一种改进的回答集逻辑程序分割方法及程序化简研究

霍子伟

中山大学软件学院

2015 年 5 月 10 日

主要内容

- 研究背景
- 改进的程序分割方法
- 程序化简
- 实验与分析
- 总结与展望

研究背景

1. 什么是 ASP 逻辑程序
2. 分割集和程序分割方法
3. 分割理论的局限性

什么是 ASP 逻辑程序

ASP 逻辑程序形成与发展历史：

- 70 年代，Kowalski 提出逻辑可以作为程序语言的基础的观点
- 1978 年，Clark 提出失败即否定和 Clark 完备
- 1979 年，Colmerauer 发明了第一种逻辑程序语言：PROLOG
- 1988 年，Gelfond 和 Lifschitz 提出稳定模型语义
- 90 年代，回答集编程 (ASP：Answer Set Program) 形成
- 90 年代至今，一系列 ASP 求解器诞生，ASP 语义也在不断扩展

什么是 ASP 逻辑程序

ASP 逻辑程序形成与发展历史：

- 70 年代，Kowalski 提出逻辑可以作为程序语言的基础的观点
- 1978 年，Clark 提出失败即否定和 Clark 完备
- 1979 年，Colmerauer 发明了第一种逻辑程序语言：PROLOG
- 1988 年，Gelfond 和 Lifschitz 提出稳定模型语义
- 90 年代，回答集编程 (ASP：Answer Set Program) 形成
- 90 年代至今，一系列 ASP 求解器诞生，ASP 语义也在不断扩展

ASP 中的一个研究热点是：ASP 求解的提速。

分割集和程序分割方法

- **分割集 (Splitting Set)**：给定一个 ASP 逻辑成 P，其分割集时一个原子集 U，U 满足：对于任意的规则 $r \in P$ 有 $head(r) \cap U \neq \emptyset$ 蕴涵 $Atoms(r) \subseteq U$。
- **底部和顶部 (bottom & top)**：标记 P 关于 U 的底部为 $b_U(P)$，顶部为 $t_U(P)$，并定义为：
  - $b_U(P) = \{r \in P \mid head(r) \cap U \neq \emptyset\}$
  - $t_U(P) = P \setminus b_U(P)$
- **化简操作** $e_U(P, X)$：其中 X 为原子集，对 P 执行以下两个步骤：
  - 删除符合以下条件的规则 r：$head(r) \cap X \neq \emptyset$ 且 $body^+(r) \cap U \nsubseteq X$，或者 $(body^-(r) \cap U) \cap X \neq \emptyset$；
  - 在剩下的规则的体部中把所有形如 a 或 $not\ a$ 的文字删掉，其中 $a \in U$。

分割集和程序分割方法

- **方案 (Solution)：** P 关于 U 的一个方案是一组原子集 $\langle X, Y\rangle$，具体地：
  - X 是 $b_U(P)$ 的回答集；
  - Y 是 $e_U(P \backslash b_U(P), X)$ 的回答集。

Theorem (分割集理论)

已知 ASP 逻辑程序 P 和其分割集 U，则一个原子集 S 是 P 的回答集，当且仅当 $S = X \cup Y$，其中 $\langle X, Y\rangle$ 是 P 关于 U 的一个方案。

分割理论的局限性

改进的程序分割方法

1. NLP 的新程序分割方法
2. DLP 的新程序分割方法
3. 强程序分割方法
4. 计算复杂性分析

NLP 的新程序分割方法

数据类型

字段	字段说明	提取说明
user_id	用户标记	抽样&字段加密
Time	行为时间	精度到天级别&隐藏年份
action_type	用户对品牌的行为类型	包括点击、购买、加入购物车、收藏4种行为 (点击：0 购买：1 收藏：2 购物车：3)
brand_id	品牌数字ID	抽样&字段加密

- 用户对任意商品的行为都会映射为一行数据。
- 其中所有商品 ID 都已汇总为商品对应的品牌 ID
- 用户和品牌都分别做了一定程度的数据抽样，且数字 ID 都做了加密
- 所有行为的时间都精确到天级别 (隐藏年份)

数据类型

1	user_id	brand_id	type	visit_datetime
2	10944750	13451	0	6月4日
3	10944750	13451	2	6月4日
4	10944750	13451	2	6月4日
5	10944750	13451	0	6月4日
6	10944750	13451	0	6月4日
7	10944750	13451	0	6月4日
8	10944750	13451	0	6月4日
9	10944750	13451	0	6月4日
10	10944750	21110	0	6月7日

第一阶段的数据规模：

- 总条数：182880
- 用户数：884
- 品牌数：9531

评估指标：F_1-Score

准确率：$precision = \frac{\sum_i^N hitBrands_i}{\sum_i^N pBrands_i}$

注：

- N 为参赛队预测的用户数
- $pBrands_i$ 为用户 i 预测他 (她) 会购买的品牌列表个数
- $hitBrands_i$ 对用户 i 预测的品牌列表与用户 i 真实购买的品牌交集的个数

召回率：$recall = \frac{\sum_i^M hitBrands_i}{\sum_i^M bBrands_i}$

注：

- M 为实际产生成交的用户数量
- $bBrands_i$ 为用户 i 真实购买的品牌个数
- $hitBrands_i$ 预测的品牌列表与用户 i 真实购买的品牌交集的个数

F_1-Score: $F_1 = \frac{2 \times P \times R}{P+R}$

问题分析

我们把该问题看作是一个分类问题——根据前 4 个月出现过的 <user, brand> 对的数据，来预测下一个月，该 <user, brand> 对是否发生购买

尝试过的做法：

- 基于矩阵分解的协同过滤（SVDFeature）
- 基于规则的推荐
- 多模型融合（规则、SVM_Perf、LR、MLP、SAE）

方案 1：SVDFeature

SVDFeature 是上海交大 Apex 实验室参加 KDDCUP 2011 期间开发的，可以完成 feature-based matrix factorization。

具体过程：

1. 将每个月的数据，构造成 <user, brand> 的矩阵（缺失的）M_1, M_2, M_3, M_4
2. 对每个月的矩阵使用 SVDFeature 进行矩阵补全，补全后的矩阵为 S_1, S_2, S_3, S_4
3. 学习一个转移矩阵 W（S_i 变换到 S_{i+1}）

方案 1：SVDFeature

结果：效果不好，F_1-Score 只有 0.09%

原因：

1. 数据规模小
2. 补全前的矩阵 M_1, M_2, M_3, M_4 过于稀疏
3. 无法处理“异常”的数据（如，购买/总操作数特别高的用户，只点击但是不购买的用户……）

方案 2：基于规则的推荐

我们设计了以下的规则：

1. 4 个月所有的购买记录作为我们最后的预测（F_1-Score $= 5.41\%$）
2. 只出现在前 2 个半月购买，后面 1 个半月没有出现购买的 <user, brand> 对，我们把它过滤掉
3. 在最后的 1 个半月中，一个用户至少 5 天点击了该商品，并且他没有购买这个商品我们将该商品推荐给他
4. 在最后的 1 个半月中，一个用户点击某个商品大于 10 次，并且他没有购买这个商品，我们将该商品推荐给他
5. 在最后的 1 个月中，一个用户在一天中点击某个商品至少 7 次，并且当天他没有购买过别的商品，我们将该商品推荐给他

结果：F_1-Score 6.41%

方案 3：多模型融合

特征构造：

1. 对每个 <user, brand> 对按每一周分别统计点击、购买、收藏、加入购物车的总次数
2. 每个 <user, brand> 对的统计数据，按周依此排列成一行，每周有 4 个数据，分别对应点击总次数、购买总次数、收藏总次数、加入购物车总次数

例如：

用户 i 在第一周对品牌 j 一共点击了 12 次，购买了 1 次，在第二周对它点击了 2 次，收藏了 3 次，那么，$< i, j >$ 对的前面 8 维分别是：

12 1 0 0 2 0 3 0

方案 3：多模型融合

训练与预测的过程：

1. 训练集由 2 部分组成，第一部分是第 1～12 周的统计数据做为输入特征，第 13～16 周做为对应的 Label；第二部分是第 3～14 周的统计数据做为输入特征，第 15～18 周做为对应的 Label
2. 对于前 12 周的一个 <user, brand> 对，如果在接下来的 4 周中出现购买记录，则对应的 Label 为 1，反之对应的 Label 为 0
3. 最后 12 周的统计数据做为预测用的输入特征，其输出的结果为我们最终的预测结果

方案 3：多模型融合

我们最终的模型是通过融合 6 个模型的结果得到的，其中包括:

- 两个 MLP（多层感知机）
- SAE（稀疏自编码网络）
- LR（逻辑式回归）
- svm_perf
- 基于规则的模型

方案 3：多模型融合

预处理：由于正负样本不平衡，并且数据量少，因此需要复制正样本，使得正负样本的数目大致平衡状态

逻辑式回归：

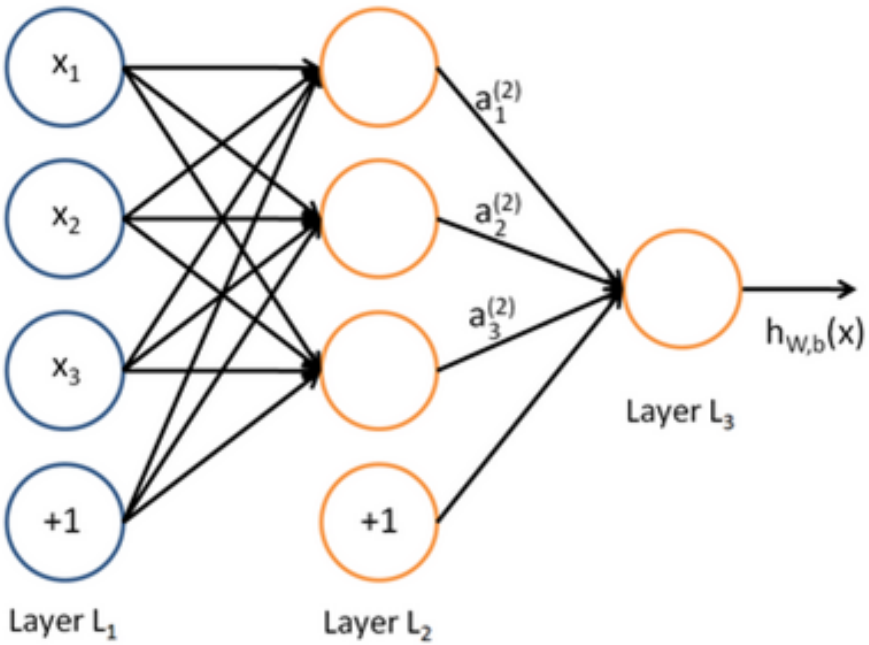
- 结果：$precision = 6.49\%$

svm_perf:

- 该 SVM 算法有多种选项，可以让目标模型优化 precision，或者优化 F_1-Score
- 估计结果：$precision = 7\%$

方案 3：多模型融合

MLP（多层感知机）



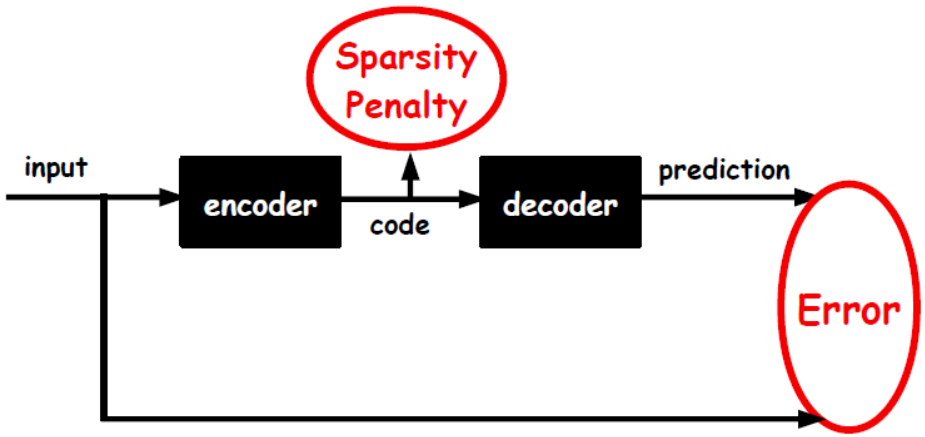
- 模型：三层网络结构，中间一层位隐含层 (20 个节点)，输出层 1 个节点，采用交叉熵损失
- 学习能力比 LR 强，可以学习到非线性的关系
- 结果:

  $\mathit{precision} = 8\%,$

  $F_1\text{-Score} = 6.6\%$

方案 3：多模型融合

SAE（自编码网络）



- 通过逐层学习自编码网络来初始化权重
- 自编码是无监督的神经网络模型，通常加入稀疏性惩罚以学习到更有意义的特征
- 模型：2 层自编码编码层 $+1$ 层 sigmoid(softmax) 输出层
- 结果：*precision* $= 6.32\%$, F_1-Score $= 6.5\%$

方案 3：多模型融合

融合的过程（投票）：

1. 采用其中一个 MLP 模型（预测数目很少，但 precision 有 11.5%）作为基本的模型
2. 其他的 5 个模型，对其求差集
3. 对这 5 个差集中的 <user, brand> 对进行投票
4. 投票结果大于等于 3 的 <user, brand> 对合并上我们基本的模型就是我们最终所要预测的结果

结果：F_1-Score $= 6.74\%$

“天池”平台

第二阶段的比赛规则是：参赛者须使用“天池”平台（阿里巴巴自主研发的分布式计算平台），访问海量的天猫数据，并利用 MapReduce、SQL 及各种平台集成的机器学习算法包调试模型、提交结果。

举办方会提供一个虚拟机给参赛队伍，参赛队伍通过 Windows 的远程桌面连接便可登陆。

“天池”平台

ODPS（Open Data Processing Service）是阿里巴巴自主研发的海量数据离线数据处理平台。主要服务于实时性要求相对不高的批量结构化数据的储存 和计算，可以提供海量数据仓库的解决方案以及针对大数据的分析建模服务。

ODPS 有许多的功能，我们这次比赛主要用到的是它所提供的计算及分析任务的功能:

- ODPS SQL
- ODPS XLab/XLib
- MapReduce

ODPS SQL

ODPS SQL 采用的是类似于 SQL 的语法，可以看作是标准 SQL 的子集，但不能因此简单地把 ODPS SQL 等价成一个数据库，它在很多方面并不具备数据库的特征，如 事务、主键约束等。

从文档上也可以看出来，ODPS SQL 的操作不支持单行的 update 和 delete。（原因：实时性、吞吐量、容错……）

类似的开源软件：Hive

Hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的 SQL 查询功能，可以将 SQL 语句转换为 MapReduce 任务进行运行。

ODPS SQL

```
ODPS
Type `help` for more information.
odps:tianchi_0245> sql
odps:sql:tianchi_0245> select count(*) from t_alibaba_bigdata_user_brand_total_1;
InstanceId: 20140526092457606glowz0z5
SQL: ..
Summary of SQL:
Inputs:
        tianchi_0245.t_alibaba_bigdata_user_brand_total_1: 571906480 (8251 bytes)
Outputs:
M1_Stg1_tianchi_0245_20140526092457606glowz0z5_SQL_0_0_0_job0:
        Worker Count:37
        Input Records:
                input: 571906480 (min: 14662137, max: 16434028, avg: 15451723)
        Output Records:
                R2_1_Stg1: 37 (min: 1, max: 1, avg: 1)
R2_1_Stg1_tianchi_0245_20140526092457606glowz0z5_SQL_0_0_0_job0:
        Worker Count:1
        Input Records:
                input: 37 (min: 37, max: 37, avg: 37)
        Output Records:
                R2_1_Stg1FS_1263012: 1 (min: 1, max: 1, avg: 1)


+------------+
| _c0        |
+------------+
| 571906480  |
+------------+
```

ODPS SQL

通过 ODPS SQL，我们可以知道第二阶段的数据规模为：

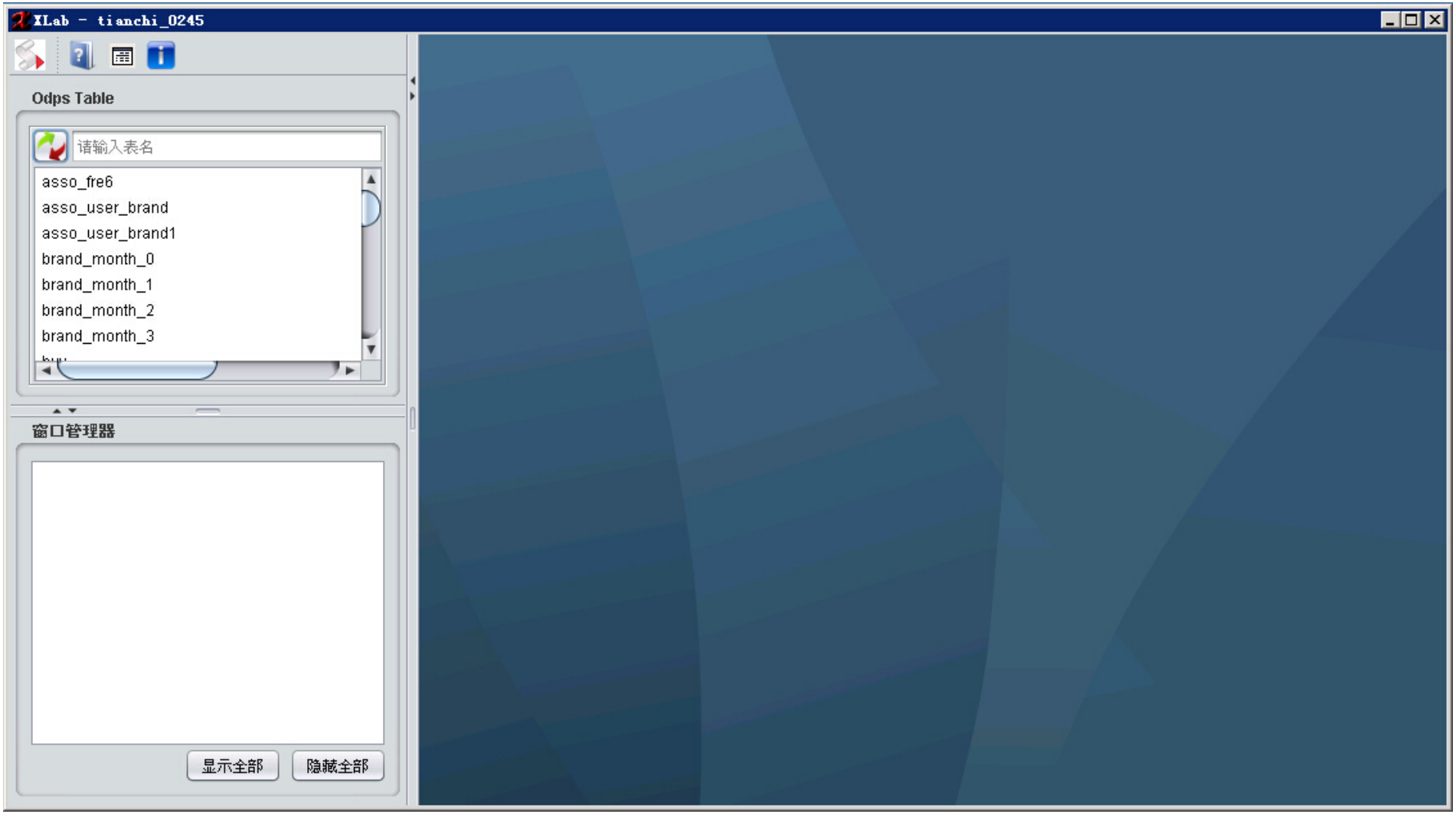
- 总条数：571906480
- 用户数：12500984
- 品牌数：29706

ODPS XLab/XLib

XLab/XLib 可以帮助用户轻松处理海量数据，包括了统计、机器学习、矩阵等常用计算功能。

- XLab 是客户端。无论您是否有大数据分析的基础，都可以通过 XLab图形界面，轻松上手；XLab 还提供了脚本编辑执行功能，灵活方便、帮您成为大数据分析的高手。
- XLib 是 XLab 的后台分布式算法库。可以通过 XLab 或 ODPS 客户端调用；由于二者使用相同的函数定义，XLab 上的函数命令和脚本可以再 ODPS 客户端上执行执行。

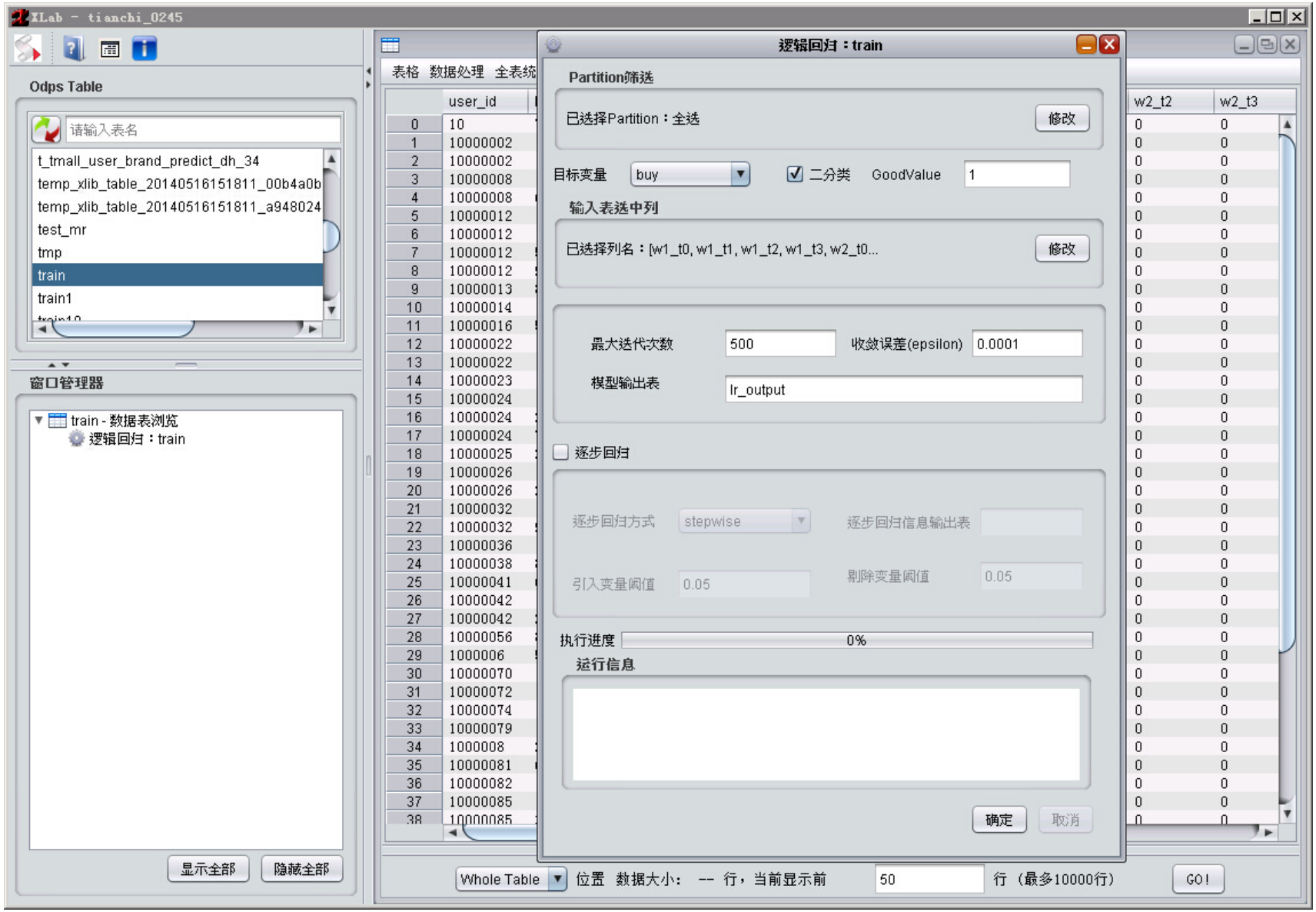
ODPS XLab/XLib



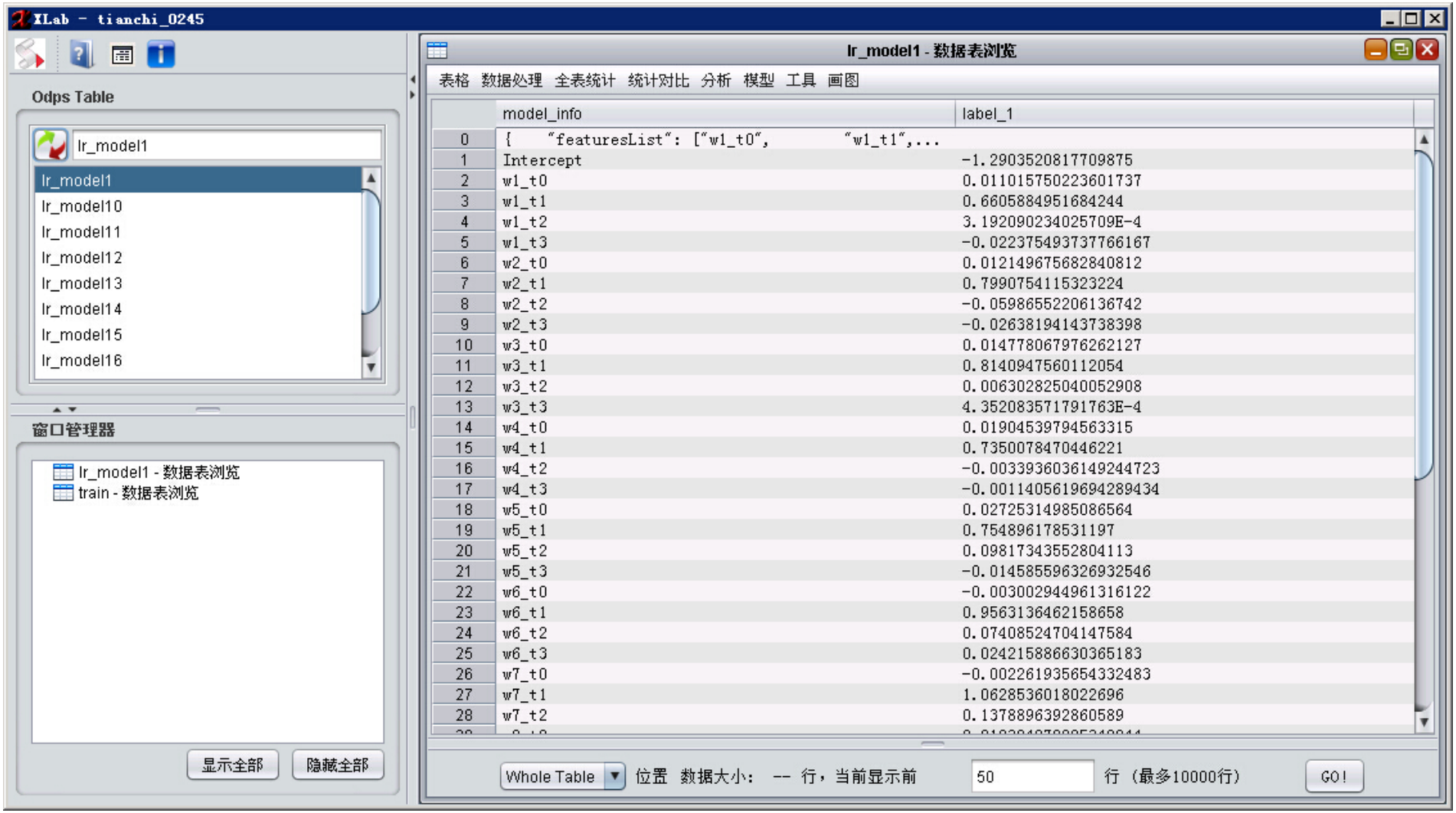
ODPS XLab/XLib

ODPS XLab/XLib

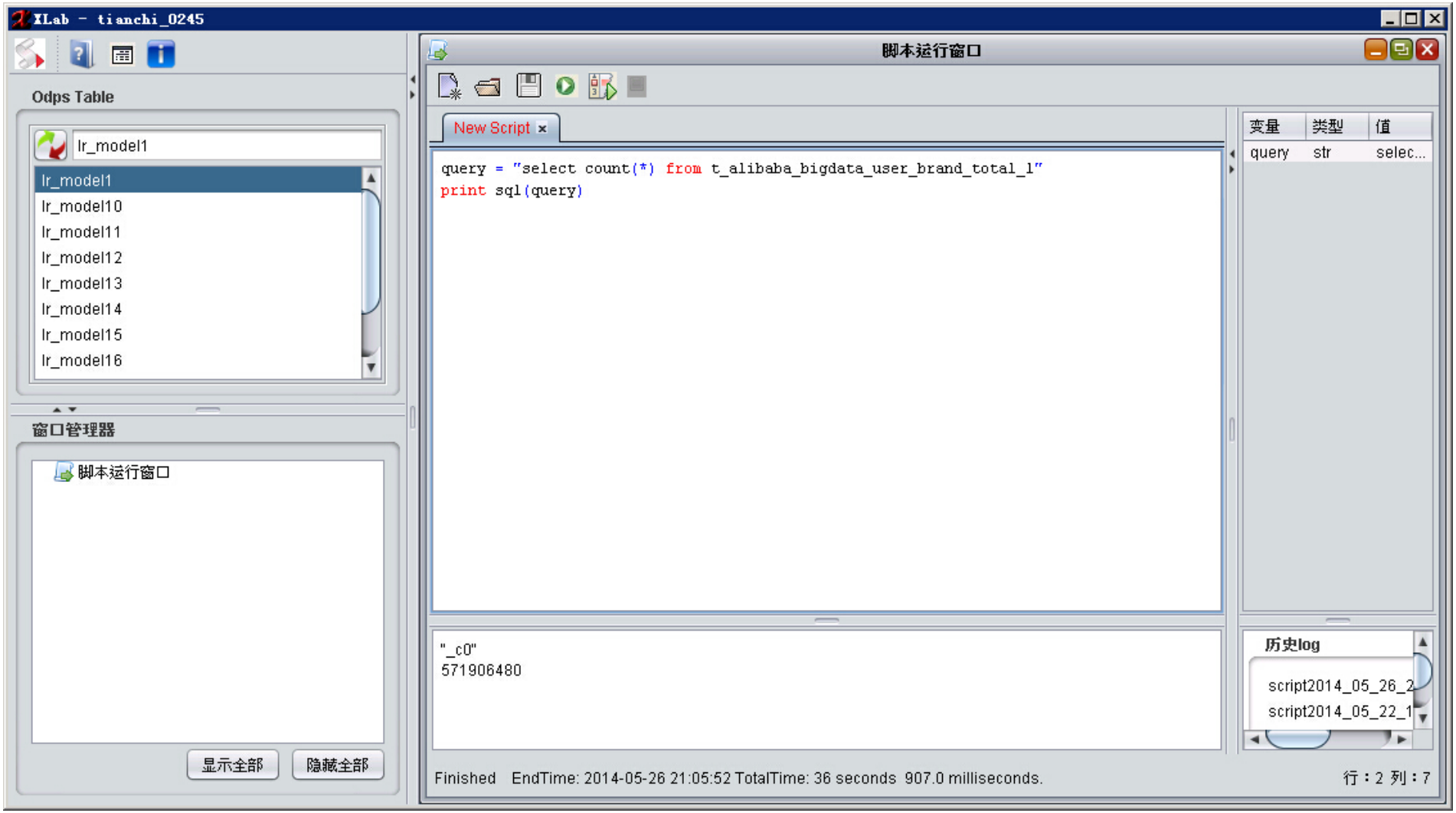
ODPS XLab/XLib



ODPS XLab/XLib



ODPS XLab/XLib

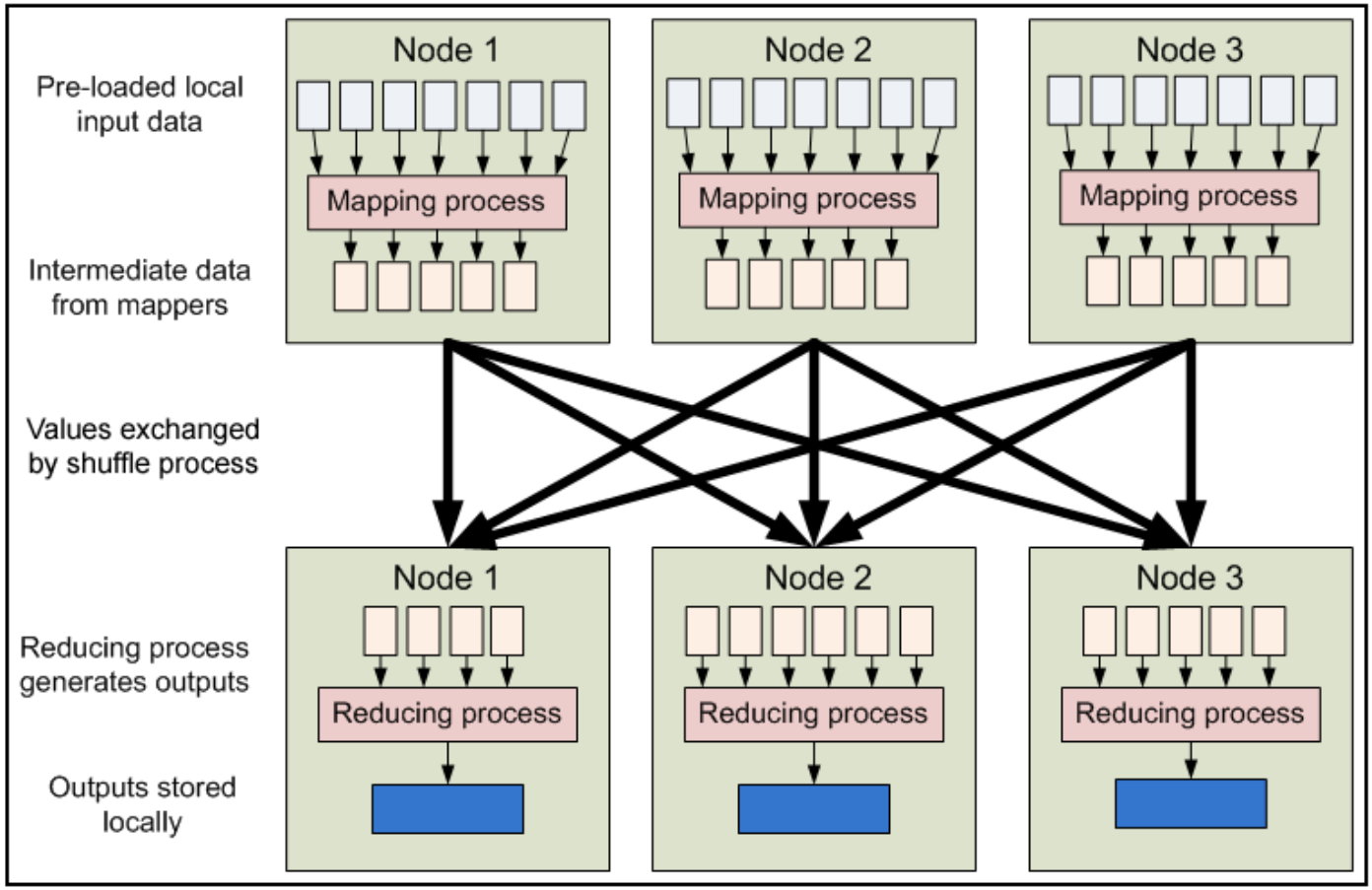


类似的开源软件：Mahout

Mahout 是 Apache Software Foundation（ASF）旗下的一个开源项目，提供一些可扩展的机器学习领域经典算法的实现，旨在帮助开发人员更加方便快捷地创建智能应用程序。

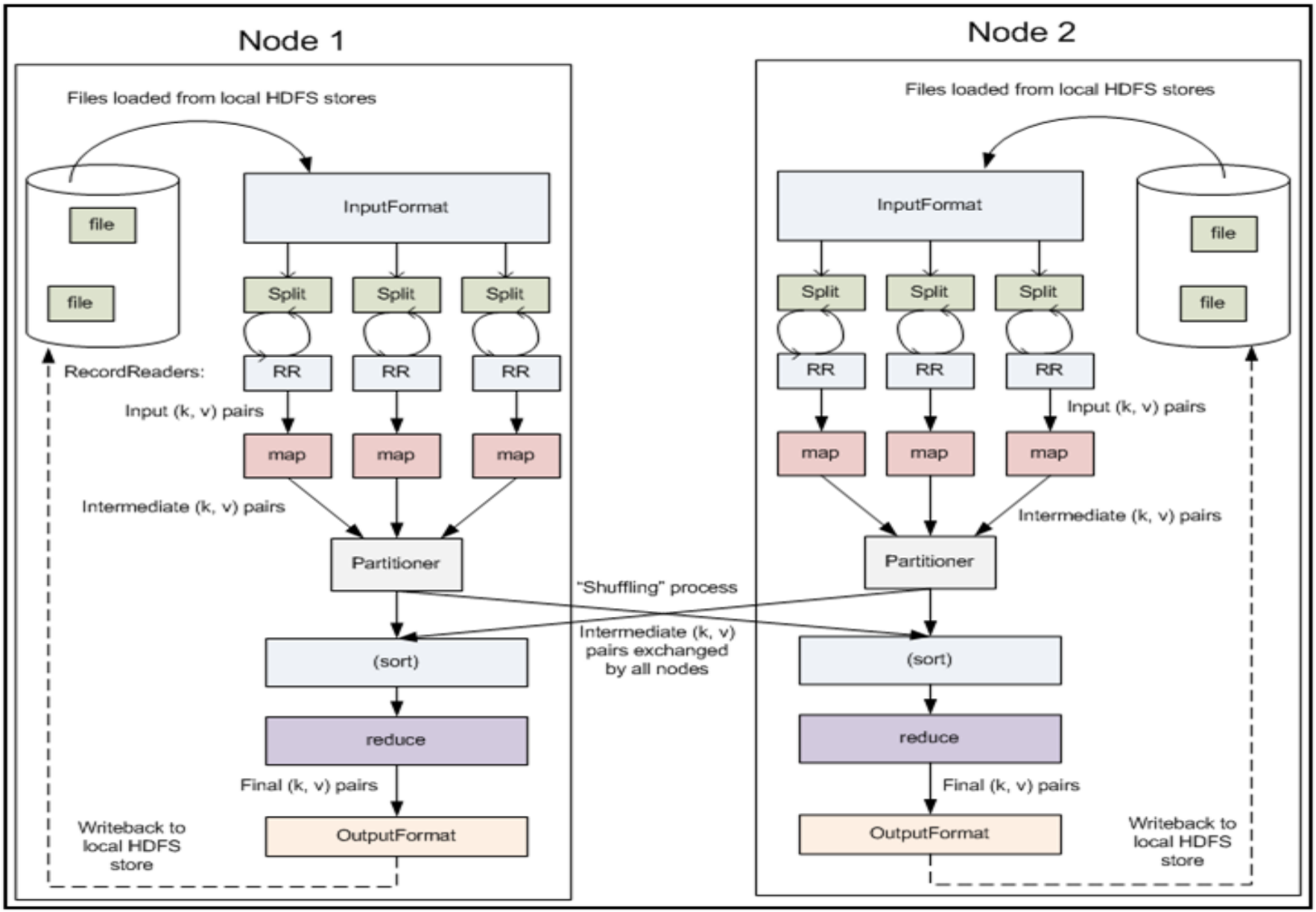
Mahout 包含许多实现，包括聚类、分类、推荐过滤、频繁子项挖掘。此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到云中。

MapReduce



- 分布式并行计算框架，把大数据集分割成多个小数据集，分而治之，各个击破
- 函数式编程范式，开发者需自定义 map 和 reduce 函数

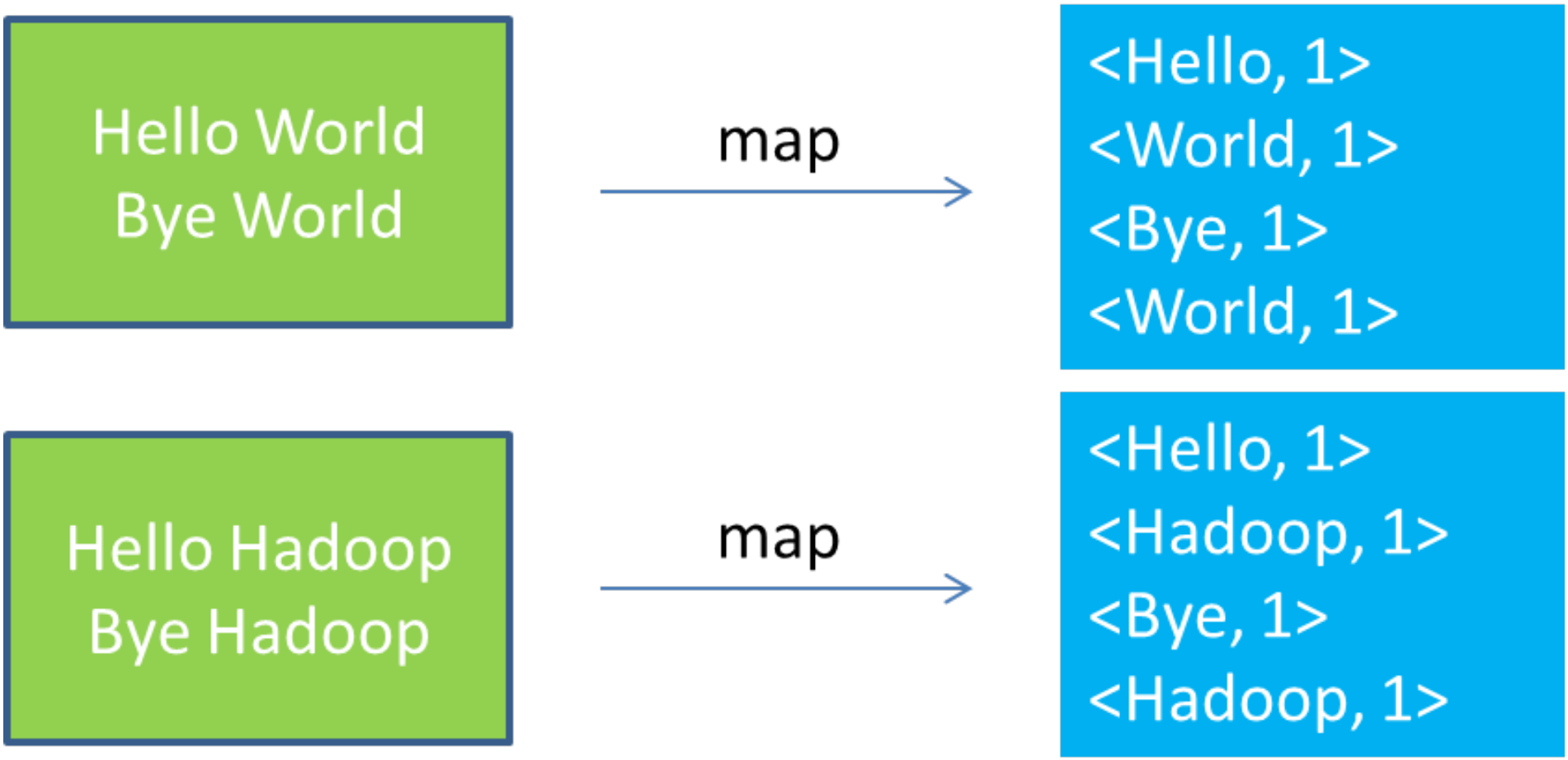
MapReduce


Node 1
Files loaded from local HDFS stores
file
file
InputFormat
Split
Split
Split
RecordReaders:
RR
RR
RR
Input (k, v) pairs
map
map
map
Intermediate (k, v) pairs
Partitioner
"Shuffling" process
Intermediate (k, v) pairs exchanged by all nodes
(sort)
reduce
Final (k, v) pairs
OutputFormat
Writeback to local HDFS store
Node 2
Files loaded from local HDFS stores
InputFormat
file
file
Split
Split
Split
RR
RR
RR
Input (k, v) pairs
map
map
map
Intermediate (k, v) pairs
Partitioner
(sort)
reduce
Final (k, v) pairs
OutputFormat
Writeback to local HDFS store

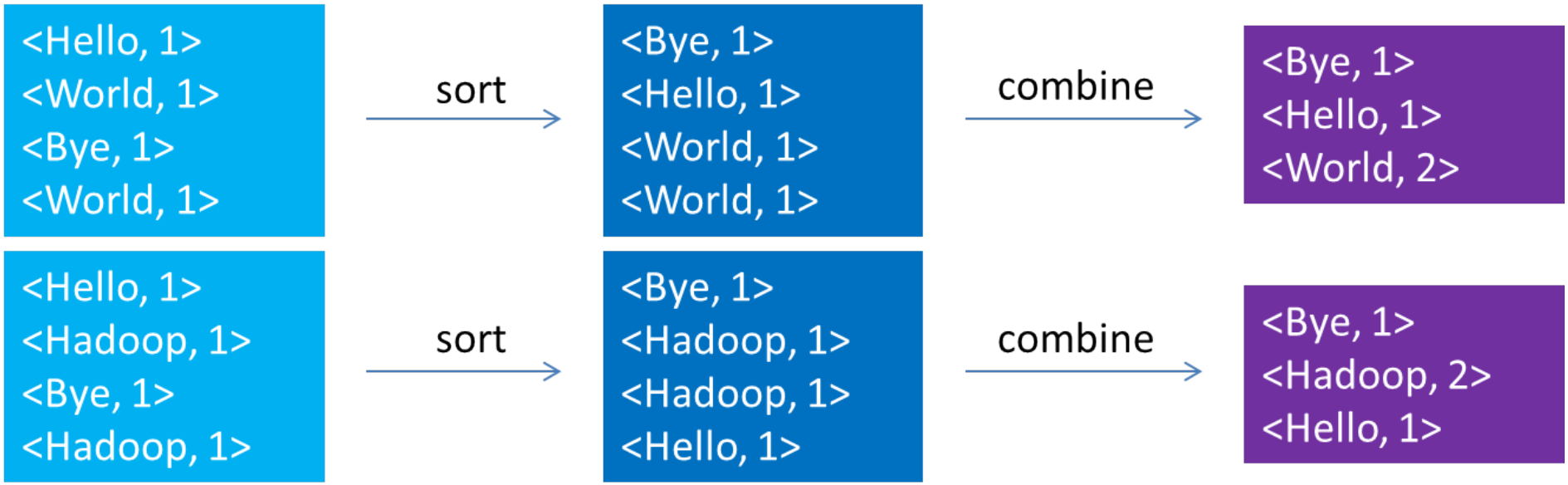
MapReduce

- 文件被划分为多个数据块存在 HDFS 文件系统中，每个 Mapper 取本地或靠近它的数据块进行 map 操作
- MapReduce 不支持修改数据
- Mapper 调用 map 处理文件的每一行数据，输入 (k1, v1)，输出 (k2, v2)
- Reducer 调用 reduce 进行归约，输入 (k, list of values)

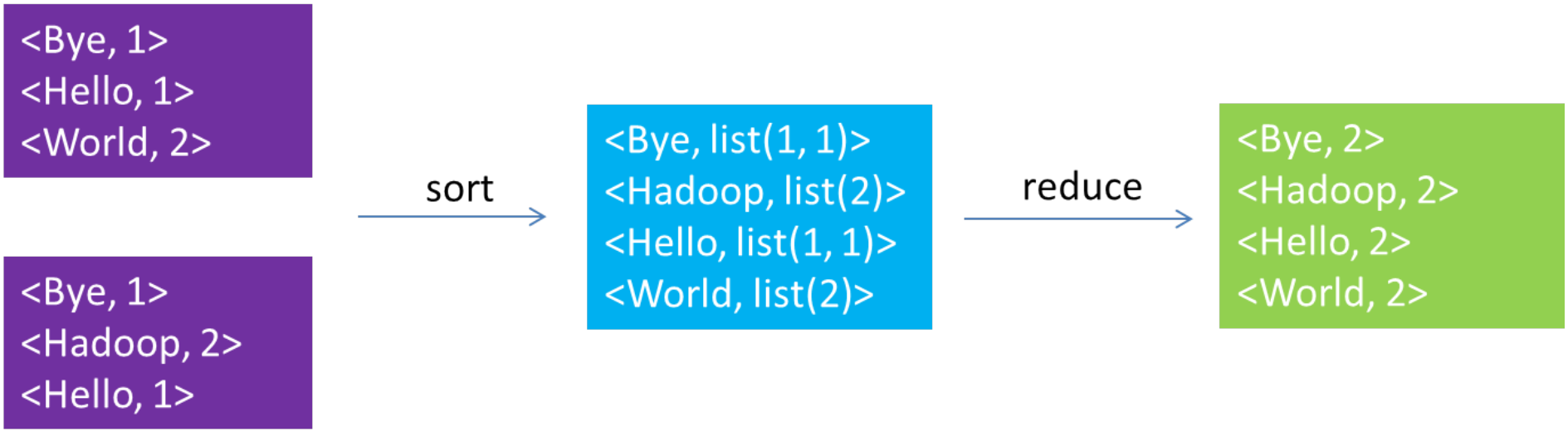
MapReduce Example: 统计词频



MapReduce Example: 统计词频



MapReduce Example: 统计词频



注：前面的 sort-combine 是在 Mapper 上做的，这里的 sort-reduce 是在 Reducer 上做的

目前进展

采取第 $1 \sim 14$ 周的统计数据做为输入特征，第 $15 \sim 18$ 周做为 Label

由于正负样本极度不平衡，并且数据量很大，因此，我们对负样本进行下采样，使得正负样本的数目 大致平衡，以这样的方式训练 50 次，得到 50 个 LR 的模型，最后对这 50 个 LR 的模型求平均，得到我们最终的模型

对预测结果抽取 top 300 万的数据做为我们最后的所提交的结果

截止到今天的排名是 156 名。

展望

- 融入用户和商品特征，而不仅仅是行为序列特征
- 尝试 MLP、RF、GBDT 等模型
- 处理非平衡数据的另外方法：不抽样，带权重的损失函数
- 推荐没有行为的商品：UserBased or ItemBased CF
- 其他？

感谢

感谢潘老师这 2 个多月来的指导，感谢刘冶师兄深夜帮忙跑模型，感谢在第一阶段的最后几天帮忙“扫雷”的同学！