Paul 4 - initial dom…

FINISHED

```
%pyspark

# Zeppelin notebook to create domain summaries based on the May/Jun/Jul 2017 CommonCraw
# 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=10000 # 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", Strin
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]
```

FINISHED

```
%pyspark

# 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-ju
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

# 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())
```

FINISHED

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

```
%pyspark

# 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()
#pr_df.count() # Should be 91M
```

FINISHED

```
+-------+--------+-------+-------------------+-------------+
|#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("ds
# 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()
```

```
|6679|accountant.clarke...|       1|
|6811|accountant.demo-f...|       1|
|6878| accountant.donyasas|       1|
|6886| accountant.downhedt|       1|
|7348|accountant.glucop...|       1|
|7411|    accountant.hfyvr|       3|
|7498|accountant.infore...|       1|
|7674|accountant.la-fucker|      18|
|7862|    accountant.mksye|       1|
|8006|   accountant.odocet|       1|
|8371|    accountant.qoxjb|       1|
|8455|    accountant.rwxtb|       1|
|8762|accountant.tousac...|       1|
|8828|     accountant.ubuc|       1|
|8918|    accountant.vcvqv|       3|
+----+--------------------+--------+
only showing top 100 rows
DataFrame[ID: string, PLD: string, NumHosts: bigint]
```

```
%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()
```

```
|3027|             ac.thai|     249|      true|
|8873|accountant.unlockpro|     222|      true|
|3178|              ac.uoj|     169|      true|
|1919|            ac.labos|     121|      true|
|1715|           ac.island|      95|      true|
|4454|         academy.jnv|      83|      true|
|9328|  accountants.portal|      67|      true|
|1095|ac.digitaluniversity|      53|      true|
|1534|          ac.hoikuen|      47|     false|
|2347|           ac.o-hara|      40|      true|
|2624|          ac.regency|      33|      true|
| 383|              ac.acs|      33|      true|
|1734|             ac.itss|      31|     false|
|3088|              ac.tyo|      30|      true|
|1830|              ac.kcu|      28|      true|
+----+--------------------+--------+----------+
only showing top 20 rows
DataFrame[ID: string, PLD: string, NumHosts: bigint, PLDisHost?: string]
```

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

```
+----+--------------------+--------+----------+------------------+--------+
|  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                                                                    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='tru
```

```
%pyspark                                                                    FINISHED
```