Tom 2 Wiki Topic ...

```
%pyspark                                                                    FINISHED

# PySpark CommonCrawl Topic Modelling
# Tom V / Paul J - 13/2/2018

# SET THE spark.driver.maxResultSize PROPERTY TO 3G

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

#wetlist = sc.textFile("s3://commoncrawl/crawl-data/CC-MAIN-2017-04/wet.paths.gz") # Ap
# Latest blog/documentation: http://commoncrawl.org/2017/10/october-2017-crawl-archive-
wetlist = sc.textFile("s3://commoncrawl/crawl-data/CC-MAIN-2017-43/wet.paths.gz") # Oct

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 cha
    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
                lang = detect2(text[:300]) # Use PyCLD2, not langdetect, which was kill
                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:
                count=count+1
                # TODO: Output total size of pages, rather than number of pages
                # Histogram.
            except Exception as e:
                pass
        #print(count)
```

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

READY

```
'en'
```

```
%pyspark

# PARAMETER - number of input files
nfiles = 1 # Total 89100

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

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

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

READY

```
320
PythonRDD[102] 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].topicDist
    return pd.Series(dict(zip(topicLabels, scores)))

# Try it on an arbitary sentence
print(score_topics("This is the latest news about North Korea and their involvement in
```

```
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             0.430010
zealand new grand auckland prix                       0.001216
Length: 100, dtype: float64
```

```
%pyspark                                                                    READY

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

```
+--------------------+--------------------+--------------------+----+
|                null|                null|Software-Info: ia...|  en|
|1000daysofwriting...|http://1000daysof...|1000 Days of Writ...|  en|
|100unhappydays.bl...|http://100unhappy...|100 Unhappy Days:...|  en|
|          10in30.com|http://10in30.com...|LearnOutLoud_300x...|  en|
|123-free-download...|http://123-free-d...|MusicBoxTool - [3...|  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,[88,152,33...|
|1000daysofwriting...|http://1000daysof...|1000 Days of Writ...|  en|[days, of, writin..
.|[days, writing, d...|(20000,[0,2,3,5,7...|
```

```
%pyspark                                                                    READY

# Create topic distribution vectors and tidy upp
scores = ldaModel.transform(vec)
scores2 = scores.drop("url").drop("text").drop("lang").drop("words").drop("filtered").d
scores2.show(5)
```

```
+--------------------+--------------------+
|                host|   topicDistribution|
+--------------------+--------------------+
|                null|[0.13351995547840...|
|1000daysofwriting...|[3.26238961502545...|
|100unhappydays.bl...|[5.81230792144732...|
|          10in30.com|[8.42785330446217...|
|123-free-download...|[6.44166735802645...|
+--------------------+--------------------+
only showing top 5 rows
```

```
%pyspark                                                                    READY

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

Load saved scores from nfiles of WET files

FINISHED

```
%pyspark

# Restart here
nfiles=1
scores2 = spark.read.parquet('s3://billsdata.net/CommonCrawl/topic_model_%d_files/cc_pa
scores2.show(5)
```

```
+--------------------+--------------------+
|              domain|   topicDistribution|
+--------------------+--------------------+
|                null|[0.12615256587902...|
|1000daysofwriting...|[3.26238961502545...|
|100unhappydays.bl...|[5.81230792144733...|
|          10in30.com|[8.42785330446217...|
|123-free-download...|[6.44166735802645...|
+--------------------+--------------------+
only showing top 5 rows
```

FINISHED

```
%pyspark

# Aggregate page-scores per Host for now (will be same process for aggregating host-sco
scores3=scores2.rdd.map(lambda x: (x['domain'], (1,x['topicDistribution']))).reduceByKe

# 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", "averageTopicDis
scores4.show(5)
{s.name: type(s.dataType) for s in scores4.schema}
```

```
+--------------------+------------------------+
|                host|averageTopicDistribution|
+--------------------+------------------------+
|   www.superdrug.com|    [0.01613454968703...|
|       kiteforum.com|    [1.11136619419678...|
|www.virtualmuseum.ca|    [1.80597394573963...|
|ukrainianstartups...|    [2.46670116225146...|
| lauragravestock.com|    [4.49484962755617...|
+--------------------+------------------------+
only showing top 5 rows
{'averageTopicDistribution': <class 'pyspark.ml.linalg.VectorUDT'>, 'host': <class 'pysp
ark.sql.types.StringType'>}
```

FINISHED

```
%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
```

FINISHED

```
# TODO: Enrich each row with the corresponding PLD (using code from Paul J, but pickle
saveURI="s3://billsdata.net/CommonCrawl/hyperlinkgraph/cc-main-2017-aug-sep-oct/domaing
pld_df=spark.read.load(saveURI)
pld_df.show(3)
pld_df.cache()
```

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

```
%pyspark
```

ERROR

```
# Next, we'll construct a local dictionary from of all the PLDS (key is the PLD, value
# This is our truth-table of known PLDs that we'll use when counting hosts
# Create a bloom filter using a pure python package (might be a little slow)
from pybloom import BloomFilter
pld_bf = BloomFilter(capacity=94000000, error_rate=0.005) # was 91M

for row in pld_df.rdd.collect(): # limit(10000000) # TODO: Still bad (and exceeds spark
    pld_bf.add(row['PLD'])

print(pld_df.rdd.take(3))
print(pld_df.rdd.take(3)[2]['PLD'])
#pld_bf.add(pld_df.rdd.take(3)[2]['PLD'])
print("aaa.aaa" in pld_bf) # Should be True

import sys
print(sys.getsizeof(pld_bf))
print(len(pld_bf)) # Should match number of items entered
```

```
# Broadcast the bloom filter so it's available on all the slave nodes - we don't need t
# it any more so it's fine being immutable.
pld_bf_distrib=sc.broadcast(pld_bf)

print("aaa.aaa" in pld_bf) # Should be true
print("aaa.aaa.bla" in pld_bf) # Should be false
print("aaa.aaa" in pld_bf_distrib.value) # Should be true
print("aaa.aaa.bla" in pld_bf_distrib.value) # Should be false
```

```
org.apache.thrift.transport.TTransportException
        at org.apache.thrift.transport.TIOStreamTransport.read(TIOStreamTransport.java:1
32)
        at org.apache.thrift.transport.TTransport.readAll(TTransport.java:86)
        at org.apache.thrift.protocol.TBinaryProtocol.readAll(TBinaryProtocol.java:429)
        at org.apache.thrift.protocol.TBinaryProtocol.readI32(TBinaryProtocol.java:318)
        at org.apache.thrift.protocol.TBinaryProtocol.readMessageBegin(TBinaryProtocol.j
ava:219)
        at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:69)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.recv_i
nterpret(RemoteInterpreterService.java:266)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.interp
ret(RemoteInterpreterService.java:250)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreter.interpret(RemoteInte
rpreter.java:373)
        at org.apache.zeppelin.interpreter.LazyOpenInterpreter.interpret(LazyOpenInterpr
eter.java:97)
        at org.apache.zeppelin.notebook.Paragraph.jobRun(Paragraph.java:406)
        at org.apache.zeppelin.scheduler.Job.run(Job.java:175)
        at org.apache.zeppelin.scheduler.RemoteScheduler$JobRunner.run(RemoteScheduler.j
ava:329)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$2
01(ScheduledThreadPoolExecutor.java:180)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(Sche
duledThreadPoolExecutor.java:293)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:114
9)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:62
4)
        at java.lang.Thread.run(Thread.java:748)
```

ERROR

```
%pyspark

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 dictio
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"))
print(is_a_pld("aaa.aaa")) # Should be true
```

```
org.apache.thrift.transport.TTransportException
        at org.apache.thrift.transport.TIOStreamTransport.read(TIOStreamTransport.java:1
32)
        at org.apache.thrift.transport.TTransport.readAll(TTransport.java:86)
        at org.apache.thrift.protocol.TBinaryProtocol.readAll(TBinaryProtocol.java:429)
        at org.apache.thrift.protocol.TBinaryProtocol.readI32(TBinaryProtocol.java:318)
        at org.apache.thrift.protocol.TBinaryProtocol.readMessageBegin(TBinaryProtocol.j
ava:219)
        at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:69)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.recv_i
nterpret(RemoteInterpreterService.java:266)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.interp
ret(RemoteInterpreterService.java:250)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreter.interpret(RemoteInte
rpreter.java:373)
        at org.apache.zeppelin.interpreter.LazyOpenInterpreter.interpret(LazyOpenInterpr
eter.java:97)
        at org.apache.zeppelin.notebook.Paragraph.jobRun(Paragraph.java:406)
        at org.apache.zeppelin.scheduler.Job.run(Job.java:175)
        at org.apache.zeppelin.scheduler.RemoteScheduler$JobRunner.run(RemoteScheduler.j
ava:329)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$2
01(ScheduledThreadPoolExecutor.java:180)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(Sche
duledThreadPoolExecutor.java:293)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:114
9)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:62
4)
        at java.lang.Thread.run(Thread.java:748)
```

```
%pyspark                                                                        ERROR

# Function to reverse hostnames
from pyspark.sql.functions import udf
def reverse_domain(domain):
    return '.'.join(reversed(domain.split('.')))
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("host_rev",udf_reverse_domain(udf_convert_hostname(udf_rever
scores5.show(10)
```

```
java.net.SocketException: Broken pipe (Write failed)
        at java.net.SocketOutputStream.socketWrite0(Native Method)
        at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:111)
        at java.net.SocketOutputStream.write(SocketOutputStream.java:155)
        at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:82)
        at java.io.BufferedOutputStream.write(BufferedOutputStream.java:121)
        at org.apache.thrift.transport.TIOStreamTransport.write(TIOStreamTransport.java:
145)
        at org.apache.thrift.protocol.TBinaryProtocol.writeString(TBinaryProtocol.java:2
02)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterContext$RemoteInterpr
eterContextStandardScheme.write(RemoteInterpreterContext.java:1133)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterContext$RemoteInterpr
eterContextStandardScheme.write(RemoteInterpreterContext.java:992)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterContext.write(RemoteI
nterpreterContext.java:882)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$interpret_arg
s$interpret_argsStandardScheme.write(RemoteInterpreterService.java:6501)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$interpret_arg
s$interpret_argsStandardScheme.write(RemoteInterpreterService.java:6424)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$interpret_arg
s.write(RemoteInterpreterService.java:6351)
        at org.apache.thrift.TServiceClient.sendBase(TServiceClient.java:63)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.send_i
nterpret(RemoteInterpreterService.java:260)
        at org.apache.zeppelin.interpreter.thrift.RemoteInterpreterService$Client.interp
ret(RemoteInterpreterService.java:249)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreter.interpret(RemoteInte
rpreter.java:373)
        at org.apache.zeppelin.interpreter.LazyOpenInterpreter.interpret(LazyOpenInterpr
eter.java:97)
        at org.apache.zeppelin.notebook.Paragraph.jobRun(Paragraph.java:406)
        at org.apache.zeppelin.scheduler.Job.run(Job.java:175)
        at org.apache.zeppelin.scheduler.RemoteScheduler$JobRunner.run(RemoteScheduler.j
ava:329)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$2
01(ScheduledThreadPoolExecutor.java:180)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(Sche
duledThreadPoolExecutor.java:293)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:114
9)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:62
4)
        at java.lang.Thread.run(Thread.java:748)
```

```
%pyspark

# TODO: Now we can aggregate page-scores per PLD, using a map-reduce similar to the hos
```

ERROR

```
java.net.ConnectException: Connection refused (Connection refused)
        at java.net.PlainSocketImpl.socketConnect(Native Method)
        at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
        at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.jav
a:206)
        at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
        at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
        at java.net.Socket.connect(Socket.java:589)
        at org.apache.thrift.transport.TSocket.open(TSocket.java:182)
        at org.apache.zeppelin.interpreter.remote.ClientFactory.create(ClientFactory.jav
a:51)
        at org.apache.zeppelin.interpreter.remote.ClientFactory.create(ClientFactory.jav
a:37)
        at org.apache.commons.pool2.BasePooledObjectFactory.makeObject(BasePooledObjectF
actory.java:60)
        at org.apache.commons.pool2.impl.GenericObjectPool.create(GenericObjectPool.java
:861)
        at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPoo
l.java:435)
        at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPoo
l.java:363)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreterProcess.getClient(Rem
oteInterpreterProcess.java:92)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreter.interpret(RemoteInte
rpreter.java:352)
        at org.apache.zeppelin.interpreter.LazyOpenInterpreter.interpret(LazyOpenInterpr
eter.java:97)
        at org.apache.zeppelin.notebook.Paragraph.jobRun(Paragraph.java:406)
        at org.apache.zeppelin.scheduler.Job.run(Job.java:175)
        at org.apache.zeppelin.scheduler.RemoteScheduler$JobRunner.run(RemoteScheduler.j
ava:329)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$2
```

```
01(ScheduledThreadPoolExecutor.java:180)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(Sche
duledThreadPoolExecutor.java:293)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:114
9)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:62
4)
        at java.lang.Thread.run(Thread.java:748)
```

```
%pyspark

# TODO: Save pld topic distributions in parquet format for Tom to play with (and to fig
# Maybe a numpy argmax to get the index of the 'top' topic for each PLD with a score.
```

READY