# Mahout谱聚类

该模块主要为在Mahout中进行Spectral Clustering准备数据。

主要实现：对用户和宾馆ID进行映射；构建<hid, uid, rating>三元组；填补三元组形成方阵；利用Mahout的Spectral Clustering进行谱聚类。

涉及到的数据文件有：

* Original matrix: Clustering/originalMatrix.txt
* Parsed matrix: Clustering/parsedMatrix.txt
* squareMatrix: Clustering/squareMatrix.txt
* clusteringResult: Clustering/clusteringResult.txt
* userReflectionTable
* hotelReflectionTable

类说明如下：

1. Preprocessing/BuildMatrix.java
   1. 用途：从原始文件中提取user与hotel的信息，构建<hid, uid, rating >三元组，即original matrix
   2. 步骤：
      1. 从HotelUserInfo中将数据读入内存
      2. 与blacklist中的黑名单用户比较，去掉黑名单用户
      3. 提取UserID, HotelID和Overall Rating，写入文件
   3. 输入：HotelUserInfo
   4. 输出：originalMatrix.txt
   5. 输出示例： 46704行

|  |
| --- |
| 80802,gourmettwo,4  80802,LNTraveller,4  80802,Bonvoyage79,3  80802,Northcutts,4  80802,TunaSlayer,5  80802,wish71,5  80802,travelbee07,2  80802,AllieL,4  80802,BillCarl,5  80802,q4quality,3 |

1. Preprocessing/ReflectMatrix.java
   1. 用途：对用户和宾馆ID映射，形成ParsedMatrix，再补全矩阵形成SquareMatrix
   2. 步骤：
      1. 将OriginalMatrix读入内存
      2. 利用HashMap生成User和Hotel的映射表userReflectionTable和hotelReflectionTable，并对OriginalMatrix进行映射替换，生成ParsedMatrix
      3. 构建邻接矩阵，填充的规则为：

newHid = uid + # of hotels

newUid = hid + # of users

* + 1. 将生成的邻接矩阵写入SquareMatrix
  1. 输入：originalMatrix.txt
  2. 输出：parsedMatrix.txt, squarematrix.txt, userReflectionTable.txt, hotelReflectionTable.txt
  3. 输出示例：
     1. parsedMatrix.txt (46704)

|  |
| --- |
| 0,0,4  0,1,4  0,2,4  0,3,5  0,4,2 |

* + 1. squareMatrix.txt (93408)

|  |
| --- |
| 0,0,4  392,396  33,40,1,4  393,39633,4  0,2,4  394,39633,4  0,3,5  395,39633,5  0,4,2  396,39633,2 |

* + 1. userReflectionTable.txt (39633)

|  |
| --- |
| 28664 janisbanis  10902 gasLosAngeles  3323 Bridgetpianos  27818 California2  36717 BigPod  37627 roomfinder  30793 California3 28638 MaryStarbucks |

* + 1. hotelReflectionTable.txt (392)

|  |
| --- |
| 50 614244  255 613399  216 81511  82 99392  308 81514  147 80788  290 124765  105 80786  251 100554 |

1. analysis/SpectralClustering.java
   1. 用途：用Java代码调用mahout进行谱聚类
   2. 步骤：
      1. 调用SpectralKMeansDriver，用Java代码传入参数
      2. 调用SequenceFile Reader 来读取mahoutResult/kmeans\_out/clusteredPoints，转化后，将text文件内容写入clustering/clusteredPoints

0 <===> wt: 1.0 distance: 0.0 vec: 1 = [0.917, -0.000, 0.001, -0.001, 0.000, -0.004, 0.004, -0.001, 0.000, -0.007, -0.000, -0.000, -0.000, -0.005, -0.018, -0.000, -0.000, 0.007, 0.000, 0.397]

16 <===> wt: 1.0 distance: 0.0 vec: 2 = [-0.130, 0.000, 0.168, -0.028, 0.000, -0.048, 0.100, 0.010, 0.001, -0.627, 0.001, 0.064, -0.002, 0.006, -0.292, -0.000, -0.678, -0.007, -0.000, 0.028]

14 <===> wt: 1.0 distance: 0.20360232657238586 vec: 3 = [-0.032, 0.000, 0.541, -0.008, -0.000, -0.010, 0.052, 0.297, -0.000, -0.066, -0.001, -0.014, 0.011, -0.009, 0.753, -0.000, 0.114, -0.167, -0.001, 0.050]

9 <===> wt: 1.0 distance: 0.8096701427068773 vec: 4 = [-0.009, -0.000, 0.144, -0.009, -0.000, -0.000, 0.014, -0.008, 0.000, 0.525, -0.000, -0.002, -0.000, -0.007, -0.702, -0.001, -0.000, -0.459, 0.000, -0.001]

14 <===> wt: 1.0 distance: 0.8144093062895443 vec: 5 = [-0.269, -0.000, -0.003, -0.002, -0.000, -0.404, 0.005, -0.074, 0.000, -0.002, 0.000, -0.165, -0.004, -0.002, -0.007, 0.002, 0.598, 0.008, 0.000, 0.611]

12 <===> wt: 1.0 distance: 0.9304969142786569 vec: 6 = [0.023, 0.052, -0.113, 0.646, 0.051, 0.054, 0.014, 0.082, 0.119, 0.386, -0.030, -0.154, -0.248, -0.021, -0.028, 0.361, 0.082, 0.164, 0.356, -0.117]

2 <===> wt: 1.0 distance: 0.0 vec: 7 = [0.000, 0.000, -0.740, -0.007, 0.000, -0.000, 0.007, -0.005, -0.000, 0.298, 0.000, -0.001, 0.000, 0.001, 0.437, 0.000, -0.000, -0.415, -0.000, 0.000]

12 <===> wt: 1.0 distance: 0.12762090706266416 vec: 8 = [0.042, 0.049, -0.110, 0.096, 0.000, 0.059, 0.392, 0.263, -0.161, 0.063, 0.105, -0.021, -0.551, -0.506, -0.058, 0.340, 0.073, 0.018, -0.140, -0.060]

* + 1. 抽取行号为id，key为类别，写入clusteringResult文件中
  1. 输入：聚类参数
  2. 输出：clustering/clusteringResult

1:0

2:16

3:14

4:9

5:14

6:12

7:2

1. 命令行调用mahout谱聚类
   1. 用途：用命令行调用mahout进行谱聚类
   2. 步骤：
      1. 上传squareMatrix到HDFS

|  |
| --- |
| hadoop fs -put /home/claire/IdeaProjects/GroupRec/DataResource/squareMatrix /claire/mahout\_input |

* + 1. 对hdfs上的mahout\_input文件进行谱聚类，保留控制台的输出

|  |
| --- |
| mahout spectralkmeans -i hdfs://localhost:9000/claire/mahout\_input/squareMatrix -o /claire/mahout\_input/clusterResultTest1000\_20 -d 40025 -k 20 -x 100 >> /tmp/thousand20.txt 2>&1 & |

* 1. 输入： squareMatrix.txt
  2. 输出：consoleOutputFromMahout.txt
  3. 输出示例：

|  |
| --- |
| 15/12/09 10:28:40 INFO SpectralKMeansDriver: 38833: 18  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38834: 18  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38835: 18  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38836: 18  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38837: 14  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38838: 14  15/12/09 10:28:40 INFO SpectralKMeansDriver: 38839: 14 |

# 距离计算

1. Preprocessing/DurationComputing
   1. 用途：利用GoogleMap API进行两点之间到达时间计算
   2. 步骤：
      1. 创建Person对象，填入其经纬度
      2. 设置destination经纬度
      3. 创建Group对象，加入Person的list与destination
      4. GoogleMap Call得到每个Person需要的时长
   3. 输入：List<Person> + Destination
   4. 输出：durationList

# gMST

该模块需要构建表示用户与宾馆关系的初始二部图G，利用协同聚类结果补充G，生成G’，生成相应的导出图Q，在Q上利用最小生成树Prim算法得到路径，最后映射回原图G找出推荐的点。

涉及到的数据文件有：

类说明如下：

1. gmst/groupMake
   1. 用途：选取用户，创建一个小组，并形成该小组的初始二部图G
   2. 步骤：
      1. 创建一些Person 实例，形成一个Group
      2. 找到所有与该组用户相关联的宾馆
      3. 用邻接矩阵的形式表示出G：

E(user,item) = R(user, item) = overall\_rating || null

1. gmst/missingValueSupplement
   1. 用途：利用co-clustering的结果补充G的数据，形成G‘
   2. 步骤：
      1. 找到Group中每个Person所在的Class
      2. Pred(user, item) = avg (user所在Class中所有user对该item的平均值, item 所在class中所有item被该user的评分的平均值) || 0

避免用户习惯性打低分的问题

1. Gmst/weightCal
   1. 用途：计算G’中点的权重：
   2. 步骤：
      1. 计算preference部分
      2. 计算distance部分
      3. 计算权重
2. gmst/inducedGraphMake
   1. 用途：用G’生成导出图Q，其中Q 的点为所有的用户，所有能够通过一个宾馆相连的两个用户之间存在一条边，权值为两个用户相连的权值之和的最小值
3. gmst/MSTMake
   1. 用途：将Q 传入Prim算法中，生成最小生成树T
4. gmst/reflectBack
   1. 用途：分析T，结合G’，找到目标宾馆
5. Gmst/pickItem
   1. 用途：对可能找到的多个目标item，利用排序规则，挑选出最合适的一个