# Paul 4 - initial dom…

```
%pyspark                                                                    FINISHED

# Zeppelin notebook to create domain summaries based on the May/Jun/Jul 2017
    CommonCrawl graph
# as per description here: http://commoncrawl.org/2017/08/webgraph-2017-may-june-july/
# PJ - 29 Sept 2017

import boto
from pyspark.sql.types import *

LIMIT=100000 # TODO - remove temporary limit to run full summaries!

# Import the PLD vertices list as a DataFrame
pld_schema=StructType([StructField("ID", StringType(), False), StructField("PLD",
    StringType(), False)])
pld_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul
    /domaingraph/vertices.txt.gz")
temp_pld = pld_txt.map(lambda k: k.split()) # By default, splits on whitespace, which
    is what we want
pld_df=temp_pld.toDF(pld_schema).limit(LIMIT)
```

```
+---+-------+
| ID|    PLD|
+---+-------+
|  0|  aaa.a|
|  1| aaa.aa|
|  2|aaa.aaa|
+---+-------+
only showing top 3 rows
DataFrame[ID: string, PLD: string]
```

```
%pyspark                                                                    FINISHED

# Next import the PLD edges as a DataFrame
pld_edges_schema=StructType([StructField("src", StringType(), False), StructField
    ("dst", StringType(), False)])
pld_edges_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may
    -jun-jul/domaingraph/edges.txt.gz")
temp_edges_pld = pld_edges_txt.map(lambda k: k.split()) # By default, splits on
    whitespace, which is what we want
```

```
+---+--------+
|src|     dst|
+---+--------+
|  2| 9193244|
| 20|75600973|
| 21|46356172|
+---+--------+
only showing top 3 rows
DataFrame[src: string, dst: string]
```

```
%pyspark
```

FINISHED

```
# Load the host-level graph vertices in the same way
host_schema=StructType([StructField("hostid", StringType(), False), StructField("host"
    , StringType(), False)])
host_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun
    -jul/hostgraph/vertices.txt.gz")
temp_host = host_txt.map(lambda k: k.split()) # By default, splits on whitespace,
    which is what we want
host_df=temp_host.toDF(host_schema).limit(LIMIT*10) # TODO - remove temporary limit to
    run full summaries!
```

```
+------+-------+
|hostid|   host|
+------+-------+
|     0|  aaa.a|
|     1| aaa.aa|
|     2|aaa.aaa|
+------+-------+
only showing top 3 rows
DataFrame[hostid: string, host: string]
```

```
%pyspark
```

FINISHED

```
# Load in harmonic centrality and page-ranks, and join based on reverse domain name
# Format: #hc_pos #hc_val #pr_pos #pr_val #host_rev
#pr_schema=StructType([StructField("hc_pos", StringType(), False), StructField
    ("hc_val", StringType(), False), StructField("pr_pos", StringType(), False),
    StructField("pr_val", StringType(), False), StructField("host_rev", StringType(),
    False)])
pr_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul
    /domaingraph/ranks.txt.gz")
header=pr_txt.first()
pr_txt=pr_txt.filter(lambda x: x!=header)
temp_pr = pr_txt.map(lambda k: k.split()) # By default, splits on whitespace, which is
    what we want
```

```
+-------+--------+-------+-------------------+--------------+
|#hc_pos| #hc_val|#pr_pos|            #pr_val|      host_rev|
+-------+--------+-------+-------------------+--------------+
|      1|24989952|      1| 0.0155264576161686|  com.facebook|
|      2|22460880|      3|0.00866038900847366|   com.twitter|
|      3|22097514|      2| 0.0128827315785546|com.googleapis|
+-------+--------+-------+-------------------+--------------+
only showing top 3 rows
DataFrame[#hc_pos: string, #hc_val: string, #pr_pos: string, #pr_val: string, host_rev:
string]
```

```
%pyspark #--packages graphframes:graphframes:0.5.0-spark2.1-s_2.11                FINISHED

# We now have everything we need in these four dataframes to create the summaries
    based on joins and counts... it's just going to take a while!

# First, let's compute the in-degree and out-degree for each PLD, using GraphFrames

# Rename from_id and to_id in edges to src and dst for GraphFrame
#pld_edges_df2 = pld_edges_df.select(col("from_id").alias("src"), col("to_id").alias
    ("dst"))
# Make a GraphFrame and compute in-degrees
#g = GraphFrame(pld_df, pld_edges_df2)
#g.inDegrees.show(5)
```

TODO!

```
%pyspark                                                                RUNNING 0%

# Next, let's count the number of host domains for each PLD, based on joining the host
    and PLD vertex data frames
from pyspark.sql.functions import concat, col, when, lit
# TODO: This is slow because it doesn't exploit the host ordering!
pld_df_tmp=pld_df.join(host_df,(host_df.host==pld_df.PLD) | (host_df.host.startswith
    (concat(pld_df.PLD, lit("."))) ),'rightouter') # Join based on exact match, or PLD
    + "."
pld_df_tmp.show(10)
host_counts=pld_df_tmp.groupBy("PLD").count() # Counts total number of hosts,
    including when host==PLD
host_counts.show(10)
```

```
+---+--------+------+-----------------+
| ID|     PLD|hostid|             host|
+---+--------+------+-----------------+
|  0|   aaa.a|     0|            aaa.a|
|  1|  aaa.aa|     1|           aaa.aa|
|  2| aaa.aaa|     2|          aaa.aaa|
|  2| aaa.aaa|     3|      aaa.aaa.aaa|
|  2| aaa.aaa|     4|  aaa.aaa.aaa.aaa|
|  2| aaa.aaa|     5| aaa.aaa.aaa.next|
|  2| aaa.aaa|     6|aaa.aaa.aaa.other|
|  2| aaa.aaa|     7|     aaa.aaa.aaaa|
|  3|aaa.aaaa|     8|         aaa.aaaa|
|  3|aaa.aaaa|     9|     aaa.aaaa.aaa|
+---+--------+------+-----------------+
only showing top 10 rows
```

Started 15 minutes ago.

PENDING

```
%pyspark

# Next, compute whether each pld appears as a host by itself using a leftOuter join
    and append as an extra column
from pyspark.sql.functions import col, when, lit
pld_host_test = when(col("host").isNull(), lit("false")).otherwise(lit("true"))
pld_df_joined2=pld_df_joined.join(host_df, pld_df_joined.PLD==host_df.host,
    'leftOuter').drop("hostid").withColumn('PLDisHost?', pld_host_test).drop("host")
```

PENDING

```
%pyspark

# Next, join with the harmonic centrality and page-rank for each domain
pld_df_joined3=pld_df_joined2.join(pr_df, pr_df.host_rev==pld_df_joined2.PLD,
    "leftOuter").drop("#hc_pos").drop("#pr_pos").drop("host_rev").withColumnRenamed
    ("#hc_val","HarmonicCentrality").withColumnRenamed("#pr_val","PageRank")
```

```
+----+-------------------+--------+----------+------------------+--------+
|  ID|                PLD|NumHosts|PLDisHost?|HarmonicCentrality|PageRank|
+----+-------------------+--------+----------+------------------+--------+
| 120|             abc.web|       1|     false|              null|    null|
| 311|             ac.8411|       1|     false|              null|    null|
| 713|              ac.bgc|       1|     false|              null|    null|
| 871|          ac.casinos|       1|      true|              null|    null|
|1014|ac.cosmopolitanun...|       1|      true|              null|    null|
|1089|            ac.dibru|       1|      true|              null|    null|
|1435|          ac.gorilla|       1|      true|              null|    null|
|2476|          ac.philter|       1|      true|              null|    null|
|3138|              ac.ula|       1|     false|              null|    null|
|3145|         ac.umedalen|       2|      true|              null|    null|
|3373|              ac.yui|       2|      true|              null|    null|
|3484|   academy.alphastar|       1|      true|              null|    null|
|3768|    academy.cirulnik|       1|      true|              null|    null|
|3787|       academy.cocoa|       1|      true|              null|    null|
|3882|academy.dental-coach|       1|      true|              null|    null|
```

```
%pyspark

# Save final table to S3 in compressed CSV format
outputURI="s3://billsdata.net/CommonCrawl/domain_summaries/"
codec="org.apache.hadoop.io.compress.GzipCodec"
pld_df_joined3.coalesce(1).write.format('com.databricks.spark.csv').options(header
```

PENDING

```
%pyspark
```

FINISHED