# 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: originalMatrix.txt
* Parsed matrix: parsedMatrix.txt
* squareMatrix:squareMatrix.txt
* clusteringResult: 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. 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：clusteringResult.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

1. Util



1. gmst/GroupMake
   1. 用途：创建原始图
   2. 步骤：
      1. initializeWeightsByRating(): 读入评分文件，构建weight矩阵[user][item]
      2. missingValueSupplement(): 利用co-clustering的结果补充缺失值
         1. 找到属于同一类别的users和items，分别放入classUsers和classItems
         2. 对小组中的每个用户，用多线程计算其与原始图item节点的缺失值getMissWeight()，将原始图补充成完全图
         3. 将新的三元组<userId, itemId, rating>写入ratingModel中
      3. makeGraph(): 创建itemNode和Edge，并给出最终的权重。
         1. 创建iteNode和Edge
         2. initializeWeightsByDistance(): 用多线程计算两点之间需要的时间，并填入edge的duration域里
         3. normalize(): 对评分和时间进行归一化，进行线性融合最终的权重。
   3. 输入：userItemRatingPath， clusteringResult
   4. 输出：ratingModel，graph(内存中)
2. gmst/ InducedGraph
   1. 用途：创建导出图
   2. 步骤：
      1. giveUserGraphId():给userNode设置GraphID
      2. getMinDistanceOfUsers():计算两个userNode之间的跳数
      3. generateIBList():产生导出图G’
      4. generateMatrix():用矩阵表示G‘
   3. 输入：graph
   4. 输出：gMatrix
3. gmst/ prim
   1. 用途：利用Prim算法计算最小生成树
   2. 步骤：
      1. 构造一个零图，选择一个初始点加入到集合中
      2. 分别从原来顶点的集合中抽取一个顶点，选择的标准是构造成的树的权值最小
      3. 循序渐进最终生成一棵最小生成树
4. gmst/ graphRec
   1. 用途：进行组推荐
   2. 步骤：
      1. 产生原始图
      2. 产生导出图
      3. 调用Prim算法，生成最小生成树的路径，用ArrayList<MSTPair>表示
      4. 将路径映射回原始图，生成推荐结果
   3. 输入：group
   4. 输出：resultList