董西成 -2.1(2) 简易电影受众系统

1、首先下载分析文件网址为:<a href="http://grouplens.org/datasets/movielens/">http://grouplens.org/datasets/movielens/</a>(MovieLens 1M Dataset中的这个包ml-1m.zip)

2、部分文件如下:

movies.dat:

```
<pre class="lang:default decode:true ">1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
11::American President, The (1995)::Comedy|Drama|Romance
12::Dracula: Dead and Loving It (1995)::Comedy|Horror
13::Balto (1995)::Animation|Children's
14::Nixon (1995)::Drama
15::Cutthroat Island (1995)::Action|Adventure|Romance
16::Casino (1995)::Drama|Thriller
17::Sense and Sensibility (1995)::Drama|Romance
18::Four Rooms (1995)::Thriller
19::Ace Ventura: When Nature Calls (1995)::Comedy
20::Money Train (1995)::Action
21::Get Shorty (1995)::Action|Comedy|Drama
22::Copycat (1995)::Crime|Drama|Thriller
23::Assassins (1995)::Thriller
24::Powder (1995)::Drama|Sci-Fi
25::Leaving Las Vegas (1995)::Drama|Romance
26::Othello (1995)::Drama
27::Now and Then (1995)::Drama
28::Persuasion (1995)::Romance
29::City of Lost Children, The (1995)::Adventure|Sci-Fi
30::Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)::Drama
31::Dangerous Minds (1995)::Drama</pre>
```

ratings.dat

```
<pre class="lang:default decode:true ">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
1::2321::3::978302205
1::720::3::978300760
1::1270::5::978300055
1::527::5::978824195
1::2340::3::978300103</pre>
```

users.dat

```
<pre class="lang:default decode:true ">1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
11::F::25::1::04093
12::M::25::12::32793
13::M::45::1::93304
14::M::35::0::60126
15::M::25::7::22903
16::F::35::0::20670
17::M::50::1::95350
18::F::18::3::95825
19::M::1::10::48073
20::M::25::14::55113
21::M::18::16::99353
22::M::18::15::53706
23::M::35::0::90049
24::F::25::7::10023
25::M::18::4::01609
26::M::25::7::23112
27::M::25::11::19130
28::F::25::1::14607
29::M::35::7::33407</pre>
```

```
<pre class="lang:scala decode:true">package org.training.spark.core

import org.apache.spark._

/**
 * 看过 “Lord of the Rings, The (1978)” 用户年龄和性别分布
 */
object MovieUserAnalyzer {
```

```
def main(args: Array[String]) {
  var masterUrl = "local[1]"
  var dataPath = "data/ml-1m/"
  if (args.length &gt; 0) {
    masterUrl = args(0)
  } else if(args.length &gt; 1) {
    dataPath = args(1)
  }

  // Create a SparContext with the given master URL
  val conf = new SparkConf().setMaster(masterUrl).setAppName("MovieUserAnalyzer")
  val sc = new SparkContext(conf)

  /**
   * Step 1: Create RDDs
   */
  val DATA_PATH = dataPath
  val MOVIE_TITLE = "Lord of the Rings, The (1978)"
  val MOVIE_ID = "2116"


  val usersRdd = sc.textFile(DATA_PATH + "users.dat")
  val ratingsRdd = sc.textFile(DATA_PATH + "ratings.dat")


  /**
   * Step 2: Extract columns from RDDs
   */


  //users: RDD[(userID, (gender, age))]
  val users = usersRdd.map(_.split("::")).map { x =&gt;
    (x(0), (x(1), x(2)))
  }

  //rating: RDD[Array(userID, movieID, ratings, timestamp)]
  val rating = ratingsRdd.map(_.split("::"))

  //usermovie: RDD[(userID, movieID)]
  val usermovie = rating.map{ x =&gt;
    (x(0), x(1))
  }.filter(_._2.equals(MOVIE_ID))

  /**
   * Step 3: join RDDs
   */

  //useRating: RDD[(userID, (movieID, (gender, age))]
  val userRating = usermovie.join(users)

  //userRating.take(1).foreach(print)


  //movieuser: RDD[(movieID, (movieTile, (gender, age))]
    val userDistribution = userRating.map { x =&gt;
      (x._2._2, 1)
    }.reduceByKey(_ + _)

    userDistribution.foreach(println)

    sc.stop()
  }
}</pre>

<pre class="lang:scala decode:true">package org.training.spark.core

import org.apache.spark._

import scala.collection.immutable.HashSet

/**
 * 年龄段在“18-24”的男性年轻人，最喜欢看哪10部电影
 */
object PopularMovieAnalyzer {
  def main(args: Array[String]) {
    var masterUrl = "local[1]"
    var dataPath = "data/ml-1m/"
    if (args.length &gt; 0) {
      masterUrl = args(0)
    } else if(args.length &gt; 1) {
      dataPath = args(1)
    }

    // Create a SparContext with the given master URL
    val conf = new SparkConf().setMaster(masterUrl).setAppName("PopularMovieAnalyzer")
    val sc = new SparkContext(conf)

    /**
     * Step 1: Create RDDs
     */
    val DATA_PATH = dataPath
    val USER_AGE = "18"


    val usersRdd = sc.textFile(DATA_PATH + "users.dat")
    val moviesRdd = sc.textFile(DATA_PATH + "movies.dat")
    val ratingsRdd = sc.textFile(DATA_PATH + "ratings.dat")

    /**
     * Step 2: Extract columns from RDDs
     */


    //users: RDD[(userID, age)]
    val users = usersRdd.map(_.split("::")).map { x =&gt;
      (x(0), x(2))
    }.filter(_._2.equals(USER_AGE))
    //Array[String]
    val userlist = users.map(_._1).collect()

    //broadcast
    val userSet = HashSet() ++ userlist
    val broadcastUserSet = sc.broadcast(userSet)


    /**
     * Step 3: map-side join RDDs
     */

    val topKmovies = ratingsRdd.map(_.split("::")).map{ x =&gt;
      (x(0), x(1))
    }.filter { x =&gt;
      broadcastUserSet.value.contains(x._1)
    }.map{ x=&gt;
      (x._2, 1)
    }.reduceByKey(_ + _).map{ x =&gt;
      (x._2, x._1)
    }.sortByKey(false).map{ x=&gt;
      (x._2, x._1)
    }.take(10)

    /**
     * Transfrom filmID to fileName
     */
    val movieID2Name = moviesRdd.map(_.split("::")).map { x =&gt;
      (x(0), x(1))
    }.collect().toMap

    topKmovies.map(x =&gt; (movieID2Name.getOrElse(x._1, null), x._2)).foreach(println)

    println(System.currentTimeMillis())

    sc.stop()
  }
}
</pre>


<pre class="lang:scala decode:true ">package org.training.spark.core

import org.apache.spark._

import scala.collection.immutable.HashSet

/**
 * 得分最高的10部电影；看过电影最多的前10个人；女性看多最多的10部电影；男性看过最多的10
电影
 */
object TopKMovieAnalyzer {
  def main(args: Array[String]) {
    var masterUrl = "local[1]"
    var dataPath = "data/ml-1m/"
    if (args.length &gt; 0) {
      masterUrl = args(0)
    } else if(args.length &gt; 1) {
      dataPath = args(1)
    }

    // Create a SparContext with the given master URL
    val conf = new SparkConf().setMaster(masterUrl).setAppName("TopKMovieAnalyzer")
    val sc = new SparkContext(conf)

    /**
     * Step 1: Create RDDs
     */
    val DATA_PATH = dataPath

    val ratingsRdd = sc.textFile(DATA_PATH + "ratings.dat")

    /**
     * Step 2: Extract columns from RDDs
     */

    //users: RDD[(userID, movieID, score)]
    val ratings = ratingsRdd.map(_.split("::")).map { x =&gt;
      (x(0), x(1), x(2))
    }.cache


    /**
     * Step 3: analyze result
     */

    val topKScoreMostMovie = ratings.map{x =&gt;
      (x._2, (x._3.toInt, 1))
    }.reduceByKey { (v1, v2) =&gt;
      (v1._1 + v2._1, v1._2 + v2._2)
    }.map { x =&gt;
      (x._2._1.toFloat / x._2._2.toFloat, x._1)
    }.sortByKey(false).
        take(10).
        foreach(println)


    val topKmostPerson = ratings.map{ x =&gt;
      (x._1, 1)
    }.reduceByKey(_ + _).
        map(x =&gt; (x._2, x._1)).
        sortByKey(false).
        take(10).
        foreach(println)

    sc.stop()
  }
}
</pre>
```

&nbsp;