, Okla. Services will be held at 10 a.m., Saturday, November 24, 2012, in The Chapel at
 ',
             u'property': u'og:description'}],
 u'Scripts': [{u'path': u'SCRIPT@/src',
               u'url': u'//cdnjs.cloudflare.com/ajax/libs/modernizr/2.6.2/modernizr.min.js?
20160411-3'},
              {u'path': u'SCRIPT@/src',
               u'url': u'//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jquery.min.js?201604
11-3'},
              {u'path': u'SCRIPT@/src',
```

Took 0 sec. Last updated by anonymous at August 28 2017, 1:42:23 PM.

`Links` is a json leaf, but behaves like a node in that its value is a list of dicts: FINISHED

Took 0 sec. Last updated by anonymous at August 28 2017, 1:45:03 PM.

```
%pyspark

node = rec['Envelope']['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Lin

_,k = traverse(node, 0, [])

for x in node2:
    pprint.pprint(x)
```
READY

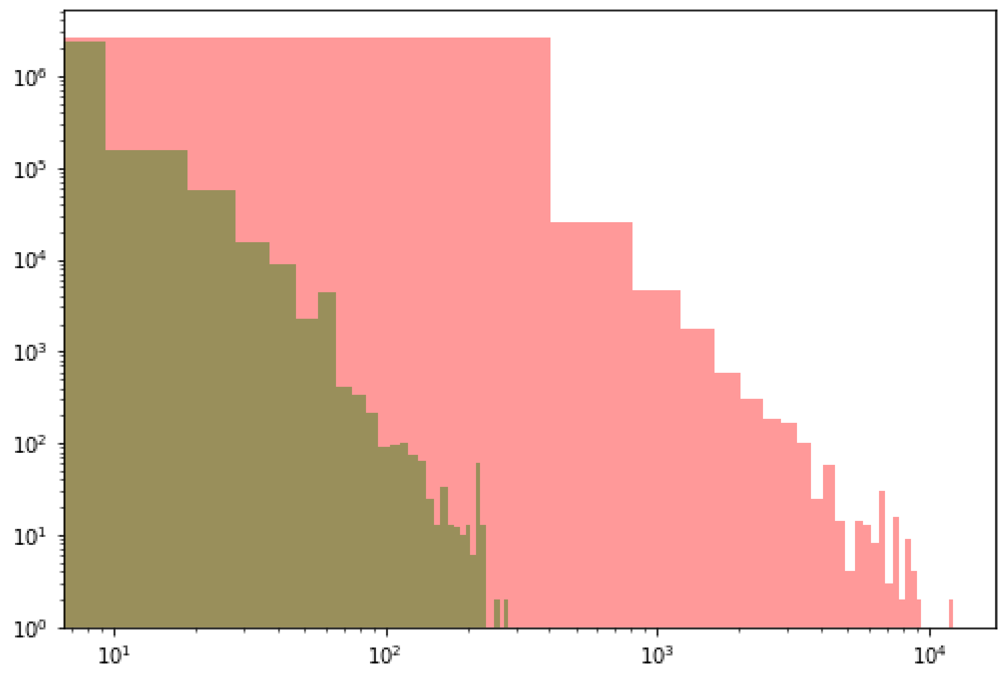
```
{u'path': u'A@/href',
 u'text': u'Feedback',
 u'url': u'mailto:yournews@ardmoreite.com'}
{u'path': u'A@/href',
 u'text': u'Terms of Use',
 u'url': u'http://www.gatehousemedia.com/terms_of_use/'}
{u'path': u'A@/href',
 u'text': u'Privacy Policy',
 u'url': u'http://www.gatehousemedia.com/privacy-policy/'}
{u'path': u'A@/href',
 u'text': u'GateHouse Media Publications',
 u'url': u'http://www.gatehousemedia.com/our-markets/'}
{u'path': u'A@/href',
 u'target': u'_blank',
 u'title': u'The Daily Ardmoreite Facebook',
 u'url': u'http://www.facebook.com/pages/The-Ardmoreite/143707431292?ref=nf'}
{u'path': u'A@/href',
 u'target': u'_blank',
```

Finally, compare the distributions (over json records) of the numbers of `'url'` values seen in a record, for the two nodes just discussed: FINISHED

Took 0 sec. Last updated by anonymous at August 28 2017, 1:47:45 PM.

```
%pyspark

import matplotlib.pyplot as plt
```
FINISHED

```
def compare_links(record):
    try:
        set1 = set([x['url'] for x in record['Envelope']['Payload-Metadata']['HTTP-Respons
        set2 = set([x['url'] for x in record['Envelope']['Payload-Metadata']['HTTP-Respons
        return [len(set1), len(set2)]
    except KeyError:
        return [0,0]

def get_link_counts(id_, iterator):
    for uri in iterator:
        file = unpack(uri)
        for record in file:
            if record['Content-Type'] == 'application/json':
                record = json.loads(record.payload.read())
                yield compare_links(record)

link_counts = files.mapPartitionsWithIndex(get_link_counts)
link_count = link_counts.collect()

llinks = [x[0] for x in link_count]
rlinks = [x[1] for x in link_count]

plt.hist(rlinks, bins=30, alpha=0.4, color='red')
plt.hist(llinks, bins=30, alpha=0.4, color='green')
plt.xscale('log')
plt.yscale('log')
plt.show()
```



Took 14 sec. Last updated by anonymous at August 28 2017, 1:56:18 PM. (outdated)

```
%pyspark
```

READY