哈尔滨工业大学

**<<大数据分析>>**

**实验报告之一**

**(2023年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **刘俊杉** |
| **学号：** | **2021112078** |
| **学院：** | **未来技术学院** |
| **教师：** | **杨东华** |

实验一 数据预处理

## 实验目的

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。理解数据 预处理各个步骤在大数据环境下的实现方式。

## 实验环境

操作系统：Windows、Linux(建议) 框架：伪分布式Hadoop环境 编程语言：Java

## 三、实验过程及结果

**3.1 数据抽样**

分层抽样所使用的属性A为user\_career属性；

实现分层抽样，使用的属性是user\_career(12个属性中的第11个)

map输入key为行首字符偏移量，value为每行的内容

Map输出key为user\_career，value为每行的内容

首先把由于输入的一行为Text类型，把其转换为String类型

将字符串以|为标志切割开

user\_career索引为list[10]

1. protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
2. String line = value.toString();
3. String[] list = line.split("\\|");
4. Text career = new Text();
5. career.set(list[10]);
6. context.write(career,value);

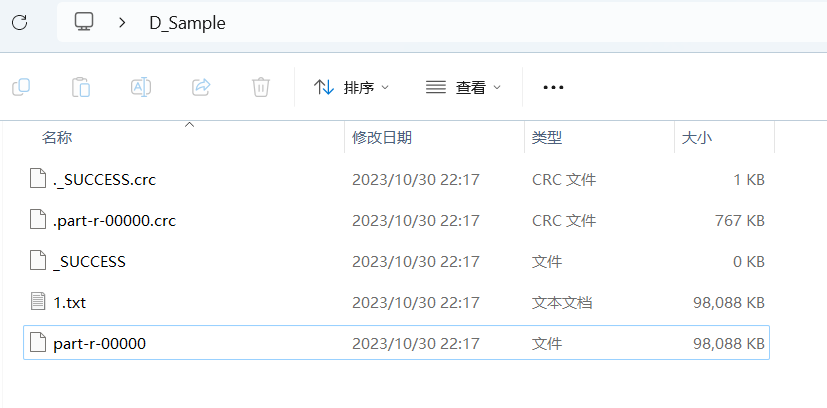
reduce阶段输入为key为user\_career，value为每行的内容

输出key为分层抽样结果

首先随机产生[0,50]的数，如果值小于10则输出，抽样比为20%

1. protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
2. for(Text item:values){
3. int num = (int)(Math.random()\*50);
4. if(num<10){
5. context.write(item,new Text(""));
6. }
7. }
8. }

运行结果：



文本

描述已自动生成

**3.2 数据过滤**

奇异值过滤所使用的属性B为两个属性，分别是longitude和latitude。

longitude的有效范围为[8.1461259, 11.1993265]，

latitude的有效范围为[56.5824856, 57.750511]

在map阶段判断longitude，latitude是否在取值范围内，如果在范围内直接将输入的value作为输出的key，为结果

1. protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
2. String line = value.toString();
3. String[] list = line.split("\\|");
4. double longitude = Double.parseDouble(list[1]);
5. double latitude = Double.parseDouble(list[2]);
6. if (longitude >= 8.1461259 && longitude <= 11.1993265 && latitude >= 56.5824856 && latitude <= 57.750511) {
7. context.write(value, new Text(""));
8. }
9. }

输出：

文本

描述已自动生成

**3.3 数据格式转换与归一化**

数据格式属性涉及到属性user\_birthday和review\_date，这些日期字段可能使用2018-03-21、2018/03/21、March 21, 2019这些格式，转换为哪种格式取决于同学们自己；temperature有华氏与摄氏两种，同样的，目标格式取决于同学们；需要归一化的属性则是rating；

先判断日期为那种格式，如果为1，2种格式就转化为第三种格式输出

1. public static  String DateFormatConversion(String s) throws ParseException {
2. SimpleDateFormat format1 = new SimpleDateFormat("MMMM d,yyyy", Locale.ENGLISH);
3. SimpleDateFormat format2 = new SimpleDateFormat("yyyy-MM-dd");
4. SimpleDateFormat format3 = new SimpleDateFormat("yyyy/MM/dd");
5. if(s.contains(",")){
6. Date date = format1.parse(s);
7. return format3.format(date);
8. }
9. else if(s.contains("-")){
10. Date date = format2.parse(s);
11. return format3.format(date);
12. }
13. else{
14. return s;
15. }

Map 阶段：更新rating的最大最小值

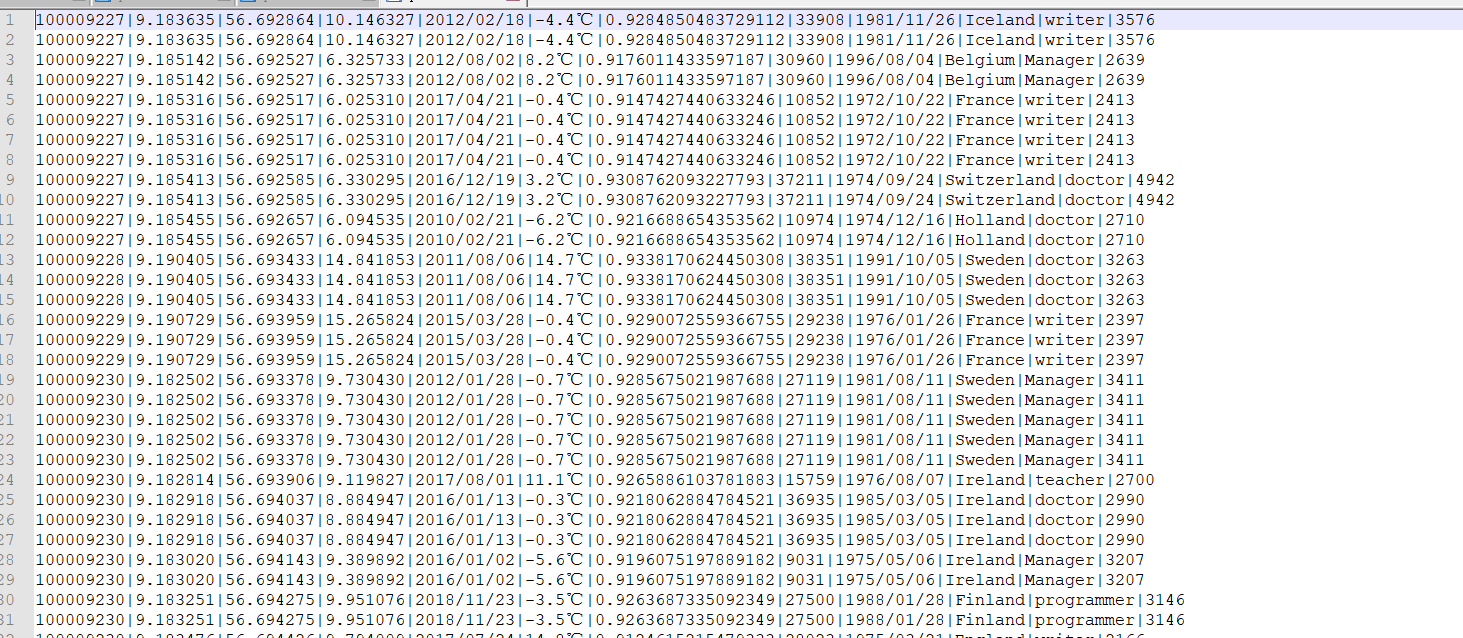
将华氏温度转化为摄氏温度，

转换日期格式

输出key为格式化的内容

Reduce 阶段：如果 rating不为空，进行归一化

rating = (rating-*min*)/(*max*-*min*);  
line = line.replace(list[6],String.*valueOf*(rating));



**3.4 数据清洗（缺失值填充）**

缺失值的属性为rating和user\_income，根据先验知识，rating近似依赖于user\_income、longitude、latitude和altitude，user\_income近似依赖于user\_nationality和user\_career。对rating和user\_income的填充可以利用这些依赖关系；

先进行user\_income的填充

Map 阶段将每个rating不为空的记录中user\_income、longitude、latitude和altitude放入一个列表中

输出的key 为输入的value

每次遇到不空的income值就将其加到sum中，num+1，记录一共多少有多少income不为空

1. String nationality = values[9];
2. String career = values[10];
3. String income = values[11];
4. String nationality\_career = nationality + career;
5. if(!income.contains("?")) {
6. if(map.containsKey(nationality\_career)) {
7. List<Double> list = map.get(nationality\_career);
8. double sum = (double)list.get(0) + Double.valueOf(income);
9. double num = (double)list.get(1) + 1.0;
10. list.set(0, sum);
11. list.set(1, num);
12. } else {
13. List<Double> list = new ArrayList<Double>();
14. list.add(Double.valueOf(income));
15. list.add(1.0);
16. map.put(nationality\_career, list);
17. }
18. }

Map阶段：如果income为空就取对应nationality、career，找到与其nationality、career相等的人，计算平均工资，如果没有就算3000

1. for(Text value : values) {
2. String line = key.toString();
3. String[] temp = line.split("\\|");
4. String income = temp[11];
5. if(income.contains("?")) {
6. String nationality = temp[9];
7. String career = temp[10];
8. String nationality\_career = nationality + career;
9. if(map.containsKey(nationality\_career)) {
10. List<Double> list = map.get(nationality\_career);
11. double incomeToFill = list.get(0)/list.get(1);
12. line = line.replace(income, String.valueOf(incomeToFill));
13. } else {
14. line = line.replace(income, String.valueOf(3000));
15. }
16. context.write(new Text(line), value);
17. } else {
18. context.write(key, value);
19. }
20. }

rating近似依赖于user\_income、longitude、latitude和altitude

利用线性规划对每个值赋予一定的权值，计算出rating

  OLSMultipleLinearRegression regression = new OLSMultipleLinearRegression();

 regression.newSampleData(y, x);

 linear = regression.estimateRegressionParameters();

Map 阶段

 ratingToFill = longitude\*linear[1] + latitude\*linear[2] + altitude\*linear[3] + income\*linear[4] + linear[0];

输出为预测rating

文本

描述已自动生成

3.5如何尽可能地减少MapReduce的轮数

Map输入为Mapper<LongWritable,Text, Text,Text>

将数据抽样，数据过滤，数据标准化和归一化合成一步mapreduce

1. public static final class Sample1Mapper extends Mapper<LongWritable,Text, Text,Text> {
2. @Override
3. protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
4. String line = value.toString();
5. String[] list = line.split("\\|");
6. *// career赋值成对应的值*
7. Text career = new Text();
8. career.set(list[10]);
9. if(!list[6].equals("?")){
10. double rating = Double.parseDouble(list[6]);
11. if(rating>max){
12. max = rating;
13. }
14. if(rating<min){
15. min = rating;
16. }
17. }
18. if(list[5].contains("℉")){
19. float temperature = Float.parseFloat(list[5].substring(0,list[5].length()-1));
20. temperature = (temperature-32)/1.8f;
21. String t = String.format("%.1f",temperature)+"℃";
22. line = line.replace(list[5],t);
23. }
24. try {
25. String review\_date = DateFormatConversion(list[4]);
26. String user\_birthday = DateFormatConversion(list[8]);
27. line = line.replace(list[4],review\_date);
28. line = line.replace(list[8],user\_birthday);
29. } catch (ParseException e) {
30. e.printStackTrace();
31. }
32. double longitude = Double.parseDouble(list[1]);
33. double latitude = Double.parseDouble(list[2]);
34. if (longitude >= 8.1461259 && longitude <= 11.1993265 && latitude >= 56.5824856 && latitude <= 57.750511) {
35. context.write(career,new Text(line));
36. }
37. }
38. }
39. public static final class Sample2Reducer extends Reducer<Text,Text,Text,Text>{
40. @Override
41. protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
42. for(Text value:values){
43. String line = value.toString();
44. *// 切割出一个个单词*
45. *// 注意|前要加转义字符*
46. String[] list = line.split("\\|");
47. if(!list[6].equals("?")){
48. double rating = Double.parseDouble(list[6]);
49. *// Min-Max归一化*
50. rating = (rating-min)/(max-min);
51. line = line.replace(list[6],String.valueOf(rating));
52. }
53. int num = (int)(Math.random()\*50);
54. if(num<10){
55. context.write(new Text(line),new Text(""));
56. }
57. }
59. }
60. }

运行结果：



## 四、实验心得

在实验完成之前学习了hadoop,在linux环境下搭建了hadoop集群，了解了hadoop的搭建过程，并且学会用mapreduce方法进行数据处理。

在实验过程中遇到了很多困难：hadoop与jdk版本不匹配，yarn配置问题等，但是在实验中都一一解决了。