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 15 2017, 11:59:35 AM.

```
%pyspark

domains_rdd.unpersist()
domain_string_rdd.unpersist()
```

FINISHED

```
PythonRDD[211] at RDD at PythonRDD.scala:48
```

Took 0 sec. Last updated by anonymous at October 15 2017, 3:18:10 PM.

```
%pyspark

from __future__ import print_function

nfiles = 1
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) for uri in x['path_set']]),
                                    sum([len(uri.encode('utf-8')) for uri in x['path_set']
                    .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])
```

FINISHED

```
Nr domains:          168033
Nr page URIs:       1782572
Nr URI chars:      63676121
Nr URI bytes:      63928095
```

Took 4 sec. Last updated by anonymous at October 15 2017, 3:18:24 PM.

FINISHED

nfiles = 1
Nr domains: 168033
Nr page URIs: 1782572
Nr URI chars: 63676121
Nr URI bytes: 63928095

nfiles = 128
Nr domains: 2626203
Nr page URIs: 71799497
Nr URI chars: 3259974688
Nr URI bytes: 3268298469

nfiles = 1024
Nr domains: 10802408
Nr page URIs: 420975127
Nr URI chars: 19980667843
Nr URI bytes: 20029236547

Write to S3 a single string for all domains:

Took 0 sec. Last updated by anonymous at October 14 2017, 8:10:18 PM.

```
%pyspark

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

FINISHED

Took 0 sec. Last updated by anonymous at October 15 2017, 3:18:48 PM. (outdated)

| 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 |

FINISHED

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_128_WAT_files/
./tmp
$ gunzip -c ./tmp/part*.gz | cat | gzip -c > ./tmp/big_domain_string_128.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 14 2017, 12:10:53 PM.

```
%pyspark

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

FINISHED

```
softportal.com
```

```
/img/draugda.png
www.seksgratka.com
/
www.canadianprogressiveworld.com
search.sify.com
/
www.okcupid.com
/
/about
/careers
/legal/privacy
/legal/safety-tips
/legal/terms
/login
/mobile
/press
/profile/Encarna530
```

Took 4 sec. Last updated by anonymous at October 15 2017, 3:19:00 PM.

```
%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()
```

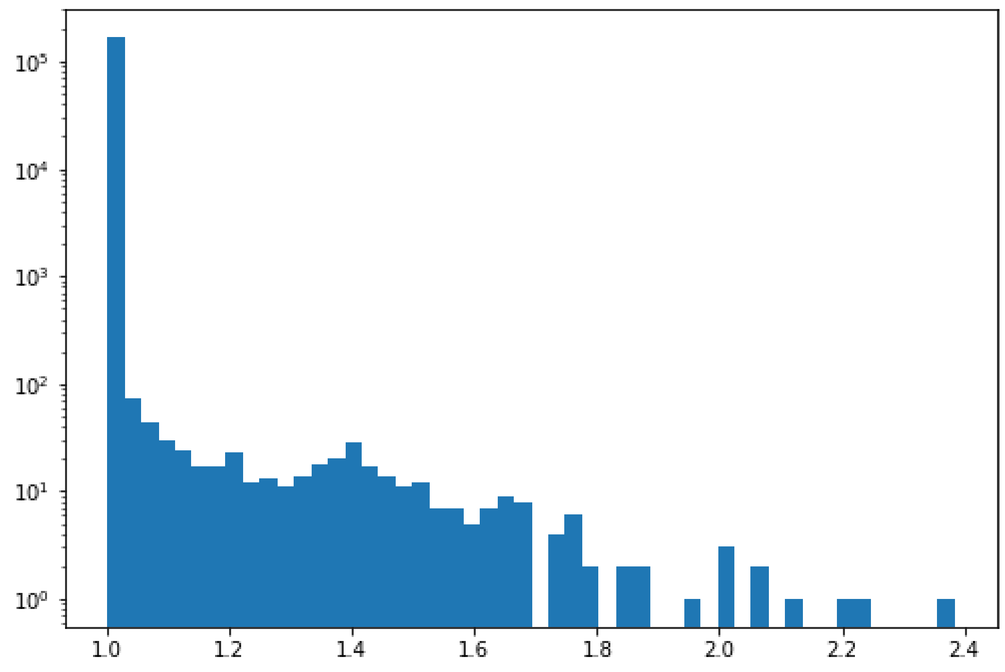
FINISHED

Took 1 sec. Last updated by anonymous at October 15 2017, 3:19:10 PM.

```
%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()
```

FINISHED

```
Nr domains:   168033
Nr non-latin:   1154
Min non-latin score: 1.0000065218
```



Took 4 sec. Last updated by anonymous at October 15 2017, 3:19:17 PM.

```
%pyspark

threshold = 1.0

nonlatin_rdd = domain_string_rdd\
        .map(lambda s: [s, nonlatin_detector(s)])\
        .filter(lambda x: x[1] > threshold)

print("Nr domains: %d" % nonlatin_rdd.count())
for x in nonlatin_rdd.takeSample(False, 10):
    print(x[0])
```

FINISHED

```
Nr domains: 1154
ja.dbpedia.org
/resource/差別
satelonline.kz
/
/about_us
/adresa-servisnih-centrov
/akcia
/akkumulyatori-5
/akkumulyatori-5/aaaaa-palchikovie-10969
/akkumulyatori-5/acer-8422
/akkumulyatori-5/alcatel-8418
/akkumulyatori-5/alkalinovie-15538
/akkumulyatori-5/anydata-18763
/akkumulyatori-5/apple-8416
/akkumulyatori-5/asus-8423
/akkumulyatori-5/benqsiemens-7339
/akkumulyatori-5/blackberry-8417
```

Took 1 sec. Last updated by anonymous at October 15 2017, 3:19:33 PM.

```
%pyspark
```

FINISHED

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

Took 20 sec. Last updated by anonymous at October 15 2017, 11:10:17 AM.

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 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(len(str))
    if N==0: return None
    U = float(len(str.encode('utf-8')))

    a = len(re.findall(r'/', str))/N
    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), a, 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 15 2017, 3:20:24 PM.

FINISHED

```
%pyspark
```

```
def feature_extractor(x):
    str_set = [s for s in x['path_set'] if (string_features_v1(s) is not None) and (string
    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()
```

```
PythonRDD[249] at RDD at PythonRDD.scala:48
```

Took 0 sec. Last updated by anonymous at October 15 2017, 3:20:39 PM.

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.

FINISHED

Took 0 sec. Last updated by anonymous at October 15 2017, 3:22:51 PM.

FINISHED

```
%pyspark

ndomains = 6
minpaths = 50

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

Took 4 sec. Last updated by anonymous at October 15 2017, 3:29:40 PM.

FINISHED

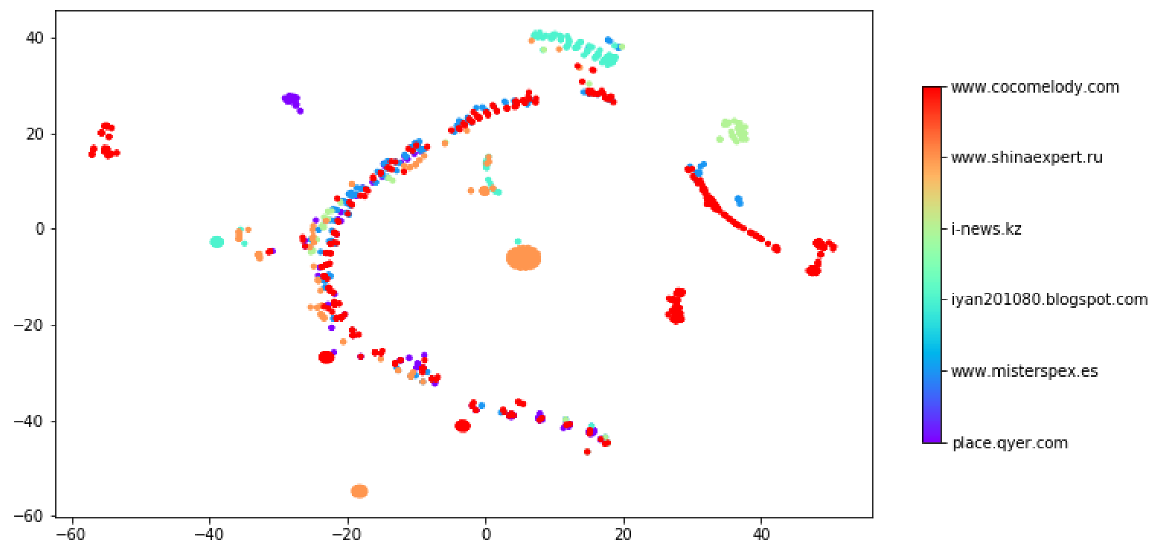
```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 1222 samples in 0.001s...
[t-SNE] Computed neighbors for 1222 samples in 0.038s...
[t-SNE] Computed conditional probabilities for sample 1000 / 1222
[t-SNE] Computed conditional probabilities for sample 1222 / 1222
[t-SNE] Mean sigma: 0.000000
[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
```

Took 45 sec. Last updated by anonymous at October 15 2017, 3:30:27 PM.

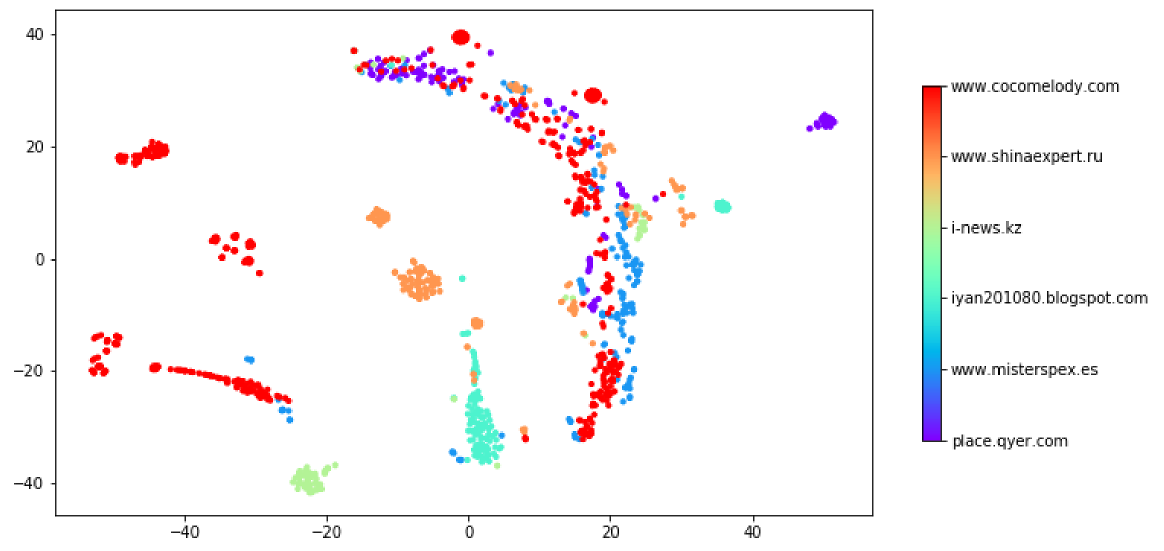
FINISHED

```
%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>]
```



[, , , , , ]



Took 4 sec. Last updated by anonymous at October 15 2017, 3:30:50 PM.

```
%pyspark

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

READY

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

READY

## Export domain feature vectors

```
%pyspark                                                                READY

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)
```

```
%pyspark                                                                READY

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,
%pyspark
```

READY