Experiment 10 - Batch Analysis using Spark

Roll No.	64
Name	Neeraj Rijhwani
Class	D15A
Subject	DS using Python Lab
LO Mapped	LO5: Design and Build an application that performs exploratory data analysis using Apache Spark

Aim:
To perform Batch Data Analysis using Spark

Introduction:

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of Apache Hadoop. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

MapReduce programming offers several benefits to help you gain valuable insights from your big data:

1. Scalability - Businesses can process petabytes of data stored in HDFS
2. Flexibility - Hadoop Enables easier access to multiple data sources and types of data.
3. Speed - With parallel processing and minimal data movement, large amounts of data can be processed quickly.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

Dataset Collection

The datasets used here are

1) Wikipedia article
   Format: Text file
   Preprocessing:
   A. The text file is loaded using SparkContext textFile method
   B. Remove Punctuation and Transform All Words to Lowercase.
   C. We use split function to separate the words in all lines .

D. We do a filtering below to exclude whitespaces.

2) Song Lyrics Dataset
   Billboard has published a Year-End Hot 100 every December since 1958. The chart measures the performance of singles in the U.S. throughout the year. Using R, I've combined the lyrics from 50 years of Billboard Year-End Hot 100 (1965-2015) into one dataset for analysis.

Approach:

Approach to count the words using Spark:

1. Let's create an RDD by using the following command
   data = sc.textFile("file_name.txt")

2. Here, pass any filename that contains the data. Now, we can read the generated result by using the following command.
   data.collect

3. Here, we split the existing data in the form of individual words by using the following command.
   splitdata= book.flatMap(lambda x: x.split()).countByValue()

4. Now, we can read the generated result by using the following command.
   splitdata.collect

5. Now, perform the map operation.
   ```
   for i, (word, count) in enumerate(word_counts.items()):
     if i == 100: break
     print(word, count)
   ```

   Here, we are assigning a value 1 to each word. Now, we can read the generated result by running the for loop.

6. Now, perform the reduce operation if needed.
   reducedata = mapdata.reduceByKey(lambda a,b : a+b)

   Here, we are summarizing the generated data.

Implementation:

Setup

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
!wget -q
https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
!tar xf spark-3.2.1-bin-hadoop3.2.tgz
```

Setting Environment Variables

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"
os.environ["PYTHONPATH"] =
"%SPARK_HOME%\python;%SPARK_HOME%\python\lib\py4j-0.10.9.3-src.zip:%PYTHONPAT
H%"
```

Setting the SparkContext

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setMaster("local").setAppName("word-counts")
sc = SparkContext(conf=conf)
```

Setting up the data

```
article = sc.textFile("Machine_Learning_Wikipedia.txt")
```

Preprocessing

```
def lower_clean_str(x):
  punc='!"#$%&\'()*+,./:;<=>?@[\\]^_`{|}~-'
  lowercased_str = x.lower()
  for ch in punc:
    lowercased_str = lowercased_str.replace(ch, '')
  return lowercased_str
article = article.map(lower_clean_str)
article=article.flatMap(lambda satir: satir.split(" "))
article = article.filter(lambda x:x!='')
```

Getting Word Count

```
article_count=article.map(lambda  word:(word,1))
article_count_RBK=article_count.reduceByKey(lambda x,y:(x+y)).sortByKey()
article_count_RBK=article_count_RBK.map(lambda x:(x[1],x[0]))
article_count_RBK.sortByKey(False).take(10)
```

Loading song lyrics dataset

```
import sys

from operator import add
from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer
from pyspark.ml.feature import StopWordsRemover
import pyspark.sql.functions as f

spark = SparkSession\
  .builder \
  .appName("PythonWordCount") \
  .getOrCreate()


data = spark.read.format('csv').options(header='true', inferSchema='true') \
  .load('billboard_lyrics_1964-2015.csv') \

print('############ CSV extract:')
data.show()

# Count and group word frequencies on the column Lyrics, when splitted by space comma
data.withColumn('word', f.explode(f.split(f.col('Lyrics'), ' '))) \
  .groupBy('word') \
  .count() \
  .sort('count', ascending=False) \
  .show()

# To remove stop words (like "I", "The", ...), we need to provide arrays of words,
not strings. Here we use APache Spark Tokenizer to do so.
# We create a new column to push our arrays of words
tokenizer = Tokenizer(inputCol="Lyrics", outputCol="words_token")
tokenized = tokenizer.transform(data).select('Rank','words_token')

print('############ Tokenized data extract:')
tokenized.show()


# Once in arrays, we can use the Apache Spark function StopWordsRemover
# A new column "words_clean" is here as an output
remover = StopWordsRemover(inputCol='words_token', outputCol='words_clean')
data_clean = remover.transform(tokenized).select('Rank', 'words_clean')

print('############ Data Cleaning extract:')
data_clean.show()
```

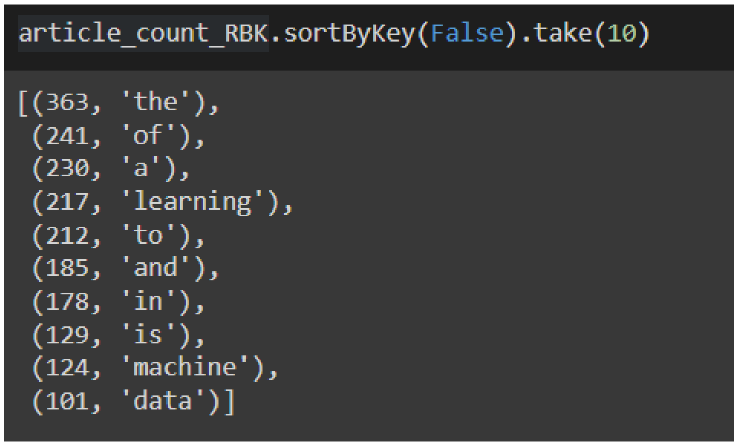
```
# Final step : like in the beginning, we can group again words and sort them by the
most used
result = data_clean.withColumn('word', f.explode(f.col('words_clean'))) \
  .groupBy('word') \
  .count().sort('count', ascending=False) \

print('############ TOP20 Most used words in Billboard songs are:')
result.show()

# Stop Spark Process
spark.stop()
```

Results:

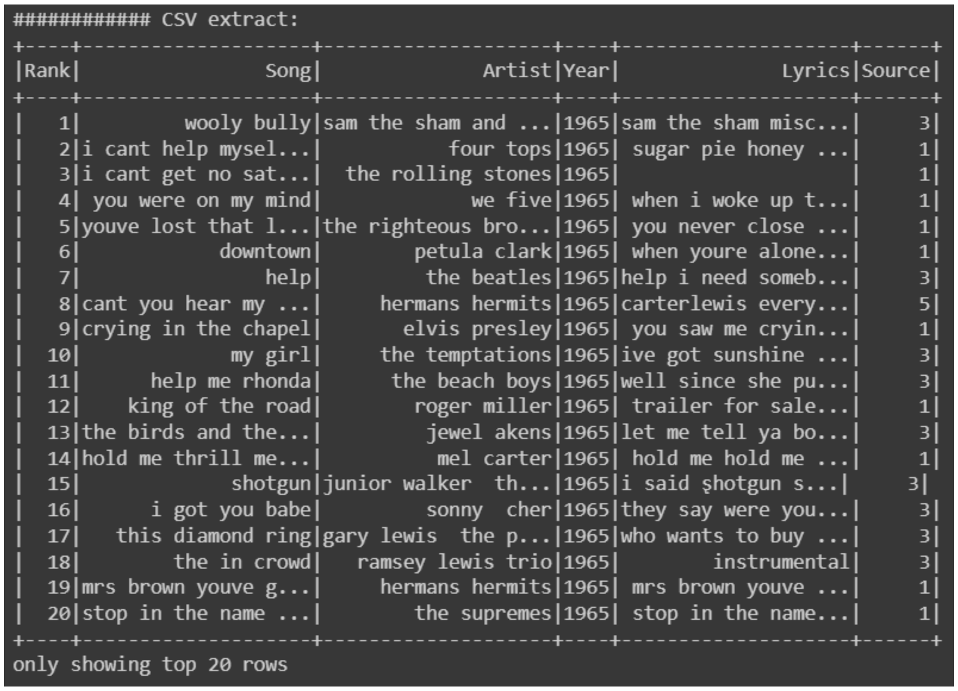
Article:



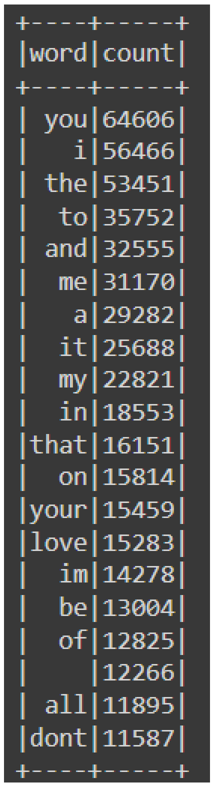
```
article_count_RBK.sortByKey(False).take(10)

[(363, 'the'),
 (241, 'of'),
 (230, 'a'),
 (217, 'learning'),
 (212, 'to'),
 (185, 'and'),
 (178, 'in'),
 (129, 'is'),
 (124, 'machine'),
 (101, 'data')]
```

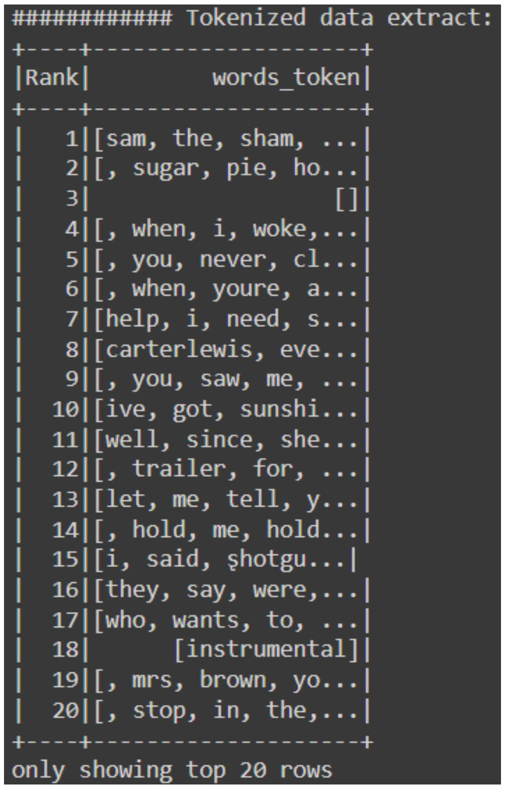
Song lyrics:



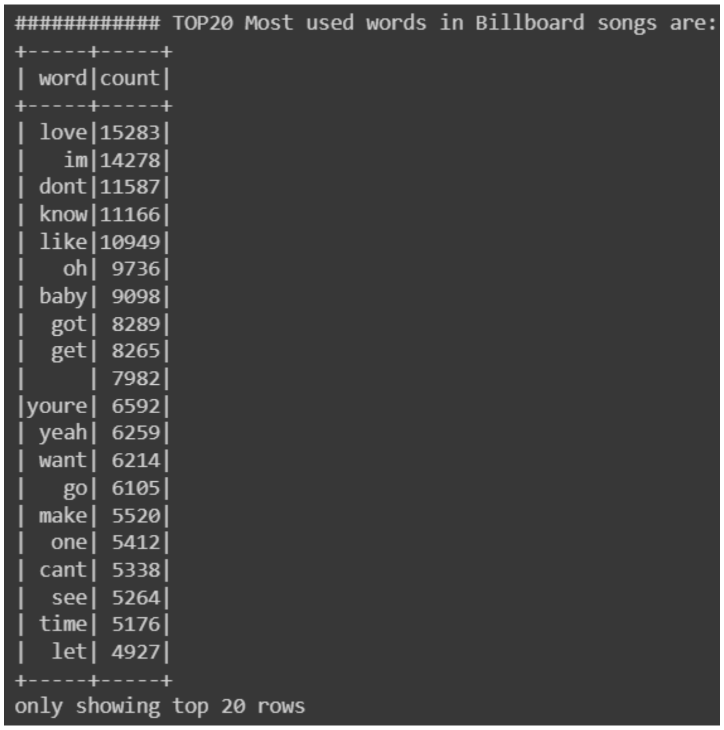
```
############ CSV extract:
+----+--------------------+--------------------+----+--------------------+------+
|Rank|                Song|              Artist|Year|              Lyrics|Source|
+----+--------------------+--------------------+----+--------------------+------+
|   1|         wooly bully|sam the sham and ...|1965|sam the sham misc...|     3|
|   2|i cant help mysel...|           four tops|1965| sugar pie honey ...|     1|
|   3|i cant get no sat...|  the rolling stones|1965|                    |     1|
|   4| you were on my mind|             we five|1965| when i woke up t...|     1|
|   5|youve lost that l...|the righteous bro...|1965| you never close ...|     1|
|   6|            downtown|        petula clark|1965| when youre alone...|     1|
|   7|                help|         the beatles|1965|help i need someb...|     3|
|   8|cant you hear my ...|     hermans hermits|1965|carterlewis every...|     5|
|   9|crying in the chapel|       elvis presley|1965| you saw me cryin...|     1|
|  10|             my girl|     the temptations|1965|ive got sunshine ...|     3|
|  11|      help me rhonda|      the beach boys|1965|well since she pu...|     3|
|  12|    king of the road|        roger miller|1965| trailer for sale...|     1|
|  13|the birds and the...|         jewel akens|1965|let me tell ya bo...|     3|
|  14|hold me thrill me...|          mel carter|1965| hold me hold me ...|     1|
|  15|             shotgun|junior walker  th...|1965|i said şhotgun s...|     3|
|  16|      i got you babe|         sonny  cher|1965|they say were you...|     3|
|  17|   this diamond ring|gary lewis  the p...|1965|who wants to buy ...|     3|
|  18|        the in crowd|   ramsey lewis trio|1965|        instrumental|     3|
|  19|mrs brown youve g...|     hermans hermits|1965| mrs brown youve ...|     1|
|  20|stop in the name ...|        the supremes|1965| stop in the name...|     1|
+----+--------------------+--------------------+----+--------------------+------+
only showing top 20 rows
```



```
+----+-----+
|word|count|
+----+-----+
| you|64606|
|   i|56466|
| the|53451|
|  to|35752|
| and|32555|
|  me|31170|
|   a|29282|
|  it|25688|
|  my|22821|
|  in|18553|
|that|16151|
|  on|15814|
|your|15459|
|love|15283|
|  im|14278|
|  be|13004|
|  of|12825|
|    |12266|
| all|11895|
|dont|11587|
+----+-----+
```



```
############ Tokenized data extract:
+----+--------------------+
|Rank|         words_token|
+----+--------------------+
|   1|[sam, the, sham, ...|
|   2|[, sugar, pie, ho...|
|   3|                  []|
|   4|[, when, i, woke,...|
|   5|[, you, never, cl...|
|   6|[, when, youre, a...|
|   7|[help, i, need, s...|
|   8|[carterlewis, eve...|
|   9|[, you, saw, me, ...|
|  10|[ive, got, sunshi...|
|  11|[well, since, she...|
|  12|[, trailer, for, ...|
|  13|[let, me, tell, y...|
|  14|[, hold, me, hold...|
|  15|[i, said, şhotgu...|
|  16|[they, say, were,...|
|  17|[who, wants, to, ...|
|  18|      [instrumental]|
|  19|[, mrs, brown, yo...|
|  20|[, stop, in, the,...|
+----+--------------------+
only showing top 20 rows
```



```
############ TOP20 Most used words in Billboard songs are:
+-----+-----+
| word|count|
+-----+-----+
| love|15283|
|   im|14278|
| dont|11587|
| know|11166|
| like|10949|
|   oh| 9736|
| baby| 9098|
|  got| 8289|
|  get| 8265|
|     | 7982|
|youre| 6592|
| yeah| 6259|
| want| 6214|
|   go| 6105|
| make| 5520|
|  one| 5412|
| cant| 5338|
|  see| 5264|
| time| 5176|
|  let| 4927|
+-----+-----+
only showing top 20 rows
```

<u>Conclusion</u>:

Thus, we have learnt what batch processing is and also learnt how to implement it using Spark.