Spark Developer Training - 3 Days

Manaranjan Pradhan
manaranjan@enablecloud.com

Working with Spark DataFrames: Analysing Movielens Data ¶

Things to learn

- Reading a file into spark dataframes
- Applying schema while reading records into a dataframe
- Displaying records
- Applying operations like grouping, sorting, aggregating, filtering etc.
- Joining multiple dataframes
- Utility functions like describing schema, showing records, rename columns, listing columns etc

Documentation for Spark DataFrame APIs are available at

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame (http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame)

In [1]:

```
sc
```

Out[1]:

```
<pyspark.context.SparkContext at 0x7fea34258240>
```

Use SQLContext to load and read structured data

In [2]:

```
## Create an sql context
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Read the tab separated file. Which contains userid, movieid, ratings and timestamp

In [3]:

```
ratings = sqlContext.read.format("com.databricks.spark.csv")           \
            .options(delimiter='\t')                                   \
            .load('file:///home/hadoop/lab/data/movies/ratings.dat')
```

Note: The details of the the csv reader is given in the url https://github.com/databricks/spark-csv (https://github.com/databricks/spark-csv)

In [4]:

```
# What is the data type of the ratings variable. It should be a dataframe.
ratings
```

Out[4]:

```
DataFrame[C0: string, C1: string, C2: string, C3: string]
```

Displaying Records

In [5]:

```
# Read the first few rows of the dataframe
ratings.show()
```

```
+---+----+---+---------+
| C0|  C1| C2|       C3|
+---+----+---+---------+
|  1|1193|  5|978300760|
|  1| 661|  3|978302109|
|  1| 914|  3|978301968|
|  1|3408|  4|978300275|
|  1|2355|  5|978824291|
|  1|1197|  3|978302268|
|  1|1287|  5|978302039|
|  1|2804|  5|978300719|
|  1| 594|  4|978302268|
|  1| 919|  4|978301368|
|  1| 595|  5|978824268|
|  1| 938|  4|978301752|
|  1|2398|  4|978302281|
|  1|2918|  4|978302124|
|  1|1035|  5|978301753|
|  1|2791|  4|978302188|
|  1|2687|  3|978824268|
|  1|2018|  4|978301777|
|  1|3105|  5|978301713|
|  1|2797|  4|978302039|
+---+----+---+---------+
only showing top 20 rows
```

Describe the schema of the records

In [6]:

```
ratings.printSchema()
```

```
root
 |-- C0: string (nullable = true)
 |-- C1: string (nullable = true)
 |-- C2: string (nullable = true)
 |-- C3: string (nullable = true)
```

In [7]:

```
ratings.schema
```

Out[7]:

```
StructType(List(StructField(C0,StringType,true),StructField(C1,StringTy
pe,true),StructField(C2,StringType,true),StructField(C3,StringType,tru
e)))
```

Apply the schema to the dataframe

In [8]:

```
from pyspark.sql.types import *
fields = [StructField("userid", IntegerType(), True),
          StructField("movieid", IntegerType(), True),
          StructField("rating", IntegerType(), True),
          StructField("timestamp", LongType(), True) ]
```

Applying the schema, while reading the records

In [9]:

```
## Read the tab separated file. Which contains userid, movieid, ratings and timestamp
ratings_df = sqlContext.read.format("com.databricks.spark.csv")                    \
            .options(delimiter='\t')                                               \
            .load('file:///home/hadoop/lab/data/movies/ratings.dat',
                  schema = StructType(fields) )
```

In [10]:

```
ratings_df.show()
```

```
+------+-------+------+---------+
|userid|movieid|rating|timestamp|
+------+-------+------+---------+
|     1|   1193|     5|978300760|
|     1|    661|     3|978302109|
|     1|    914|     3|978301968|
|     1|   3408|     4|978300275|
|     1|   2355|     5|978824291|
|     1|   1197|     3|978302268|
|     1|   1287|     5|978302039|
|     1|   2804|     5|978300719|
|     1|    594|     4|978302268|
|     1|    919|     4|978301368|
|     1|    595|     5|978824268|
|     1|    938|     4|978301752|
|     1|   2398|     4|978302281|
|     1|   2918|     4|978302124|
|     1|   1035|     5|978301753|
|     1|   2791|     4|978302188|
|     1|   2687|     3|978824268|
|     1|   2018|     4|978301777|
|     1|   3105|     5|978301713|
|     1|   2797|     4|978302039|
+------+-------+------+---------+
only showing top 20 rows
```

In [11]:

```
ratings_df.printSchema()
```

```
root
 |-- userid: integer (nullable = true)
 |-- movieid: integer (nullable = true)
 |-- rating: integer (nullable = true)
 |-- timestamp: long (nullable = true)
```

Show the list of columns in the dataframe

In [12]:

```
# Return a list of columns
ratings_df.columns
```

Out[12]:

```
['userid', 'movieid', 'rating', 'timestamp']
```

In [13]:

```
## How many records in the dataframe?
ratings_df.count()
```

Out[13]:

```
1000209
```

Drop a column

In [14]:

```
## We donot need the timestamp column.. let's drop it
ratings_df = ratings_df.drop( 'timestamp')
```

In [15]:

```
ratings_df.show()
```

```
+------+-------+------+
|userid|movieid|rating|
+------+-------+------+
|     1|   1193|     5|
|     1|    661|     3|
|     1|    914|     3|
|     1|   3408|     4|
|     1|   2355|     5|
|     1|   1197|     3|
|     1|   1287|     5|
|     1|   2804|     5|
|     1|    594|     4|
|     1|    919|     4|
|     1|    595|     5|
|     1|    938|     4|
|     1|   2398|     4|
|     1|   2918|     4|
|     1|   1035|     5|
|     1|   2791|     4|
|     1|   2687|     3|
|     1|   2018|     4|
|     1|   3105|     5|
|     1|   2797|     4|
+------+-------+------+
only showing top 20 rows
```

Applying operations like groupby() and sort()

In [16]:

```
movie_counts = ratings_df.groupBy("movieid").count()
```

In [17]:

```
from pyspark.sql.functions import *
movie_counts = movie_counts.sort(desc("count"))
```

In [18]:

```
movie_counts.show( 10 )
```

```
+-------+-----+
|movieid|count|
+-------+-----+
|   2858| 3428|
|    260| 2991|
|   1196| 2990|
|   1210| 2883|
|    480| 2672|
|   2028| 2653|
|    589| 2649|
|   2571| 2590|
|   1270| 2583|
|    593| 2578|
+-------+-----+
only showing top 10 rows
```

Applying an aggregation function to the group by

In [19]:

```
avg_ratings = ratings_df.groupBy("movieid").agg( {"rating":"avg"} )
```

In [20]:

```
avg_ratings.printSchema()
```

```
root
 |-- movieid: integer (nullable = true)
 |-- avg(rating): double (nullable = true)
```

In [21]:

```
avg_ratings = avg_ratings.sort( desc( "avg(rating)" ) )
```

In [22]:

```
avg_ratings.show( 10 )
```

```
+-------+-----------+
|movieid|avg(rating)|
+-------+-----------+
|    989|        5.0|
|   3607|        5.0|
|   3382|        5.0|
|   3172|        5.0|
|   3656|        5.0|
|    787|        5.0|
|   3881|        5.0|
|   3233|        5.0|
|   3280|        5.0|
|   1830|        5.0|
+-------+-----------+
only showing top 10 rows
```

Joining multiple dataframes

In [23]:

```
avg_ratings_count = avg_ratings.join( movie_counts,
                                      avg_ratings.movieid == movie_counts.movieid ,
                                      'inner' ).drop(movie_counts.movieid)
```

In [24]:

```
avg_ratings_count.printSchema()
```

```
root
 |-- avg(rating): double (nullable = true)
 |-- movieid: integer (nullable = true)
 |-- count: long (nullable = false)
```

Renaming a column in a dataframe

In [25]:

```
avg_ratings_count = avg_ratings_count.withColumnRenamed( "avg(rating)",
                                                         "mean_rating" )
```

In [26]:

```
avg_ratings_count.printSchema()
```

```
root
 |-- mean_rating: double (nullable = true)
 |-- movieid: integer (nullable = true)
 |-- count: long (nullable = false)
```

In [27]:

```
avg_ratings_count = avg_ratings_count                                    \
                    .withColumn( "mean_rating",
                                 round( avg_ratings_count["mean_rating"]
                                       , 2 ) )
```

In [28]:

```
avg_ratings_count = avg_ratings_count.sort( desc( "mean_rating" ) )
```

In [29]:

```
avg_ratings_count.show( 10 )
```

```
+-----------+-------+-----+
|mean_rating|movieid|count|
+-----------+-------+-----+
|        5.0|   3607|    1|
|        5.0|    989|    1|
|        5.0|   3382|    1|
|        5.0|   3172|    1|
|        5.0|    787|    3|
|        5.0|   3656|    1|
|        5.0|   3280|    1|
|        5.0|   3881|    1|
|        5.0|   3233|    2|
|        5.0|   1830|    1|
+-----------+-------+-----+
only showing top 10 rows
```

Filtering records in a dataframe based on a criteria

In [30]:

```
avg_ratings_count = avg_ratings_count.filter( avg_ratings_count["count"] > 20 )
```

In [31]:

```
avg_ratings_count = avg_ratings_count.sort( desc( "mean_rating" ) , desc( "count") )
```

In [32]:

```
avg_ratings_count.show( 10 )
```

```
+-----------+-------+-----+
|mean_rating|movieid|count|
+-----------+-------+-----+
|       4.61|   2905|   69|
|       4.56|   2019|  628|
|       4.55|    318| 2227|
|       4.52|    858| 2223|
|       4.52|     50| 1783|
|       4.52|    745|  657|
|       4.51|    527| 2304|
|       4.51|   1148|  882|
|       4.49|    922|  470|
|       4.48|   1198| 2514|
+-----------+-------+-----+
only showing top 10 rows
```

Loading movies data

In [33]:

```
movies_df = sqlContext.read.format("com.databricks.spark.csv")           \
            .options(delimiter='\t', header = True, inferSchema = True)  \
            .load('file:///home/hadoop/lab/data/movies/movies.dat')
```

In [34]:

```
movies_df.show( 10 )
```

```
+-------+------------------+--------------------+
|movieid|              name|                tags|
+-------+------------------+--------------------+
|      1|  Toy Story (1995)|Animation|Childre...|
|      2|    Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|        Comedy|Drama|
|      5|Father of the Bri...|              Comedy|
|      6|       Heat (1995)|Action|Crime|Thri...|
|      7|    Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|Adventure|Children's|
|      9| Sudden Death (1995)|              Action|
|     10|  GoldenEye (1995)|Action|Adventure|...|
+-------+------------------+--------------------+
only showing top 10 rows
```

In [35]:

```
movies_df.printSchema()
```

```
root
 |-- movieid: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- tags: string (nullable = true)
```

Joining Ratings and Movies data to find top 20 best rated movies

In [36]:

```
top_movies = avg_ratings_count.limit(20)                                \
            .join( movies_df,
                  avg_ratings_count.movieid == movies_df.movieid,
                  "inner" ).drop(movies_df.movieid)
```

In [37]:

```
top_movies_20 = top_movies.select( "movieid", "mean_rating", "count", "name" )
```

In [38]:

```
top_movies_20.collect()
```

Out[38]:

```
[Row(movieid=50, mean_rating=4.52, count=1783, name='Usual Suspects, Th
e (1995)'),
 Row(movieid=260, mean_rating=4.45, count=2991, name='Star Wars: Episod
e IV - A New Hope (1977)'),
 Row(movieid=318, mean_rating=4.55, count=2227, name='Shawshank Redempt
ion, The (1994)'),
 Row(movieid=527, mean_rating=4.51, count=2304, name="Schindler's List
(1993)"),
 Row(movieid=720, mean_rating=4.43, count=438, name='Wallace & Gromit:
The Best of Aardman Animation (1996)'),
 Row(movieid=745, mean_rating=4.52, count=657, name='Close Shave, A (19
95)'),
 Row(movieid=750, mean_rating=4.45, count=1367, name='Dr. Strangelove o
r: How I Learned to Stop Worrying and Love the Bomb (1963)'),
 Row(movieid=858, mean_rating=4.52, count=2223, name='Godfather, The (1
972)'),
 Row(movieid=904, mean_rating=4.48, count=1050, name='Rear Window (195
4)'),
 Row(movieid=922, mean_rating=4.49, count=470, name='Sunset Blvd. (a.k.
a. Sunset Boulevard) (1950)'),
 Row(movieid=1148, mean_rating=4.51, count=882, name='Wrong Trousers, T
he (1993)'),
 Row(movieid=1178, mean_rating=4.47, count=230, name='Paths of Glory (1
957)'),
 Row(movieid=1198, mean_rating=4.48, count=2514, name='Raiders of the L
ost Ark (1981)'),
 Row(movieid=1207, mean_rating=4.43, count=928, name='To Kill a Mocking
bird (1962)'),
 Row(movieid=1212, mean_rating=4.45, count=480, name='Third Man, The (1
949)'),
 Row(movieid=2019, mean_rating=4.56, count=628, name='Seven Samurai (Th
e Magnificent Seven) (Shichinin no samurai) (1954)'),
 Row(movieid=2762, mean_rating=4.41, count=2459, name='Sixth Sense, The
(1999)'),
 Row(movieid=2905, mean_rating=4.61, count=69, name='Sanjuro (1962)'),
 Row(movieid=3338, mean_rating=4.44, count=27, name='For All Mankind (1
989)'),
 Row(movieid=3435, mean_rating=4.42, count=551, name='Double Indemnity
(1944)')]
```

Saving the results into a csv file

In [41]:

```
top_movies_20.write                                          \
    .format("com.databricks.spark.csv")                      \
    .option("header", "true")                                \
    .save("file:///home/hadoop/lab/results/topmovies")
```

Exercises

- Find out 20 worst rated movies. But only consider those movies which are rated by at least 100 users.
- Find out best 10 and worst 10 movies in each category - Categories are tags given in the *movies.data* file

What we learnt

Please make a note of things that you learnt.