Tom 2 Wiki Topic …

%pyspark FINISHED

```
# PySpark CommonCrawl Topic Modelling
# Tom V / Paul J - 9/2/2018

# 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")
```

FINISHED

```
'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
```

FINISHED

```
320
PythonRDD[102] at RDD at PythonRDD.scala:48
```

```
%pyspark
```

FINISHED

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

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

FINISHED

```
%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')
```

```
%pyspark
```

FINISHED

```
# 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

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

FINISHED

```
+--------------------+--------------------+--------------------+----+
|                host|                 url|                text|lang|
|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

# 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)
```

FINISHED

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

# 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_
```

FINISHED

Load saved scores from nfiles of WET files

READY

```
%pyspark

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

```
%pyspark
```

ERROR

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

# Next, divide by the total to create averaged vectors, and TODO: convert back to a dat
vec_schema=StructType([StructField('host', StringType(), False), StructField('averageTo
scores4=scores3.map(lambda x: (x[0], (x[1][1]/x[1][0]))).toDF(vec_schema) # Failed atte
scores4.show(5)
```

```
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2029)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2050)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2069)
        at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:336)
        at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:38
)
        at org.apache.spark.sql.Dataset.org$apache$spark$sql$Dataset$$collectFromPlan(Da
taset.scala:2861)
        at org.apache.spark.sql.Dataset$$anonfun$head$1.apply(Dataset.scala:2150)
        at org.apache.spark.sql.Dataset$$anonfun$head$1.apply(Dataset.scala:2150)
        at org.apache.spark.sql.Dataset$$anonfun$55.apply(Dataset.scala:2842)
        at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.
scala:65)
        at org.apache.spark.sql.Dataset.withAction(Dataset.scala:2841)
        at org.apache.spark.sql.Dataset.head(Dataset.scala:2150)
        at org.apache.spark.sql.Dataset.take(Dataset.scala:2363)
        at org.apache.spark.sql.Dataset.showString(Dataset.scala:241)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

```
%pyspark
```

ERROR

```
# TODO: Enrich each row with the corresponding PLD (using code fropm Paul J, but pickle
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-7074246543696920832.py", line 349, in <module>
    [code.body[-(nhooks + 1)]])
IndexError: list index out of range
```

```
%pyspark
```

READY

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