Học phần Quản trị Dữ liệu lớn: Bài thực hành số 4

Phạm Tiến Lâm, Đặng Văn Báu

1. Làm quen với spark
   - Tạo file trên google colab
   - Đặt tên như sau: “Ho_va_ten__MSV”
   - Cài đặt thư viện sau:

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
```

Activity 1.

- Khai báo thư viện

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql import Row
3 from pyspark.sql import functions
```

- Tạo hàm loadMovieNames() để đọc file:

```
1  def loadMovieNames():
2      movieNames = {}
3      with open("ml-100k/u.item", encoding="latin1") as f:
4          for line in f:
5              fields = line.split('|')
6              movieNames[int(fields[0])] = fields[1]
7      return movieNames
8
9  movieNames = loadMovieNames()
10 print(movieNames)
```

- Khởi tạo hàm parseInput():

```
1 def parseInput(line):
2     fields = line.split()
3     return Row(movieID = int(fields[1]), rating = float(fields[2]))
```

- Sử dụng SparkSession để tạo DataFrame dạng bảng. Thực thi SQL trên bảng.

```
1 spark = SparkSession.builder.appName("PopularMovies").getOrCreate()
```

- Đọc file

```
2 lines = spark.sparkContext.textFile("ml-100k/u.data")
```

- Sử dụng hàm parseInput() đã khởi tạo ở trên và mapping với dữ liệu.

```
movies = lines.map(parseInput)
movies.take(10)
```

```
[Row(movieID=50, rating=5.0),
 Row(movieID=172, rating=5.0),
 Row(movieID=133, rating=1.0),
 Row(movieID=242, rating=3.0),
 Row(movieID=302, rating=3.0),
 Row(movieID=377, rating=1.0),
 Row(movieID=51, rating=2.0),
 Row(movieID=346, rating=1.0),
 Row(movieID=474, rating=4.0),
 Row(movieID=265, rating=2.0)]
```

- Khởi tạo dataFrame có tên movieDataset():

```
movieDataset = spark.createDataFrame(movies)
```

```
movieDataset.take(10)
```

```
[Row(movieID=50, rating=5.0),
 Row(movieID=172, rating=5.0),
 Row(movieID=133, rating=1.0),
 Row(movieID=242, rating=3.0),
 Row(movieID=302, rating=3.0),
 Row(movieID=377, rating=1.0),
 Row(movieID=51, rating=2.0),
 Row(movieID=346, rating=1.0),
 Row(movieID=474, rating=4.0),
 Row(movieID=265, rating=2.0)]
```

- Tính trung bình điểm rating của các movieID từ movieDataset:

```
averageRatings = movieDataset.groupBy("movieID").avg("rating")
averageRatings.take(10)
```

```
[Row(movieID=474, avg(rating)=4.252577319587629),
 Row(movieID=29, avg(rating)=2.6666666666666665),
 Row(movieID=26, avg(rating)=3.452054794520548),
 Row(movieID=964, avg(rating)=3.3333333333333335),
 Row(movieID=1677, avg(rating)=3.0),
 Row(movieID=65, avg(rating)=3.5391304347826087),
 Row(movieID=191, avg(rating)=4.163043478260869),
 Row(movieID=1224, avg(rating)=2.6666666666666665),
 Row(movieID=558, avg(rating)=3.6714285714285713),
 Row(movieID=1010, avg(rating)=3.25)]
```

- Đếm số lượt rate cho từng movieID:

```
counts = movieDataset.groupBy("movieID").count()
counts.take(10)
```

```
[Row(movieID=474, count=194),
 Row(movieID=29, count=114),
 Row(movieID=26, count=73),
 Row(movieID=964, count=9),
 Row(movieID=1677, count=1),
 Row(movieID=65, count=115),
 Row(movieID=191, count=276),
 Row(movieID=1224, count=12),
 Row(movieID=558, count=70),
 Row(movieID=1010, count=44)]
```

- Kết hợp 2 bảng counts và averageRatings:

```
averagesAndCounts = counts.join(averageRatings, "movieID")
averagesAndCounts.take(10)
```

```
[Row(movieID=26, count=73, avg(rating)=3.452054794520548),
 Row(movieID=29, count=114, avg(rating)=2.6666666666666665),
 Row(movieID=474, count=194, avg(rating)=4.252577319587629),
 Row(movieID=964, count=9, avg(rating)=3.3333333333333335),
 Row(movieID=1677, count=1, avg(rating)=3.0),
 Row(movieID=65, count=115, avg(rating)=3.5391304347826087),
 Row(movieID=191, count=276, avg(rating)=4.163043478260869),
 Row(movieID=418, count=129, avg(rating)=3.5813953488372094),
 Row(movieID=541, count=49, avg(rating)=2.877551020408163),
 Row(movieID=558, count=70, avg(rating)=3.6714285714285713)]
```

- Sắp xếp và hiển thị bảng averagesAndCounts theo avg(rating):

```
sorted_DF = averagesAndCounts.orderBy("avg(rating)")
topTen = sorted_DF.head(10)
```

```
for movie in topTen:
        print (movieNames[movie[0]], movie[1], movie[2])

# Stop the session
spark.stop()
```

```
Touki Bouki (Journey of the Hyena) (1973) 1 1.0
Amityville: Dollhouse (1996) 3 1.0
Quartier Mozart (1992) 1 1.0
Power 98 (1995) 1 1.0
Amityville: A New Generation (1993) 5 1.0
Lotto Land (1995) 1 1.0
Hostile Intentions (1994) 1 1.0
Falling in Love Again (1980) 2 1.0
The Courtyard (1995) 1 1.0
Bloody Child, The (1996) 1 1.0
```

2. Bài tập

Activity 2.

- Khai báo thư viện:

```
1  from pyspark import SparkConf, SparkContext
```

- Khởi tạo hàm loadMovieNames() để load file “u.item” và lưu vào movieNames

```
1  def loadMovieNames():
2      movieNames = {}
3      with open("ml-100k/u.item", encoding="latin1") as f:
4          for line in f:
5              fields = line.split('|')
6              movieNames[int(fields[0])] = fields[1]
7      return movieNames
8
9  movieNames = loadMovieNames()
10 print(movieNames)
```

- Sử dụng SparkContext tạo RDD từ file “u.data" với 2 trường thông tin “movieID” và “rating”. Khởi tạo hàm parseInput() để thực thi.

```
1  movieRatings = lines.map(parseInput)
```

```
1  movieRatings.take(10)
```

```
[(50, (5.0, 1.0)),
 (172, (5.0, 1.0)),
 (133, (1.0, 1.0)),
 (242, (3.0, 1.0)),
 (302, (3.0, 1.0)),
 (377, (1.0, 1.0)),
 (51, (2.0, 1.0)),
 (346, (1.0, 1.0)),
 (474, (4.0, 1.0)),
 (265, (2.0, 1.0))]
```

- Tạo RDD in ra chứa thông tin của từng movieID: (countRating, totalRating)

```
1 ratingTotalsAndCount.take(10)
```

```
[(50, (2546.0, 584.0)),
 (172, (1548.0, 368.0)),
 (242, (467.0, 117.0)),
 (302, (1236.0, 297.0)),
 (346, (459.0, 126.0)),
 (474, (825.0, 194.0)),
 (86, (591.0, 150.0)),
 (1014, (300.0, 98.0)),
 (222, (1336.0, 365.0)),
 (40, (165.0, 57.0))]
```

- Tạo RDD tính trung bình rating của từng movieID movieID: (averageRatings)

```
2 averageRatings.take(10)
3
```

```
[(50, 4.359589041095891),
 (172, 4.206521739130435),
 (242, 3.9914529914529915),
 (302, 4.161616161616162),
 (346, 3.642857142857143),
 (474, 4.252577319587629),
 (86, 3.94),
 (1014, 3.061224489795918),
 (222, 3.66027397260274),
 (40, 2.8947368421052633)]
```

- Tạo RDD sắp xếp movieID theo averageRatings

```
[(858, 1.0),
 (1334, 1.0),
 (1348, 1.0),
 (1320, 1.0),
 (314, 1.0),
 (1364, 1.0),
 (830, 1.0),
 (784, 1.0),
 (1374, 1.0),
 (1582, 1.0)]
```

Activity 3.

- Tạo một ma trận X chứa các users.
- Mỗi user là tuple (key, values)

  Với key là userID

  values là một hotVector

  (Ví dụ user50 rate cho movieID = 3 là 4*

  và movieID = 4 là 5*

  Thì hotVector có dạng là[0 0 0 4 5 0 0 0 …. len(movie)])

  (vector có kiểu dữ liệu list, với index và giá trị là movieID và số rate tương ứng)
- Tính khoảng cách giữa 1 vector với tất cả các vector còn lại.
- Recommend movie cho các user có khoảng cách lớn nhất và điểm rate của các moviedID > 3.
- Print list of watched movies for user 0
- Print list of recommended movies for user 0