Tom 2 Wiki Topic …

%pyspark READY

```
# PySpark CommonCrawl Topic Modelling
# Tom V / Paul J - 14/2/2018

# SET THE spark.driver.maxResultSize PROPERTY TO 16g

import boto
from boto.s3.key import Key
from gzipstream import GzipStreamFile
from pyspark.sql.types import *
import warc
import ujson as json
from urlparse import urlparse
from langdetect import detect_langs
import pycld2 as cld2

# Latest blog/documentation: http://commoncrawl.org/2018/01/january-2018-crawl-archive
    -now-available/
wetlist = sc.textFile("s3://commoncrawl/crawl-data/CC-MAIN-2018-05/wet.paths.gz") #
    Jan 2018 Crawl - 80000 files (9.29TB)

wetlist.cache()

def unpack(uri):
    conn = boto.connect_s3(anon=True, host='s3.amazonaws.com')
    bucket = conn.get_bucket('commoncrawl')
    key_ = Key(bucket, uri)
    file_ = warc.WARCFile(fileobj=GzipStreamFile(key_))
    return file_

def detect(x):
    try:
        return detect_langs(x)[0].lang # Maybe we can get away with looking at less
            characters, or do something less expensive?
    except Exception as e:
        return None

def detect2(x):
    try:
        isReliable, textBytesFound, details = cld2.detect(x)
        return details[0][1]
    except Exception as e:
        print(e)
        return None

def process_wet(id_, iterator):
    for uri in iterator:
        file = unpack(uri)
        for record in file: # Approx 53k web pages per WET file
```

```
            try:
                #url = record.rec_headers.get_header('WARC-Target-URI')
                #yield record, record.content_stream().read().decode('utf-8')
                url = record.url

                # TODO: Limit number of bytes read per record e.g. read(200000)

                domain = None if not url else urlparse(url).netloc
                text = record.payload.read().decode('utf-8') #.limit(100) # TODO:
                    Limit this read to ensure max length (for improving
                    parallizability)
                lang = detect2(text[:300]) # Use PyCLD2, not langdetect, which was
                    killing performance!
                yield domain, url, text, lang
            except Exception as e:
                yield e

def process_wet_simple(id_, iterator):
    count=0
    for uri in iterator:
        file = unpack(uri)
        for record in file:
            try:
```

```
%pyspark
detect2("this is a test")
```

READY

```
'en'
```

```
%pyspark

# PARAMETER - number of input files
nfiles = 8192 # Total 80000

# PARAMETER - slices / partitions of input
files = sc.parallelize(wetlist.take(nfiles)) #, numSlices=nfiles/32) # TODO: Try
    numSlices=nfiles/32 etc, or just default!

# Should parallelize
print(files.getNumPartitions())
rdd=files.mapPartitionsWithIndex(process_wet)

print(str(rdd))
docs = rdd.toDF(["host", "url", "text","lang"]) #  "lang"
#docs.cache()
```

READY

```
640
PythonRDD[114] at RDD at PythonRDD.scala:48
```

```
%pyspark
```

READY

```
# Filter for English only
docs_en = docs.filter(docs.lang == 'en')
```

Load saved vectors from Wikipedia model (created by python Wikipedia Text Processing.ipynb)

READY

```
%pyspark
from pyspark.ml import Pipeline,PipelineModel
from pyspark.ml.feature import RegexTokenizer, CountVectorizer, StopWordsRemover
from pyspark.ml.clustering import LocalLDAModel

textModel = PipelineModel.load('s3://billsdata.net/CommonCrawl/wikipedia/text_model')
ldaModel = LocalLDAModel.load('s3://billsdata.net/CommonCrawl/wikipedia/lda_model')
```

READY

```
%pyspark

# Test the models - for debugging only
import numpy as np
import pandas as pd

X=ldaModel.topicsMatrix().toArray()
vocab = np.array(textModel.stages[2].vocabulary)

topicLabels = [' '.join(vocab[np.argsort(X[:,i])[::-1][:5]]) for i in range(100)]

def score_topics(text):
    df = sqlContext.createDataFrame(pd.DataFrame({'text':[text]}))
    vec = textModel.transform(df)
    scores = ldaModel.transform(vec).select('topicDistribution').collect()[0]
        .topicDistribution.toArray()
    return pd.Series(dict(zip(topicLabels, scores)))

# Try it on an arbitary sentence
```

```
school students education university college        0.001276
season team first teams cup                        0.001276
series book published books novel                  0.001261
series show television also episode                0.001292
ship ships two navy war                            0.001220
social one may also people                         0.001240
space earth light solar star                       0.001223
species found also large may                       0.001261
station line railway service train                 0.001261
team season coach football first                   0.001253
tom oliver ghost haiti kay                         0.001183
ukrainian ukraine dog dogs stamps                  0.001198
university research professor published science    0.001323
war union soviet communist political               0.001213
water company construction new coal                0.001240
world olympics championships summer women          0.430457
zealand new grand auckland prix                    0.001216
Length: 100, dtype: float64
```

```
%pyspark

# Now score pages from our WET files
docs_en.show(5)
vec = textModel.transform(docs_en)
vec.show(5)
```

READY

```
+--------------------+--------------------+--------------------+----+
|                host|                 url|                text|lang|
|                null|                null|Software-Info: ia...|  en|
|0ncemorewithfeeli...|http://0ncemorewi...|Once More, With F...|  en|
|        100share.com|http://100share.c...|Saitek Eclipse Ba...|  en|
|101bestandroidapp...|http://101bestand...|fabien | 101 Best...|  en|
|        1045espn.com|http://1045espn.c...|Mickey Joseph on ...|  en|
+--------------------+--------------------+--------------------+----+
only showing top 5 rows
+--------------------+--------------------+--------------------+----+--------------------
-+--------------------+--------------------+
|                host|                 url|                text|lang|               word
s|            filtered|                 vec|
+--------------------+--------------------+--------------------+----+--------------------
-+--------------------+--------------------+
|                null|                null|Software-Info: ia...|  en|[software, info, ..
.|[software, info, ...|(20000,[84,152,33...|
|0ncemorewithfeeli...|http://0ncemorewi...|Once More, With F...|  en|[once, more, with..
.|[feeling, christm...|(20000,[2,3,5,7,1...|
```

```
%pyspark

# Create topic distribution vectors and tidy upp
scores = ldaModel.transform(vec)
scores2 = scores.drop("url").drop("text").drop("lang").drop("words").drop("filtered"
    ).drop("vec")
```

READY

```
+--------------------+--------------------+
|                host|   topicDistribution|
+--------------------+--------------------+
|                null|[4.59706427634550...|
|0ncemorewithfeeli...|[3.78779380156022...|
|        100share.com|[8.1625505686031E...|
|101bestandroidapp...|[1.28997469852102...|
|        1045espn.com|[6.65356474668816...|
+--------------------+--------------------+
only showing top 5 rows
```

```
%pyspark

# Save these vectors to disc, so we can just load them later
scores2.write.parquet('s3://billsdata.net/CommonCrawl/topic_model_%d_files
```

READY

Load saved scores from nfiles of WET files

READY

```
%pyspark

# Restart here
nfiles=8192
scores2 = spark.read.parquet('s3://billsdata.net/CommonCrawl/topic_model_%d_files
    /cc_page_wiki_topic_scores' % nfiles)
```

```
+--------------------+--------------------+
|                host|   topicDistribution|
+--------------------+--------------------+
|                null|[4.59706427634550...|
|        1-urlm.co.uk|[5.46671064984294...|
|          10015.info|[0.10477661497170...|
|100sajad.blogfa.c...|[0.00116257997605...|
| 101bestwebsites.com|[4.30357917602672...|
+--------------------+--------------------+
only showing top 5 rows
```

READY

```
%pyspark

# Aggregate page-scores per Host for now (will be same process for aggregating host
    -scores to PLD-scores) - first count and sum vectors
scores3=scores2.rdd.map(lambda x: (x['host'], (1,x['topicDistribution']))).reduceByKey
    (lambda acc, val: (acc[0]+val[0], acc[1]+val[1]))

# Next, divide by the total to create averaged vectors, and convert back to a
    dataframe
scores4=scores3.map(lambda x: (x[0], (x[1][1]/x[1][0]))).toDF(["host",
    "averageTopicDistribution"])
```

```
+--------------------+------------------------+
|                host|averageTopicDistribution|
+--------------------+------------------------+
|www.iplayoutside.com|    [3.33132075400783...|
|       www.naml.info|    [1.15112340833547...|
|pateriaseo.blogsp...|    [5.30444355332745...|
|  www.1stcapture.com|    [1.34253907230677...|
|online.bomageorgi...|    [5.18636606421855...|
+--------------------+------------------------+
only showing top 5 rows
111567639
2894462
```

READY

```
%pyspark

# Just playing - code to help understand the different libraries and vector types!
import pyspark.mllib.linalg as mllib
import pyspark.ml.linalg as ml
df = sc.parallelize([
```

```
    (mllib.DenseVector([1, ]), ml.DenseVector([1, ])),
    (mllib.SparseVector(1, [0, ], [1, ]), ml.SparseVector(1, [0, ], [1, ]))
]).toDF(["mllib_v", "ml_v"])
df.show()
{s.name: type(s.dataType) for s in df.schema}
```

```
+-------------+-------------+
|      mllib_v|         ml_v|
+-------------+-------------+
|        [1.0]|        [1.0]|
|(1,[0],[1.0])|(1,[0],[1.0])|
+-------------+-------------+
{'ml_v': <class 'pyspark.ml.linalg.VectorUDT'>, 'mllib_v': <class 'pyspark.mllib.linalg.
VectorUDT'>}
```

```
%pyspark
```

READY

```
# Enrich each row with the corresponding PLD (using code from Paul J)
#saveURI="s3://billsdata.net/CommonCrawl/hyperlinkgraph/cc-main-2017-aug-sep-oct
    /domaingraph/vertices/"
saveURI="s3://billsdata.net/CommonCrawl/hyperlinkgraph/cc-main-2017-18-nov-dec-jan
    /domaingraph/vertices/"
pld_df=spark.read.load(saveURI)
pld_df.show(3)
```

```
+---+-------+--------+
| ID|    PLD|NumHosts|
+---+-------+--------+
|  0|  aaa.a|       1|
|  1| aaa.aa|       1|
|  2|aaa.aaa|       4|
+---+-------+--------+
only showing top 3 rows
70367284
```

```
%pyspark
```

READY

```
# Generate a Bloom Filter of PLDs from the vertex file
from pybloom import BloomFilter

# Attempt to distribute our Bloom Filter so we can build it in parallel, and save it
    to disk as an RDD
def build_partial_bloom(capacity, error_rate):
    def _build_partial_bloom(record):
        bloom_filter = BloomFilter(capacity=capacity, error_rate=error_rate)
        for _,PLD,_ in record: # Just take the PLD field (not the ID or NumHosts)
            bloom_filter.add(PLD)
        yield (None, bloom_filter) # returns a Generator (i.e. a kind of Iterator that
            can only get called once, and doesn't stay in memory)
    return _build_partial_bloom

def merge_bloom(bloom_in1, bloom_in2):
```

```
    return bloom_in1.union(bloom_in2) # The reduce function simply becomes a Union

# Our generator function for partial Bloom Filters
generate_bloom = build_partial_bloom(capacity=94000000, error_rate=0.005) # Only 70M
    in latest web graph?

# Construct the distributed BloomFilter using the PLD dataframe
bloom_filter_rdd = pld_df.rdd.mapPartitions(generate_bloom).reduceByKey(merge_bloom)

# Save/Load the Bloom Filter RDD - not working
#saveURI="s3://billsdata.net/CommonCrawl/hyperlinkgraph/cc-main-2017-18-nov-dec-jan
    /domaingraph/vertex_bloom_filter/"
#bloom_filter_rdd.saveAsTextFile(saveURI)
#bloom_filter_rdd=spark.read.load(saveURI)

# Collect and broadcast to cluster nodes
bloom_filter=bloom_filter_rdd.collect()
print(bloom_filter)
pld_bf_distrib = sc.broadcast(bloom_filter[0][1])

# Test contents
import sys
```

```
[(None, <pybloom.pybloom.BloomFilter object at 0x7ff8a4ee0f10>)]
64
0
True
False
```

```
%pyspark                                                                    READY

from pyspark.sql.functions import udf

# Returns a Boolean to say whether PLD is a hostname in itself
def is_a_pld(hostname):
    #if hostname in pld_lookup_table:
    #if pld_lookup_table.filter(lambda a: a == hostname).count()>0:
    if hostname in pld_bf_distrib.value:
        return True
    else:
        return False

# Define a function to do the hostname->pld conversion, if the pld exists in our
    dictionary
def convert_hostname(hostname):
    # Return hostname as-is, if this is already a PLD
    #if hostname in pld_lookup_table:
    #if pld_lookup_table.filter(lambda a: a == hostname).count()>0:
    if hostname in pld_bf_distrib.value:
        return hostname
    # Otherwise we're going to have to split it up and test the parts
    try:
        parts=hostname.split('.')
        if (len(parts)>4 and is_a_pld('.'.join(parts[0:4]))):
            return '.'.join(parts[0:4])
```

```
        if (len(parts)>3 and is_a_pld('.'.join(parts[0:3]))):
            return '.'.join(parts[0:3])
        if (len(parts)>2 and is_a_pld('.'.join(parts[0:2]))):
            return '.'.join(parts[0:2])
        if (len(parts)>1 and is_a_pld('.'.join(parts[0:1]))):
            return '.'.join(parts[0:1])
        return "ERROR" # Couldn't find a corresponding PLD - this should never happen!
    except:
        return "ERROR"

udf_convert_hostname = udf(convert_hostname, StringType())

# Test
print(convert_hostname("aaa.aaa"))
```

```
aaa.aaa
True
```

```
%pyspark
```

READY

```
# Function to reverse hostnames
from pyspark.sql.functions import udf
def reverse_domain(domain):
    try:
        ret =  '.'.join(reversed(domain.split('.')))
        return ret
    except:
        return "NODOMAIN"
print(reverse_domain("com.facebook"))
udf_reverse_domain = udf(reverse_domain, StringType())

# Convert hosts in Topic DF to PLDs using convert_hostname function from Paul 5.
scores5=scores4.withColumn("pld",udf_reverse_domain(udf_convert_hostname
    (udf_reverse_domain("host")))) # Reverse the hostnames prior to lookup, then back
```

```
facebook.com
+--------------------+------------------------+--------------------+
|                host|averageTopicDistribution|                 pld|
+--------------------+------------------------+--------------------+
|www.iplayoutside.com|    [3.33132075400783...|    iplayoutside.com|
|www.harlequinbook...|    [2.18651285593280...|harlequinbooks.co...|
|pateriaseo.blogsp...|    [5.30444355332745...|         blogspot.in|
|  www.1stcapture.com|    [1.34253907230677...|      1stcapture.com|
|online.bomageorgi...|    [5.18636606421855...|     bomageorgia.org|
|www.rodingtonpc.o...|    [4.53516223682002...|  rodingtonpc.org.uk|
|     capitalvets.net|    [5.77909467395404...|     capitalvets.net|
|dheluestarini.wor...|    [0.29064496117340...|       wordpress.com|
|     darkcandles.com|    [0.04991811456460...|     darkcandles.com|
|www.agentlemother...|    [1.31513676399448...|   agentlemother.com|
+--------------------+------------------------+--------------------+
only showing top 10 rows
```

```
%pyspark
```

READY

```
# Now we can aggregate page-scores per PLD, using a map-reduce similar to the host
    aggregation above - first count and sum vectors
scores6=scores5.rdd.map(lambda x: (x['pld'], (1,x['averageTopicDistribution']
    ))).reduceByKey(lambda acc, val: (acc[0]+val[0], acc[1]+val[1]))

# Next, divide by the total to create averaged vectors, and convert back to a
    dataframe
scores7=scores6.map(lambda x: (x[0], (x[1][1]/x[1][0]))).toDF(["pld_rev",
    "averageTopicDistribution"])
```

```
+-------------------+-----------------------+
|             pld_rev|averageTopicDistribution|
+-------------------+-----------------------+
|thetruthmatters.blog|    [4.64008309641444...|
|      cappelleria.eu|    [0.00557376059687...|
|     capitalvets.net|    [5.77909467395404...|
|    safariguides.net|    [2.95713650633799...|
| ymcachattanooga.org|    [4.46676317269227...|
+-------------------+-----------------------+
only showing top 5 rows
2894462
1757198
```

READY

```
%pyspark

# Save pld topic distributions in parquet format for Tom to play with (and to figure
    out how to create a PLD topic summary from this).
# TODO: Maybe a numpy argmax to get the index of the 'top' topic for each PLD with a
```

Load saved PLD vectors from nfiles of WET files

FINISHED

```
%pyspark

# Restart here
nfiles=8192
scores7 = spark.read.parquet('s3://billsdata.net/CommonCrawl/topic_model_%d_files
    /cc_pld_wiki_topic_scores' % nfiles)
```

```
+-------------------+-----------------------+
|             pld_rev|averageTopicDistribution|
+-------------------+-----------------------+
|          scrtec.org|    [0.00101137139816...|
|corporatefitnessc...|    [1.16246367421139...|
|        1centweb.com|    [1.94489585421988...|
|   mygaragestory.net|    [2.04009774069444...|
|     gitedeville.com|    [1.18827049530001...|
+-------------------+-----------------------+
only showing top 5 rows
```

```
%pyspark
```

ERROR

```
# Next run t-SNE embedding on the vectors to reduce from indlen dimensions to 2

import numpy as np

from sklearn.manifold import TSNE

#X = np.array([[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 1], [1, 1, 0]])

X=np.array(scores7.select('averageTopicDistribution').take(10000))
```

```
[[2.88131243e-05 2.75475952e-05 3.04710806e-05 ... 2.81830024e-05
  2.68391742e-05 2.78737638e-05]]
 [[7.76791115e-04 7.42672922e-04 8.21489002e-04 ... 7.59803265e-04
  7.23574156e-04 7.51466306e-04]]]
[[1.01137140e-03 9.66949978e-04 1.06956744e-03 ... 9.89253450e-04
  9.42083646e-04 9.78398843e-04]
 [1.16246367e-04 1.11140598e-04 1.22935382e-04 ... 1.13704145e-04
  1.08282478e-04 1.12456523e-04]
 [1.94489585e-04 1.85947220e-04 2.05680848e-04 ... 1.90236241e-04
  1.81165354e-04 1.88148870e-04]
 ...
 [1.80705922e-04 1.72768963e-04 1.91104049e-04 ... 1.76754016e-04
  1.68325992e-04 1.74814579e-04]
 [2.88131243e-05 2.75475952e-05 3.04710806e-05 ... 2.81830024e-05
  2.68391742e-05 2.78737638e-05]
 [7.76791115e-04 7.42672922e-04 8.21489002e-04 ... 7.59803265e-04
  7.23574156e-04 7.51466306e-04]]
(10000, 2)

Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-387383354023936293.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-387383354023936293.py", line 365, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 9, in <module>
  File "/usr/lib/zeppelin/interpreter/lib/python/backend_zinline.py", line 303, in displ
ayhook
    show()
  File "/usr/lib/zeppelin/interpreter/lib/python/backend_zinline.py", line 72, in __call
__
    manager.show(**kwargs)
  File "/usr/local/lib64/python2.7/site-packages/matplotlib/backend_bases.py", line 2680
, in show
    raise NonGuiException()
NonGuiException
```

```
%pyspark
```

FINISHED

```
# Let's try the same thing in UMAP (BEWARE: install umap-learn, not umap!)
```

```
#import numpy as np
import umap

#X=np.array(scores7.select('averageTopicDistribution').take(1000))
#print(X)
#new_X = np.array([i[0] for i in X])
#print(new_X)
U_embedded = umap.UMAP().fit_transform(new_X)
```

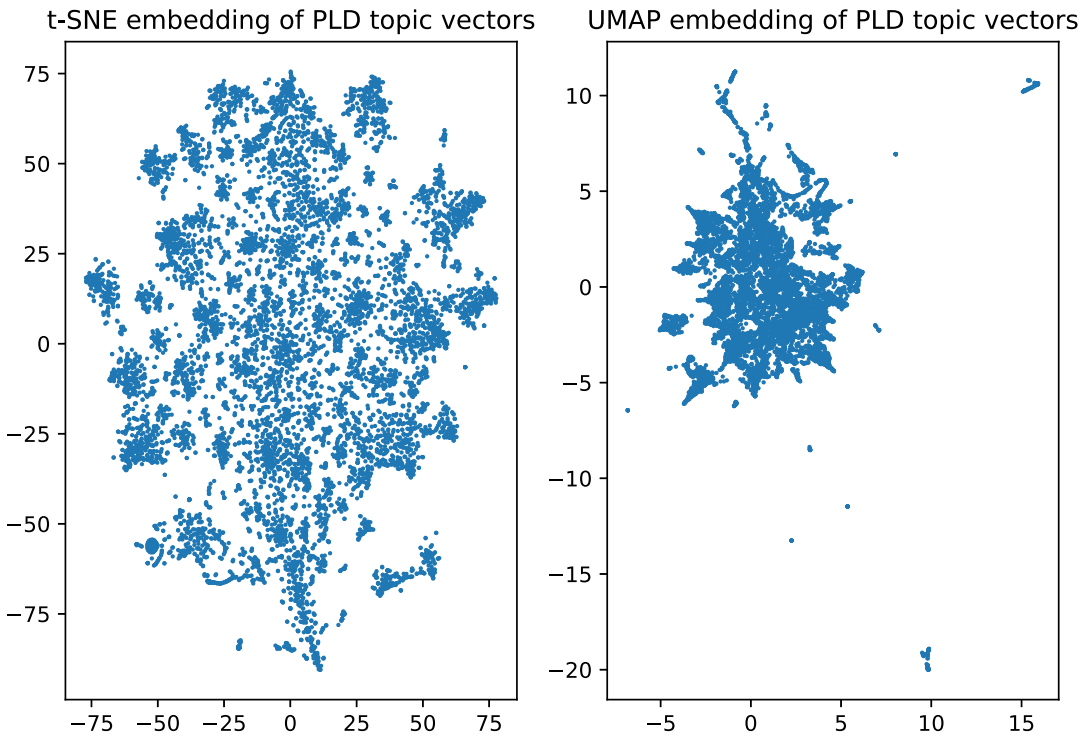
```
(10000, 2)
```

```
%pyspark
```

ERROR

```
# Now plot the t-SNE embedding using matplotlib
import matplotlib
import StringIO

# Turn off interactive mode - we really want this but it leads to dependency errors on
    AWS linux!
matplotlib.use('agg',warn=False, force=True) # Removes Tkinter error (no interactive
    plots on AWS EMR)
from matplotlib import pyplot as plt
print "Switched to:",matplotlib.get_backend()

fig, ax = plt.subplots(nrows=1, ncols=2) #, figsize=(40,16))
#print(X_embedded)

#colors=np.array(pld_codes_with_vec_sc.select('StatusColour').collect())
#colors2 = np.array([i[0] for i in colors])
#groups=np.array(pld_codes_with_vec_sc.select('StatusClass').collect())
#groups2 = np.array([i[0] for i in groups])
#zip_list = np.array(zip(pld_codes_with_vec_sc.collect(),X_embedded))
#x_emb = zip_list[:,1]
#x=[row[0] for row in x_emb]
#y=[row[1] for row in x_emb]

#ax.scatter(X_embedded[:,0], X_embedded[:,1], s=1.0, c=colors2, edgecolors='face',
    cmap='rainbow')
#ax.scatter(x, y, s=1.0, c=colors2, edgecolors='face', cmap='rainbow')
ax[0].set_title("t-SNE embedding of PLD topic vectors")
ax[0].scatter(X_embedded[:,0], X_embedded[:,1], s=1.0)
ax[1].set_title("UMAP embedding of PLD topic vectors")
ax[1].scatter(U_embedded[:,0], U_embedded[:,1], s=1.0)

# plt.show() doesn't seem to work so this is a workaround
def show(p):
    img = StringIO.StringIO()
    p.savefig(img, format='svg')
    img.seek(0)
    print "%html <div style='width:600px'>" + img.buf + "</div>"
```

```
Switched to: agg
```




```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-387383354023936293.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-387383354023936293.py", line 365, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 17, in <module>
  File "/usr/lib/zeppelin/interpreter/lib/python/backend_zinline.py", line 303, in displ
ayhook
    show()
  File "/usr/lib/zeppelin/interpreter/lib/python/backend_zinline.py", line 72, in __call
__
    manager.show(**kwargs)
  File "/usr/local/lib64/python2.7/site-packages/matplotlib/backend_bases.py", line 2680
, in show
    raise NonGuiException()
NonGuiException
```

```
%pyspark
```

READY