

# Paul 4 - initial dom...

%pyspark

FINISHED

```
# Zeppelin notebook to create domain summaries based on the May/Jun/Jul 2017 CommonCrawl
# 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", String
pld_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul/
temp_pld = pld_txt.map(lambda k: k.split()) # By default, splits on whitespace, which i
pld_df=temp_pld.toDF(pld_schema).limit(LIMIT)
pld_df.show(3)
pld_df.cache()
# Should have 91M domains
#print(pld_df.count())
```

```
+---+-----+
```

```
| 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"
pld_edges_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul/
temp_edges_pld = pld_edges_txt.map(lambda k: k.split()) # By default, splits on whitesp
pld_edges_df=temp_edges_pld.toDF(pld_edges_schema).limit(LIMIT*10) # TODO - remove temp
pld_edges_df.show(3)
pld_edges_df.cache()
```

```

+---+-----+
|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",
host_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul.
temp_host = host_txt.map(lambda k: k.split()) # By default, splits on whitespace, which
host_df=temp_host.toDF(host_schema).limit(LIMIT*10) # TODO - remove temporary limit to
host_df.show(3)
host_df.cache()
# Should have 1.3B hosts
#print(host_df.count())

```

```

+-----+-----+
|hostid|  host|
+-----+-----+
|      0| aaa.a|
|      1| aaa.aal
|      2|aaa.aaal
+-----+-----+
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"
("pr_val", StringType(), False), StructField("host_rev", StringType(), False)])
pr_txt=sc.textFile("s3://commoncrawl/projects/hyperlinkgraph/cc-main-2017-may-jun-jul/d
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
pr_df=temp_pr.toDF(header.split()).limit(LIMIT*100).withColumnRenamed("#host_rev", "host.
pr_df.show(3)
pr_df.cache()
#print(pr_df.count()) # Should be 91M

```

```

+-----+-----+-----+-----+-----+
|#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
```

```
# 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("d
```

```
# Make a GraphFrame and compute in-degrees
```

```
#g = GraphFrame(pld_df, pld_edges_df2)
```

```
#g.inDegrees.show(5)
```

```
# TODO: Figure out how to use GraphFrames with Zeppelin!
```

```
print("TODO!")
```

TODO!

```
%pyspark
```

FINISHED

```
# Next, let's count the number of host domains for each PLD, based on joining the host d
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(co
pld_df_tmp.show(10)
```

```
host_counts=pld_df_tmp.groupBy("PLD").count() # Counts total number of hosts, including
host_counts.show(10)
```

```
# Now rejoin the host counts with our original dataframe
```

```
pld_df_joined=pld_df.join(host_counts, pld_df.PLD==host_counts.PLD).drop(pld_df.PLD).wi
pld_df_joined.show(100)
```

```
pld_df_joined.cache()
```

```

| 4709| academy.newtown| 1|
| 5224| academy.talk| 2|
| 6064| accountant.buy-mo...| 1|
| 6198| accountant.buyfur...| 1|
| 6445| accountant.c0eul| 4|
| 6483| accountant.canadi...| 1|
| 6776| accountant.czmht| 1|
| 6826| accountant.devoch...| 1|
| 6863| accountant.dkeiil| 1|
| 7269| accountant.generi...| 1|
| 7285| accountant.generi...| 1|
| 7369| accountant.gvykeed| 3|
| 7386| accountant.hbpnj| 1|
| 7412| accountant.hghgro...| 1|
| 8048| accountant.online...| 1|
| 8236| accountant.petoug| 6|
| 8417| accountant.rhjmheq| 3|
| 8482| accountant.semdzof| 1|

```

```
%pyspark
```

FINISHED

```

# Next, compute whether each pld appears as a host by itself using a leftOuter join and
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')
pld_df_joined2.sort("NumHosts", ascending=False).show(20)
pld_df_joined2.cache()
#pld_df_joined2.groupBy("PLDisHost?").count().show()

```

```

+-----+-----+-----+-----+
| ID|          PLD|NumHosts|PLDisHost?|
+-----+-----+-----+-----+
|81492|      ar.com| 523509|      true|
|69342|      am.do| 10416|      true|
|15549| ae.blogspot|  9702|      true|
|50431|      ai.id|  7574|      true|
| 1426|      ac.go|  4106|      true|
|73298|      am.mm|  3384|      true|
|82254| ar.com.2sex|  2101|      true|
|98239| ar.com.anun|  1956|      true|
|68693|      am.cms|  1792|      true|
|65544|      am.190|  1754|      true|
|77376|      am.web|  1660|      true|
|73281|      am.mj|  1626|      true|
|52047|      ai.of|  1445|      true|
|83250| ar.com.a-e-a|  1363|     false|
|75471| am.schools|  1243|      true|

```

```
%pyspark
```

FINISHED

```
# 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",
    "HarmonicCentrality").withColumnRenamed("#pr_val", "PageRank")
pld_df_joined3.show(20)
```

110891	ac.dibrui	11	true1	null1	null1
114351	ac.gorilla1	11	true1	null1	null1
124761	ac.philter1	11	true1	null1	null1
131381	ac.ula1	11	false1	null1	null1
131451	ac.umedalen1	21	true1	1209300914.69402844088012e-091	
133731	ac.yuil	21	true1	1210521715.09160908953513e-091	
134841	academy.alphastar1	11	true1	null1	null1
137681	academy.cirulnik1	11	true1	null1	null1
137871	academy.cocoal	11	true1	null1	null1
138821	academy.dental-coach1	11	true1	null1	null1
141571	academy.ger1	11	true1	null1	null1
144101	academy.investmen...1	21	true1	1229918011.31584108265683e-081	
147691	academy.newtown1	11	true1	null1	null1
152241	academy.talk1	21	true1	null1	null1
160641	accountant.buy-mo...1	11	true1	null1	null1

+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
only showing top 20 rows

DataFrame[ID: string, PLD: string, NumHosts: bigint, PLDisHost?: string, HarmonicCentral

```
%pyspark
```

FINISHED

```
# 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='true',
```

```
%pyspark
```

FINISHED