Tom 1 Topic Mode…

FINISHED

```
%pyspark

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

wetlist = sc.textFile("s3://commoncrawl/crawl-data/CC-MAIN-2017-04/wet.paths.gz")
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[:300])[0].lang
    except Exception as e:
        return None

def process_wet(id_, iterator):
    for uri in iterator:
        file = unpack(uri)
        for record in file:
            try:
                #url = record.rec_headers.get_header('WARC-Target-URI')
                #yield record, record.content_stream().read().decode('utf-8')
                url = record.url
                domain = None if not url  else urlparse(url).netloc
                text = record.payload.read().decode('utf-8')
                lang = detect(text)
                yield domain, url, lang, text
            except Exception as e:
                yield e
```

FINISHED

```
%pyspark
# PARAMETER - number of input files
nfiles = 4
```

```
# PARAMETER - slices / partitions of input
files = sc.parallelize(wetlist.take(nfiles), numSlices=16)

print(files.getNumPartitions())

rdd = files.mapPartitionsWithIndex(process_wet)

docs = rdd.toDF(["domain", "url", "lang", "text"])
docs.cache()

docs.show()
```

```
|            08.od.ua|http://08.od.ua/t...|   ru|4в компания, ооо ...|
|             0lik.ru|http://0lik.ru/te...|   ru|Free Arthur Radle...|
|             0lik.ru|http://0lik.ru/te...|   ru|Новогодняя рамка ...|
|             0lik.ru|http://0lik.ru/te...|   ru|Коллекция из 3 св...|
|           1.163.com|http://1.163.com/...|   ko|【平安金】平安银行 富贵平安金碗筷...|
|           1.163.com|http://1.163.com/...|zh-cn|夺宝记录 - 网易1元夺宝
欢迎来...|
|         100.ufc.com|http://100.ufc.co...|   en|UFC 100 Official ...|
|         10000km.com|http://10000km.co...|   ja|海外「この炊飯器のどこに9万円の価...|
|      1000designs.ru|http://1000design...|   ru|Экзотический скра...|
|      1000designs.ru|http://1000design...|   ru|Летняя рамка на д...|
|           1000ff.de|http://1000ff.de/...|   en|Neues Spielen?
10...|
|         1000form.ru|http://1000form.r...|   ru|Коллективный дого...|
|1000fragrances.bl...|http://1000fragra...|   en|1000fragrances: A...|
|           1000mg.jp|http://1000mg.jp/...|   ja|ニワトリの長すぎるロングトーンｗ  ...|
+--------------------+--------------------+-----+--------------------+
only showing top 20 rows
```

```
%pyspark                                                                    FINISHED

docs_en = docs.filter(docs.lang == 'en')

# PARAMETER - possibly set partitions?

docs_en = docs_en.repartition(16)
```

```
%pyspark                                                                    FINISHED

stopwords_english = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
    , 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their',
    'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
    'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for
    'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
    'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more'
    'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'shoul
    'didn', 'doesn', 'hadn', 'hasn', 'haven', 'isn', 'ma', 'mightn', 'mustn', 'needn',

from pyspark.ml import Pipeline
from pyspark.ml.feature import RegexTokenizer, CountVectorizer, StopWordsRemover
```

```
# PARAMETER - regex tokenization
tokenizer = RegexTokenizer(inputCol="text", outputCol="words", pattern="\\p{L}{2,}", ga
stopwordRemover = StopWordsRemover(inputCol="words", outputCol="filtered",stopWords=sto

# PARAMETER - vocab size, min and max doc frequency
cv = CountVectorizer(inputCol="filtered", outputCol="vec",vocabSize=20000, minDF=50)

pipeline = Pipeline(stages=[tokenizer, stopwordRemover, cv])

model = pipeline.fit(docs_en)

vecs = model.transform(docs_en)

vecs.show()
```

```
%pyspark                                                                    FINISHED

#Run the topic modelling

from pyspark.ml.clustering import LDA
#inputCol="vec", outputCol="ldaVec", k=3, optimizer="online"

lda = LDA(k=300, maxIter=100, featuresCol="vec")

ldaModel = lda.fit(vecs)
```

```
%pyspark                                                                    FINISHED

#Save the models

ldaModel.save('s3://billsdata.net/CommonCrawl/topic_model_4files/ldamodel')
pipeline.save('s3://billsdata.net/CommonCrawl/topic_model_4files/textpipeline')
```

```
%pyspark                                                                    FINISHED

# Get topic vectors for index pages (estimate of topic vec per domain)

vecs_index = vecs.filter("url LIKE '%index.html'")
results = ldaModel.transform(vecs_index)

# Drop text cols
results2=results.drop('text').drop('words').drop('filtered')

# Save domain topic vecs
results2.write.parquet('s3://billsdata.net/CommonCrawl/topic_model_4files/cc_index_page
```

```
%pyspark                                                                    FINISHED

# Create a dataset containing just the host, url and top 3 topic labels & scores
```

```
import pandas as pd
topicIndices = ldaModel.describeTopics(maxTermsPerTopic = 5).collect()
vocab = model.stages[2].vocabulary

topic_labels = []
for i, (topic, terms, termWeights) in enumerate(topicIndices):
    topwords = pd.Series(dict(zip([vocab[t] for t in terms], termWeights))).sort_values(
    topic_labels.append('_'.join(topwords.index.values))

topic_labels = np.array(topic_labels)

def topTopics(x):
    labels = topic_labels[np.argsort(x.topicDistribution)[::-1][:3]]
    scores = np.sort(x.topicDistribution)[::-1][:3]
    return (x.domain, x.url, str(labels[0]), float(scores[0]), str(labels[1]), float(sc

results3 = results2.rdd.map(topTopics)
results3 = results3.toDF(["host", "url", "topic1", "score1", "topic2", "score2", "topic

results3.write.parquet('s3://billsdata.net/CommonCrawl/topic_model_4files/cc_index_page_
results3.show()
```

```
%pyspark
```

READY