哈尔滨工业大学

<<大数据分析>>

实验报告之一

(2019 年度春季学期)

姓名:	王丙昊
学号:	1160300302
学院:	计算机学院
教师:	杨东华

实验一 数据预处理

一、实验目的

1. 掌握数据预处理的步骤和方法，包括数据抽样、数据过滤、数据标准化和归一化、数据清洗。
2. 理解数据预处理各个步骤在大数据环境下的实现方式。
3. 整合 MapReduce 程序，缩短四个步骤完成时间。
4. 使用比默认值更合理的缺失值填充方法来填充数据中的缺失值。

二、实验环境

Linux 操作系统，伪分布式 Hadoop 环境，Java 语言。

三、实验过程及结果

**********。

1. 数据抽样

数据抽样过程使用一个 MapReduce 程序来实现，在 Map 阶段以“career”作为 key，以原记录内容作为 value 进行输出。

Map 阶段：分层

value in：从 DFS 中读取的一条记录的内容，类型为 Text

key out：该记录中的“career”，类型为 Text

value out：该条记录的内容，类型为 Text

举例：

输入：

144552912|9.349849|56.740876|17.052772|2011/06/27|18.5|83.91|38267|1974-06-08|Switzerland|programmer|5042

输出：

(**programmer**,144552912|9.349849|56.740876|17.052772|2011/06/27|18.5|83.91|38267|1974-06-08|Switzerland|programmer|5042)

Reduce 阶段：对每层进行**系统抽样**

执行过程：首先设置抽样率，如本实验中设为 0.01，也就是 100 个里面抽取一个。首先在前 100 个样本中随机抽取一个，然后每隔 100 个抽取下一个，直到本层样本抽取完毕。

Key in：对应 Map 阶段的 key out，类型为 Text，表示该样本中“Career”

Value in：对应 Map 阶段的 value out，类型为 Text，表示一个样本记录

Key out：被抽中的样本记录，类型为 Text

Value out：空。

举例：

输入：

(programmer,144552912|9.349849|56.740876|17.052772|2011/06/27|18.5|83.91|38267|1974-06-08|Switzerland|programmer|5042)

输出：

(144552912|9.349849|56.740876|17.052772|2011/06/27|18.5|83.91|38267|1974-06-08|Switzerland|programmer|5042，null)

2. 数据过滤

数据过滤阶段使用了两个 MapReduce 程序，第一个 MapReduce 程序读取数据抽样的结果 D_Sample，找到其 rating 取值前 1%和 99%的阈值；第二个 MapReduce 程序对原数据进行过滤，包括不合法的经纬度以及 rating 在阈值外的记录。

MapReduce1. Map：

功能：从 DFS 中读取 D_Sample，将所有的元组发送到一个集群节点上。

执行过程：执行判断，如果所读如的记录中的 **rating 不为空**，则按照下面的格式进行输出。

Value in: 从 DFS 中读取的一条记录的内容，类型为 Text

Key out: 类型为 IntWritable 的某个常数，用于将所有记录发送到同一个 Reduce 节点上处理。

Value out: 该样本中 rating 项的内容，类型为 DoubleWritable

举例：

输入：

114740919|9.607819|57.275233|5.816675|2016/05/04|12.5℃|80.20|8635|1992-09-29|Denmark|Manager|3965

输出：

（CONST，80.20）

MapReduce1.Reduce：

功能：找到 D_Sample 中 rating 前 1%和 99%的阈值

执行过程：在 Map 阶段，D_Sample 中所有记录的 rating 均在一个 reduce 节点上，只需要对其进行排序，便可以按照下标找到阈值。

输入：key 和 value 均为 Map 阶段的 key out 和 value out。

输出：由于该 MapReduce 程序只是为了找到 D_Sample 文件中 Rating 的阈值，所以无需输出。

MapReduce2.Map：

功能：根据原数据的经纬度信息，对其进行过滤。

执行过程：从读如的 Text 中提取 longitude 及 latitude 属性，判断合法则输出。

Value in：从 DFS 中读取的记录的内容，类型为 Text

Key out：记录中的 rating，由于可能有空值的形况，类型为 Text

Value out：所读入记录的内容，类型为 Text

MapReduce2.Reduce：

功能：根据原数据的 Rating 信息，结合上一个 MapReduce 程序中得到的 D_Sample 中 Rating 的阈值，对原数据进行过滤。

执行过程：如果 Rating 为“缺失值”，直接输出；否则判断其是否在 D_Sample 确定的阈值之内，是则输出。

输入：key in 和 value in 分别对应 Map 阶段的 key out 和 value out

Key out：记录的内容，类型为 Text

Value out：空。

3. 数据标准化和归一化

数据标准化和归一化通过一轮 MapReduce 程序来实现。Map 阶段实现了日期和温度的标准化，Reduce 阶段实现了 Rating 的归一化。将日期格式都改为 yyyy/MM/dd 格式，将温度都改为摄氏度；归一化采用的是 min-max 归一化。

Map 阶段：

功能：首先将日期和温度标准化，执行过程中，同时获取 D_Filter 中的最大值和最小值，用于 Reduce 阶段的归一化。

执行过程：通过正则表达式来判断读如的记录中日期和温度的格式，如果不是要求的输出类型，则修改后，将其输出。

输入：将数据过滤后的结果 D_Filter 从 DFS 逐行读入，作为 Map 阶段的输入。

输出：

Key out：将 rating 作为输出的 key，类型为 Text

Value out：将日期和温度标准化后，形成一条新的纪录作为输出的 value，类行为 Text。

举例：

输入：

130583012|8.571642|56.762345|2.328474|January 1,2016|31.3℉|59.35|36959|1980-07-29|Spain|artist|1408

输出：

(59.35，130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|59.35|36959|1980/07/29|Spain|artist|1408)

Reduce 阶段：

功能：对 Rating 属性进行归一化处理

执行过程：采用的是 **min-max 归一化方法**，min 和 max 的值在 map 阶段已经得到，只需要判断 rating 是否是缺失值，如果不是缺失值的话对其进行归一化处理，否则直接输出。

输出：key，将 value in 的 rating 归一化后形成的新的记录

Value，为空。

举例：

输入：

(59.35，130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|59.35|36959|1980/07/29|Spain|artist|1408)

输出：

(130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|0.61|36959|1980/07/29|Spain|artist|1408，null)

4. 数据清洗

数据清洗过程通过两轮 MapReduce 程序实现，第一轮 MapReduce 用于填写缺失的 Income 属性，第二轮 MapReduce 用于填写缺失的 rating 属性。

第一轮 MapReduce：

通过“同一国家同一职业的人收入相近”这个依赖关系来处理缺失的 Income 属性，思想就是将同一国家同一职业的记录都放在同一个节点上进行处理，用非缺失值对缺失值进行估计，这里采用了**指数加权移动平均**的思想。

该缺失值填充方法介绍：在 reduce 方法中使用，仅需要处理该节点上的数据一遍，用之前已经读取过的非缺失值来估计当前遇到的缺失值。如果遇到的数据是非缺失值，加权更新用于填充缺失值的数据：

```
if(!income.equals("?")) {
    income_fill = 0.95 * income_fill + 0.05 * Double.parseDouble(income);
    context.write(val, null);
```

如果遇到的是缺失值，则用当前的 income_fill 进行填充。

Map 阶段：以记录中的 nation 和 career 作为 Key，原纪录作为 value，进行输出。

举例：

输入：

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|0.61|15594|1986/10/12|France|artist|?

输出：

(**France artist**,

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|0.61|15594|1986/10/12|France|artist|?)

Reduce 阶段：对 Income 的缺失值进行填充，**使用指数加权移动平均的方法来更新用于填充的值 Income_fill。**

执行过程：首先需要初始化 Income_fill，用于处理第一个数据是缺失值的情况。从前向后读取该节点上的数据一遍，如果是非缺失值，更新 Income_fill，否

则用当前的 Income_fill 来填充缺失值。

输入：输入的 key、value 与 Map 阶段的输出相同

Key out：填充 Income 后的记录，类型为 Text

Value out：为空

举例：

输入：

(**France artist**,

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|0.61|15594|1986/10/12|France|artist|?)

输出：

(25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|0.61|15594|1986/10/12|France|artist|3513，null)

第二轮 MapReduce:

根据“收入相近的人对同一地点的评分相近”这一依赖关系，结合 **KNN 方法**来对 Rating 的缺失值进行填充。处理方法与 Income 时相同，将对同一节点的评价记录都放在同一节点上进行处理。

Map 阶段：以记录中的 latitude、longitude 以及 altitude 作为 key，原纪录作为 Value 进行输出。

举例：

输入：

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|**?**|15594|1986/10/12|France|artist|3513

输出：

(10.634075|57.738730|1.630424，

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|**?**|15594|1986/10/12|France|artist|3513)

Reduece 阶段：使用 **KNN** 的方法，对缺失的 Rating 进行填充。

执行过程：首先遍历一遍 Reduce 节点上的所有记录，分别将无缺失值的 Rating **值**和有有缺失值的**记录**分别保存起来。然后遍历所有有缺失值的记录，将其中 Income 与所有 Rating 值进行差运算、取绝对值并从小到达排序取前 K 个，这也就是与缺失值的记录最相近的 K 条记录（**一维的 KNN**），取它们的平均值对 Rating 进行填充。

Key out：将 Rating 填充后新的记录，类型为 Text

Value out：为空。

格式举例：

输入：

(10.634075|57.738730|1.630424，

25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|**?**|15594|1986/10/12|France|artist|3513)

输出：

(25706427|10.634075|57.738730|1.630424|2013/06/26|19.5℃|**0.61**|15594|1986/10/12|France|artist|3513, null)

5. 整合预处理过程中的 MapReduce 程序

整合时的主要思想是将数据过滤与数据标准化和归一化一并处理。

第一轮 MapReduce 执行分层抽样，第二轮 MapReduce 从抽样结果中获取奇异值的临界值，这两轮 MapReduce 没有改变。

第三轮 MapReduce **将数据过滤与数据标准化和归一化进行了整合**，原本这两个过程需要读取两遍原文件，整合后只需要在一轮 MapReduce 中就可以完成这两项工作，也就是少读了一遍原文件，详细操作：

Map 阶段：对日期和温度进行标准化，并过滤掉经纬度不在所要求范围内的记录。输出时，以 Rating 作为 Key，标准化后的数据作为 Value 进行输出。

举例：

输入：

130583012|8.571642|56.762345|2.328474|January 1,2016|31.3℉|59.35|36959|1980-07-29|Spain|artist|1408

输出：

(59.35，130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|59.35|36959|1980/07/29|Spain|artist|1408)

Reduce 阶段：判读 Key 值（Rating），如果 Rating 为缺失值（？），将该 Reduce 节点上的所有记录输出；如果 Rating 不是缺失值，判断其是否在正常范围内，如果在正常范围内，将该节点上的所有记录输出，否则过滤掉该节点上的记录。

举例：

输入：

(59.35，130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|59.35|36959|1980/07/29|Spain|artist|1408)

输出：

(130583012|8.571642|56.762345|2.328474|2016/1/1 |31.3℃|59.35|36959|1980/07/29|Spain|artist|1408, null)

四、实验心得

**********。

环境搭建：

参考：http://dblab.xmu.edu.cn/blog/install-hadoop/

程序编写：

该实验编写时，合理分配好各个 Map 阶段和 Reduce 阶段所要执行的任务，然后再去编写每一个模块，这样既可以降低编写的难度，思路也会更清晰。