1. 贝叶斯分类器理论介绍

贝叶斯分类的实质其实是概率的计算。假如现在有 3 个类别 A，B，C，并且有测试文档 t，那么我们如何把文档 t 分类？我们需要分别计算文档 t 属于 A，B，C 的概率 p(A|t)、p(B|t)和 p(C|t)，然后比较大小，选择概率最大的那个，我们就认为文档属于该分类。

根据贝叶斯分类的公式：

$$\mathrm{p(C|t)} = \frac{p(t|C)p(C)}{p(t)}$$

公式的意思是：测试文档 t 属于 C 类的概率等于 C 类中 t 出现的概率乘以 C 类出现的概率除以测试文档 t 出现的概率。其中 p(t)即测试文档出现的概率，因为每个测试文档出现的概率都是一样的，都是 1/测试文档总数，因为分类只是比概率大小，所以 p(t)我们可以忽略了。只剩下$p(t|C)p(C)$，其中 p(C)叫做先验概率，这个很好计算，每个类别的 p(C)等于 C 类的文档/总文档个数，这个我们可以先算好。主要就是 p(t|C)的计算，计算公式为：

$$\mathrm{p(t|C)} = \mathrm{p(t1|C)} * \mathrm{p(t2|C)} * \ldots\ldots$$

意思是：$\mathrm{p(t|C)}$等于 C 类中出现 t1，t2，……单词的概率相乘的乘积，其中 t1，t2，……都是文档 t 中的单词。一个类中出现某单词的次数我们可以从训练集中统计到，那么这个概率我们就能计算到。

因为类中某个单词出现的概率都是小数，连乘后可能会导致浮点数溢出，那么我们可以对连乘取 log 操作，因为我们是比较概率大小，取 log 后不会改变大小顺序，而且也避免了浮点数的溢出。那么最后我们其实是计算：

$$\mathrm{Max}[\log p(C) + \sum \log p(t_k|C)]$$

对这个式子取最大值，这个值最大，那么我们就认为文档 t 属于此分类。

需要注意的是，在计算$\mathrm{p(t|C)}$时，如果一个单词没有在 C 类出现过，那么我认为那个概率为 0 吗？因为 0 乘以任何数都为 0，这样会影响整个计算。为了防止这个错误，我们把分子和分母都加 1，即平滑操作，这样就能避免这个错误。

对整个贝叶斯分类举个例子：下表是我的训练文档 t，

ID	文档中的单词	是否属于 C 类

1	Chinese Beijing Chinese	Yes
2	Chinese Chinese Shanghai	Yes
3	Chinese Macao	Yes
4	Tokyo Japan Chinese	No

现在我们有测试文档 5 如下，如何判断文档 5 是否属于 C 类?

5	Chinese Chinese Chinese Tokyo Japan	?

根据贝叶斯分类的原理，我们要计算文档属于 C 类的概率和不属于 C 类的概率，算出来的结果哪个更大，我们就认为结果是什么。

先验概率：

$$\mathrm{p(C)} = \frac{3}{4}$$

$$\mathrm{p}(\bar{C}) = \frac{1}{4}$$

条件概率（平滑操作）：

$$\mathrm{p(Chinese|C)} = \frac{5+1}{8+6} = \frac{3}{7}$$

$$\mathrm{p(Tokyo|C)} = \mathrm{p(Japen|C)} = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\mathrm{p(Chinese}|\bar{C}) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\mathrm{p(Tokyo}|\bar{C}) = \mathrm{p(Japen}|\bar{C}) = \frac{1+1}{3+6} = \frac{2}{9}$$

那么：

$$\mathrm{p(C|t)} = \frac{3}{4} * \left(\frac{3}{7}\right)^3 * \left(\frac{1}{14}\right)^2 \approx 0.0003$$

$$\mathrm{p}(\bar{C}|\mathrm{t}) = \frac{1}{4} * \left(\frac{2}{9}\right)^5 \approx 0.0001$$

因为文档 t 属于 C 类的概率大于文档 t 不属于 C 类的概率，那么我们就将文档 t 划分为 C 类，这就是贝叶斯分类的原理。

2. 数据集说明

数据集的选择，我选择了 NBCorpus\Country 下三个文件数量最多的目录，如下表所示：

类别	目录包含文件数	训练集	测试集

AUSTR	305	229	76
BRAZ	200	150	50
CANA	263	197	66

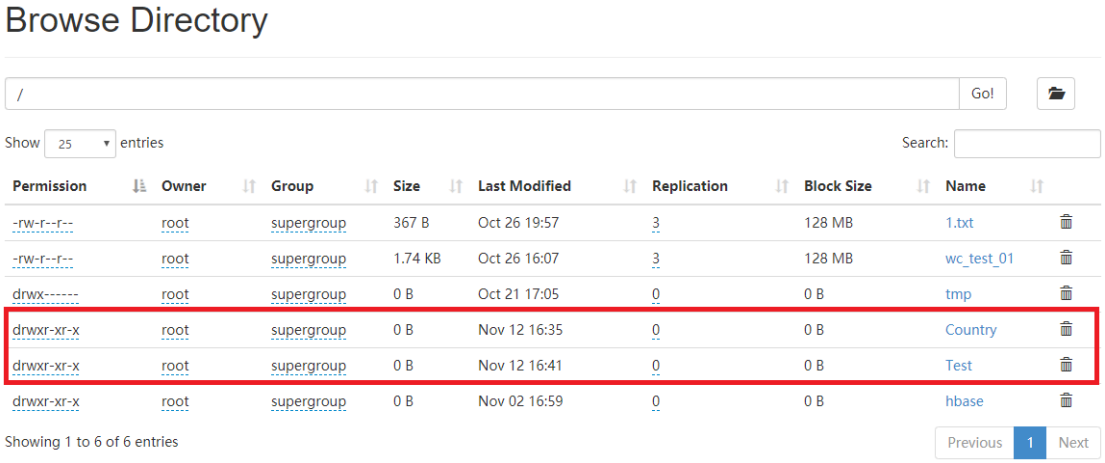
其中，测试集文件的数量约占该类总文件数量的 1/4。

首先将文件上传到 HDFS 上，命令：

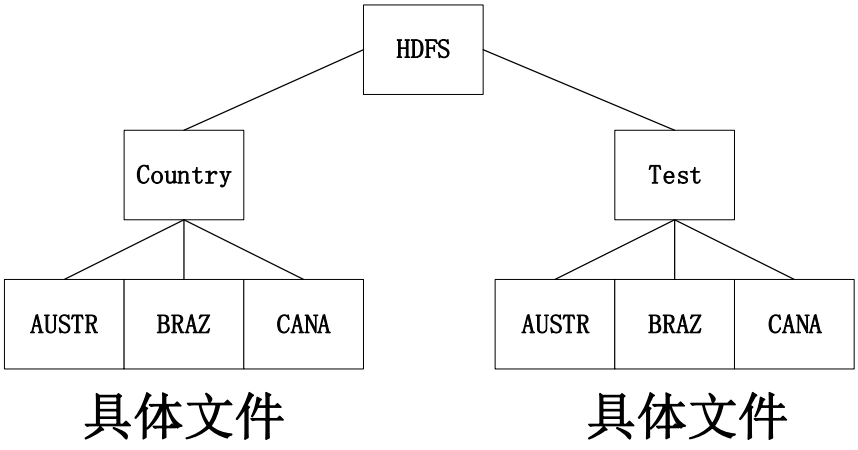
hadoop fs -put /root/Desktop/Country/ hdfs://hadoop01:9000/Country

hadoop fs -put /root/Desktop/Test/ hdfs://hadoop01:9000/Test

上传成功如图所示：



整个 HDFS 中的目录结构如图：



在程序运行过程中，除了计算文件外，没有产生任何的中间文件。

3. MapReduce 算法设计

我使用了 3 个 Job 即 3 个 MapReduce 程序，均使用了自定义的 InputFormat，其中 Job2 和 Job3 使用的是同一个 InputFormat。

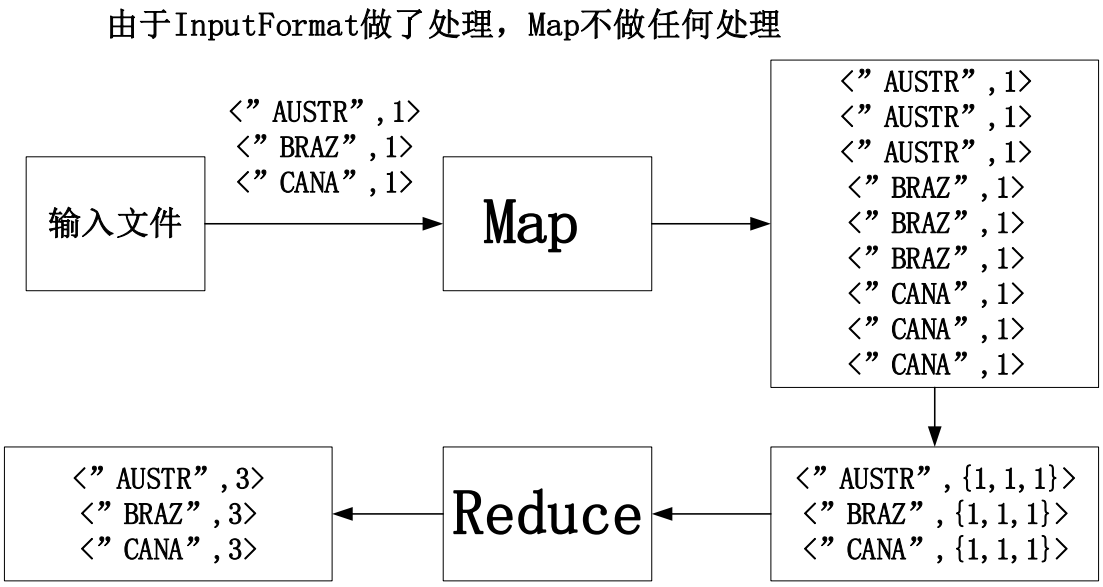
3.1 ClassFileSumJob

此 Job 的作用是统计每个类别有多少数据，用于计算先验概率。使用的是 ClassNameInputFormat 和 ClassNameRecordReader 作为 Map 的输入读写。

Job 的输入：hdfs://hadoop01:9000/Country（训练集的总目录）

Job 的输出：hdfs://hadoop01:9000/Bayes/ClassSum

数据流图如下图所示：



Map 的输入：<“AUSTR”,1>，即类名和个数 1。

Map 的输出：直接输出，不做任何处理。

Reduce 的输入：<“AUSTR”,{1,1,1}>，即将同 key 的数据 group 起来。

Reduce 的输出：即最后的结果，类名加总数。

编程技巧：

在阅读 TextInputFormat 的源码的基础上，弄清楚 InputFormat 和 RecordReader 的流程，虽然我的 Job 的输入目录是：

hdfs://hadoop01:9000/Country

但是这个目录被分成了 3 个 InputSplit，即自动划分了子目录，这样就比较好处理了。学会看日志的输出，除了 Main 函数以外，自定义的 System.out 输出在/logs/userLogs/application_id/contain_id 中，学会看日志才能找到问题和错误！

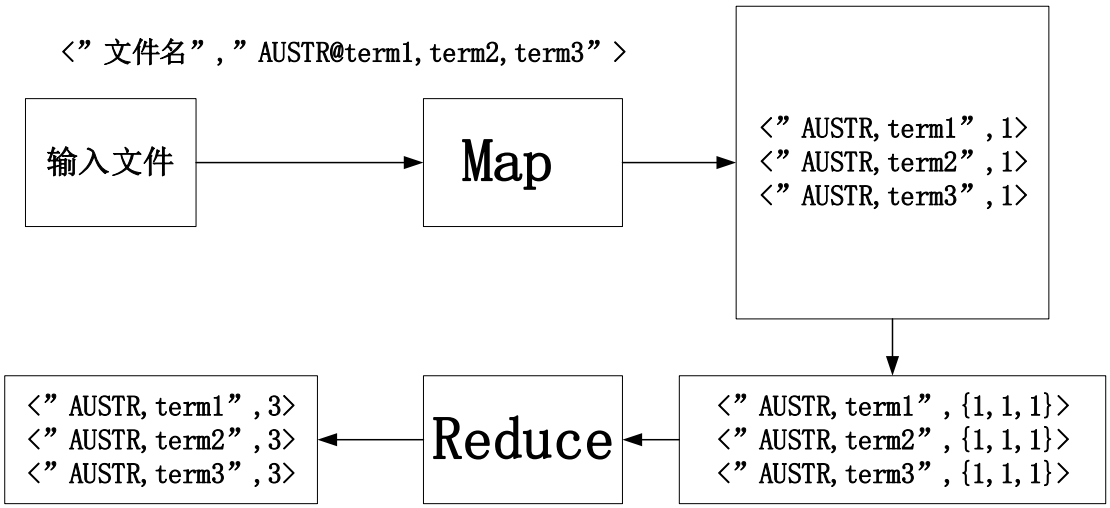
3.2 TermStatisticJob

此 Job 的作用是统计每个类别中的单词的个数，用于计算条件概率。使用的是 TermStatisticInputFormat 和 TermStatisticRecordReader 作为 Map 的输入读写。

Job 的输入：hdfs://hadoop01:9000/Country（训练集的总目录）

Job 的输出：hdfs://hadoop01:9000/Bayes/Term

数据流图如下图所示：



Map 的输入：<“文件名”,“AUSTR@term1,term2,term3”>，即文件名和当前的类名加分隔符“@”，再加上当前文档的全部单词，用“,”分隔（输入的 key 是多余的，但是为了第 3 个 Job 能使用，就加上了）。

Map 的输出：<“AUSTR,term1”,1>，即类名加单词作为联合 key，然后是数量 1

Reduce 的输入：<“AUSTR,term1”,{1,1,1}>

Reduce 的输出：<“AUSTR,term1”,3>，即最后的统计结果

编程技巧：

使用 Scanner 类按行读取整个文档的字符串，并用 StringBuilder 类累加字符串，最后 toString()转化为字符串。

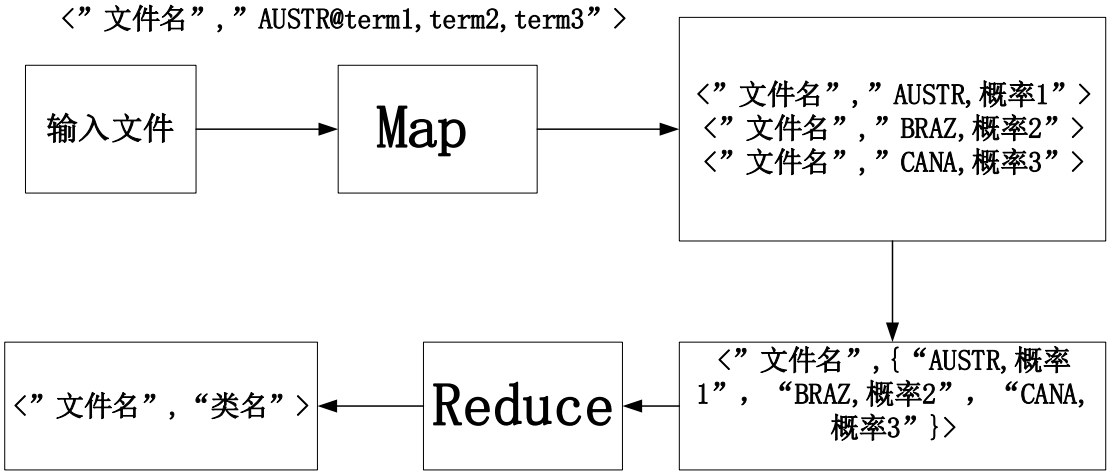
3.3 PreditionJob

此 Job 的作用是预测测试文档。使用的是 TermStatisticInputFormat 和 TermStatisticRecordReader 作为 Map 的输入读写（和上一个 Job 使用的同一个 InputFormat）。

Job 的输入：hdfs://hadoop01:9000/Test（测试集的总目录）

Job 的输出：hdfs://hadoop01:9000/Bayes/Predict

数据流图如下图所示：



Map 的输入：<“文件名”,“AUSTR@term1,term2,term3”>，即文件名和当前的类名加分隔符“@”，再加上当前文档的全部单词，用“,”分隔。

Map 的输出：<“文件名”,“类,概率”>，即分别计算该文件属于 3 个类的概率。

Reduce 的输入：<“文件名”,{3 个类的概率}>

Reduce 的输出：取最大值操作，输出最大概率的<“文件名”,“类名”>

编程技巧：

保存测试集中所有的单词集合时，使用 Set 集合，因为 Set 集合可以自动去重复。最后的计算结果用一个 Map 集合保存即可。

4. 程序运行说明

整个环境的搭建：

节点数	共 2 台节点，都是 CentOS 系统，1 台 Master，1 台 Slave

JDK	Java 1.8
Hadoop	版本 2.8.4
开发	基于 Maven 的 Java 程序

ClassFileSumJob 程序运行图：

```
File  Edit  View  Search  Terminal  Help
[root@hadoop01 sbin]# hadoop jar /root/Desktop/bayes.jar zhangchao.bayes.job.ClassFileSumJob
18/11/14 00:48:03 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
18/11/14 00:48:04 INFO client.RMProxy: Connecting to ResourceManager at hadoop01/192.168.8.2:8032
18/11/14 00:48:05 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
tion with ToolRunner to remedy this.
18/11/14 00:48:05 INFO input.FileInputFormat: Total input files to process : 3
18/11/14 00:48:05 INFO mapreduce.JobSubmitter: number of splits:3
18/11/14 00:48:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1542180877510_0002
18/11/14 00:48:06 INFO impl.YarnClientImpl: Submitted application application_1542180877510_0002
18/11/14 00:48:06 INFO mapreduce.Job: The url to track the job: http://hadoop01:8088/proxy/application_1542180877510_0002/
18/11/14 00:48:06 INFO mapreduce.Job: Running job: job_1542180877510_0002
18/11/14 00:48:21 INFO mapreduce.Job: Job job_1542180877510_0002 running in uber mode : false
18/11/14 00:48:21 INFO mapreduce.Job:  map 0% reduce 0%
18/11/14 00:48:37 INFO mapreduce.Job:  map 33% reduce 0%
18/11/14 00:48:38 INFO mapreduce.Job:  map 100% reduce 0%
18/11/14 00:48:50 INFO mapreduce.Job:  map 100% reduce 100%
18/11/14 00:48:51 INFO mapreduce.Job: Job job_1542180877510_0002 completed successfully
18/11/14 00:48:51 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=6571
                FILE: Number of bytes written=643879
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=295
                HDFS: Number of bytes written=28
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=3
                Launched reduce tasks=1
                Other local map tasks=3
                Total time spent by all maps in occupied slots (ms)=41768
                Total time spent by all reduces in occupied slots (ms)=10295
                Total time spent by all map tasks (ms)=41768
                Total time spent by all reduce tasks (ms)=10295
                Total vcore-milliseconds taken by all map tasks=41768
                Total vcore-milliseconds taken by all reduce tasks=10295
                Total time spent by all map tasks (ms)=41768
                Total time spent by all reduce tasks (ms)=10295
                Total vcore-milliseconds taken by all map tasks=41768
                Total vcore-milliseconds taken by all reduce tasks=10295
                Total megabyte-milliseconds taken by all map tasks=42770432
                Total megabyte-milliseconds taken by all reduce tasks=10542080
        Map-Reduce Framework
                Map input records=576
                Map output records=576
                Map output bytes=5413
                Map output materialized bytes=6583
                Input split bytes=295
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=6583
                Reduce input records=576
                Reduce output records=3
                Spilled Records=1152
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
                GC time elapsed (ms)=977
                CPU time spent (ms)=2360
                Physical memory (bytes) snapshot=687865856
                Virtual memory (bytes) snapshot=8240300032
                Total committed heap usage (bytes)=386990080
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=28
[root@hadoop01 sbin]#
```

TermStatisticJob 程序运行图：

```
File  Edit  View  Search  Terminal  Help
root@hadoop01 sbin]# hadoop jar /root/Desktop/bayes.jar zhangchao.bayes.job.TermStatisticJob
18/11/14 04:44:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
18/11/14 04:44:17 INFO client.RMProxy: Connecting to ResourceManager at hadoop01/192.168.8.2:8032
18/11/14 04:44:18 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
tion with ToolRunner to remedy this.
18/11/14 04:44:19 INFO input.FileInputFormat: Total input files to process : 3
18/11/14 04:44:19 INFO mapreduce.JobSubmitter: number of splits:3
18/11/14 04:44:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1542194304292_0002
18/11/14 04:44:19 INFO impl.YarnClientImpl: Submitted application application_1542194304292_0002
18/11/14 04:44:19 INFO mapreduce.Job: The url to track the job: http://hadoop01:8088/proxy/application_1542194304292_0002/
18/11/14 04:44:19 INFO mapreduce.Job: Running job: job_1542194304292_0002
18/11/14 04:44:34 INFO mapreduce.Job: Job job_1542194304292_0002 running in uber mode : false
18/11/14 04:44:34 INFO mapreduce.Job:  map 0% reduce 0%
18/11/14 04:45:15 INFO mapreduce.Job:  map 100% reduce 0%
18/11/14 04:45:35 INFO mapreduce.Job:  map 100% reduce 100%
18/11/14 04:45:36 INFO mapreduce.Job: Job job_1542194304292_0002 completed successfully
18/11/14 04:45:36 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1819762
                FILE: Number of bytes written=4270281
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=799482
                HDFS: Number of bytes written=356651
                HDFS: Number of read operations=588
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=3
                Launched reduce tasks=1
                Other local map tasks=3
                Total time spent by all maps in occupied slots (ms)=107673
                Total time spent by all reduces in occupied slots (ms)=13500
                Total time spent by all map tasks (ms)=107673
                Total time spent by all reduce tasks (ms)=13500
                Total vcore-milliseconds taken by all map tasks=107673
                Total vcore-milliseconds taken by all reduce tasks=13500
                Total megabyte-milliseconds taken by all map tasks=110257152
                Total megabyte-milliseconds taken by all reduce tasks=13824000
        Map-Reduce Framework
                Map input records=576
                Map output records=97215
                Map output bytes=1625326
                Map output materialized bytes=1819774
                Input split bytes=295
                Combine input records=0
                Combine output records=0
                Reduce input groups=22636
                Reduce shuffle bytes=1819774
                Reduce input records=97215
                Reduce output records=22636
                Spilled Records=194430
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
                GC time elapsed (ms)=1787
                CPU time spent (ms)=8600
                Physical memory (bytes) snapshot=603193344
                Virtual memory (bytes) snapshot=8243003392
                Total committed heap usage (bytes)=387428352
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=799187
        File Output Format Counters
                Bytes Written=356651
[root@hadoop01 sbin]#
```

PreditionJob 程序运行图：

三个 Job 都使用了 3 个 Map 任务和 1 个 Reduce 任务，因为 Map 任务的数量和 InputSplit 有关，由于 Country 目录下有 3 个子目录，程序运行时产生了 3 个进程分别处理 3 个 InputSplit。因为没有写 Partitioner 函数，所以默认就 1 个 Reducer，最后只输出到一个文件。

5. 实验结果分析

三个 Job 产生的结果文件为 classSum、term 和 predict（都重命名过），分别为 ClassFileSumJob、TermStatisticJob 和 PreditionJob 计算出的文件，放在附件中。

计算准确率：

TP：每个文档的真实类别为 C 且预测值也为 C

FN：每个文档的真实类别为 C 但预测类型不为 C

FP：每个文档的真实类别不为 C 但是最后的预测结果为 C

TN：每个文档的真实类别不为 C 且最后的判断结果也不为 C

准确率的计算：P = TP/(TP+FP)

召回率的计算：R = TP/(TP+FN)

调和平均值 F1：F1 = 2PR/(P+R)

类 AUSTR：

TP	66
FN	10
FP	7
TN	109
P	90.41%
R	86.84%
F1	88.59%

类 BRAZ：

TP	49
FN	1
FP	3
TN	139
P	94.23%

R	98.00%
F1	96.08%

类 CANA：

TP	58
FN	8
FP	9
TN	117
P	86.57%
R	87.88%
F1	87.22%

总结：类 AUSTR 的分类准确率为 88.59%，类 BRAZ 和类 CANA 分类的准确率分别为 96.08%和 87.22%。分类的准确率挺高的。

整个实验的难度就在于自定义 InputFormat，搭建集群的时候可能会有点小问题，但是通过百度都能解决，写 InputFormat 的时候，要多输出日志，然后去 logs 文件下看日志文件，然后进一步分析，就能知道 split 当前处理的是什么目录。其它的思路都是按照老师的 PPT 上的设计思路走的，就是 key 和 value 的设计。