READY

Extracting web links from WAT

See https://github.com/commoncrawl/cc-pyspark (https: //github.com/commoncrawl/cc-pyspark) for pre-canned solution. Here let's derive an RDD from the observations above.

For each json record, we seen that the current URL is contained in the value
`['Envelope']['WARC-Header-Metadata']['WARC-Target-URI']`

See https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/ (https: //iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/). In particular:

> **WARC-Target-URI**
> The original URI whose capture gave rise to the information content in this record. In the context of web harvesting, this is the URI that was the target of a crawler's retrieval request. For a 'revisit' record, it is the URI that was the target of a retrieval request. Indirectly, such as for a 'metadata', or 'conversion' record, it is a copy of the WARC-Target-URI appearing in the original record to which the newer record pertains. The URI in this value shall be properly escaped according to [RFC3986] and written with no internal whitespace.

Some outgoing links are contained in
`['Envelope']['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Head']['Link']`
which is itself a list of dicts with keys
`'path', 'rel', 'url'`
As the documentation below shows, these are not links of interest – these are contained in
`['Envelope']['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Links']`
whose dicts have keys
`'path','target','text','url'`

See
https://webarchive.jira.com/wiki/spaces/Iresearch/pages/13467719/Web+Archive+Metadata+File+Specifi
(https:
//webarchive.jira.com/wiki/spaces/Iresearch/pages/13467719/Web+Archive+Metadata+File+Specificatior
In particular:

> **HTML-Metadata**

```
"HTML-Metadata": {
  "Head": {
   "Metas": [
    {
     "content": "Jim DeMint - U.S. Senate South Carolina",
     "name": "description"
    },
    {
     "content": "demint, jim deMint, senate, south carolina, republi
     "name": "keywords"
    }
   ],
   "Title": "Jim DeMint - U.S. Senate"
  },
  "Links": [
   {
    "path": "TABLE@/background",
    "url": "/demint_images/top_bg1.gif"
   },
   {
    "path": "A@/href",
    "text": "clicking here.",
    "url": "http://jimdemint.com/demint_contents/issues/jobs/"
   }]
}
```

Links
Indicates the absolute URI of an outgoing link from the capture, the URI of the link as it appears on the page, the type of outgoing link (link, embed, redirect or other), XPath-suffix of link (best-effort), the alt attribute and anchor-text (truncated to 100 bytes)
Head
Attributes and values of HTML head elements: title, base, style, link, meta and script

```
%pyspark

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

import ujson as json

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

FINISHED

```
s3://commoncrawl/crawl-data/CC-MAIN-2017-04/wat.paths.gz MapPartitionsRDD[1] at textFile at
 NativeMethodAccessorImpl.java:0
```

Took 31 sec. Last updated by anonymous at September 02 2017, 10:35:37 AM.

```
%pyspark
```

ERROR

```
from __future__ import print_function

nfiles = 1024
files = sc.parallelize(watlist.take(nfiles))

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

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

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

See below for timings. READY

FINISHED

```
%pyspark

import urlparse
from collections import Counter

def parse_links(record):

    try:
        page_url = record['WARC-Header-Metadata']['WARC-Target-URI']
        page_domain = urlparse.urlparse(page_url).netloc
        links = record['Payload-Metadata']['HTTP-Response-Metadata']['HTML-Metadata']['Lin
        out_links = Counter([urlparse.urlparse(url['url']).netloc for url in links])
        return (page_domain, out_links)
    except:
        return None

links_rdd = json_rdd\
            .map(parse_links)\
            .filter(lambda x: x is not None)\
            .reduceByKey(lambda x,y: x+y)\
            .map(lambda x: {'domain': x[0], 'out': dict(x[1])})
links_rdd.cache()
json_rdd.unpersist()
```

```
PythonRDD[4] at RDD at PythonRDD.scala:48
```

Took 0 sec. Last updated by anonymous at September 02 2017, 11:07:59 AM.

FINISHED

```
%pyspark
```

```
from __future__ import print_function

page_link_count = links_rdd\
                    .map(lambda x: (1, sum(x['out'].values())))\
                    .reduceByKey(lambda x,y: x+y)\
                    .collect()[0][1]

print("Nr page links:", page_link_count)
print("Nr domain links:", links_rdd.count())
```

```
Nr page links: 8102680487
Nr domain links: 814114
```

Took 56 min 14 sec. Last updated by anonymous at September 02 2017, 12:04:16 PM.

Timings: FINISHED

Cluster	nr files	json record count	page/domain link count
16 x m3.2xlarge	128	21.0M in 11 min 39 sec	1.0B --> 199k in 18 min 18 sec
16 x m3.2xlarge	256	41.9M in 22 min 42 sec	2.0B --> 283k in 39 min 38 sec
16 x m4.2xlarge	512	83.8M in 13 min 9 sec	4.0B --> 432k in 27 min 7 sec
16 x m4.2xlarge	1024	167M in 27 min 40 sec	8.1B --> 814k in 56 min 14 sec

Let's eyeball what's in `links_rdd`. Note that most domains also contain the empty string in their links. This is output by `urlparse.urlparse`, and suggests that the linked URL was in the same domain, e.g. a local file path.

See https://docs.python.org/2/library/urlparse.html (https: //docs.python.org/2/library/urlparse.html).

Took 0 sec. Last updated by anonymous at September 02 2017, 2:45:33 PM.

FINISHED

```
%pyspark

outputURI = "s3://billsdata.net/CommonCrawl/webgraph_%d_WAT_files" % nfiles

codec = "org.apache.hadoop.io.compress.GzipCodec"
links_rdd.saveAsTextFile(outputURI, codec)
```

Took 4 sec. Last updated by anonymous at September 02 2017, 12:15:42 PM. (outdated)

Read from the S3 stored files: FINISHED

Took 0 sec. Last updated by anonymous at September 02 2017, 2:47:21 PM.

FINISHED

```
%pyspark

from __future__ import print_function

inputURI = "s3://billsdata.net/CommonCrawl/webgraph_1024_files/"

#links_rdd = sc.textFile(inputURI).map(eval)
#links_rdd.cache()

domain_link_count = links_rdd\
                    .map(lambda x: [len(x['out']), sum(x['out'].values())])\
                    .aggregate((0, 0, 0),
                               lambda acc, value: (acc[0] + 1, acc[1] + value[0], acc[2]
                               lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1],
print("Nr domains: %18d" % domain_link_count[0])
print("Nr domain links: %13d" % domain_link_count[1])
```

```
Nr domains:             814114
Nr domain links:      24412865
Nr page links:      8102680487
```

Took 0 sec. Last updated by anonymous at September 02 2017, 3:25:07 PM.

FINISHED

```
%pyspark

from pprint import pprint

sample = links_rdd.take(10)
for x in sample:
    #x['out'].pop('', None)
    pprint(x)
```

```
{'domain': u'vkaraoke.org',
 'out': {'': 11651,
         u'mc.yandex.ru': 100,
         u'ok.ru': 100,
         u'vk.com': 100,
         u'vkaraoke.org': 3051,
         u'www.facebook.com': 100}}
{'domain': u'kidsactivitycenter.com', 'out': {u'mcc.godaddy.com': 1}}
{'domain': u'kkbelter.hu',
 'out': {'': 9,
         u'belsoepiteszet.kkbelter.hu': 2,
         u'foto.kkbelter.hu': 2,
         u'www.kkbelter.hu': 1}}
{'domain': u'parismp.com', 'out': {'': 11, u'www.pcdepot.co.jp': 1}}
{'domain': u'adonizm.com',
 'out': {'': 233,
         u'adonizm.com': 11443,
         u'adonizmdotcom.tumblr.com': 81
```

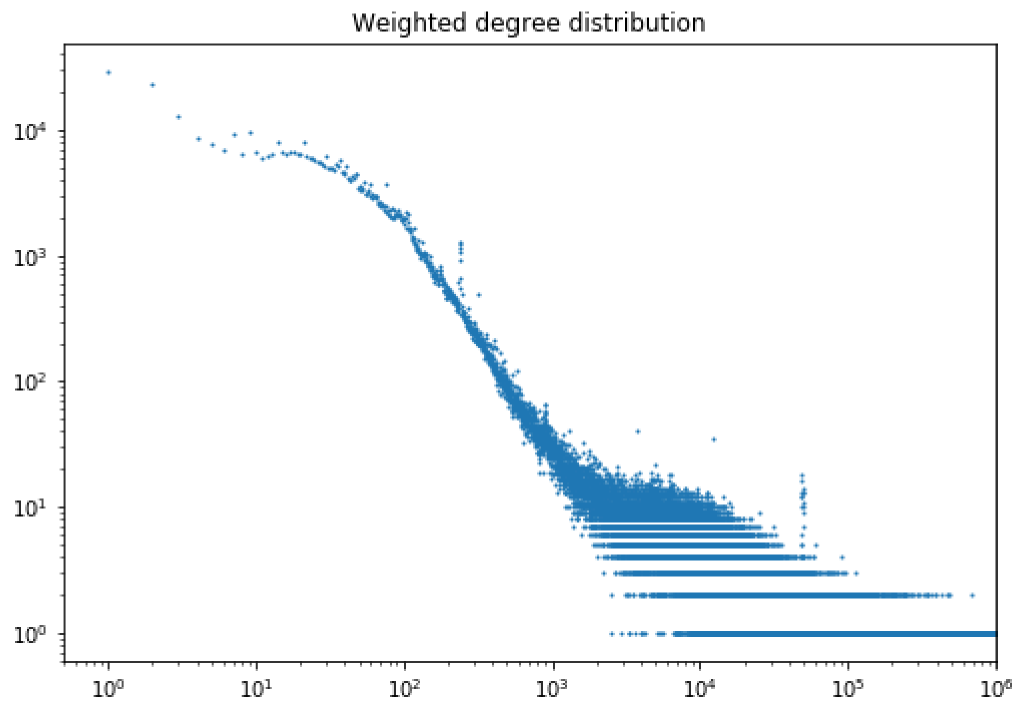
Took 0 sec. Last updated by anonymous at September 02 2017, 2:51:35 PM.

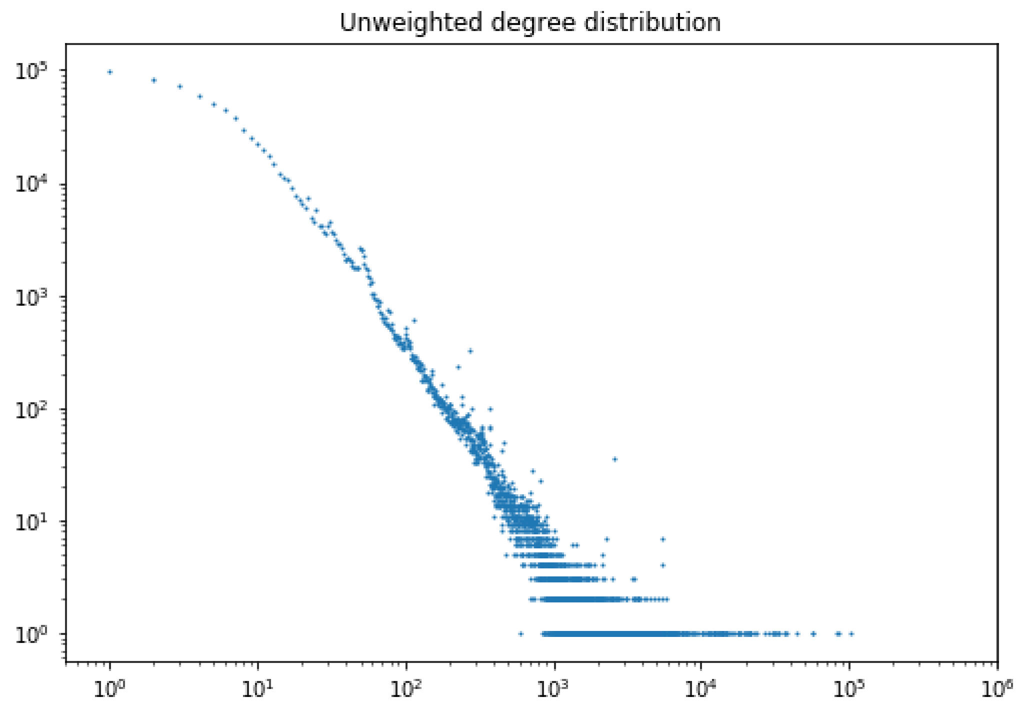
Let's view the out-degree distribution:

READY

FINISHED

```
%pyspark

from collections import Counter
import matplotlib.pyplot as plt

def degree(record):
    record.pop('', None)
    return [len(record['out'].keys()), sum(record['out'].values())]

out_degree = links_rdd.map(degree)
wtd_degree = out_degree.map(lambda x: x[1]).collect()
unwtd_degree = out_degree.map(lambda x: x[0]).collect()

wtd_distribution = Counter(wtd_degree)
unwtd_distribution = Counter(unwtd_degree)

plt.scatter(wtd_distribution.keys(), wtd_distribution.values(), s=1.0)
plt.xlim([0.5,1e06])
plt.xscale("log")
plt.yscale("log")
plt.title("Weighted degree distribution")
plt.show()

plt.scatter(unwtd_distribution.keys(), unwtd_distribution.values(), s=1.0)
plt.xlim([0.5,1e06])
plt.xscale("log")
plt.yscale("log")
plt.title("Unweighted degree distribution")
plt.show()

"""
plt.scatter(unwtd_degree, wtd_degree, s=0.2)
plt.xscale("log")
plt.yscale("log")
plt.title("Weighted against unweighted degrees")
plt.show()
"""
```





'\nplt.scatter(unwtd_degree, wtd_degree, s=0.2)\nplt.xscale("log")\nplt.yscale("log")\nplt.title("Weighted against unweighted degrees")\nplt.show()\n'

Took 6 sec. Last updated by anonymous at September 02 2017, 3:26:14 PM.

```
%pyspark

links_rdd.unpersist()
```

FINISHED

```
PythonRDD[53] at RDD at PythonRDD.scala:48
```

Took 0 sec. Last updated by anonymous at September 02 2017, 3:26:23 PM.

```
%pyspark
```

READY