
# Hadoop应用开发实战案例 第5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：推荐系统概述
- 需求分析：推荐系统指标设计
- 算法模型：Hadoop并行算法
- 架构设计：推荐系统架构
- 程序开发：MapReduce程序实现

# 广告的发展趋势

- 传统：纸媒体广告，电视广告，电台广告，户外广告，电梯广告，车身广告，短信广告
- 互联网：旗帜广告，EDM，弹窗
- 精准广告（精准投放，精准测量效果）日益成为主流，滥发的广告方式将被抛弃
- 诞生一门新的研究学科《计算广告学》
- 斯坦福课程：http://www.stanford.edu/class/msande239/

# 项目背景：推荐系统概述

- 电子商务网站是个性化推荐系统重要地应用的领域之一，亚马逊就是个性化推荐系统的积极应用者和推广者，亚马逊的推荐系统深入到网站的各类商品，为亚马逊带来了至少30%的销售额。

- 不光是电商类，推荐系统无处不在。QQ，人人网的好友推荐；新浪微博的你可能感兴趣的人；优酷，土豆的电影推荐；豆瓣的图书推荐；大从点评的餐饮推荐；世纪佳缘的相亲推荐；天际网的职业推荐等。

# 项目背景：推荐算法分类

- 按数据使用划分：
  - 协同过滤算法：UserCF, ItemCF, ModelCF
  - 基于内容的推荐: 用户内容属性和物品内容属性
  - 社会化过滤：基于用户的社会网络关系

- 按模型划分：
  - 最近邻模型:基于距离的协同过滤算法
  - Latent Factor Mode(SVD)：基于矩阵分解的模型
  - Graph：图模型，社会网络图模型

# 当当网上的推荐





张量分析[第二版] 70万种图书音像5折封顶！20万种科教类书6.9折封顶！




商品编号：8763182

抢购 ¥23.50 还剩 1 天 5 小时 35 分结束

当 当 价：¥28.00

定　　价：¥34.00　　折扣：6.9折

顾客评分：★★★★★　已有486人评论，98.4%推荐

配 送 至：广东广州市海珠区　有货

下周二(12月17日)可送达，请在16小时6分钟内下单并选择“普通快递送货上门”

运费说明 >>

作　　者：黄克智 等

出 版 社：清华大学出版社

出版时间：2003-7-1

版　　次：　　页　　数：　　字

印刷时间：　　开　　本：　　纸

# 汽车之家上的推荐

当前位置：首页 > 中型车 > 奥迪A4L > 综述

一汽-大众CC

## 一汽-大众奥迪-奥迪A4L

全部 | 停售

车系首页 | 参数配置 | 图片实拍 | 报价 | 口碑 | 车型详解 | 文章 | 维修保养



新车指导价：27.28-57.81万 (9款车型) 询价

二手车价格：19.00-46.00万 (133个车源)

车身颜色： 更多颜色

发 动 机：1.8T 2.0T 3.0T

变 速 箱：手动 无级 双离合　　车身结构：三厢

用户评分：4.55分(1175条口碑)

奥迪A4L口碑排行第2名　　详细排行

| 口碑排行 | 用户评分 |
|---|---|
| 1. 奥迪A6L | 4.55分 |
| 2. 奥迪A4L | 4.55分 |
| 3. 宝马3系 | 4.37分 |
| 4. 一汽-大众CC | 4.35分 |
| 5. 迈腾 | 4.35分 |

单项评分及排名

| | | |
|---|---|---|
| 空间： | 4.52分 | 第3名 |
| 动力： | 4.51分 | 第4名 |
| 操控： | 4.56分 | 第3名 |
| 油耗： | 4.40分 | 第1名 |
| 舒适性： | 4.45分 | 第2名 |
| 外观： | 4.84分 | 第2名 |
| 内饰： | 4.58分 | 第2名 |
| 性价比： | 4.50分 | 第1名 |

奥迪A4L论坛　　更多>>

- 再次发提车作业申请认证，不把妹子的黑丝　12月15日

最新文章 | 买车必看　　新闻 导购 评测

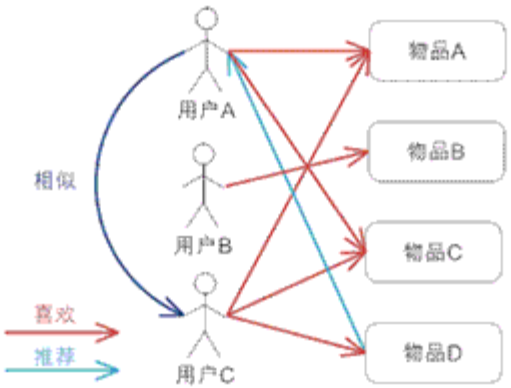
- 新闻 | CTS降10万 本周豪华品牌中型车降价排行　11月30日

# Dataguru的推荐

# 项目背景：基于用户的协同过滤算法UserCF

- 基于用户的协同过滤，通过不同用户对物品的评分来评测用户之间的相似性，基于用户之间的相似性做出推荐。
- 简单来讲就是：给用户推荐和他兴趣相似的其他用户喜欢的物品。

| 用户/物品 | 物品A | 物品B | 物品C | 物品D |
|---|---|---|---|---|
| 用户A | √ | | √ | 推荐 |
| 用户B | | √ | | |
| 用户C | √ | | √ | √ |



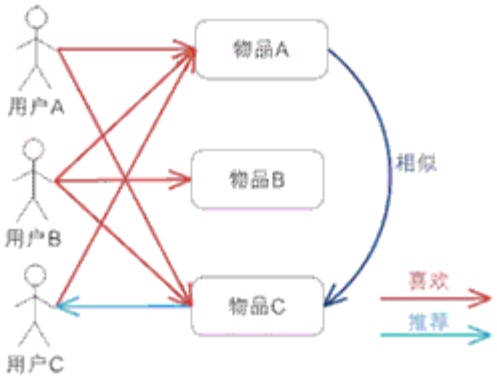
# 项目背景：基于用户的协同过滤算法UserCF

- 基于UserCF 的基本思想相当简单，基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。
- 计算上，就是将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，找到 K 邻居后，根据邻居的相似度权重以及他们对物品的偏好，预测当前用户没有偏好的未涉及物品，计算得到一个排序的物品列表作为推荐。
- 上图给出了一个例子，对于用户 A，根据用户的历史偏好，这里只计算得到一个邻居 – 用户 C，然后将用户 C 喜欢的物品 D 推荐给用户 A。

# 项目背景：基于物品的协同过滤算法ItemCF

- 基于item的协同过滤，通过用户对不同item的评分来评测item之间的相似性，基于item之间的相似性做出推荐。
- 简单来讲就是：给用户推荐和他之前喜欢的物品相似的物品。

| 用户/物品 | 物品A | 物品B | 物品C |
|---|---|---|---|
| 用户A | √ | | √ |
| 用户B | √ | √ | √ |
| 用户C | √ | | 推荐 |



# 项目背景：基于物品的协同过滤算法ItemCF

- 基于ItemCF 的原理和基于UserCF 类似，只是在计算邻居时采用物品本身，而不是从用户的角度，即基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。
- 从计算的角度看，就是将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度，得到物品的相似物品后，根据用户历史的偏好预测当前用户还没有表示偏好的物品，计算得到一个排序的物品列表作为推荐。
- 上图给出了一个例子，对于物品 A，根据所有用户的历史偏好，喜欢物品 A 的用户都喜欢物品 C，得出物品 A 和物品 C 比较相似，而用户 C 喜欢物品 A，那么可以推断出用户 C 可能也喜欢物品 C。

- 上文中图片和对应图片解释，摘自：
  https://www.ibm.com/developerworks/cn/web/1103_zhaoct_recommstudy2/

# 项目背景：基于物品的协同过滤算法实现

- **分为2个步骤**
- 1. 计算物品之间的相似度
- 2. 根据物品的相似度和用户的历史行为给用户生成推荐列表

- 注：基于物品的协同过滤算法，是目前商用最广泛的推荐算法。

# 需求分析：案例介绍

- 互联网某电影点评网站，主要产品包括 电影介绍，电影排行，网友对电影打分，网友影评，影讯&购票，用户在看|想看|看过的电影，猜你喜欢(推荐)。

- 用户在完成注册后，可以浏览网站的各种电影介绍，看电影排行榜，选择自己喜欢的分类，找到自己想看的电影，并设置为“想看”，同时对自己已经看过的电影写下影评，并打分。

# 需求分析：案例介绍

- 通过简短的描述，我们可以粗略地看出，这个网站提供个性化推荐电影服务：
- 核心点：
  - 网站提供所有电影信息，吸引用户浏览
  - 网站收集用户行为，包括浏览行为，评分行为，评论行为，从而推测出用户的爱好。
  - 网站帮助用户找到，用户还没有看过，并满足他兴趣的电影列表。
  - 网站通过海量数据的积累了，预测未来新片的市场影响和票房

- **电影推荐** 将成为这个网站的核心功能。

# 需求分析：推荐系统指标设计

在真实的环境中设计推荐的时候，要全面考量数据量，算法性能，结果准确度等的指标。

- 推荐算法选型：基于物品的协同过滤算法ItemCF，并行实现
- 数据量：基于Hadoop架构，支持GB,TB,PB级数据量
- 算法检验：可以通过 准确率，召回率，覆盖率，流行度 等指标评判。
- 结果解读：通过ItemCF的定义，合理给出结果解释

# 算法模型：测试数据集

- Mahout In Action书里，第一章第六节基于物品的协同过滤算法进行实现。
- 测试数据集:small.csv
- 每行3个字段，依次是用户ID,电影ID,用户对电影的评分(0-5分，每0.5分为一个评分点！)

```
1,101,5.0
1,102,3.0
1,103,2.5
2,101,2.0
2,102,2.5
2,103,5.0
2,104,2.0
3,101,2.0
3,104,4.0
3,105,4.5
3,107,5.0
4,101,5.0
4,103,3.0
4,104,4.5
4,106,4.0
5,101,4.0
5,102,3.0
5,103,2.0
5,104,4.0
5,105,3.5
5,106,4.0
```

- 注：数据集可以自己从互联网获得

# 算法模型: 算法的思想

- 1. 建立物品的同现矩阵
- 2. 建立用户对物品的评分矩阵
- 3. 矩阵计算推荐结果

# 算法模型: 1). 建立物品的同现矩阵

- 按用户分组，找到每个用户所选的物品，单独出现计数及两两一组计数。

| | [101] | [102] | [103] | [104] | [105] | [106] | [107] |
|---|---|---|---|---|---|---|---|
| [101] | 5 | 3 | 4 | 4 | 2 | 2 | 1 |
| [102] | 3 | 3 | 3 | 2 | 1 | 1 | 0 |
| [103] | 4 | 3 | 4 | 3 | 1 | 2 | 0 |
| [104] | 4 | 2 | 3 | 4 | 2 | 2 | 1 |
| [105] | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| [106] | 2 | 1 | 2 | 2 | 1 | 2 | 0 |
| [107] | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

# 算法模型: 2). 建立用户对物品的评分矩阵

- 按用户分组，找到每个用户所选的物品及评分

```
       V3
[101] 2.0
[102] 0.0
[103] 0.0
[104] 4.0
[105] 4.5
[106] 0.0
[107] 5.0
```

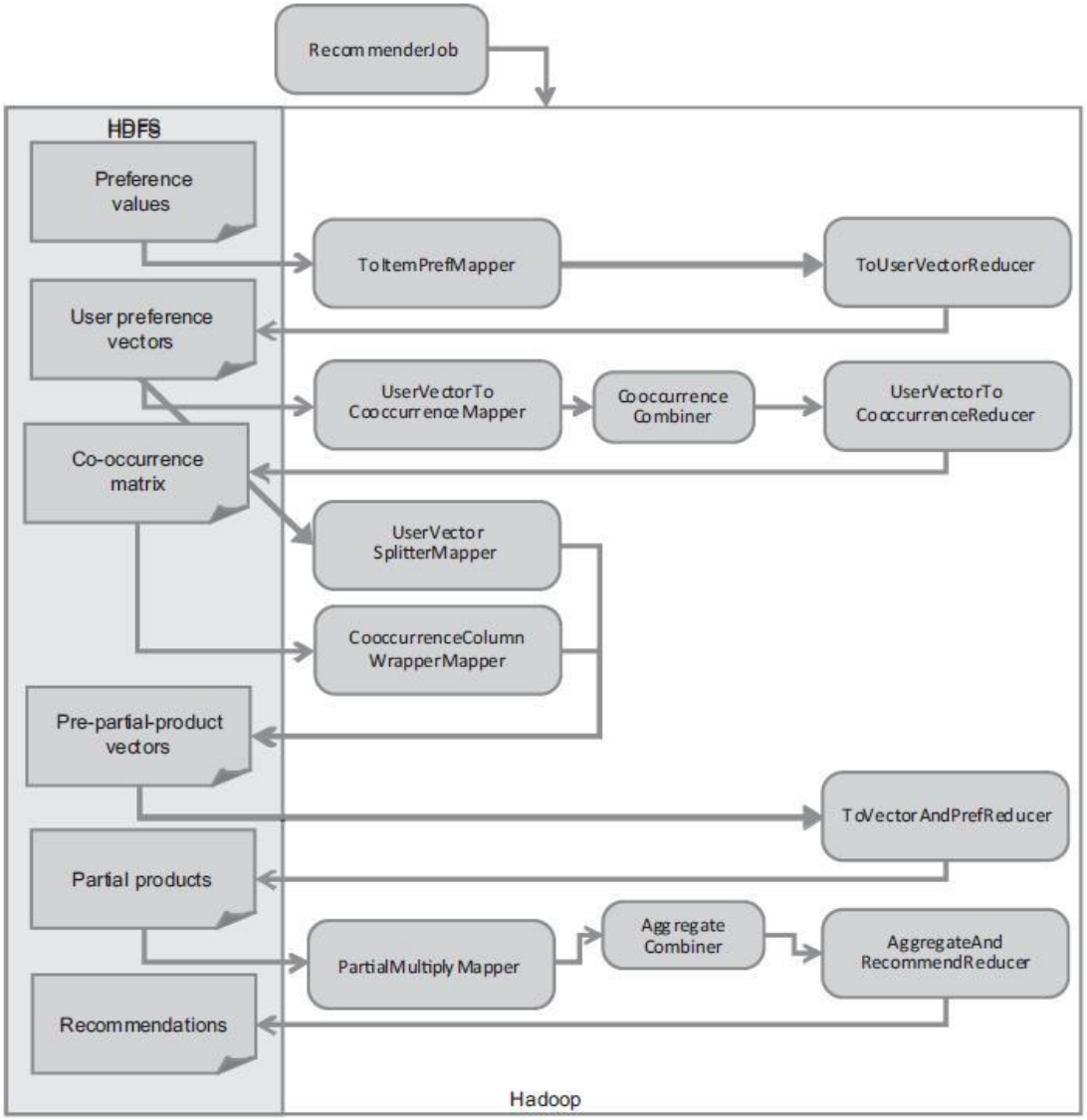
# 算法模型: 3). 矩阵计算推荐结果

- 同现矩阵*评分矩阵=推荐结果

| | 101 | 102 | 103 | 104 | 105 | 106 | 107 | | U3 | | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 5 | 3 | 4 | 4 | 2 | 2 | 1 | | 2.0 | | 40.0 |
| 102 | 3 | 3 | 3 | 2 | 1 | 1 | 0 | | 0.0 | | 18.5 |
| 103 | 4 | 3 | 4 | 3 | 1 | 2 | 0 | X | 0.0 | = | 24.5 |
| 104 | 4 | 2 | 3 | 4 | 2 | 2 | 1 | | 4.0 | | 40.0 |
| 105 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | | 4.5 | | 26.0 |
| 106 | 2 | 1 | 2 | 2 | 1 | 2 | 0 | | 0.0 | | 16.5 |
| 107 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | | 5.0 | | 15.5 |

图片摘自 Mahout In Action

# 算法模型: MapReduce任务设计

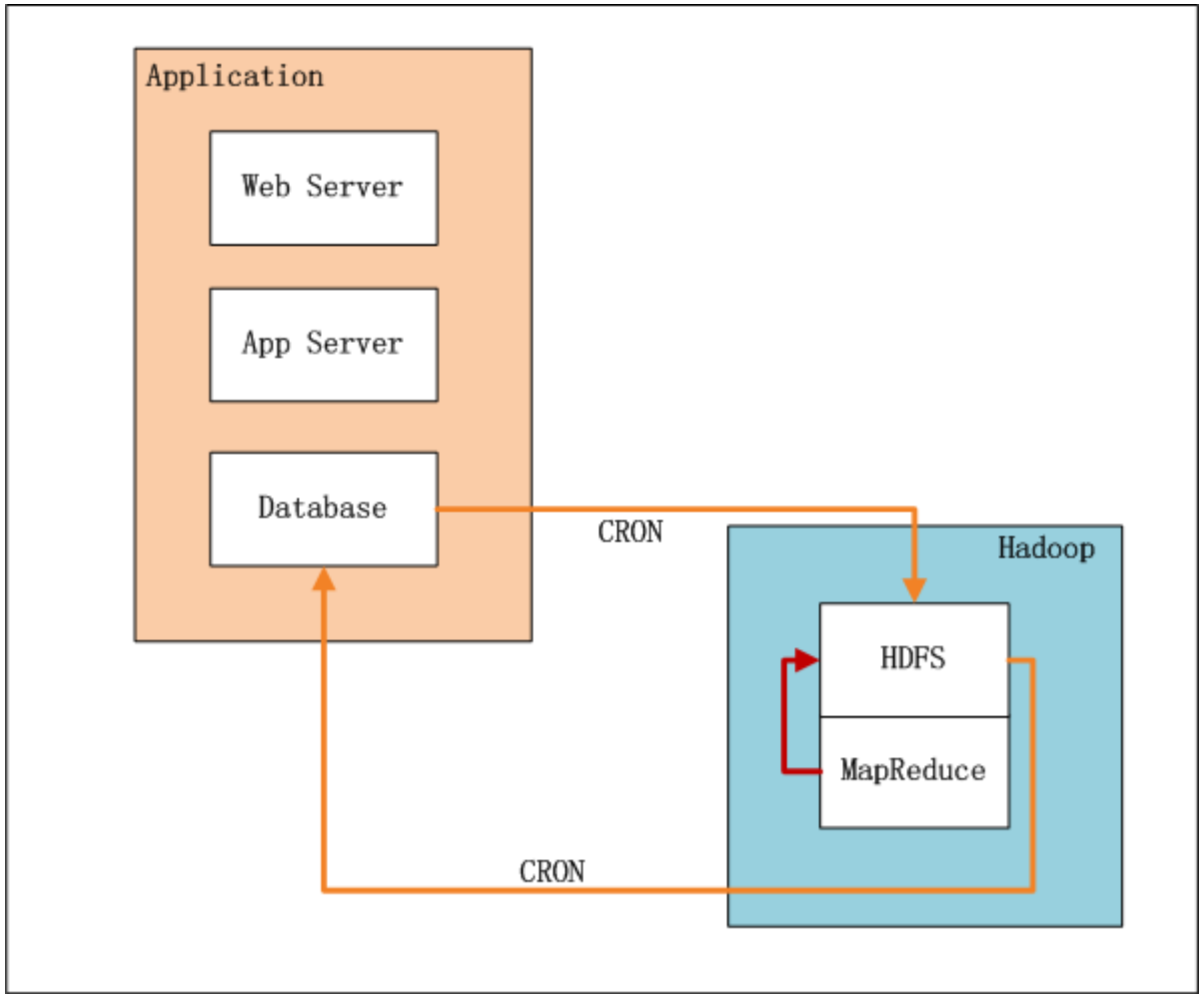



图片摘自 Mahout In Action

# 算法模型: 解读MapRduce任务

- 步骤1: 按用户分组，计算所有物品出现的组合列表，得到用户对物品的评分矩阵
- 步骤2: 对物品组合列表进行计数，建立物品的同现矩阵
- 步骤3: 合并同现矩阵和评分矩阵
- 步骤4: 计算推荐结果列表

# 架构设计：推荐系统架构


Application
Web Server
App Server
Database
CRON
Hadoop
HDFS
MapReduce
CRON

# 架构设计：推荐系统架构说明

- 上图中，左边是Application业务系统，右边是Hadoop的HDFS, MapReduce。
- 业务系统记录了用户的行为和对物品的打分
- 设置系统定时器CRON，每xx小时，增量向HDFS导入数据(userid,itemid,value,time)。
- 完成导入后，设置系统定时器，启动MapReduce程序，运行推荐算法。
- 完成计算后，设置系统定时器，从HDFS导出推荐结果数据到数据库，方便以后的及时查询。

# 程序开发：系统环境

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

- Hadoop集群系统环境：
  - Linux: Ubuntu 12.04.2 LTS 64bit Server
  - Java: 1.6.0_29
  - Hadoop: hadoop-1.0.3，单节点，IP:192.168.1.210

- 请参考文章：用Maven构建Hadoop项目

# 程序开发： MapReduce程序实现

新建Java类：

- Recommend.java，主任务启动程序
- Step1.java，按用户分组，计算所有物品出现的组合列表，得到用户对物品的评分矩阵
- Step2.java，对物品组合列表进行计数，建立物品的同现矩阵
- Step3.java，对同现矩阵和评分矩阵转型
- Step4.java，合并矩阵，并计算推荐结果列表
- HdfsDAO.java，HDFS操作工具类

# 程序开发： Recommend.java

```
public class Recommend {

    public static final String HDFS = "hdfs://192.168.1.210:9000";
    public static final Pattern DELIMITER = Pattern.compile("[\t,]");

    public static void main(String[] args) throws Exception {
        Map<String, String> path = new HashMap<String, String>();
        path.put("data", "logfile/small.csv");
        path.put("Step1Input", HDFS + "/user/hdfs/recommend");
        path.put("Step1Output", path.get("Step1Input") + "/step1");
        path.put("Step2Input", path.get("Step1Output"));
        path.put("Step2Output", path.get("Step1Input") + "/step2");
        path.put("Step3Input1", path.get("Step1Output"));
        path.put("Step3Output1", path.get("Step1Input") + "/step3_1");
        path.put("Step3Input2", path.get("Step2Output"));
        path.put("Step3Output2", path.get("Step1Input") + "/step3_2");
        path.put("Step4Input1", path.get("Step3Output1"));
        path.put("Step4Input2", path.get("Step3Output2"));
        path.put("Step4Output", path.get("Step1Input") + "/step4");

        Step1.run(path);
        Step2.run(path);
        Step3.run1(path);
        Step3.run2(path);
        Step4.run(path);
        System.exit(0);
```

# 程序开发： Step1.java，源代码

- 按用户分组，计算所有物品出现的组合列表，得到用户对物品的评分矩阵

```
public class Step1 {

    public static class Step1_ToItemPreMapper extends MapReduceBase implements Mapper {
        private final static IntWritable k = new IntWritable();
        private final static Text v = new Text();

        @Override
        public void map(Object key, Text value, OutputCollector output, Reporter reporter) t
            String[] tokens = Recommand.DELIMITER.split(value.toString());
            int userID = Integer.parseInt(tokens[0]);
            String itemID = tokens[1];
            String pref = tokens[2];
            k.set(userID);
            v.set(itemID + ":" + pref);
            output.collect(k, v);
        }
    }

    public static class Step1_ToUserVectorReducer extends MapReduceBase implements Reducer
        private final static Text v = new Text();

        @Override
        public void reduce(IntWritable key, Iterator values, OutputCollector output, Reporte
            StringBuilder sb = new StringBuilder();
            while (values.hasNext()) {
                sb.append("," + values.next());
            }
            v.set(sb.toString().replaceFirst(",", ""));
            output.collect(key, v);
        }
    }
```

# 程序开发： Step1.java，计算结果

```
~ hadoop fs -cat /user/hdfs/recommand/step1/part-00000

1	102:3.0,103:2.5,101:5.0
2	101:2.0,102:2.5,103:5.0,104:2.0
3	107:5.0,101:2.0,104:4.0,105:4.5
4	101:5.0,103:3.0,104:4.5,106:4.0
5	101:4.0,102:3.0,103:2.0,104:4.0,105:3.5,106:4.0
```

# 程序开发： Step2.java，源代码

- 对物品组合列表进行计数，建立物品的同现矩阵。

```
public class Step2 {
    public static class Step2_UserVectorToCooccurrenceMapper extends MapReduceBase implement
        private final static Text k = new Text();
        private final static IntWritable v = new IntWritable(1);

        @Override
        public void map(LongWritable key, Text values, OutputCollector output, Reporter repo
            String[] tokens = Recommand.DELIMITER.split(values.toString());
            for (int i = 1; i < tokens.length; i++) {
                String itemID = tokens[i].split(":")[0];
                for (int j = 1; j < tokens.length; j++) {
                    String itemID2 = tokens[j].split(":")[0];
                    k.set(itemID + ":" + itemID2);
                    output.collect(k, v);
                }
            }
        }
    }

    public static class Step2_UserVectorToConoccurrenceReducer extends MapReduceBase impleme
        private IntWritable result = new IntWritable();

        @Override
        public void reduce(Text key, Iterator values, OutputCollector output, Reporter repor
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            result.set(sum);
            output.collect(key, result);
        }
    }
```

# 程序开发： Step2.java，计算结果

```
~ hadoop fs -cat /user/hdfs/recommand/step2/part-00000

101:101 5
101:102 3
101:103 4
101:104 4
101:105 2
101:106 2
101:107 1
102:101 3
102:102 3
102:103 3
102:104 2
102:105 1
102:106 1
103:101 4
103:102 3
103:103 4
103:104 3
103:105 1
103:106 2
104:101 4
104:102 2
104:103 3
104:104 4
104:105 2
104:106 2
104:107 1
```

# 程序开发： Step3.java，源代码

- 对同现矩阵和评分矩阵转型

```
public class Step3 {

    public static class Step31_UserVectorSplitterMapper extends MapReduceBase implements Map
        private final static IntWritable k = new IntWritable();
        private final static Text v = new Text();

        @Override
        public void map(LongWritable key, Text values, OutputCollector output, Reporter repo
            String[] tokens = Recommand.DELIMITER.split(values.toString());
            for (int i = 1; i < tokens.length; i++) {
                String[] vector = tokens[i].split(":");
                int itemID = Integer.parseInt(vector[0]);
                String pref = vector[1];

                k.set(itemID);
                v.set(tokens[0] + ":" + pref);
                output.collect(k, v);
            }
        }
    }
```

```
    public static class Step32_CooccurrenceColumnWrapperMapper extends MapReduceBase impleme
        private final static Text k = new Text();
        private final static IntWritable v = new IntWritable();

        @Override
        public void map(LongWritable key, Text values, OutputCollector output, Reporter repo
            String[] tokens = Recommand.DELIMITER.split(values.toString());
            k.set(tokens[0]);
            v.set(Integer.parseInt(tokens[1]));
            output.collect(k, v);
        }
    }
```

# 程序开发： Step3.java，计算结果

```
~ hadoop fs -cat /user/hdfs/recommand/step3_1/part-00000

101	5:4.0
101	1:5.0
101	2:2.0
101	3:2.0
101	4:5.0
102	1:3.0
102	5:3.0
102	2:2.5
103	2:5.0
103	5:2.0
103	1:2.5
103	4:3.0
104	2:2.0
104	5:4.0
104	3:4.0
104	4:4.5
105	3:4.5
105	5:3.5
106	5:4.0
106	4:4.0
107	3:5.0
```

```
~ hadoop fs -cat /user/hdfs/recommand/step3_2/part-00000

101:101 5
101:102 3
101:103 4
101:104 4
101:105 2
101:106 2
101:107 1
102:101 3
102:102 3
102:103 3
102:104 2
102:105 1
102:106 1
103:101 4
103:102 3
103:103 4
103:104 3
103:105 1
103:106 2
104:101 4
104:102 2
104:103 3
```

# 程序开发： Step4.java，源代码

```
public class Step4 {

    public static class Step4_PartialMultiplyMapper extends MapReduceBase implements Mapper
        private final static IntWritable k = new IntWritable();
        private final static Text v = new Text();

        private final static Map> cooccurrenceMatrix = new HashMap>();

        @Override
        public void map(LongWritable key, Text values, OutputCollector output, Reporter repo
            String[] tokens = Recommand.DELIMITER.split(values.toString());

            String[] v1 = tokens[0].split(":");
            String[] v2 = tokens[1].split(":");

            if (v1.length > 1) {// cooccurrence
                int itemID1 = Integer.parseInt(v1[0]);
                int itemID2 = Integer.parseInt(v1[1]);
                int num = Integer.parseInt(tokens[1]);

                List list = null;
                if (!cooccurrenceMatrix.containsKey(itemID1)) {
                    list = new ArrayList();
                } else {
                    list = cooccurrenceMatrix.get(itemID1);
                }
                list.add(new Cooccurrence(itemID1, itemID2, num));
                cooccurrenceMatrix.put(itemID1, list);
            }
            if (v2.length > 1) {// userVector
                int itemID = Integer.parseInt(tokens[0]);
                int userID = Integer.parseInt(v2[0]);
                double pref = Double.parseDouble(v2[1]);
                k.set(userID);
                for (Cooccurrence co : cooccurrenceMatrix.get(itemID)) {
                    v.set(co.getItemID2() + "," + pref * co.getNum());
                    output.collect(k, v);
                }

            }
        }
    }
```

# 程序开发： Step4.java，计算结果

```
~ hadoop fs -cat /user/hdfs/recommand/step4/part-00000

1	107,5.0
1	106,18.0
1	105,15.5
1	104,33.5
1	103,39.0
1	102,31.5
1	101,44.0
2	107,4.0
2	106,20.5
2	105,15.5
2	104,36.0
2	103,41.5
2	102,32.5
2	101,45.5
3	107,15.5
3	106,16.5
3	105,26.0
3	104,38.0
3	103,24.5
3	102,18.5
3	101,40.0
4	107,9.5
4	106,33.0
4	105,26.0
4	104,55.0
```

| | 101 | 102 | 103 | 104 | 105 | 106 | 107 | | U3 | | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **101** | 5 | 3 | 4 | 4 | 2 | 2 | 1 | | 2.0 | | 40.0 |
| **102** | 3 | 3 | 3 | 2 | 1 | 1 | 0 | | 0.0 | | 18.5 |
| **103** | **4** | **3** | **4** | **3** | **1** | **2** | **0** | **X** | **0.0** | **=** | **24.5** |
| **104** | 4 | 2 | 3 | 4 | 2 | 2 | 1 | | 4.0 | | 40.0 |
| **105** | 2 | 1 | 1 | 2 | 2 | 1 | 1 | | 4.5 | | 26.0 |
| **106** | 2 | 1 | 2 | 2 | 1 | 2 | 0 | | 0.0 | | 16.5 |
| **107** | 1 | 0 | 0 | 1 | 1 | 0 | 1 | | 5.0 | | 15.5 |

# 程序开发： HdfsDAO.java，源代码

```
public class HdfsDAO {

    private static final String HDFS = "hdfs://192.168.1.210:9000/";

    public HdfsDAO(Configuration conf) {
        this(HDFS, conf);
    }

    public HdfsDAO(String hdfs, Configuration conf) {
        this.hdfsPath = hdfs;
        this.conf = conf;
    }

    private String hdfsPath;
    private Configuration conf;

    public static void main(String[] args) throws IOException {
        JobConf conf = config();
        HdfsDAO hdfs = new HdfsDAO(conf);
        hdfs.copyFile("datafile/item.csv", "/tmp/new");
        hdfs.ls("/tmp/new");
    }
```

- 详细解释，请参考文章：Hadoop编程调用HDFS

# 程序开发： 源代码下载

- 完整的MapReduce实现，在github上面：
  - https://github.com/bsspirit/maven_hadoop_template/releases/tag/recommend

- 补充资料：
- RHadoop的实现方案
  - RHadoop实践系列之三 R实现MapReduce的协同过滤算法
- Mahout的实现方案
  - Mahout分步式程序开发 基于物品的协同过滤ItemCF

# 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