華中科技大學

研究生（Hadoop/MapReduce 编程模型与应用）课程报告

题 目：基于 MapReduce 框架实现朴素贝叶斯分类

学　　　　号：	M201977184
姓　　　　名：	程诚
专　　　　业：	计算机技术
指 导 教 师：	辜希武　讲师
院（系、所）：	计算机科学与技术学院

目　录

1 朴素贝叶斯算法理论介绍

朴素贝叶斯算法（Naive Bayes）是一种属于监督学习的分类模型生成算法，实现简单，没有迭代，基于贝叶斯理论，其中的“朴素”指的是假设每个样本的各个特征之间强相互独立，即每个特征与其他特征都不相关。显然这个假设在实际中不可能为真，但朴素贝叶斯算法在实际运用中依旧有不错的分类效果，是个十分经典的机器学习算法。

朴素贝叶斯分类的内容可以简单表示如下：

1) 设特征集合为$T=\{t_1,t_2\cdots t_n\}$，而已知$X=\{t_1,t_2\cdots t_s\}$是一个待分类项

2) 已知类别集合$C=\{C_1,C_2\cdots C_m\}$

3) 计算条件概率$p(C_1\,|\,X),p(C_2\,|\,X)\cdots p(C_m\,|\,X)$，即已知待分类项 X，求其属于$C_1,C_2\cdots C_m$的概率

4) 设条件概率最大值为$p(C_k\,|\,X)=\max\{p(C_1\,|\,X),p(C_2\,|\,X)\cdots p(C_m\,|\,X)\}$，则认为$X$属于类别$C_k$

朴素贝叶斯分类重点在于计算条件概率$p(C_i\,|\,X)$，由贝叶斯公式可得：

$$p(C_i\,|\,X)=\frac{p(X\,|\,C_i)\cdot p(C_i)}{p(X)} \tag{1.1}$$

由于最后是取最大概率$p(C_k\,|\,X)$，而每个条件概率$p(C_i\,|\,X)$在式(1.1)中分母相同，故有：

$$p(C_i\,|\,X)\propto p(X\,|\,C_i)\cdot p(C_i) \tag{1.2}$$

即计算条件概率$p(C_i\,|\,X)$可以转换成计算概率$p(X\,|\,C_i)\cdot p(C_i)$，又由于假设各个特征之间强相互独立，故有：

$$p(X\,|\,C_i)=p(t_1,t_2\cdots t_s\,|\,C_i)=\prod_{j=1}^{s}p(t_j\,|\,C_i) \tag{1.3}$$

即：

$$p(C_i\,|\,X)\propto p(C_i)\times\prod_{j=1}^{s}p(t_j\,|\,C_i) \tag{1.4}$$

其中先验概率 $p(C_i)$ 为每个类别出现的概率，条件概率 $p(t_j \mid C_i)$ 为特征 t_j 在类别 C_i 中出现的概率。

建立朴素贝叶斯分类器的过程实际上就是利用训练集求每个类别出现的概率 $p(C_i)$ 和每个特征 t_j 在各个类别中出现的概率 $p(t_j \mid C_i)$ 的过程，其中 $1 \le i \le m$，$1 \le j \le n$，对待分类项 X 的分类过程，实际上就是求其在各个类别中出现的概率，取最大概率对应的类别作为其分类类别。

2 MapReduce 算法设计

本次实验的主要内容是使用朴素贝叶斯算法针对文档训练集建立文档分类模型，并使用训练后的模型对文档测试集进行分类，最后计算并输出分类结果的各项评价指标。

模型建立分为三个部分，第一个部分统计各类文档中各个单词的数目，第二个部分统计类文档的单词总数和文档总数，用于之后计算各类文档出现的概率，即先验概率 $p(C_i)$，第三部分计算各类文档各个单词出现的概率，即条件概率 $p(t_j \mid C_i)$。

其中整体实验分为数据预处理、建立分类器、分类测试三个部分，其中建立分类器所使用的 MapReduce 程序分为三个 Job，具体内容如下所示：

2.1 数据预处理

为了使得各类文档的先验概率随机，在进行 MapReduce 程序设计之前，将原始数据集的所有文档名中加上文档类别，使得修改后的文件名为“文档类别—文档 ID”的格式，然后将整个数据集按 1:1 的比例随机分为训练集和测试集。

在 Hadoop 集群中使用 MapReduce 程序处理小文件时，由于每个小文件都有占有一个 block，都会形成一个 split，都需要一个 MapTask 和 ReduceTask 来进行处理，因此会产生大量的 Task，整个 MapReduce 程序会在资源的申请和释放上花费大量的时间，同时大量 block 的原数据会增加 NameNode 的内存压力。

为了消除训练集中小型文档对 Hadoop 集群运行 MapReduce 程序产生的负面影响，提高生成分类模型的效率，在建立分类器模型之前，单机环境下离线运行单个 MapReduce Job 将训练集聚集生成单个 SequenceFile 文件，然后上传到 HDFS 集群上，便于之后的 Job 读取。在运行 MapReduce 程序生成 SequenceFile 文件时，使用自定义的 CombineFileInputFormat 输入格式整合所有小文件 block 尽可能形成少量的 split，减少 MapTask 的数量，提高整合效率。

2.2 MapReduce Job1

Job1：统计各类文档中各个单词的数量

输入路径：训练集 SequenceFile 文件路径

输出路径：自定义 Job1 输出路径，如：/src/Output/Job1/

备注：输入的文件为 SequenceFile 文件格式，使用 SequenceFileInputFormat 作为输入格式

Job1Mapper：

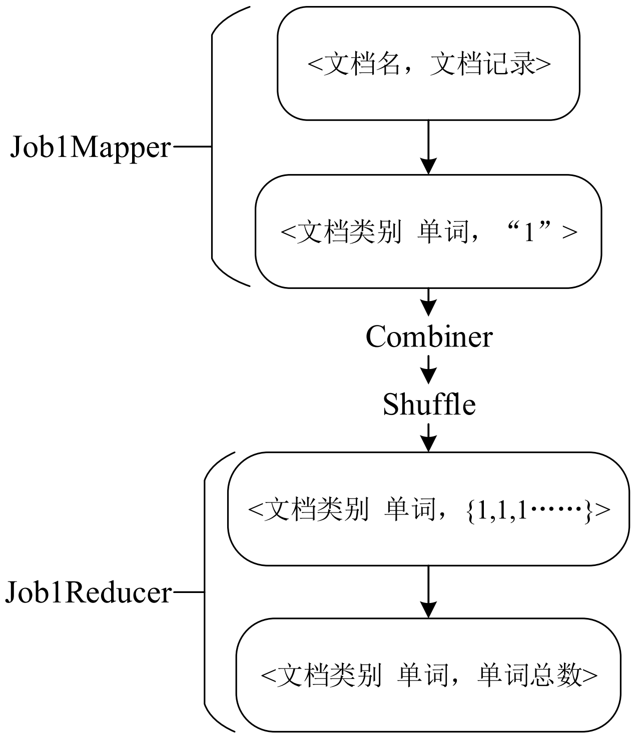
输入：<文档名 ，整个文档记录>，**输出**：<文档类别　单词 ，"1">

备注：其中“文档类别”通过划分文件名取其前缀获得，“单词”通过对文档记录划分可得，文档类别和单词之间使用制表符“\t”划分

Job1Combiner：即 Reducer

Job1Reducer：

输入：<文档类别　单词，{1,1,1,1,1,1……}>，**输出**：<文档类别　单词，单词总数>



2.3 MapReduce Job2

Job2：统计出各类文档中单词总数和各类文档总数

输入路径：训练集 SequenceFile 文件路径

输出路径：自定义 Job2 输出路径，如：/src/Output/Job2/

备注：输入的文件为 SequenceFile 文件格式，使用 SequenceFileInputFormat 作为输入格式

Job2Mapper：

输入：<文档名 ，整个文档>，**输出**：<文档类别 ，本文档中单词总数>

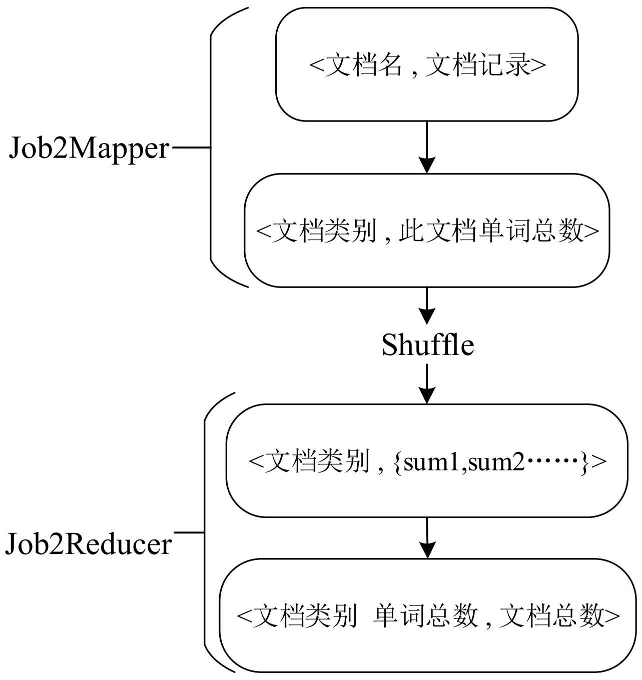
Job2Combiner：No Combiner

备注：此种场景不能使用 Combiner

Job2Reducer：

输入：<文档类别，{sum1，sum2……}>，**输出**：<文档类别 本类文档单词总数，本类文档个数>

备注：其中“本类文档个数”由 sum 的个数来计算，每有一个 sum 则说明有一个对应的文档属于当前类别，因为是每个一个文档作为一个记录进行处理



Job2Mapper
<文档名 , 文档记录>
<文档类别 , 此文档单词总数>
Shuffle
Job2Reducer
<文档类别 , {sum1,sum2……}>
<文档类别 单词总数 , 文档总数>

2.4 MapReduce Job3

Job3：计算各个类别文档中各种单词出现的条件概率

输入路径：Job1 的输出路径，即获取各个类别中各个单词的总数

输出路径：自定义 Job3 输出路径，如：/src/Output/Job3/

备注：MapReduce 程序会自动忽略以下划线"_"开头的文件，不用担心_success 文件，本来_success 文件也是空文件

Job3Mapper：

输入：<记录偏移量 LongWritable，<文档类别　单词　单词总数>>

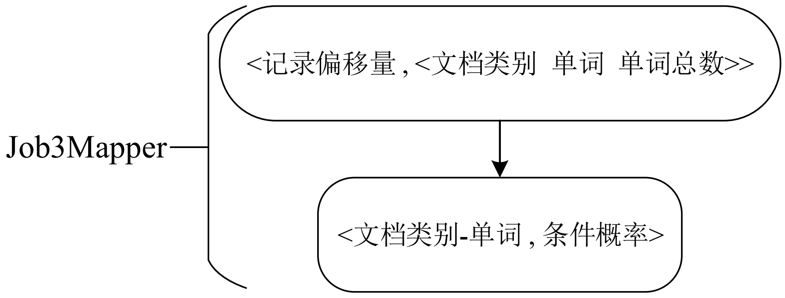
输出：<文档类别-单词，条件概率>

描述：通过读取 Job2 的输出文档获得各个类别对应的单词总数，通过读取 Job1 的输出文档获取各个类别各个单词的总数，由此来计算各个类别中各个单词的出现的条件概率

备注：在 Job3 的 Mapper 类中重写 setup 方法，每次启动 MapTask 时读取 Job2 的输出文件，在 Mapper 中读取对应文档类别的单词总数，计算单词出现概率

Job3Combiner：No Combiner

Job3Reducer：No Reducer



3 源代码清单

3.1 Job1.java

```
package com.TomAndersen.hadoop.BayesClassification;

import
com.TomAndersen.hadoop.HDFSTools.CombineSmallfileInputFormat;
import com.TomAndersen.hadoop.HDFSTools.WholeFileInputFormat;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import
org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;

import java.io.IOException;

/**
 * @Author TomAndersen
 * @Version
 * @Date 2019/11/4
 * @Description : Job1 统计出各类文档中各种单词的数目
 * Job1:
 * 将一个文档作为一个记录处理
 * 输入路径：训练集
 * 输出路径：自定义 Job1 输出路径，如：/src/OutPut/Job1/
 * Mapper:
 * 输入：<文档名,整个文档记录>，输出：<文档类别 单词，1>
 * 其中文档类别通过读取文件名前缀获得，单词通过对 text 划分可得
 * Conbiner: 即 Reducer
 * Reducer:
 * 输入: <文档类别 单词, {1,1,1,1,1,1……}>, 输出: <文档类别 单词, sum>
 */

public class Job1 extends Configured implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        // 重写 run 方法, 外部使用 ToolRunner 启动 Job
        // 第一个参数为训练集, 第二个参数为输出路径
        String InputPath = args[0];
        String OutputPath = args[1];
        Configuration configuration = new Configuration();
        Job         job         =         Job.getInstance(configuration,
this.getClass().getName());
        job.setInputFormatClass(SequenceFileInputFormat.class);//使用
SequenceFile 作为输入
        job.setJarByClass(Job1.class);//设置主类
        job.setMapperClass(Job1Mapper.class);// 设置 Mapper
        job.setCombinerClass(Job1Reducer.class);// 设置 Combiner
        job.setReducerClass(Job1Reducer.class);// 设置 Reducer

        //默认使用的 OutputFormat 是 TextOutputFormat, 使用时一定要指定 Map
输出的 Key 和 Value 类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //设置 job 输出的 key 类型
        job.setOutputKeyClass(Text.class);
        //设置 job 输出的 value 类型
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(InputPath));// 添
加输入路径
        FileOutputFormat.setOutputPath(job, new Path(OutputPath));//
设置输出路径
        return job.waitForCompletion(true) ? 0 : 1;
    }

    //Mapper
    public    static    class    Job1Mapper    extends    Mapper<Text,
BytesWritable, Text, IntWritable> {
        private final static Text KEYOUT = new Text();//预定义 KeyOut
        private    final    static    IntWritable    VALUEOUT    =    new
IntWritable(1);//预定义 ValueOut

        @Override
        public  void  map(Text  KeyIn,  BytesWritable  ValueIn,  Context
context)
                throws IOException, InterruptedException {//KeyIn 是文档
名，ValueIn 是文档内容
            // Text, BytesWritable, Text, IntWritable 输入输出键值对类型
            // 将文档内容按照回车分割，即分割成一个个单词
            String[]                contents            =            new
String(ValueIn.copyBytes()).split("\r\n|\n|\r");
            // 以各种系统的换行形式作为分隔符进行切分
            // 获取文档类别，文档名中已有类别戳
            String fileClass = KeyIn.toString().split("-")[0];

            for (String word : contents) {
                // 对每个单词加上类别名
                KEYOUT.set(fileClass + "\t" + word);//设置 KeyOut，使用制
表符来间隔文档类别和单词
                //输出格式为<类别名 单词，"1">
                context.write(KEYOUT, VALUEOUT);
            }
        }
    }

    //Reducer
    public    static    class    Job1Reducer    extends    Reducer<Text,
IntWritable, Text, IntWritable> {
        private    final    static    IntWritable    VALUEOUT    =    new
IntWritable();//预定义 ValueOut

        @Override
        public    void    reduce(Text    KeyIn,    Iterable<IntWritable>
ValuesIn, Context context)
                throws IOException, InterruptedException {

            int sum = 0;
            for (IntWritable Value : ValuesIn) {
                sum  +=  Value.get();//  要使用 combiner 就不能只是单纯的计数
+1，而应该是取值相加
            }
            // 输出<类别名  单词，单词总数>
            VALUEOUT.set(sum);
```

```
                context.write(KeyIn, VALUEOUT);
            }
        }
    }
```

3.2 Job2.java

```java
package com.TomAndersen.hadoop.BayesClassification;

import
com.TomAndersen.hadoop.HDFSTools.CombineSmallfileInputFormat;
import com.TomAndersen.hadoop.HDFSTools.WholeFileInputFormat;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;

import java.io.IOException;

/**
 * @Author TomAndersen
 * @Version
 * @Date 2019/11/4
 * Job2:
 * 输入路径：训练集
 * 输出路径：自定义 Job2 输出路径，如：/src/OutPut/Job2/
 * Mapper:
 * 输入：<文档名,整个文档>，输出：<文档类别，本文档中单词总数 sum>
 * Reducer:
 * 输入：<文档类别，{sum1，sum2……}>，输出：<文档类别 本类文档单词总数，本类文
档个数>
 * 其中"本类文档个数"由 sum 的个数来计算，每有一个 sum 则说明有一个对应的文档属
于当前类别，因为是一个文档作为一个记录进行处理
 */
public class Job2 extends Configured implements Tool {
    @Override
    public int run(String[] args) throws Exception {
        // 重写 run 方法，外部使用 ToolRunner 启动 Job

        // 第一个参数为训练集，第二个参数为输出路径
        String InputPath = args[0];
        String OutputPath = args[1];

        Configuration configuration = new Configuration();
        Job job = Job.getInstance(configuration,
this.getClass().getName());

        // 设置输入格式
        //job.setInputFormatClass(WholeFileInputFormat.class);//使用
整个文件内容作为记录输入

//job.setInputFormatClass(CombineSmallfileInputFormat.class);//使用
整合小文件 block 的方式输入
        job.setInputFormatClass(SequenceFileInputFormat.class);//使用
SequenceFile 作为输入

        job.setJarByClass(Job2.class);//设置主类

        job.setMapperClass(Job2Mapper.class);// 设置 Mapper
        // 不能设置 combiner
        job.setReducerClass(Job2Reducer.class);// 设置 Reducer
        job.setNumReduceTasks(1);// 设置单个 ReduceTask，确保输出文件只有
一个，便于后面的 Job 进行读取

        // 默认使用的 OutputFormat 是 TextOutputFormat，使用时一定要指定 Map
和 Reduce 输出的 Key-Value 类型
        // 设置 Map 输出 Key Value 类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //设置 job 输出的 key 类型
        job.setOutputKeyClass(Text.class);
        //设置 job 输出的 value 类型
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(InputPath));// 添
加输入路径
        FileOutputFormat.setOutputPath(job, new Path(OutputPath));//
设置输出路径
```

```
        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static class Job2Mapper extends Mapper<Text,
BytesWritable, Text, IntWritable> {
        private final static Text KEYOUT = new Text();//预定义
KeyOut，避免多次 New
        private final static IntWritable VALUEOUT = new
IntWritable();//预定义 ValueOut，避免多次 New

        @Override
        public void map(Text KeyIn, BytesWritable ValueIn, Context
context)
                throws IOException, InterruptedException {
            // 对整个文档内容按回车、换行、空字符进行分割，分割产生的碎片数量即
为本文档中单词总数
            String[] contents = new
String(ValueIn.copyBytes()).split("\r\n|\n|\r");
            // 本文档单词总数作为 ValueOut
            VALUEOUT.set(contents.length);

            // 文件名中包含类别信息，需要分离出来
            // 获取文档类别，文档名中已有类别戳
            String fileClass = KeyIn.toString().split("-")[0];
            // 文档类别作为 KeyOut
            KEYOUT.set(fileClass);
            context.write(KEYOUT, VALUEOUT);

        }
    }


    public static class Job2Reducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
        private final static Text KEYOUT = new Text();// 预定义 KeyOut
        private final static IntWritable VALUEOUT = new
IntWritable();// 预定义 ValueOut

        //不能使用 combiner
        public void reduce(Text KeyIn, Iterable<IntWritable>
ValuesIn, Context context)
                throws IOException, InterruptedException {
```

```
            int sumOfWords = 0;// 本类别文档单词总数
            int sumOfFiles = 0;// 本类别文档总数

            for (IntWritable value : ValuesIn) {
                //每有一个 value 则有一个此类别文档
                sumOfWords += value.get();
                sumOfFiles += 1;
            }
            // KeyOut 为<文档类别 本类单词总数>,KeyIn 为文档类别
            String fileClass = KeyIn.toString();
            // 以制表符 \t 作为分隔符
            KEYOUT.set(fileClass + "\t" + sumOfWords);
            // ValueOut 为本类文档个数
            VALUEOUT.set(sumOfFiles);
            // 写入
            context.write(KEYOUT, VALUEOUT);
        }
    }
}
```

3.3 Job3.java

```java
package com.TomAndersen.hadoop.BayesClassification;

import com.TomAndersen.hadoop.HDFSTools.BayesTools;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;

import java.io.IOException;
import java.util.HashMap;

/**
 * @Author TomAndersen
 * @Version
 * @Date 2019/11/4
 * Job3:
 * 输入路径：Job1 的输出路径（MapReduce 程序会自动忽略以下划线"_"开头的文件，
不用担心_success 文件）
 * 输出路径：自定义 Job3 输出路径，如：/src/OutPut/Job3/
 * Mapper:
 * 输入：< LongWritable，<文档类别 单词 sum>>
 * 输出：<文档类别 单词，条件概率 p>
 * 在 Mapper 中读取对应文档类别的单词总数，计算单词出现概率
 * No Reducer
 */
public class Job3 extends Configured implements Tool {

    // public static HashMap fileClassToSumOfWords = null;// 存放文档
类别到单词总数的映射，键值对类型为<String,String>
    // public static HashMap fileClassToSumOfFiles = null;// 存放文档
类别到文档总数的映射

    @Override
    public int run(String[] args) throws Exception {
        // 获取配置信息
        Configuration configuration = new Configuration();
        //下面是对 Job3 的配置
        String InputPath = args[0]; // 输入路径为 Job1 的输出路径
        String OutputPath = args[1];// 输出路径
        // 获取 Job 实例
        Job job = Job.getInstance(configuration,
this.getClass().getName());

        // 设置输入格式
        job.setInputFormatClass(TextInputFormat.class);
        // 设置 Job3 的主类
        job.setJarByClass(Job3.class);
        // 设置 Mapper
        job.setMapperClass(Job3Mapper.class);
        // No Combiner,no Reducer，使用默认的 Reducer
        job.setNumReduceTasks(1);// 设置单个 ReduceTask，确保输出文件只有
一个，便于后面的 Job 进行读取
        // 设置 Mapper 的输出 Key-value 类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(DoubleWritable.class);

        // 添加输入路径
        FileInputFormat.addInputPath(job, new Path(InputPath));
        // 设置输出路径
        FileOutputFormat.setOutputPath(job, new Path(OutputPath));
        // 返回执行状态
        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static class Job3Mapper extends Mapper<LongWritable,
Text, Text, DoubleWritable> {

        private final static Text KEYOUT = new Text();//预定义
KeyOut，避免多次 New
        private final static DoubleWritable VALUEOUT = new
DoubleWritable();//预定义 ValueOut，避免多次 New
        private static HashMap fileClassToSumOfWords = null;// 存放文
档类别到单词总数的映射，键值对类型为<String,String>

        // 重载 setup 方法，在 setup 方法中初始化 fileClassToSumOfWords，
setup 方法每次启动 Task 时由 run 方法调用，只执行一次
        @Override
        public void setup(Context context) throws IOException,
InterruptedException {
            super.setup(context);
            Configuration configuration = context.getConfiguration();
            HashMap[] myMaps =
BayesTools.getKeyValuesByReadFile(JobsInitiator.Job2_OutputPath +
"part-r-00000",
                    configuration, "\t");
            fileClassToSumOfWords = myMaps[0];
        }

        // 输入：<NullWritable，<文档类别   单词   单词总数>>
        // 输出：<文档类别 单词   条件概率>
        @Override
        public void map(LongWritable KeyIn, Text ValueIn, Context
context)
                throws IOException, InterruptedException {
            // 对输入的一行文本数据按制表符 \t 进行分割
            // ValueIn 格式为<文档类别   单词   单词总数>，KeyIn 为记录偏移量
            String[] ValuesIn = ValueIn.toString().split("\t");
            String fileClass = ValuesIn[0];// 获取文档类别
            String word = ValuesIn[1];// 获取当前单词
            Double sumOfWords = Double.valueOf(ValuesIn[2]);// 获取当
前单词总数
            // 获取对应类别所有单词的总数
            Double sumOfClassWord = Double.valueOf((String)
                    Job3Mapper.fileClassToSumOfWords.get(fileClass));
            // 计算单词的条件概率
            // 设置输出的 Key 值，为<文档类别-单词>，虽然实现了 TextPair，为了
便于后续读取还是直接用"-"分割
            KEYOUT.set(fileClass + "-" + word);
            // 设置输出的 Value 值，为单词条件概率
            VALUEOUT.set(sumOfWords / sumOfClassWord);
            context.write(KEYOUT, VALUEOUT);
        }
    }
}
```

3.4 JobsInitiator.java

```
package com.TomAndersen.hadoop.BayesClassification;

import com.TomAndersen.hadoop.HDFSTools.BayesTools;
import org.apache.hadoop.util.ToolRunner;


/**
 * @Author
 * @Version
 * @Date 2019/11/5
 * 用于启动所有的 Job，本类中只进行作业调度
 */
public class JobsInitiator {

    // 单机测试下的本地文件系统相对路径，千万注意此路径集群中不能使用，集群中路
径前面需要加/
    public static final String Job1_OutputPath =
"src/Output/Job1/";
    public static final String Job2_OutputPath =
"src/Output/Job2/";
    public static final String Job3_OutputPath =
"src/Output/Job3/";


    /*// 集群测试下的 HDFS 文件路径
    public static final String Job1_OutputPath =
"/src/Output/Job1/";
    public static final String Job2_OutputPath =
"/src/Output/Job2/";
    public static final String Job3_OutputPath =
"/src/Output/Job3/";
    */

    public static void main(String[] args) throws Exception {
        if (args.length < 1) {
            System.out.println("***************At least one
parameters are required!***************");
            System.exit(1);
        } else if (args.length > 1) {
            System.out.println("***************Only one parameters
are required!***************");
            System.exit(1);
        }

        // 第一个参数为训练集文档路径
        final String TrainDataSetPath = args[0];

        int JobexitCode = 0;
        BayesTools.CheckOutputPath(Job1_OutputPath);//检查 Job1 输出路
径是否为空
        JobexitCode += ToolRunner.run(new Job1(), new
String[]{TrainDataSetPath, Job1_OutputPath});

        BayesTools.CheckOutputPath(Job2_OutputPath);//检查 Job2 输出路
径是否为空
        JobexitCode += ToolRunner.run(new Job2(), new
String[]{TrainDataSetPath, Job2_OutputPath});

        BayesTools.CheckOutputPath(Job3_OutputPath);//检查 Job2 输出路
径是否为空
        JobexitCode += ToolRunner.run(new Job3(), new
String[]{Job1_OutputPath, Job3_OutputPath});

        System.exit(JobexitCode);// 当参数为 0 时表示正常终止 JVM，为非 0 时
表示异常终止

        // 使用训练好的模型进行分类
        // BayesTools.BayesClassifier(TestDataSetPath);
        // 将分类器建模和分类器测试分为两个程序比较好，就不放在这里运行了

    }
}
```

3.5 其余代码

本报告只介绍了朴素贝叶斯分类器构建的 MapReduce 代码，关于测试集分类其余详细工程源代码请见：

https://github.com/TomAndersen-cc/MyFirstHadoopProject

4 数据集说明

本实验所用具体数据集如表 4.1 所示。其中训练集、测试集文档按照比例 1:1 从中随机选取。

表 4.1 实验所用数据集

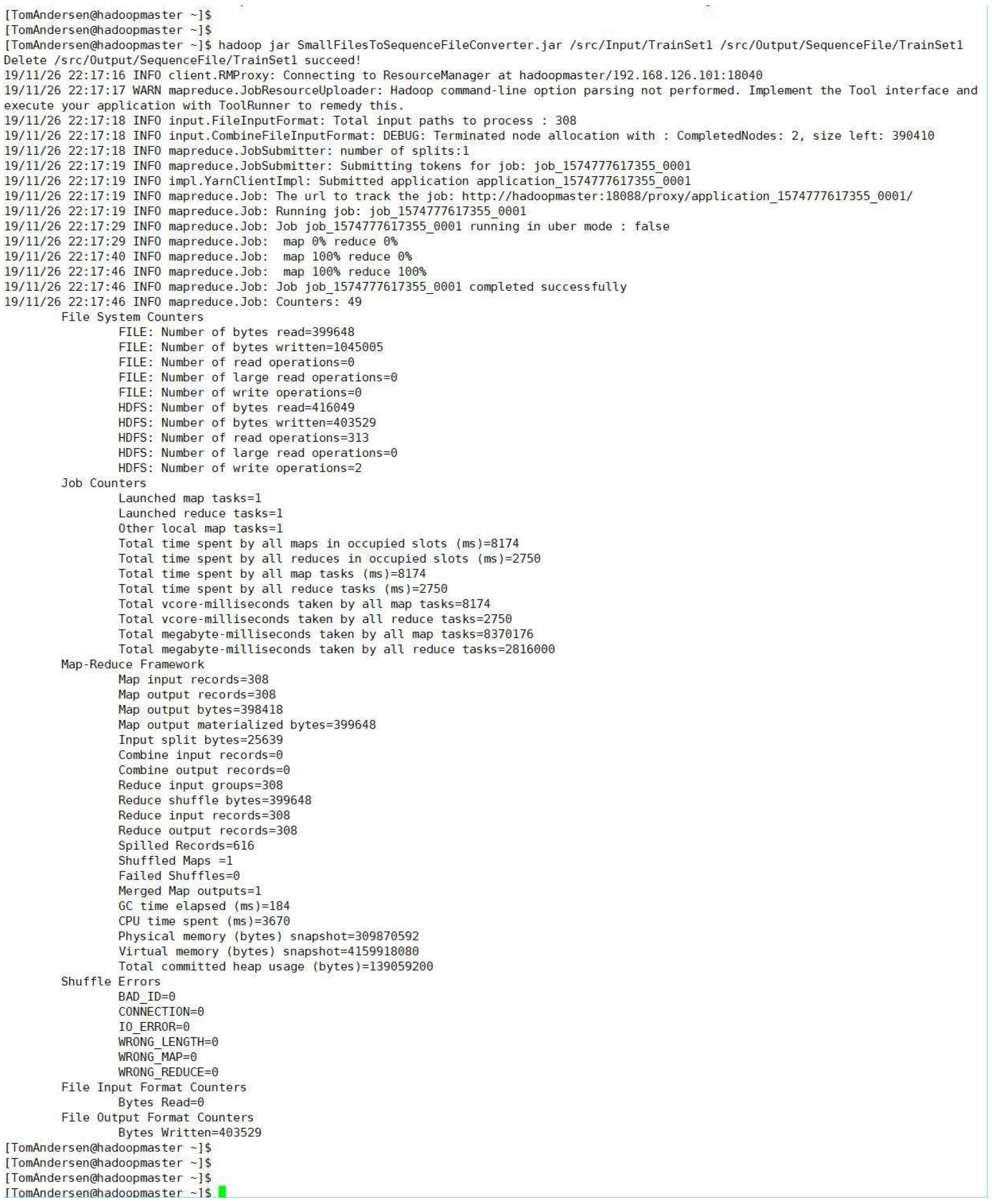
	BELG	BRAZ	CANA	总计
文档数量	154	200	263	617

5 程序运行说明

在本次实验中，每个 MapReduce Job 运行时都只有一个 MapTask 和 ReduceTask。首先本次实验中为了提高对小文件的处理效率，采用了生成 SequenceFile 文件的方式整合了数据集，由此大大减小了集群在 Task 资源调度时和频繁地将中间结果写入磁盘时所浪费的时间。其次由于文档数量有限，数据量太小，生成的 SequenceFile 文件大小依旧无法填满一个 block，所以最后实际执行时，每个 Job 最多只有一个 MapTask 和 ReduceTask。

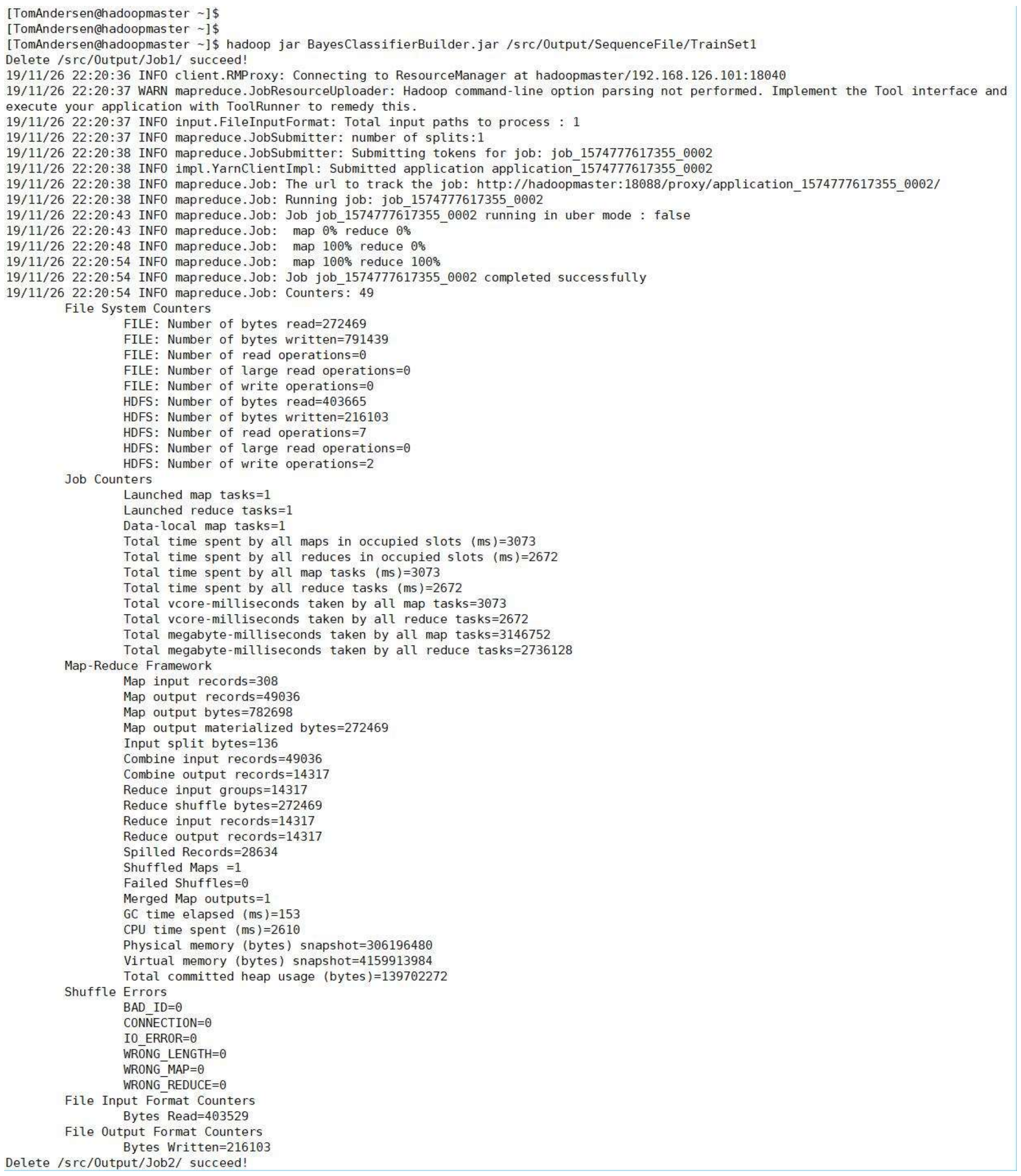
其中关于程序运行时 Web UI 页面作业监控截图及程序运行命令行截图具体如下所示：

5.1 SmallFilesToSequenceFileConverter



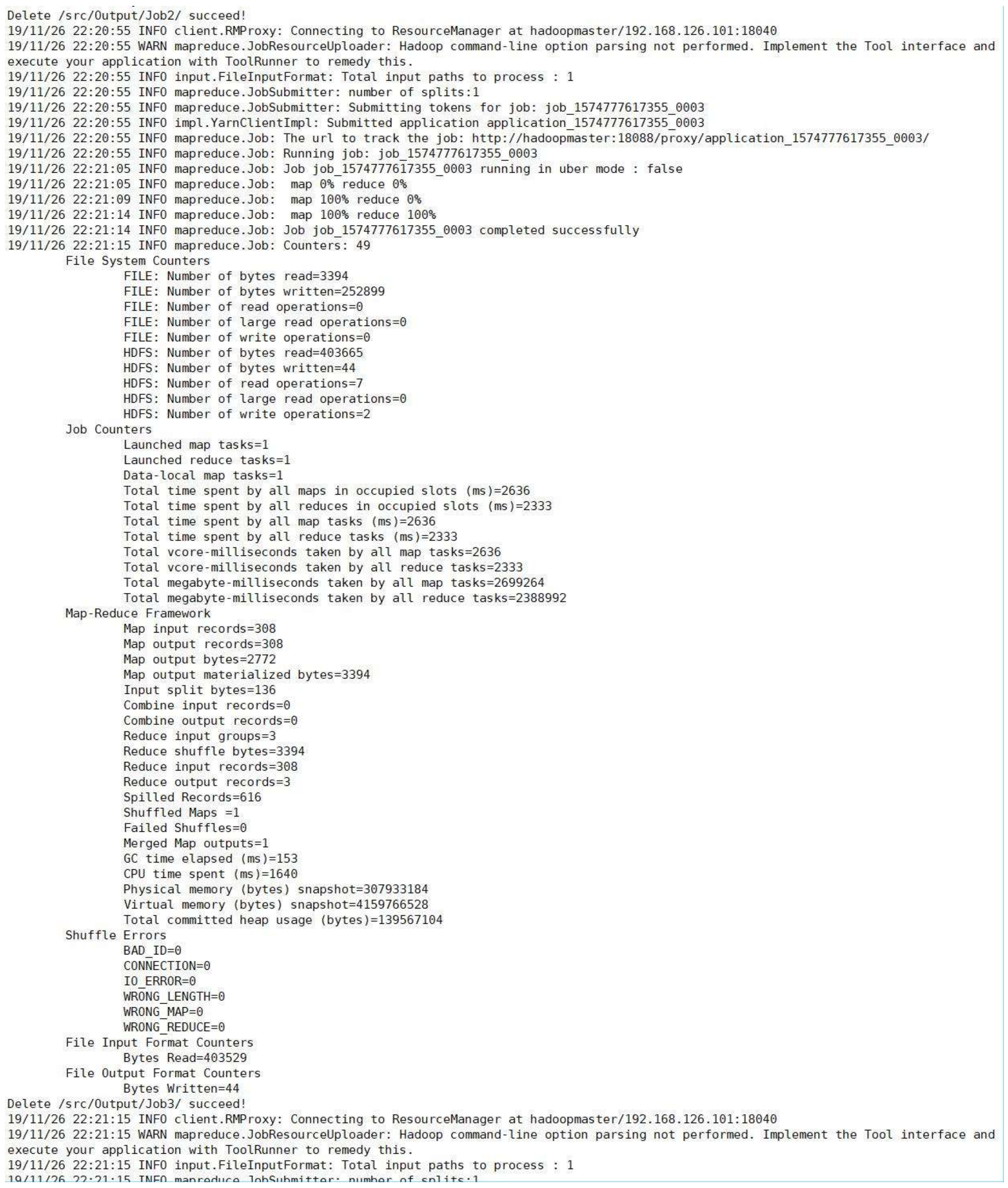
```
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$ hadoop jar SmallFilesToSequenceFileConverter.jar /src/Input/TrainSet1 /src/Output/SequenceFile/TrainSet1
Delete /src/Output/SequenceFile/TrainSet1 succeed!
19/11/26 22:17:16 INFO client.RMProxy: Connecting to ResourceManager at hadoopmaster/192.168.126.101:18040
19/11/26 22:17:17 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
19/11/26 22:17:18 INFO input.FileInputFormat: Total input paths to process : 308
19/11/26 22:17:18 INFO input.CombineFileInputFormat: DEBUG: Terminated node allocation with : CompletedNodes: 2, size left: 390410
19/11/26 22:17:18 INFO mapreduce.JobSubmitter: number of splits:1
19/11/26 22:17:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1574777617355_0001
19/11/26 22:17:19 INFO impl.YarnClientImpl: Submitted application application_1574777617355_0001
19/11/26 22:17:19 INFO mapreduce.Job: The url to track the job: http://hadoopmaster:18088/proxy/application_1574777617355_0001/
19/11/26 22:17:19 INFO mapreduce.Job: Running job: job_1574777617355_0001
19/11/26 22:17:29 INFO mapreduce.Job: Job job_1574777617355_0001 running in uber mode : false
19/11/26 22:17:29 INFO mapreduce.Job:  map 0% reduce 0%
19/11/26 22:17:40 INFO mapreduce.Job:  map 100% reduce 0%
19/11/26 22:17:46 INFO mapreduce.Job:  map 100% reduce 100%
19/11/26 22:17:46 INFO mapreduce.Job: Job job_1574777617355_0001 completed successfully
19/11/26 22:17:46 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=399648
                FILE: Number of bytes written=1045005
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=416049
                HDFS: Number of bytes written=403529
                HDFS: Number of read operations=313
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8174
                Total time spent by all reduces in occupied slots (ms)=2750
                Total time spent by all map tasks (ms)=8174
                Total time spent by all reduce tasks (ms)=2750
                Total vcore-milliseconds taken by all map tasks=8174
                Total vcore-milliseconds taken by all reduce tasks=2750
                Total megabyte-milliseconds taken by all map tasks=8370176
                Total megabyte-milliseconds taken by all reduce tasks=2816000
        Map-Reduce Framework
                Map input records=308
                Map output records=308
                Map output bytes=398418
                Map output materialized bytes=399648
                Input split bytes=25639
                Combine input records=0
                Combine output records=0
                Reduce input groups=308
                Reduce shuffle bytes=399648
                Reduce input records=308
                Reduce output records=308
                Spilled Records=616
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=184
                CPU time spent (ms)=3670
                Physical memory (bytes) snapshot=309870592
                Virtual memory (bytes) snapshot=4159918080
                Total committed heap usage (bytes)=139059200
        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=403529
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
```

5.2 MapReduce Job1



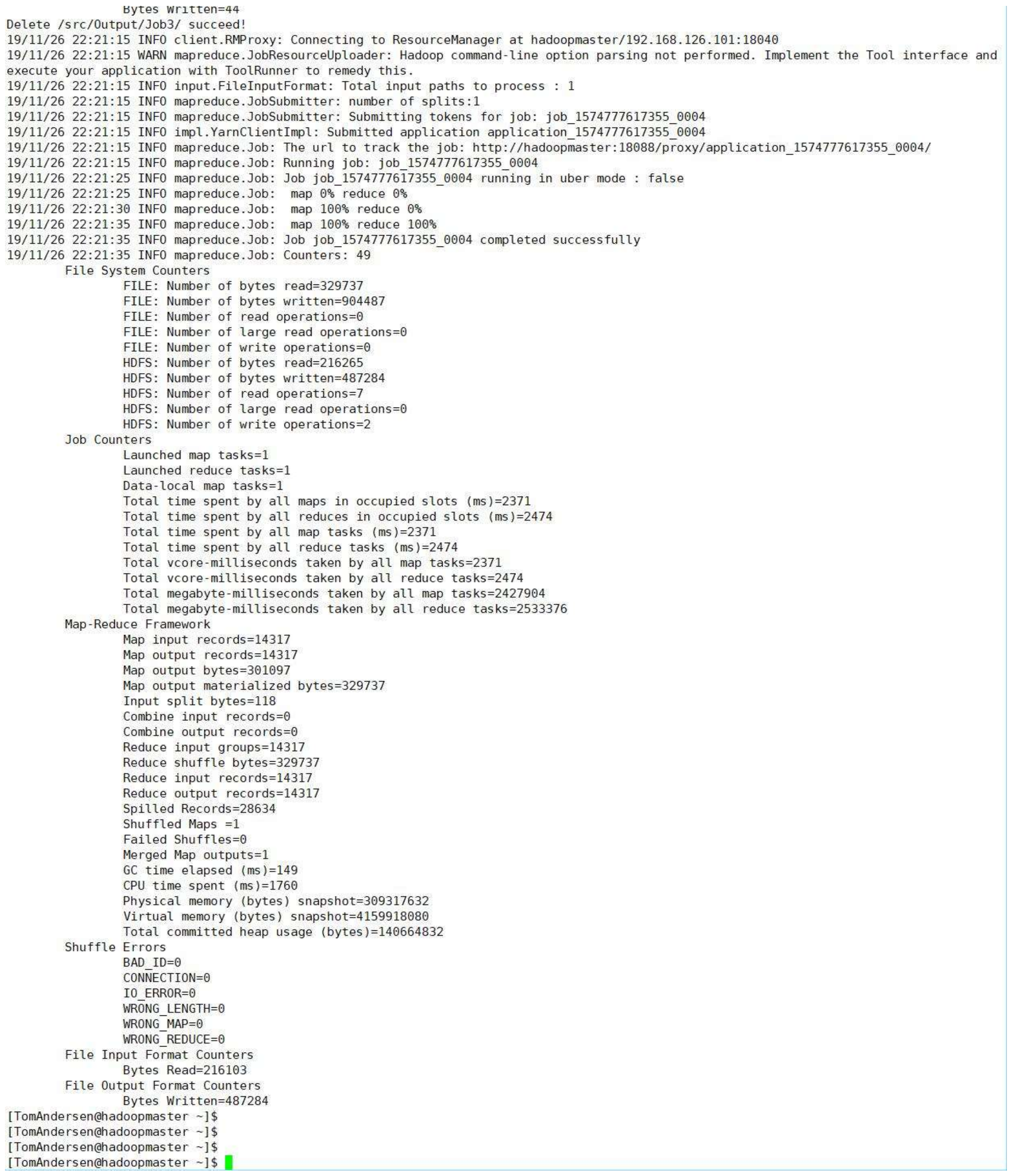
```
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$ hadoop jar BayesClassifierBuilder.jar /src/Output/SequenceFile/TrainSet1
Delete /src/Output/Job1/ succeed!
19/11/26 22:20:36 INFO client.RMProxy: Connecting to ResourceManager at hadoopmaster/192.168.126.101:18040
19/11/26 22:20:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
19/11/26 22:20:37 INFO input.FileInputFormat: Total input paths to process : 1
19/11/26 22:20:37 INFO mapreduce.JobSubmitter: number of splits:1
19/11/26 22:20:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1574777617355_0002
19/11/26 22:20:38 INFO impl.YarnClientImpl: Submitted application application_1574777617355_0002
19/11/26 22:20:38 INFO mapreduce.Job: The url to track the job: http://hadoopmaster:18088/proxy/application_1574777617355_0002/
19/11/26 22:20:38 INFO mapreduce.Job: Running job: job_1574777617355_0002
19/11/26 22:20:43 INFO mapreduce.Job: Job job_1574777617355_0002 running in uber mode : false
19/11/26 22:20:43 INFO mapreduce.Job:  map 0% reduce 0%
19/11/26 22:20:48 INFO mapreduce.Job:  map 100% reduce 0%
19/11/26 22:20:54 INFO mapreduce.Job:  map 100% reduce 100%
19/11/26 22:20:54 INFO mapreduce.Job: Job job_1574777617355_0002 completed successfully
19/11/26 22:20:54 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=272469
                FILE: Number of bytes written=791439
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=403665
                HDFS: Number of bytes written=216103
                HDFS: Number of read operations=7
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3073
                Total time spent by all reduces in occupied slots (ms)=2672
                Total time spent by all map tasks (ms)=3073
                Total time spent by all reduce tasks (ms)=2672
                Total vcore-milliseconds taken by all map tasks=3073
                Total vcore-milliseconds taken by all reduce tasks=2672
                Total megabyte-milliseconds taken by all map tasks=3146752
                Total megabyte-milliseconds taken by all reduce tasks=2736128
        Map-Reduce Framework
                Map input records=308
                Map output records=49036
                Map output bytes=782698
                Map output materialized bytes=272469
                Input split bytes=136
                Combine input records=49036
                Combine output records=14317
                Reduce input groups=14317
                Reduce shuffle bytes=272469
                Reduce input records=14317
                Reduce output records=14317
                Spilled Records=28634
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=153
                CPU time spent (ms)=2610
                Physical memory (bytes) snapshot=306196480
                Virtual memory (bytes) snapshot=4159913984
                Total committed heap usage (bytes)=139702272
        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=403529
        File Output Format Counters
                Bytes Written=216103
Delete /src/Output/Job2/ succeed!
```

5.3 MapReduce Job2



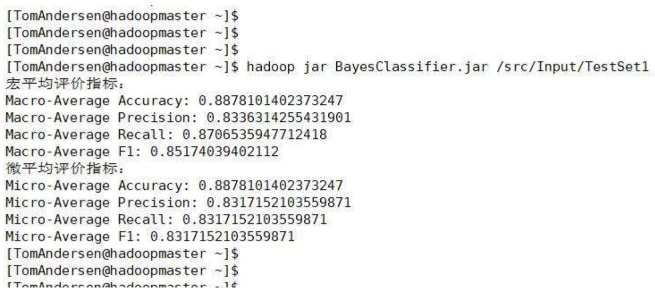
```
Delete /src/Output/Job2/ succeed!
19/11/26 22:20:55 INFO client.RMProxy: Connecting to ResourceManager at hadoopmaster/192.168.126.101:18040
19/11/26 22:20:55 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
19/11/26 22:20:55 INFO input.FileInputFormat: Total input paths to process : 1
19/11/26 22:20:55 INFO mapreduce.JobSubmitter: number of splits:1
19/11/26 22:20:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1574777617355_0003
19/11/26 22:20:55 INFO impl.YarnClientImpl: Submitted application application_1574777617355_0003
19/11/26 22:20:55 INFO mapreduce.Job: The url to track the job: http://hadoopmaster:18088/proxy/application_1574777617355_0003/
19/11/26 22:20:55 INFO mapreduce.Job: Running job: job_1574777617355_0003
19/11/26 22:21:05 INFO mapreduce.Job: Job job_1574777617355_0003 running in uber mode : false
19/11/26 22:21:05 INFO mapreduce.Job:  map 0% reduce 0%
19/11/26 22:21:09 INFO mapreduce.Job:  map 100% reduce 0%
19/11/26 22:21:14 INFO mapreduce.Job:  map 100% reduce 100%
19/11/26 22:21:14 INFO mapreduce.Job: Job job_1574777617355_0003 completed successfully
19/11/26 22:21:15 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=3394
                FILE: Number of bytes written=252899
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=403665
                HDFS: Number of bytes written=44
                HDFS: Number of read operations=7
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2636
                Total time spent by all reduces in occupied slots (ms)=2333
                Total time spent by all map tasks (ms)=2636
                Total time spent by all reduce tasks (ms)=2333
                Total vcore-milliseconds taken by all map tasks=2636
                Total vcore-milliseconds taken by all reduce tasks=2333
                Total megabyte-milliseconds taken by all map tasks=2699264
                Total megabyte-milliseconds taken by all reduce tasks=2388992
        Map-Reduce Framework
                Map input records=308
                Map output records=308
                Map output bytes=2772
                Map output materialized bytes=3394
                Input split bytes=136
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=3394
                Reduce input records=308
                Reduce output records=3
                Spilled Records=616
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=153
                CPU time spent (ms)=1640
                Physical memory (bytes) snapshot=307933184
                Virtual memory (bytes) snapshot=4159766528
                Total committed heap usage (bytes)=139567104
        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=403529
        File Output Format Counters
                Bytes Written=44
Delete /src/Output/Job3/ succeed!
19/11/26 22:21:15 INFO client.RMProxy: Connecting to ResourceManager at hadoopmaster/192.168.126.101:18040
19/11/26 22:21:15 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
19/11/26 22:21:15 INFO input.FileInputFormat: Total input paths to process : 1
19/11/26 22:21:15 INFO mapreduce.JobSubmitter: number of splits:1
```

5.4 MapReduce Job3



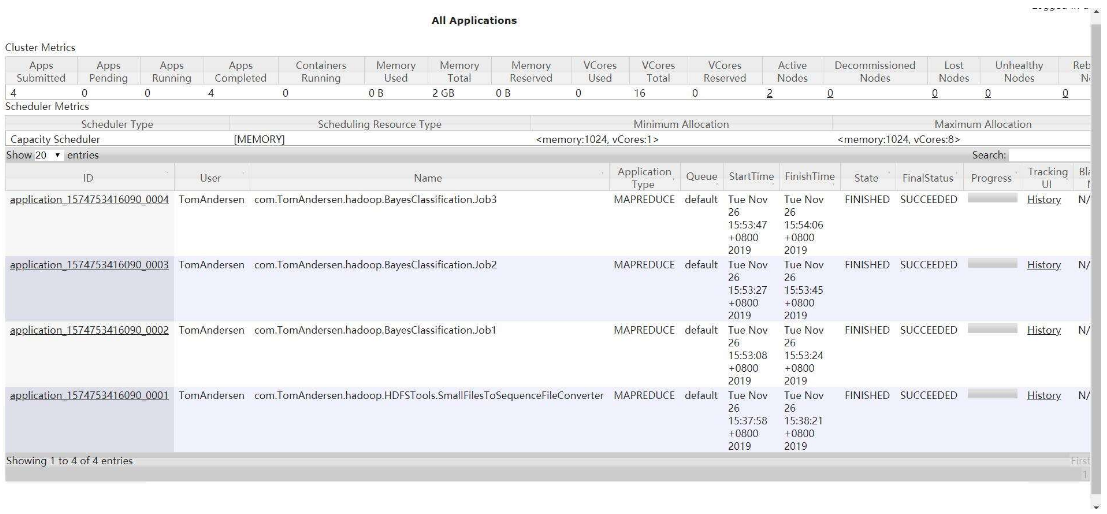
```
                Bytes Written=44
Delete /src/Output/Job3/ succeed!
19/11/26 22:21:15 INFO client.RMProxy: Connecting to ResourceManager at hadoopmaster/192.168.126.101:18040
19/11/26 22:21:15 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
19/11/26 22:21:15 INFO input.FileInputFormat: Total input paths to process : 1
19/11/26 22:21:15 INFO mapreduce.JobSubmitter: number of splits:1
19/11/26 22:21:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1574777617355_0004
19/11/26 22:21:15 INFO impl.YarnClientImpl: Submitted application application_1574777617355_0004
19/11/26 22:21:15 INFO mapreduce.Job: The url to track the job: http://hadoopmaster:18088/proxy/application_1574777617355_0004/
19/11/26 22:21:15 INFO mapreduce.Job: Running job: job_1574777617355_0004
19/11/26 22:21:25 INFO mapreduce.Job: Job job_1574777617355_0004 running in uber mode : false
19/11/26 22:21:25 INFO mapreduce.Job:  map 0% reduce 0%
19/11/26 22:21:30 INFO mapreduce.Job:  map 100% reduce 0%
19/11/26 22:21:35 INFO mapreduce.Job:  map 100% reduce 100%
19/11/26 22:21:35 INFO mapreduce.Job: Job job_1574777617355_0004 completed successfully
19/11/26 22:21:35 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=329737
                FILE: Number of bytes written=904487
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=216265
                HDFS: Number of bytes written=487284
                HDFS: Number of read operations=7
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2371
                Total time spent by all reduces in occupied slots (ms)=2474
                Total time spent by all map tasks (ms)=2371
                Total time spent by all reduce tasks (ms)=2474
                Total vcore-milliseconds taken by all map tasks=2371
                Total vcore-milliseconds taken by all reduce tasks=2474
                Total megabyte-milliseconds taken by all map tasks=2427904
                Total megabyte-milliseconds taken by all reduce tasks=2533376
        Map-Reduce Framework
                Map input records=14317
                Map output records=14317
                Map output bytes=301097
                Map output materialized bytes=329737
                Input split bytes=118
                Combine input records=0
                Combine output records=0
                Reduce input groups=14317
                Reduce shuffle bytes=329737
                Reduce input records=14317
                Reduce output records=14317
                Spilled Records=28634
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=149
                CPU time spent (ms)=1760
                Physical memory (bytes) snapshot=309317632
                Virtual memory (bytes) snapshot=4159918080
                Total committed heap usage (bytes)=140664832
        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=216103
        File Output Format Counters
                Bytes Written=487284
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
```

5.5　测试集分类结果评价



```
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$ hadoop jar BayesClassifier.jar /src/Input/TestSet1
宏平均评价指标:
Macro-Average Accuracy: 0.8878101402373247
Macro-Average Precision: 0.8336314255431901
Macro-Average Recall: 0.8706535947712418
Macro-Average F1: 0.85174039402112
微平均评价指标:
Micro-Average Accuracy: 0.8878101402373247
Micro-Average Precision: 0.8317152103559871
Micro-Average Recall: 0.8317152103559871
Micro-Average F1: 0.8317152103559871
[TomAndersen@hadoopmaster ~]$
[TomAndersen@hadoopmaster ~]$
```

5.6　Web UI 界面监控截图



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Reb N
4	0	0	4	0	0 B	2 GB	0 B	0	16	0	2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:1024, vCores:8>

Show 20 entries　　Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Bla
application_1574753416090_0004	TomAndersen	com.TomAndersen.hadoop.BayesClassification.Job3	MAPREDUCE	default	Tue Nov 26 15:53:47 +0800 2019	Tue Nov 26 15:54:06 +0800 2019	FINISHED	SUCCEEDED		History	N/
application_1574753416090_0003	TomAndersen	com.TomAndersen.hadoop.BayesClassification.Job2	MAPREDUCE	default	Tue Nov 26 15:53:27 +0800 2019	Tue Nov 26 15:53:45 +0800 2019	FINISHED	SUCCEEDED		History	N/
application_1574753416090_0002	TomAndersen	com.TomAndersen.hadoop.BayesClassification.Job1	MAPREDUCE	default	Tue Nov 26 15:53:08 +0800 2019	Tue Nov 26 15:53:24 +0800 2019	FINISHED	SUCCEEDED		History	N/
application_1574753416090_0001	TomAndersen	com.TomAndersen.hadoop.HDFSTools.SmallFilesToSequenceFileConverter	MAPREDUCE	default	Tue Nov 26 15:37:58 +0800 2019	Tue Nov 26 15:38:21 +0800 2019	FINISHED	SUCCEEDED		History	N/

Showing 1 to 4 of 4 entries

6 实验结果分析

本次实验中使用 BELG、GRAZ、CANA 三类共计 617 个文档，其中测试集和训练集比例为 1:1，最终使用训练集构建的朴素贝叶斯分类器对测试集分类结果的各项评价指标计算如下所示。

从分类结果来看，实验所构建的朴素贝叶斯分类器效果显著。

宏平均评价指标：

```
Macro-Average Accuracy: 0.8878101402373247
Macro-Average Precision: 0.8336314255431901
Macro-Average Recall: 0.8706535947712418
Macro-Average F1: 0.85174039402112
```

微平均评价指标：

```
Micro-Average Accuracy: 0.8878101402373247
Micro-Average Precision: 0.8317152103559871
Micro-Average Recall: 0.8317152103559871
Micro-Average F1: 0.8317152103559871
```