Hadoop应用开发实战案例 第6周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

课程大纲

- 项目背景：职位推荐系统介绍
- 需求分析：KPI指标设计
- 算法模型：基于Mahout推荐算法
- 架构设计：职位推荐引擎架构
- 程序开发：基于Mahout的推荐算法实现
- 补充资料

项目背景: 职位推荐系统介绍

- 随着大数据思想实施的落地，推荐系统也开始倍受关注。不光是电商，各种互联网应用都开始应用推荐系统，像搜索，社交网络，音乐，餐饮，地图服务等等。

- 有一类的垂直社交网站，以“职业”社交定位，通过维护用户的职业社交圈，提升用户自身的价值，从而帮助用户找个更适合的工作。主要的用户分别3类，普通用户，猎头，企业HR。

- 目前，最大的职业社交网站是www.linkedin.com。

项目背景: 推荐引擎与规则引擎

- 在以前，我们没有使用推荐算法的时候，我们是通过设置各种约束条件，匹配数据的自然属性呈现给用户，这种就是基于规则的系统。
- 比如，用户购买了一个商品，我们会推荐同类别的其他商品，通过类别属性作为推荐的规则。后来问题就出现了，当用户一次性买了多种类别的不同商品的时候，前一条规则就失败了，我们要进一步设计规则，IT类别优先推荐，价格高的产品优先推荐.....几个回合下来，我们要不停的增加规则，以至于规则有可能的会前后冲突，增加一条新的规则会让推荐结果越来越不好，而且还无法解释是为什么。
- 针对于规则引擎的各种问题，推荐引擎出现了。推荐引擎是基于用户的访问行为的算法，不依赖于用户属性，如果有用户属性可以增加算法的维度。

项目背景: 推荐引擎Mahout

- Mahout是Hadoop的子项目。
- Mahout框架包含了一套完整的推荐系统引擎，标准化的数据结构，多样的算法实现，简单的开发流程。Mahout推荐的推荐系统引擎是模块化的，分为5个主要部分组成：数据模型，相似度算法，近邻算法，推荐算法，算法评分器。

需求分析：案例介绍

- 互联网某职业社交网站，主要产品包括 个人简历展示页，人脉圈，微博及分享链接，职位发布，职位申请，教育培训等。
- 用户在完成注册后，需要完善自己的个人信息，包括教育背景，工作经历，项目经历，技能专长等等信息。然后，你要告诉网站，你是否想找工作！！当你选择“是”（求职中），网站会从数据库中为你推荐你可能感兴趣的职位。

- 通过简短的描述，我们可以粗略地看出，这家职业社交网站的定位和主营业务。
- 核心点有2个：
  - 用户：尽可能多的保存有效完整的用户资料
  - 服务：帮助用户找到工作，帮助猎头和企业找到员工

- 因此，**职位推荐引擎** 将成为这个网站的核心功能。

需求分析： KPI指标设计

- 通过推荐带来的职位浏览量: 用户的PV
- 通过推荐带来的职位申请量: 用户的转化

需求分析： 功能需求

为了完成KPI的指标，我们把问题转化为“功能需求”：

- 1. 组合使用推荐算法，选出“评估推荐器”验证得分较高的算法
- 2. 人工验证推荐结果
- 3. 职位有时效性，推荐的结果应该是发布2013年内的发布职位
- 4. 工资的标准，应不低于用户浏览职位工资的平均值的80%

算法模型： 测试数据集

- pv.csv: 职位被浏览的信息
  - 2列数据：用户ID，职位ID(userid,jobid)
  - 记录数:2500条
  - 用户数:1000个，用户ID:1-1000
  - 职位数:200个，职位ID：1-200

- job.csv: 职位基本信息
  - 3列数据：职业ID，发布时间，工资标准(jobid,create_date,salary)
  - 职位数:200个，职位ID：1-200

算法模型： 数据样例

- pv.csv

```
1, 11
2, 136
2, 187
3, 165
3, 1
3, 24
4, 8
4, 199
5, 32
5, 100
6, 14
7, 59
7, 147
8, 92
9, 165
9, 80
9, 171
10, 45
10, 31
10, 1
10, 152
```

- job.csv

```
1, 2013-01-24, 5600
2, 2011-03-02, 5400
3, 2011-03-14, 8100
4, 2012-10-05, 2200
5, 2011-09-03, 14100
6, 2011-03-05, 6500
7, 2012-06-06, 37000
8, 2013-02-18, 5500
9, 2010-07-05, 7500
10, 2010-01-23, 6700
11, 2011-09-19, 5200
12, 2010-01-19, 29700
13, 2013-09-28, 6000
14, 2013-10-23, 3300
15, 2010-10-09, 2700
16, 2010-07-14, 5100
17, 2010-05-13, 29000
18, 2010-01-16, 21800
19, 2013-05-23, 5700
20, 2011-04-24, 5900
```

算法模型： 算法组合

- 我们选择UserCF,ItemCF,SlopeOne的 3种推荐算法，进行7种组合的测试。

- userCF1: LogLikelihoodSimilarity + NearestNUserNeighborhood + GenericBooleanPrefUserBasedRecommender
- userCF2: CityBlockSimilarity+ NearestNUserNeighborhood + GenericBooleanPrefUserBasedRecommender
- userCF3: CityBlockSimilarity+ NearestNUserNeighborhood + GenericBooleanPrefUserBasedRecommender
- itemCF1: LogLikelihoodSimilarity + GenericBooleanPrefItemBasedRecommender
- itemCF2: CityBlockSimilarity+ GenericBooleanPrefItemBasedRecommender
- itemCF3: CityBlockSimilarity+ GenericBooleanPrefItemBasedRecommender
- slopeOne：SlopeOneRecommender

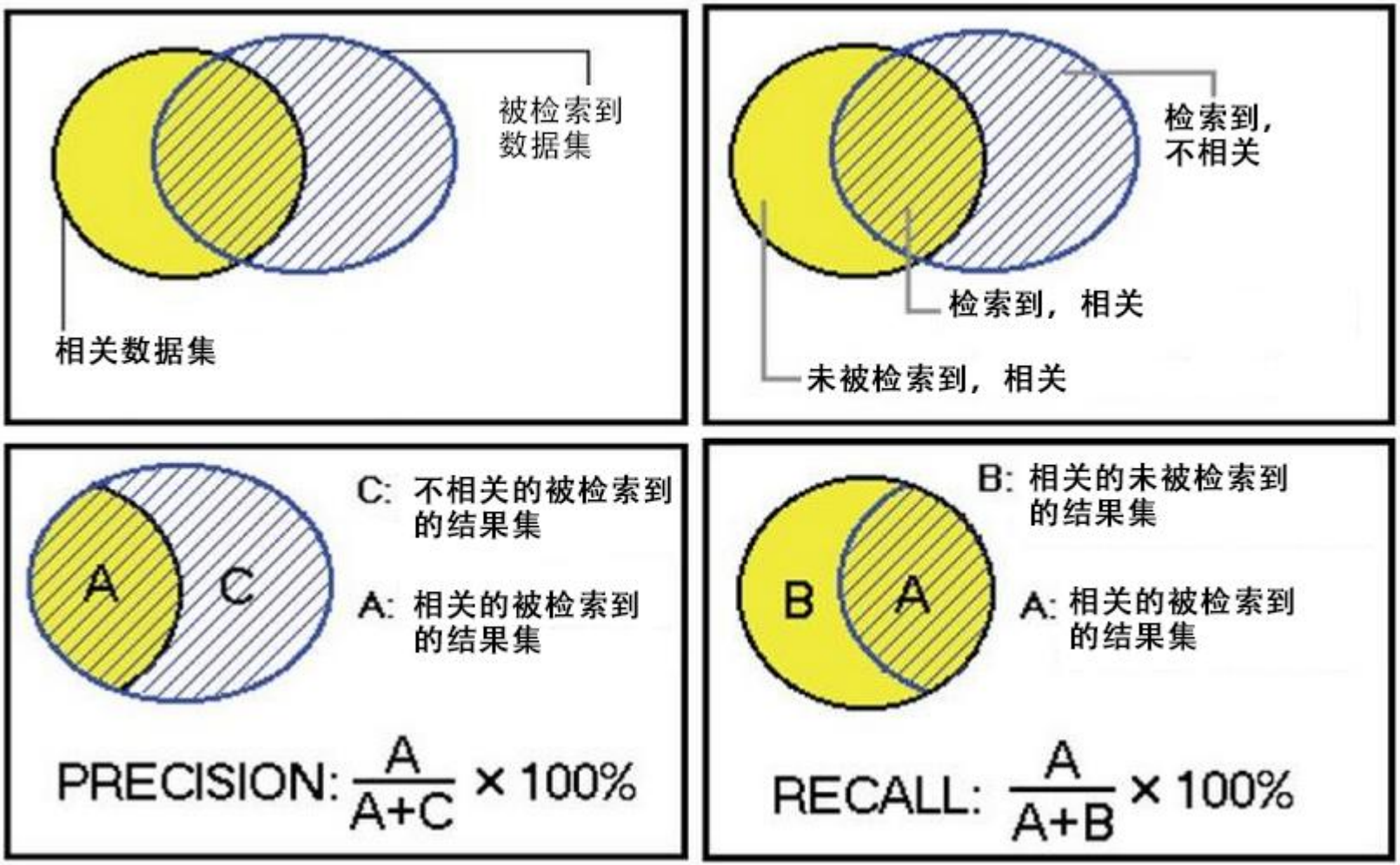
算法模型: 算法评估1

- Mahout提供了2个评估推荐器的指标，查准率和召回率（查全率），这两个指标是搜索引擎中经典的度量方法。

	相关	不相关
检索到	A	C
未检索到	B	D

- A：检索到的，相关的（搜到的也想要的）
- B：未检索到的，但是相关的（没搜到，然而实际上想要的）
- C：检索到的，但是不相关的（搜到的但没用的）
- D：未检索到的，也不相关的（没搜到也没用的）

算法模型: 算法评估2


被检索到
数据集
相关数据集
检索到，
不相关
检索到，相关
未被检索到，相关
A
C
C: 不相关的被检索到
的结果集
A: 相关的被检索到
的结果集
PRECISION: A/(A+C) × 100%
B
A
B: 相关的未被检索到
的结果集
A: 相关的被检索到
的结果集
RECALL: A/(A+B) × 100%

算法模型: 算法评估3

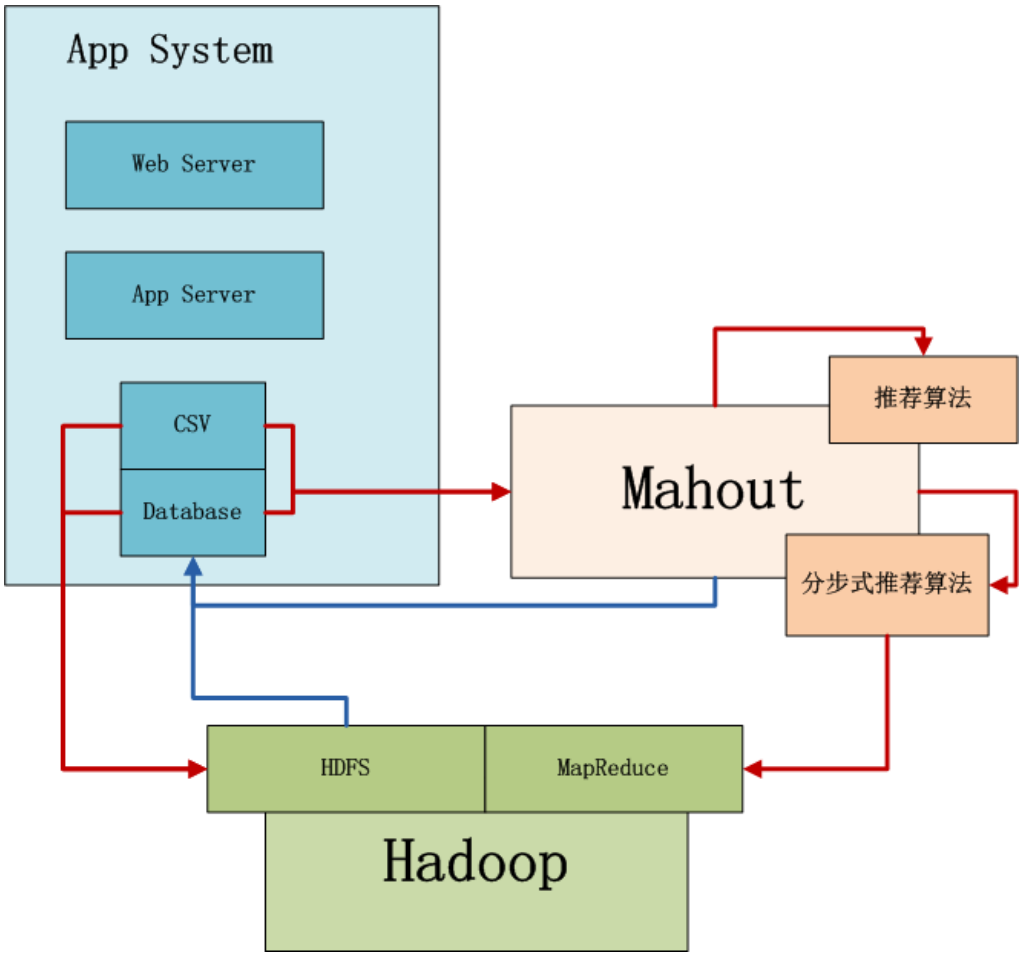
- 被检索到的越多越好，这是追求“查全率”，即A/(A+B)，越大越好。
  被检索到的，越相关的越多越好，不相关的越少越好，这是追求“查准率”，即A/(A+C)，越大越好。

- 在大规模数据集合中，这两个指标是相互制约的。当希望索引出更多的数据的时候，查准率就会下降，当希望索引更准确的时候，会索引更少的数据。

算法模型： 算法组合解释

- 关于的推荐算法的详细介绍，请参考文章：Mahout推荐算法API详解
- 关于算法的组合的详细介绍，请参考文章：从源代码剖析Mahout推荐引擎

架构设计：职位推荐引擎系统架构



架构设计：架构说明

上图中，左边是Application业务系统，右边是Mahout，下边是Hadoop集群。

- 1. 当数据量不太大，算法复杂时：直接选择用Mahout读取CSV或者Database数据，在单机内存中进行计算。Mahout是多线程的应用，会并行使用单机所有系统资源。

- 2. 当数据量很大时，并行化算法(ItemCF)：先把业务系统的数据导入到Hadoop的HDFS中，然后用Mahout访问HDFS实现并行算法。算法的性能与整个Hadoop集群有关。

- 3. 计算后的结果，保存到数据库中，方便以后查询！

程序开发: 用Maven构建Mahout项目

- 开发环境
  - Win7 64bit
  - Java 1.6.0_45
  - Maven3
  - Eclipse Juno Service Release 2
  - Mahout-0.8
  - Hadoop-1.1.2

- 请参考文章：用Maven构建Mahout项目

程序开发: 新建Java类

- RecommendFactory，构造数据结构
- RecommenderEvaluator.java, 选出“评估推荐器”验证得分较高的算法
- RecommenderResult.java, 对指定数量的结果人工比较
- RecommenderFilterOutdateResult.java，排除过期职位
- RecommenderFilterSalaryResult.java，排除工资过低的职位

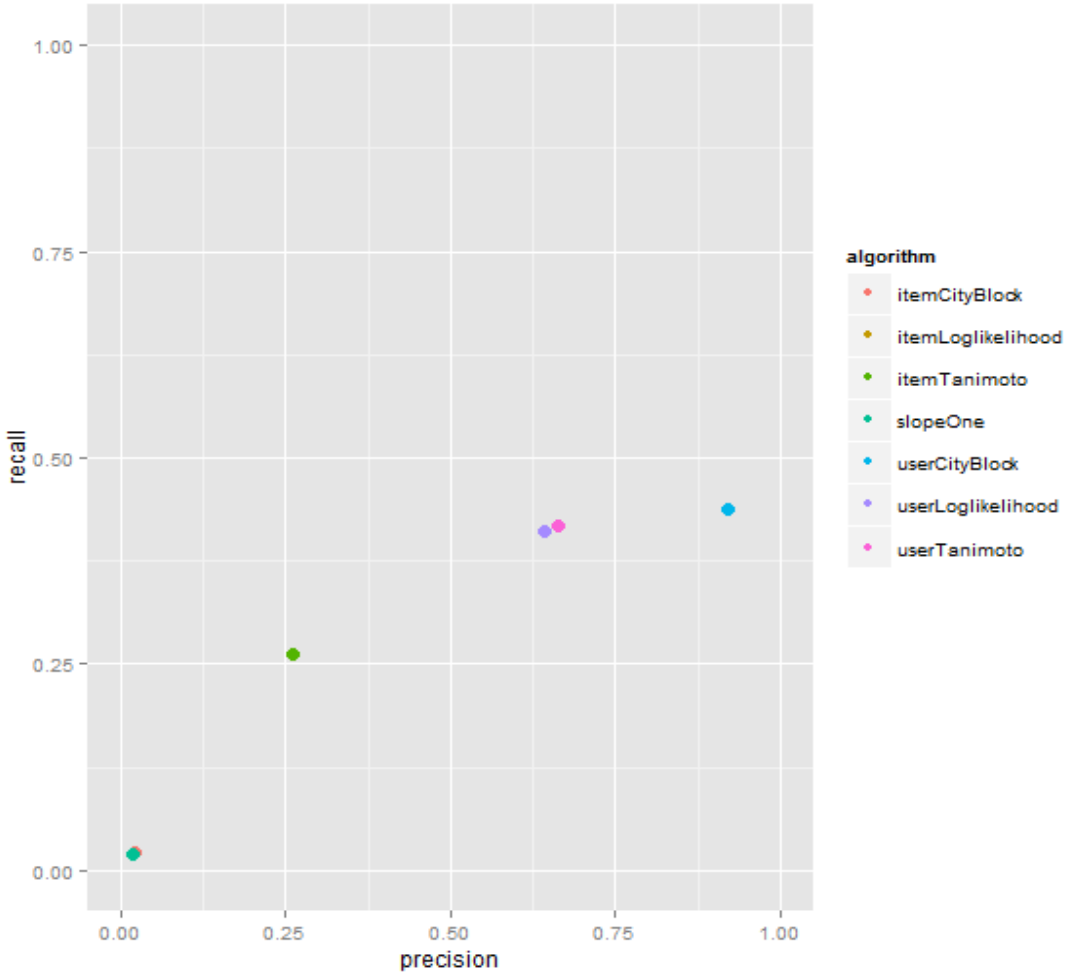
程序开发: RecommenderEvaluator.java

```
public class RecommenderEvaluator {

    final static int NEIGHBORHOOD_NUM = 2;
    final static int RECOMMENDER_NUM = 3;

    public static void main(String[] args) throws T
        String file = "datafile/job/pv.csv";
        DataModel dataModel = RecommendFactory.buil
        userLoglikelihood(dataModel);
        userCityBlock(dataModel);
        userTanimoto(dataModel);
        itemLoglikelihood(dataModel);
        itemCityBlock(dataModel);
        itemTanimoto(dataModel);
        slopeOne(dataModel);
    }

    public static RecommenderBuilder userLoglikeli
        System.out.println("userLoglikelihood");
        UserSimilarity userSimilarity = RecommendF
        UserNeighborhood userNeighborhood = Recomm
        RecommenderBuilder recommenderBuilder = Re

        RecommendFactory.evaluate(RecommendFactory.EVALUATOR.AVERAGE_ABSOLUTE_DIFFERENCE, re
        RecommendFactory.statsEvaluator(recommenderBuilder, null, dataModel, 2);
        return recommenderBuilder;
    }

    public static RecommenderBuilder userCityBlock(DataModel dataModel) throws TasteExceptio
        System.out.println("userCityBlock");
        UserSimilarity userSimilarity = RecommendFactory.userSimilarity(RecommendFactory.SIM
        UserNeighborhood userNeighborhood = RecommendFactory.userNeighborhood(RecommendFacto
        RecommenderBuilder recommenderBuilder = RecommendFactory.userRecommender(userSimilar

        RecommendFactory.evaluate(RecommendFactory.EVALUATOR.AVERAGE_ABSOLUTE_DIFFERENCE, re
        RecommendFactory.statsEvaluator(recommenderBuilder, null, dataModel, 2);
        return recommenderBuilder;
    }

    public static RecommenderBuilder userTanimoto(DataModel dataModel) throws TasteExceptio
        System.out.println("userTanimoto");
        UserSimilarity userSimilarity = RecommendFactory.userSimilarity(RecommendFactory.SIM
        UserNeighborhood userNeighborhood = RecommendFactory.userNeighborhood(RecommendFacto
        RecommenderBuilder recommenderBuilder = RecommendFactory.userRecommender(userSimilar

        RecommendFactory.evaluate(RecommendFactory.EVALUATOR.AVERAGE_ABSOLUTE_DIFFERENCE, re
        RecommendFactory.statsEvaluator(recommenderBuilder, null, dataModel, 2);
        return recommenderBuilder;
    }
```

程序开发:控制台输出

```
userLoglikelihood
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.2741487771272658
Recommender IR Evaluator: [Precision:0.6424242424242422,Recall:0.4098360655737705]
userCityBlock
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.575306732961736
Recommender IR Evaluator: [Precision:0.919580419580419,Recall:0.4371584699453552]
userTanimoto
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.5546485136181523
Recommender IR Evaluator: [Precision:0.6625766871165644,Recall:0.41803278688524603]
itemLoglikelihood
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.5398332608612343
Recommender IR Evaluator: [Precision:0.26229508196721296,Recall:0.26229508196721296]
itemCityBlock
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.9251437840891661
Recommender IR Evaluator: [Precision:0.02185792349726776,Recall:0.02185792349726776]
itemTanimoto
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.9176432856689655
Recommender IR Evaluator: [Precision:0.26229508196721296,Recall:0.26229508196721296]
slopeOne
AVERAGE_ABSOLUTE_DIFFERENCE Evaluater Score:0.0
Recommender IR Evaluator: [Precision:0.01912568306010929,Recall:0.01912568306010929]
```

程序开发: 可视化输出 – 算法验证器



- Recall和Precision，都是越接近1越好。
- UserCityBlock算法评估的结果是最好的，基于UserCF的算法比ItemCF都要好，SlopeOne算法几乎没有得分。

程序开发: RecommenderResult.java

- 为得到差异化结果，我们分别取UserCityBlock itemLoglikelihood，对推荐结果人工比较。

```
public class RecommenderResult {

    final static int NEIGHBORHOOD_NUM = 2;
    final static int RECOMMENDER_NUM = 3;

    public static void main(String[] args) throws TasteException, IOException {
        String file = "datafile/job/pv.csv";
        DataModel dataModel = RecommendFactory.buildDataModelNoPref(file);
        RecommenderBuilder rb1 = RecommenderEvaluator.userCityBlock(dataModel);
        RecommenderBuilder rb2 = RecommenderEvaluator.itemLoglikelihood(dataModel);

        LongPrimitiveIterator iter = dataModel.getUserIDs();
        while (iter.hasNext()) {
            long uid = iter.nextLong();
            System.out.print("userCityBlock    =>");
            result(uid, rb1, dataModel);
            System.out.print("itemLoglikelihood=>");
            result(uid, rb2, dataModel);
        }
    }

    public static void result(long uid, RecommenderBuilder recommenderBuilder, DataModel da
        List list = recommenderBuilder.buildRecommender(dataModel).recommend(uid, RECOMMENDI
        RecommendFactory.showItems(uid, list, false);
    }

}
```

程序开发: 控制台输出

- 控制台输出：只截取部分结果

```
...
userCityBlock    =>uid:968, (61, 0.333333)
itemLoglikelihood=>uid:968, (121, 1.429362)(153, 1.239939)(198, 1.207726)
userCityBlock    =>uid:969,
itemLoglikelihood=>uid:969, (75, 1.326499)(30, 0.873100)(85, 0.763344)
userCityBlock    =>uid:970,
itemLoglikelihood=>uid:970, (13, 0.748417)(156, 0.748417)(122, 0.748417)
userCityBlock    =>uid:971,
itemLoglikelihood=>uid:971, (38, 2.060951)(104, 1.951208)(83, 1.941735)
userCityBlock    =>uid:972,
itemLoglikelihood=>uid:972, (131, 1.378395)(4, 1.349386)(87, 0.881816)
userCityBlock    =>uid:973,
itemLoglikelihood=>uid:973, (196, 1.432040)(140, 1.398066)(130, 1.380335)
userCityBlock    =>uid:974, (19, 0.200000)
itemLoglikelihood=>uid:974, (145, 1.994049)(121, 1.794289)(98, 1.738027)
...
```

程序开发: 人工对数据进行分析

- 我们查看uid=974的用户推荐信息：
- 搜索pv.csv

```
> pv[which(pv$userid==974),]
     userid jobid
2426    974   106
2427    974   173
2428    974    82
2429    974   188
2430    974    78
```

- 搜索job.csv

```
> job[job$jobid %in% c(145,121,98,19),]
    jobid create_date salary
19     19  2013-05-23   5700
98     98  2010-01-15   2900
121   121  2010-06-19   5300
145   145  2013-08-02   6800
```

程序开发: RecommenderFilterOutdateResult.java

- 排除过期职位

```
public class RecommenderFilterOutdateResult {

    final static int NEIGHBORHOOD_NUM = 2;
    final static int RECOMMENDER_NUM = 3;

    public static void main(String[] args) throws TasteException, IOException {
        String file = "datafile/job/pv.csv";
        DataModel dataModel = RecommendFactory.buildDataModelNoPref(file);
        RecommenderBuilder rb1 = RecommenderEvaluator.userCityBlock(dataModel);
        RecommenderBuilder rb2 = RecommenderEvaluator.itemLoglikelihood(dataModel);

        LongPrimitiveIterator iter = dataModel.getUserIDs();
        while (iter.hasNext()) {
            long uid = iter.nextLong();
            System.out.print("userCityBlock    =>");
            filterOutdate(uid, rb1, dataModel);
            System.out.print("itemLoglikelihood=>");
            filterOutdate(uid, rb2, dataModel);
        }
    }

    public static void filterOutdate(long uid, RecommenderBuilder recommenderBuilder, DataM
        Set jobids = getOutdateJobID("datafile/job/job.csv");
        IDRescorer rescorer = new JobRescorer(jobids);
        List list = recommenderBuilder.buildRecommender(dataModel).recommend(uid, RECOMMENDE
        RecommendFactory.showItems(uid, list, true);
    }
```

程序开发:控制台输出

- 控制台输出：只截取部分结果

```
...
itemLoglikelihood=>uid:965, (200,0.829600)(122,0.748417)(170,0.736340)
userCityBlock    =>uid:966, (114,0.250000)
itemLoglikelihood=>uid:966, (114,1.516898)(101,0.864536)(99,0.858057)
userCityBlock    =>uid:967,
itemLoglikelihood=>uid:967, (105,0.873100)(114,0.725016)(168,0.707119)
userCityBlock    =>uid:968,
itemLoglikelihood=>uid:968, (174,0.735004)(39,0.696716)(185,0.696171)
userCityBlock    =>uid:969,
itemLoglikelihood=>uid:969, (197,0.723203)(81,0.710230)(167,0.668358)
userCityBlock    =>uid:970,
itemLoglikelihood=>uid:970, (13,0.748417)(122,0.748417)(28,0.736340)
userCityBlock    =>uid:971,
itemLoglikelihood=>uid:971, (28,1.540753)(174,1.511881)(39,1.435575)
userCityBlock    =>uid:972,
itemLoglikelihood=>uid:972, (14,0.800605)(60,0.794088)(163,0.710230)
userCityBlock    =>uid:973,
itemLoglikelihood=>uid:973, (56,0.795529)(13,0.712680)(120,0.701026)
userCityBlock    =>uid:974, (19,0.200000)
itemLoglikelihood=>uid:974, (145,1.994049)(89,1.578694)(19,1.435193)
...
```

程序开发: 人工对数据进行分析

- 我们查看uid=974的用户推荐信息：
- 搜索pv.csv

```
> pv[which(pv$userid==974),]
     userid jobid
2426    974   106
2427    974   173
2428    974    82
2429    974   188
2430    974    78
```

- 搜索job.csv

```
> job[job$jobid %in% c(19,145,89),]
    jobid create_date salary
19     19  2013-05-23   5700
89     89  2013-06-15   8400
145   145  2013-08-02   6800
```

程序开发: 排除过期的职位比较

排除过期的职位比较

- userCityBlock结果都是19，
- itemLoglikelihood的第2，3的结果被替换为了得分更低的89和19。

```
userCityBlock    =>uid:974, (19,0.200000)
itemLoglikelihood=>uid:974, (145,1.994049)(121,1.794289)(98,1.738027)
```

```
> job[job$jobid %in% c(145,121,98,19),]
    jobid create_date salary
19     19  2013-05-23   5700
98     98  2010-01-15   2900
121   121  2010-06-19   5300
145   145  2013-08-02   6800
```

```
userCityBlock    =>uid:974, (19,0.200000)
itemLoglikelihood=>uid:974, (145,1.994049)(89,1.578694)(19,1.435193)
```

```
> job[job$jobid %in% c(19,145,89),]
    jobid create_date salary
19     19  2013-05-23   5700
89     89  2013-06-15   8400
145   145  2013-08-02   6800
```

程序开发: RecommenderFilterSalaryResult.java

- 我们查看uid=974的用户，浏览过的职位。

```
> job[job$jobid %in% c(106,173,82,188,78),]
    jobid create_date salary
78     78  2012-01-29   6800
82     82  2010-07-05   7500
106   106  2011-04-25   5200
173   173  2013-09-13   5200
188   188  2010-07-14   6000
```

- 平均工资为6140，我们觉得用户的浏览职位的行为，一般不会看比自己现在工资低的职位。因此设计算法，排除工资低于平均工资80%的职位，即排除工资小于4912的推荐职位(6140*0.8=4912)

- 请大家自己尝试解决这个问题！

程序开发: 补充资料

- 程序源代码下载：
- https://github.com/bsspirit/maven_mahout_template/releases/tag/mahout_recommend_v1

- 补充资料：
- http://blog.fens.me/hadoop-mahout-maven-eclipse/
- http://blog.fens.me/mahout-recommendation-api/
- http://blog.fens.me/hadoop-mahout-mapreduce-itemcf/
- http://blog.fens.me/mahout-recommend-engine/
- http://blog.fens.me/hadoop-mahout-recommend-job/

关于作者

- 张丹, 编程爱好者(Java,R,PHP,Javascript)
- DataguruID: bsspirit
- Weibo: @Conan_Z
- Blog：http://blog.fens.me
- Email: bsspirit@gmail.com

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间