FINISHED

Building domain features from WAT

What this notebook does:

Extracts domain string signaturs and uses these to construct feature vectors for domains.

Took 0 sec. Last updated by anonymous at October 20 2017, 3:31:55 PM.

FINISHED

```
%pyspark

from __future__ import print_function

nfiles = 1024
inputURI = "s3://billsdata.net/CommonCrawl/domain_paths_from_%d_WAT_files/" % nfiles

domains_rdd = sc.textFile(inputURI).map(eval)

domain_uri_count = domains_rdd\
                    .map(lambda x: [len(x['path_set']),
                                    sum( [ len(uri[0]) for uri in x['path_set'] ] ),
                                    sum([len(uri[0].encode('utf-8')) for uri in x['path_se
                    .aggregate((0, 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: %15d" % domain_uri_count[0])
print("Nr page URIs: %13d" % domain_uri_count[1])
print("Nr URI chars: %13d" % domain_uri_count[2])
print("Nr URI bytes: %13d" % domain_uri_count[3])
```

```
Nr domains:         4285451
Nr page URIs:     420643485
Nr URI chars:   20332763778
Nr URI bytes:   20381326882
```

Took 1 min 55 sec. Last updated by anonymous at October 20 2017, 4:26:00 PM.

FINISHED

nfiles = 1
Nr domains: 99807
Nr page URIs: 1691436
Nr URI chars: 63947884
Nr URI bytes: 64199858

nfiles = 128
Nr domains: 1197074
Nr page URIs: 70897255
Nr URI chars: 3300575852
Nr URI bytes: 3308899496

nfiles = 1024
Nr domains: 4285451
Nr page URIs: 420643485
Nr URI chars: 20332763778
Nr URI bytes: 20381326882

Write to S3 a single string for all domains:

Took 0 sec. Last updated by anonymous at October 20 2017, 4:18:22 PM.

FINISHED

```
%pyspark

def domain_string(domain, path_set):
    """
    Takes domain and concatenates with path URIs separated by newlines.
    """
    out = domain + '\n' + '\n'.join(sorted([x[0] for x in list(path_set)])) + '\n\n\n'
    return out

domain_string_rdd = domains_rdd\
    .map(lambda x: domain_string(x['domain'], x['path_set']))
domain_string_rdd.cache()

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

```
PythonRDD[386] at RDD at PythonRDD.scala:48
```

Took 0 sec. Last updated by anonymous at October 20 2017, 4:30:16 PM. (outdated)

FINISHED

Cluster	nr files	nr domains	nr page URIs	nr chars	time
16 x m4.large	1	168k	1.8M	63.7M	6 sec
16 x m3.xlarge	128	2.6M	71.8M	3.26B	1 min 4 sec
16 x m4.large	128	2.6M	71.8M	3.26B	48 sec

To concatenate into a single gzip file (may need to mount extra local disk space):

```
$ aws s3 sync
s3://billsdata.net/CommonCrawl/domain_string_from_1024_WAT_files/ ./tmp
$ gunzip -c ./tmp/part*.gz | cat | gzip -c > ./tmp/big_domain_string_1024.gz
$ rm ./tmp/part* ./tmp/_SUCCESS
$ aws s3 sync ./tmp s3://billsdata.net/CommonCrawl/
$ rm -r ./tmp
```

Took 0 sec. Last updated by anonymous at October 20 2017, 4:42:33 PM.

FINISHED

```
%pyspark

for x in domain_string_rdd.takeSample(False, 10):
    print(x)
```

```
trezvyi64.ru
/2011/04/
/2011/05/
/2011/10/
/2011/12/
```

```
/2012/01/
/2012/03/
/2012/04/
/2012/05/
/2012/06/
/2012/09/
/2012/12/
/2013/03/
/2013/04/
/2013/05/
/2013/06/
/2013/07/
```

Took 2 min 31 sec. Last updated by anonymous at October 20 2017, 4:32:54 PM.

FINISHED

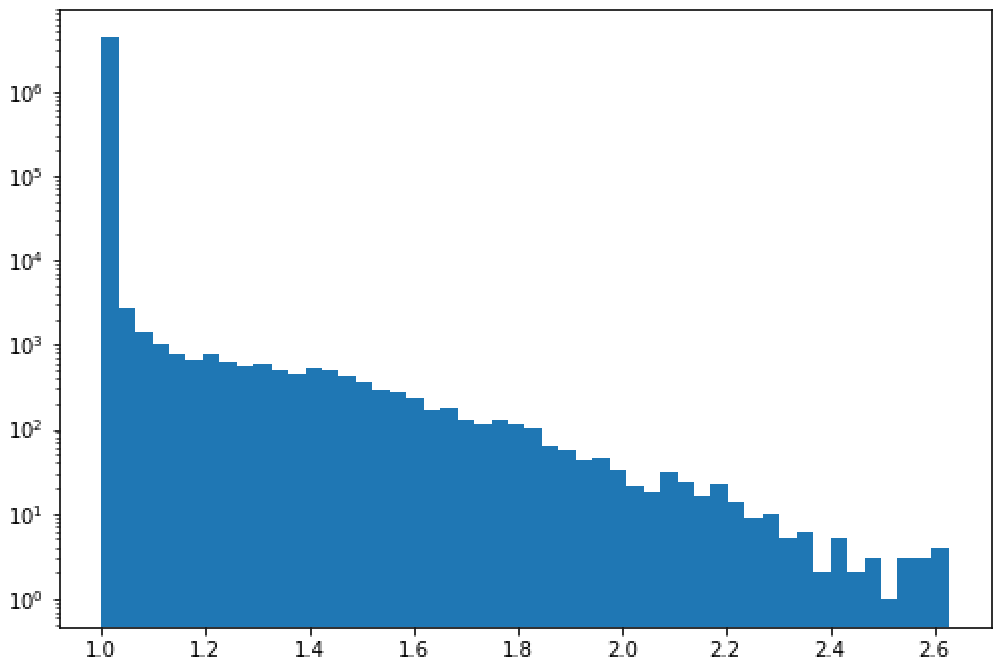
```
%pyspark

def nonlatin_detector(str):
    """
    Computes the excess nr bytes over nr characters in a string.
    """
    N = len(str)
    return float(len(str.encode('utf-8')))/N

nonlatin_dist = domain_string_rdd.map(nonlatin_detector).collect()
```

Took 4 sec. Last updated by anonymous at October 20 2017, 4:33:19 PM.

FINISHED

```
%pyspark

import matplotlib.pyplot as plt

nonlatin = [x for x in nonlatin_dist if x > 1.0]

print("Nr domains: %8d" % len(nonlatin_dist))
print("Nr non-latin: %6d" % len(nonlatin))
print("Min non-latin score: %.10f" % min(nonlatin))

plt.hist(nonlatin_dist, bins=50)
plt.yscale("log")
plt.show()
```

```
Nr domains:  4285451
Nr non-latin:  51969
Min non-latin score: 1.0000000237
```



Took 4 sec. Last updated by anonymous at October 20 2017, 4:33:31 PM.

FINISHED

```
%pyspark

threshold = 1.05

nonlatin_rdd = domain_string_rdd\
        .filter(lambda x: nonlatin_detector(x) > threshold)

print("Nr domains: %d" % nonlatin_rdd.count())
for x in nonlatin_rdd.takeSample(False, 10):
    print(x)
```

```
Nr domains: 12287
gateway-hotel.co.jp
/
www.voebb.de
/aDISWeb/app
/
korona-m.com
/images/cart.png
/images/facebook.png
/images/logo.png
/images/magnifier.png
/products/thumbs/bireni-fastatsi-100gr-2016-05-24-11-48-39.jpg
/products/thumbs/bireni-fastatsi-200gr-2016-05-24-11-49-35.jpg
/products/thumbs/parzhena-tsarevitsa-100-gr-2016-05-24-11-56-36.jpg
/products/thumbs/parzhena-tsarevitsa-200-gr-2016-05-24-11-57-14.jpg
/products/thumbs/pechen-sham-fastak-100-gr-2016-05-24-11-55-47.jpg
/products/thumbs/pechen-sham-fastak-200-gr-2016-05-24-11-56-11.jpg
```

Took 10 sec. Last updated by anonymous at October 20 2017, 4:44:21 PM.

FINISHED

```
%pyspark

outputURI = "s3://billsdata.net/CommonCrawl/domain_string_nonlatin_1pt05_12287_from_%d_WAT
```

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

Took 6 sec. Last updated by anonymous at October 20 2017, 4:44:59 PM.

Now let's look at basic statistics of the path URI for a domain… READY

FINISHED

```
%pyspark

import re
from math import log
from collections import Counter

def hx(i):
    """
    Normalised 2-char hex representation of 0-255
    """
    a = hex(i)[2:]
    if len(a)<2: a = ''.join(['0',a])
    return a
hexabet = [hx(x) for x in range(256)]

def depth(uri):
    return uri[:-1].count('/')

def length(uri):
    return len(uri) - uri.count('/')

def byte_count(str):
    out = dict([(x,0) for x in hexabet])
    ct = dict(Counter([c.encode('hex') for c in str.encode('utf-8')]))
    for k in out.keys():
        if k in ct.keys():
            out[k] += ct[k]
    out = [v[1] for v in sorted(out.iteritems(), key=lambda (k,v): k)]
    out = [float(x)/sum(out) for x in out]
    return out

def string_features_v1(str):
    """
    Coarse first version of a feature vector for a string.
    A placeholder for stronger versions.
    """
    N = float(length(str))
    if N==0: return None
    U = float(len(str.encode('utf-8')))
    D = depth(str)
    b = len(re.findall(r'\.', str))/N
    c = len(re.findall(r'-', str))/N
    d = len(re.findall(r'_', str))/N
    cap = len(re.findall(r'[A-Z]', str))/N
    num = len(re.findall(r'[0-9]', str))/N
    return [log(N), log(U), D, b, c, d, num, cap]

def string_features_v2(str):
    """
    Version 2: combine the byte distribution with the previous string statistics.
    """
    return byte_count(str) + string_features_v1(str)
```

Took 0 sec. Last updated by anonymous at October 20 2017, 5:04:01 PM.

```
%pyspark                                                                            FINISHED

def feature_extractor(x):
    str_set = [s[0] for s in x['path_set'] if (string_features_v1(s[0]) is not None) and (
    a = [string_features_v1(s) for s in str_set]
    b = [string_features_v2(s) for s in str_set]
    return (x['domain'], a, b)

page_feature_rdd = domains_rdd.map(feature_extractor)
page_feature_rdd.cache()

page_feature_rdd.take(1)
```

```
[(u'urbanhippie.co.nz', [[4.174387269895637, 4.248495242049359, 5, 0.015384615384615385, 0.
09230769230769231, 0.0, 0.2923076923076923, 0.06153846153846154]], [[0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.08571428571428572, 0.014285714285714285, 0.07142857142857142, 0.04
285714285714286, 0.02857142857142857, 0.02857142857142857, 0.02857142857142857, 0.014285714
285714285, 0.04285714285714286, 0.04285714285714286, 0.02857142857142857, 0.0, 0.0142857142
85714285, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.014285714285714285,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.014285714285714285, 0.0, 0.0, 0.014285714285714285, 0.0,
 0.0, 0.014285714285714285, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.014285714285714285, 0.0, 0.02857142857142857, 0.014285714285714285, 0.0428571428571428
6, 0.014285714285714285, 0.014285714285714285, 0.0, 0.02857142857142857, 0.0142857142857142
85, 0.0, 0.02857142857142857, 0.0, 0.02857142857142857, 0.07142857142857142, 0.057142857142
85714, 0.0, 0.02857142857142857, 0.02857142857142857, 0.02857142857142857, 0.02857142857142
857, 0.0, 0.014285714285714285, 0.0, 0.014285714285714285, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
```

Took 43 min 58 sec. Last updated by anonymous at October 20 2017, 5:52:12 PM.

READY

The plot below takes a random sample of domains, and computes feature vectors v1 and v2 from the path URIs for each domain.
Dots are URIs, colours are domains.

```
%pyspark                                                                            ERROR

ndomains = 6
minpaths = 50

some_domains = page_feature_rdd\
                 .filter(lambda x: len(x[1]) >= minpaths)\
                 .takeSample(False, ndomains)
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-7418246470775012421.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-7418246470775012421.py", line 360, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 5, in <module>
  File "/usr/lib/spark/python/pyspark/rdd.py", line 479, in takeSample
    initialCount = self.count()
  File "/usr/lib/spark/python/pyspark/rdd.py", line 1041, in count
    return self.mapPartitions(lambda i: [sum(1 for _ in i)]).sum()
  File "/usr/lib/spark/python/pyspark/rdd.py", line 1032, in sum
```

```
    return self.mapPartitions(lambda x: [sum(x)]).fold(0, operator.add)
  File "/usr/lib/spark/python/pyspark/rdd.py", line 906, in fold
    vals = self.mapPartitions(func).collect()
  File "/usr/lib/spark/python/pyspark/rdd.py", line 809, in collect
    port = self.ctx._jvm.PythonRDD.collectAndServe(self._jrdd.rdd())
```

Took 2 sec. Last updated by anonymous at October 20 2017, 5:05:52 PM.

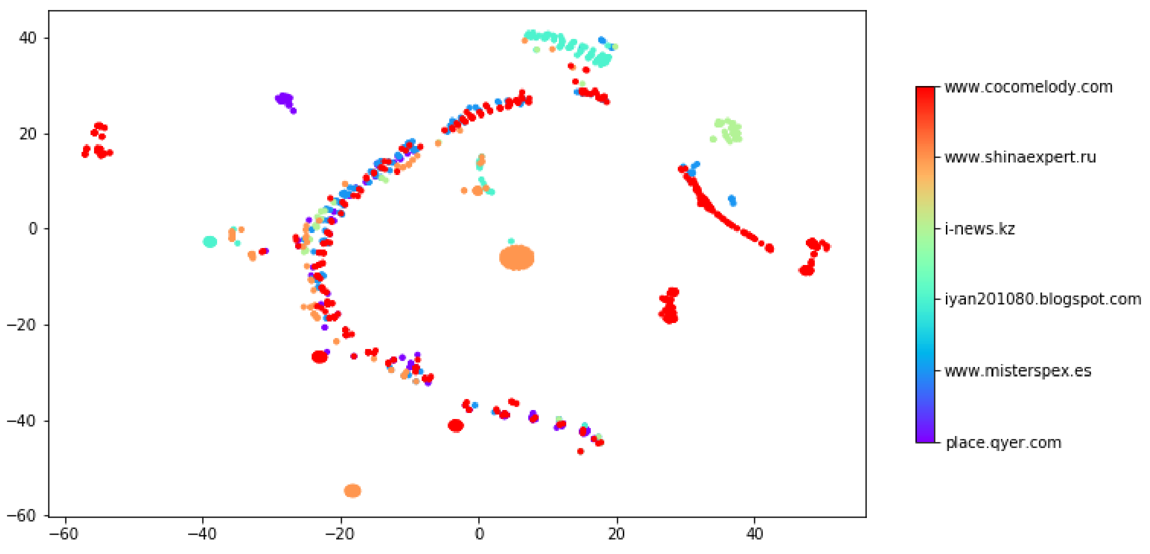
READY

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 1222 samples in 0.001s...
[t-SNE] Computed neighbors for 1222 samples in 0.038s...
[t-SNE] Computed conditional probabilities for sample 1000 / 1222
[t-SNE] Computed conditional probabilities for sample 1222 / 1222
[t-SNE] Mean sigma: 0.000000
[t-SNE] KL divergence after 250 iterations with early exaggeration: 48.648224
[t-SNE] Error after 1000 iterations: 0.210676
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 1222 samples in 0.008s...
[t-SNE] Computed neighbors for 1222 samples in 0.302s...
[t-SNE] Computed conditional probabilities for sample 1000 / 1222
[t-SNE] Computed conditional probabilities for sample 1222 / 1222
[t-SNE] Mean sigma: 0.080659
[t-SNE] KL divergence after 250 iterations with early exaggeration: 52.923714
[t-SNE] Error after 1000 iterations: 0.406656
```

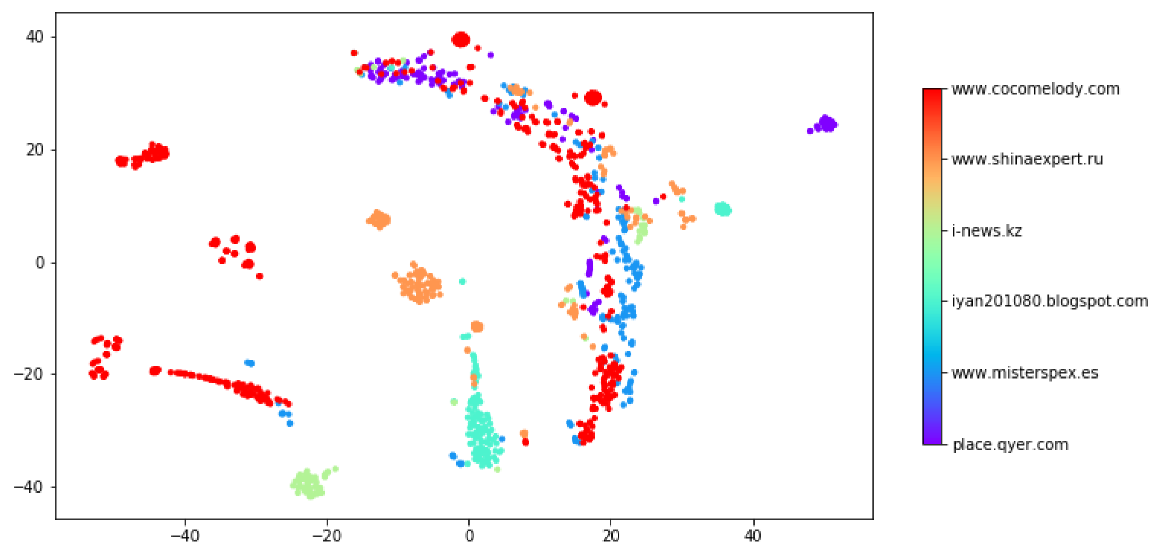
READY

```
%pyspark

import matplotlib.pyplot as plt
for proj in [proj_2d_v1, proj_2d_v2]:
    fig, ax = plt.subplots(figsize=(12,6))
    cax = ax.scatter(proj[:,0], proj[:,1], s=10.0, c=col, edgecolors='face', cmap='rainbow
    cbar = fig.colorbar(cax, ticks=range(ndomains), shrink=0.7)
    cbar.ax.set_yticklabels([dom[0] for dom in some_domains])  # vertically oriented colorl
    plt.show()
```

```
[<matplotlib.text.Text object at 0x7f1d16015110>, <matplotlib.text.Text object at 0x7f1d160
208d0>, <matplotlib.text.Text object at 0x7f1d15fdaad0>, <matplotlib.text.Text object at 0x
7f1d15fd1610>, <matplotlib.text.Text object at 0x7f1d15fda450>, <matplotlib.text.Text objec
t at 0x7f1d15fe4390>]
```



```
[, , , , , ]
```



READY

```
%pyspark

page_feature_rdd.unpersist()
domains_rdd.unpersist()
```

```
PythonRDD[70] at RDD at PythonRDD.scala:48
```

READY

Export domain feature vectors

READY

```
%pyspark

nfiles = 1024
inputURI = "s3://billsdata.net/CommonCrawl/domain_paths_from_%d_WAT_files/" % nfiles
domains_rdd = sc.textFile(inputURI).map(eval)
domains_rdd.cache()

def domain_features(domain, path_set):
    """
    Takes domain + set of paths as output by parse_urls() and
    applies extracts statistics of the signature string.
    """
    return string_features_v2(domain_string(domain, path_set))

def feature_extractor(x):
    return (x['domain'], domain_features(x['domain'], x['path_set']))

domain_feature_rdd = domains_rdd.map(feature_extractor)
```

READY

```
%pyspark

outputURI = "s3://billsdata.net/CommonCrawl/domain_basic_string_feature_vectors_from_%d_WA
codec = "org.apache.hadoop.io.compress.GzipCodec"
```

```
domain_feature_rdd.saveAsTextFile(outputURI, codec)
```

Timings: READY

Cluster	nr files	nr domains	time
16 x m4.large	128	2.6M	40 min 7 sec

Let's check what we've just written:

```
%pyspark                                                                READY

inputURI = "s3://billsdata.net/CommonCrawl/domain_hex_feature_vectors_from_%d_WAT_files" %
features_rdd = sc.textFile(inputURI).map(eval)
print("Nr domains:", features_rdd.count())
print(features_rdd.take(1))
```

```
('Nr domains:', 2626203)
[(u'www.iggl.de', [3.6375861597263857, 0.5, 0.0, 0.0, 0.02564102564102564, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.358974358974359, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.05128205128205128, 0.0, 0.0, 0.0, 0.05
128205128205128, 0.0, 0.02564102564102564, 0.02564102564102564, 0.15384615384615385, 0.2051
2820512820512, 0.0, 0.02564102564102564, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.02564102564102564, 0.0,
 0.05128205128205128, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])]
```

```
%pyspark                                                                READY
```