**成都信息工程大学**

**实验报告**

**总结报告**

|  |
| --- |
| **学院： 软件工程学院** |
| **专业： 软件工程** |
| **班级： 软工182** |
| **姓名： 徐莘杰、邓棋、孟令迪** |
| **学号： 2018081065、2018081062、2018081050** |

目录

[**1.** **实验原理** 3](#_Toc61167480)

[1.1Hadoop下的MapReduce基本原理 3](#_Toc61167481)

[1.2朴素贝叶斯基本思想 5](#_Toc61167482)

[1.3情感分析原理 5](#_Toc61167483)

[**2.** **实验过程** 5](#_Toc61167484)

[2.1实验搭建环境 5](#_Toc61167485)

[2.2maven搭建环境 12](#_Toc61167486)

[2.3创建hadoop项目以及引入依赖 13](#_Toc61167487)

[2.4代码编写 13](#_Toc61167488)

[**3.** **实验结论及结果** 26](#_Toc61167489)

[**4.** **感受心得** 26](#_Toc61167490)

[**5.** **答辩** 28](#_Toc61167491)

1. **实验原理**

1.1Hadoop下的MapReduce基本原理

MapReduce是Hadoop中的完成数据计算任务的核心框架

1. MapReduce 组成实体

     （1）Client节点：此节点上运行MapReduce程序和JobClient实例对象，负责提交MapReduce作业。

     （2）JobTracker：协调调度，主控节点，一个Hadoop集群仅有一个JobTracker节点

     （3）Map TaskTracker：执行 Map任务，一个Hadoop集群有多个TaskTracker节点

     （4）Reduce TaskTracker：执行Reduce任务，一个Hadoop集群有多个TaskTracker节点

     （5）HDFS，存储数据文件，配置文件

2. 作业流程分布讲解

   （1）作业启动：

                                     由Client节点运行MapReduce程序，创建JobClient实例

                                                                    ↓

                                     JobClient向JobTracker发出请求，获得一个JobID，用于标识本次MapReduce作业

                                                                    ↓

                                     JobClient将运行作业需要的相关资源（配置文件、输入数据分片数量、包含Mapper类和Reducer类的JAR文件）

                                     放入作业对应的HDFS目录， 计算分片数量和map任务数量

                                                                    ↓

                                     向JobTracker提交作业，并获得作业的状态对象句柄

   （2）作业初始化：

                                                               切分数据

                                    对于多个JobClient的请求，JobTracker使用队列机制，由作业调度器进行调度

                                                                     ↓

                                    JobTracker创建一个代表此作业的JobInProgress实例，用于后续跟踪和调度此作业

                                                                     ↓

                                    JobTracker从HDFS中获得输入数据的分片信息，决定创建Map任务的数量，并对应创建出一批TaskInProgress，以监控和调度Map任务

   （3）作业/任务调度：

                                    TaskTracker向JobTracker 发送心跳消息，主动pull 任务

                                                                     ↓

                                    JobTracker向有闲置资源的TaskTracker派发任务

   （4）Map执行：

                                    TaskTracker从HDFS提取相关资源（Jar包、数据）

                                                                 ↓

                                    TaskTracker 创建一个TaskRunner实例运行Map任务

                                                                 ↓

                                   TaskRunner在一个单独的JVM中启动MapTask执行map函数

                                    （使用单独的JVM是为了避免MapTask的异常影响TaskTracker的正常运行）

                                                                 ↓

　　　　　　　　　　　　　定期存入缓存，缓存已满时存入本地磁盘

                                                                 ↓

                                    MapTask定期向TaskTracker报告进度

                                                                 ↓

                                                  输出键值对 word：1

 （5）Shuffle：

                                    汇总相同的key的1值信息，确保相同的key交由同一个reduce任务进行合并处理

 （6）Reduce：

                                     JobTracker分配Reduce任务

                                                      ↓

                                    创建ReduceRunner运行Reduce任务

                                                      ↓

                                    ReduceRunner在单独的JVM中启动ReduceTask执行reduce函数

                                                      ↓

                                   从Map节点下载中间结果数据

                                                      ↓

                                   当所有Map任务完成后，JobTracker会通知所有的Reduce TaskTracker节点开始Reduce任务执行

                                                      ↓

                                   ReduceTask定期向TaskTracker报告进度

 （7）完成作业：

                                   JobTracker通过每一个TaskTracker和其之间的心跳消息，全部任务完成

                                                      ↓

                                   JobClient轮询获知任务完成

                                                      ↓

                                   JobClient通知用户程序整个作业完成，显示必要信息。

1.2朴素贝叶斯基本思想

朴素贝叶斯方法是在[贝叶斯](https://baike.baidu.com/item/%E8%B4%9D%E5%8F%B6%E6%96%AF/1405899)算法的基础上进行了相应的简化，即假定给定目标值时属性之间相互条件独立。也就是说没有哪个属性变量对于决策结果来说占有着较大的比重，也没有哪个属性变量对于决策结果占有着较小的比重。虽然这个简化方式在一定程度上降低了贝叶斯分类算法的分类效果，但是在实际的应用场景中，极大地简化了贝叶斯方法的复杂性。

1.3情感分析原理

上传训练集数据至hdfs,清洗数据,得到训练模型。上传测试集数据至hdfs,基于得出的训练模型采用朴素贝叶斯算法进行分类。

1. **实验过程**

2.1实验搭建环境

搭建Hadoop集群环境

环境搭建

实验设备与环境：JDK1.8, VMware® Workstation 16 Pro，Hadoop-2.7.4，ubuntu-20.04.1

/etc/profile文件

1. # /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
2. # and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
4. if [ "${PS1-}" ]; then
5. if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
6. # The file bash.bashrc already sets the default PS1.
7. # PS1='\h:\w\$ '
8. if [ -f /etc/bash.bashrc ]; then
9. . /etc/bash.bashrc
10. fi
11. else
12. if [ "`id -u`" -eq 0 ]; then
13. PS1='# '
14. else
15. PS1='$ '
16. fi
17. fi
18. fi
20. if [ -d /etc/profile.d ]; then
21. for i in /etc/profile.d/\*.sh; do
22. if [ -r $i ]; then
23. . $i
24. fi
25. done
26. unset i
27. fi
29. export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64
30. export JRE\_HOME=$JAVA\_HOME/jre
31. export CLASSPATH=.:$CLASSPATH:$JAVA\_HOME/lib:$JRE\_HOME/lib
32. export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin
34. export PATH=$PATH:/usr/local/hadoop/bin:/usr/local/hadoop/sbin

core-site.xml文件

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<?xml-stylesheet** type="text/xsl" href="configuration.xsl"**?>**
3. <!--
4. Licensed under the Apache License, Version 2.0 (the "License");
5. you may not use this file except in compliance with the License.
6. You may obtain a copy of the License at
8. http://www.apache.org/licenses/LICENSE-2.0
10. Unless required by applicable law or agreed to in writing, software
11. distributed under the License is distributed on an "AS IS" BASIS,
12. WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13. See the License for the specific language governing permissions and
14. limitations under the License. See accompanying LICENSE file.
15. --**>**
17. <!-- Put site-specific property overrides in this file. -->
19. **<configuration>**
20. **<property>**
21. **<name>**fs.defaultFS**</name>**
22. **<value>**hdfs://master:9000**</value>**
23. **</property>**
24. **<property>**
25. **<name>**hadoop.tmp.dir**</name>**
26. **<value>**file:/usr/local/hadoop/tmp**</value>**
27. **<description>**Abase for other temporary directories.**</description>**
28. **</property>**
29. **</configuration>**

hdfs-site.xml文件

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<?xml-stylesheet** type="text/xsl" href="configuration.xsl"**?>**
3. <!--
4. Licensed under the Apache License, Version 2.0 (the "License");
5. you may not use this file except in compliance with the License.
6. You may obtain a copy of the License at
8. http://www.apache.org/licenses/LICENSE-2.0
10. Unless required by applicable law or agreed to in writing, software
11. distributed under the License is distributed on an "AS IS" BASIS,
12. WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13. See the License for the specific language governing permissions and
14. limitations under the License. See accompanying LICENSE file.
15. --**>**
17. <!-- Put site-specific property overrides in this file. -->
19. **<configuration>**
20. **<property>**
21. **<name>**dfs.namenode.secondary.http-address**</name>**
22. **<value>**master:50090**</value>**
23. **</property>**
24. **<property>**
25. **<name>**dfs.replication**</name>**
26. **<value>**3**</value>**
27. **</property>**
28. **<property>**
29. **<name>**dfs.namenode.name.dir**</name>**
30. **<value>**file:/usr/local/hadoop/tmp/dfs/name**</value>**
31. **</property>**
32. **<property>**
33. **<name>**dfs.datanode.data.dir**</name>**
34. **<value>**file:/usr/local/hadoop/tmp/dfs/data**</value>**
35. **</property>**
36. **</configuration>**

mapred-site.xml文件

1. **<?xml** version="1.0"**?>**
2. **<?xml-stylesheet** type="text/xsl" href="configuration.xsl"**?>**
3. <!--
4. Licensed under the Apache License, Version 2.0 (the "License");
5. you may not use this file except in compliance with the License.
6. You may obtain a copy of the License at
8. http://www.apache.org/licenses/LICENSE-2.0
10. Unless required by applicable law or agreed to in writing, software
11. distributed under the License is distributed on an "AS IS" BASIS,
12. WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13. See the License for the specific language governing permissions and
14. limitations under the License. See accompanying LICENSE file.
15. --**>**
17. <!-- Put site-specific property overrides in this file. -->
19. **<configuration>**
20. **<property>**
21. **<name>**mapreduce.framework.name**</name>**
22. **<value>**yarn**</value>**
23. **</property>**
24. **<property>**
25. **<name>**mapreduce.jobhistory.address**</name>**
26. **<value>**master:10020**</value>**
27. **</property>**
28. **<property>**
29. **<name>**mapreduce.jobhistory.webapp.address**</name>**
30. **<value>**master:19888**</value>**
31. **</property>**
32. **</configuration>**

yarn-site.xml文件

1. **<?xml** version="1.0"**?>**
2. <!--
3. Licensed under the Apache License, Version 2.0 (the "License");
4. you may not use this file except in compliance with the License.
5. You may obtain a copy of the License at
7. http://www.apache.org/licenses/LICENSE-2.0
9. Unless required by applicable law or agreed to in writing, software
10. distributed under the License is distributed on an "AS IS" BASIS,
11. WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12. See the License for the specific language governing permissions and
13. limitations under the License. See accompanying LICENSE file.
14. --**>**
15. **<configuration>**
17. <!-- Site specific YARN configuration properties -->
18. <!-- 结点管理器-->
19. **<property>**
20. **<name>**yarn.nodemanager.aux-services**</name>**
21. **<value>**mapreduce\_shuffle**</value>**
22. **</property>**
23. **<property>**
24. **<name>**yarn.nodemanager.aux-services.mapreduce.shuffle.class**</name>**
25. **<value>**org.apache.hadoop.mapred.ShuffleHandler**</value>**
26. **</property>**
28. **<property>**
29. **<name>**yarn.resourcemanager.address**</name>**
30. **<value>**master:8032**</value>**
31. **</property>**
32. **<property>**
33. **<name>**yarn.resourcemanager.scheduler.address**</name>**
34. **<value>**master:8030**</value>**
35. **</property>**
36. **<property>**
37. **<name>**yarn.resourcemanager.resource-tracker.address**</name>**
38. **<value>**master:8031**</value>**
39. **</property>**

42. **<property>**
43. **<name>**yarn.resourcemanager.webapp.address**</name>**
44. **<value>**master:8088**</value>**
45. **</property>**
46. **<property>**
47. **<name>**yarn.resourcemanager.webapp.https.address**</name>**
48. **<value>**master:8090**</value>**
49. **</property>**
50. **<property>**
51. **<name>**yarn.resourcemanager.admin.address**</name>**
52. **<value>**master:8033**</value>**
53. **</property>**
54. **</configuration>**

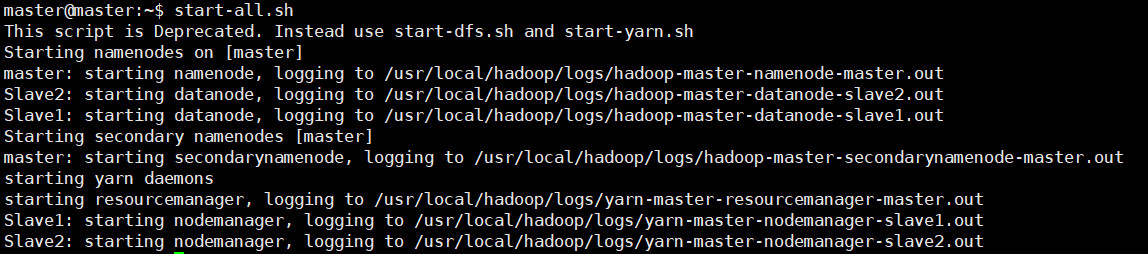
节点名称和地址的配置

192.168.111.128 master

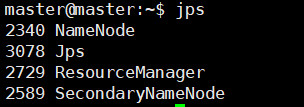
192.168.111.129 slave1

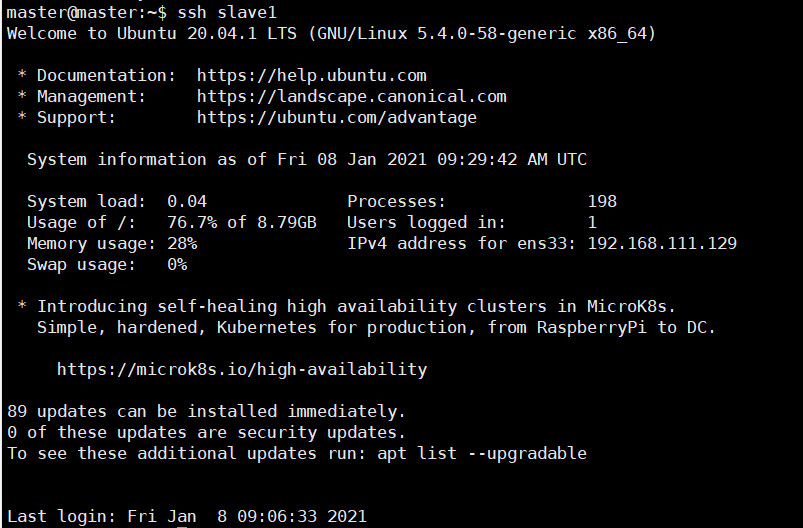
192.168.111.130 slave2

进入启动目录并用start-all.sh来启动dfs和yarn

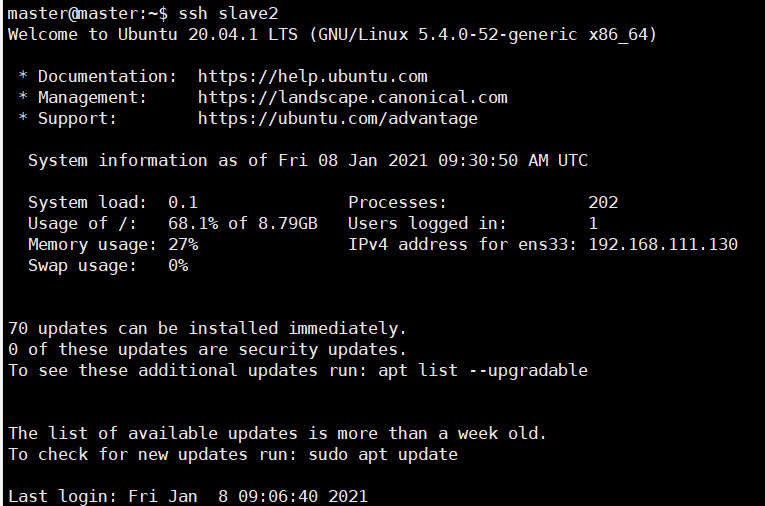


查看master节点的进程

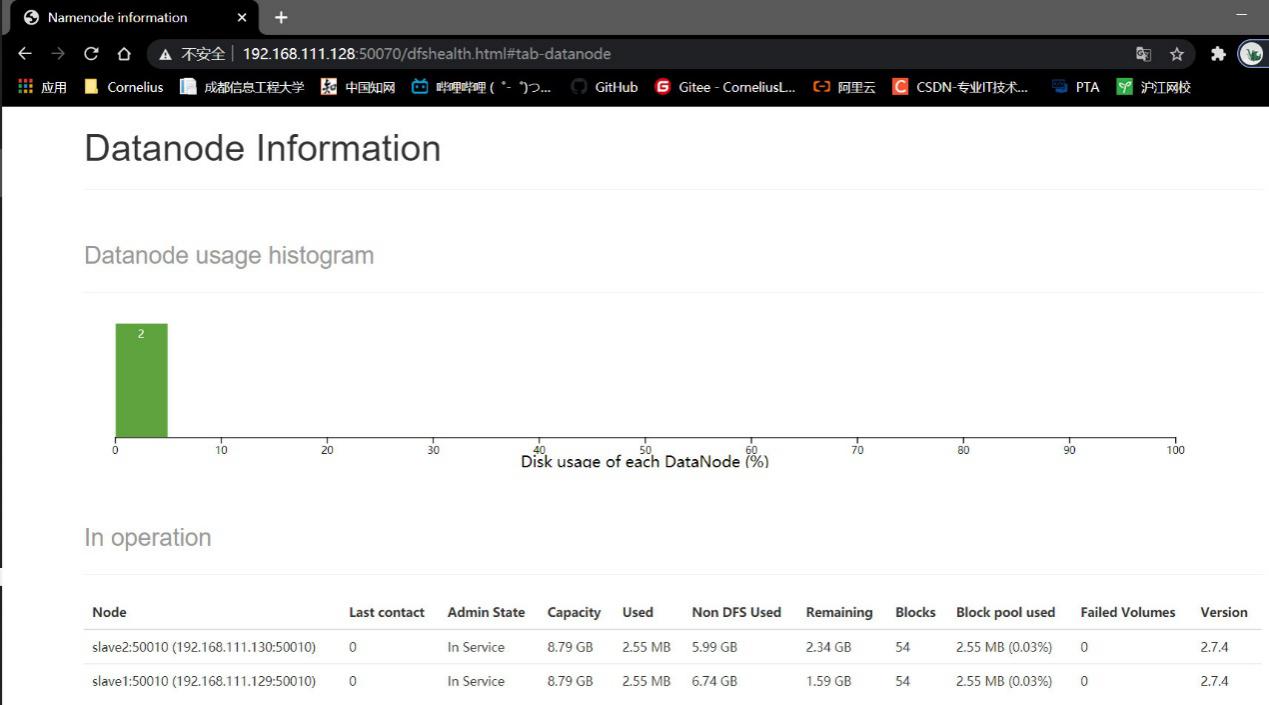


ssh slave1并且查看进程

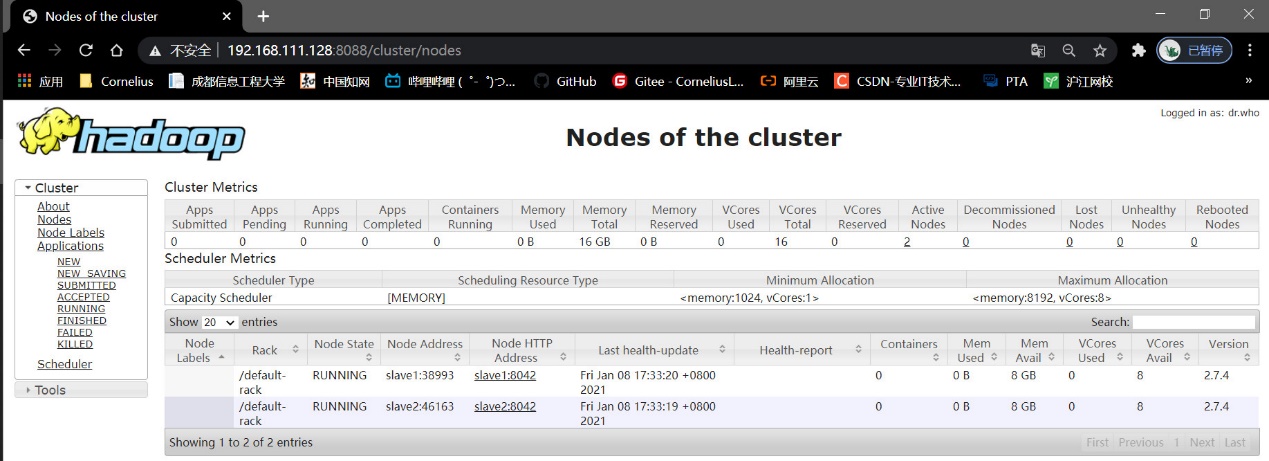
ssh slave2并查看进程



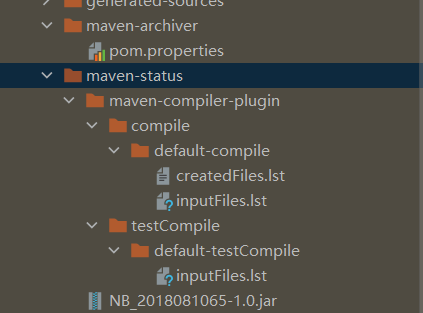
50070查看存活节点



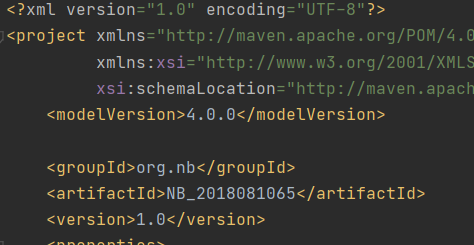
8088查看存活节点

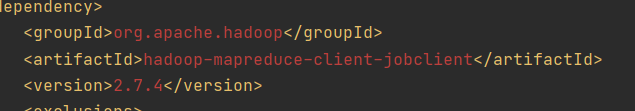


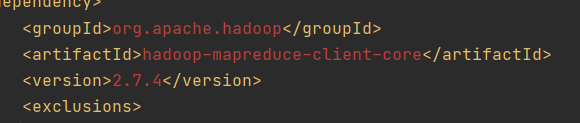
2.2maven搭建环境

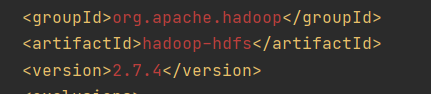


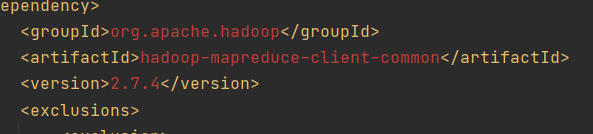
2.3创建hadoop项目以及引入依赖











2.4代码编写

2.4.1 NBMapper&NBReduce文件

NBMapper

package mapper;  
  
import java.io.IOException;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
  
public class NBMapper extends Mapper<LongWritable, Text, LongWritable, Text> {  
 public NBMapper() {  
 }  
  
 protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, LongWritable, Text>.Context context) throws IOException, InterruptedException {  
 String[] contents = value.toString().split("\t");  
 System.out.println(key.get());  
 context.write(key, new Text(contents[1]));  
 }  
}

NBReducer

package reducer;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Set;  
import mapper.TrainMapper;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.FSDataInputStream;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
public class NBReducer extends Reducer<LongWritable, Text, LongWritable, Text> {  
 static Map<String, Integer> *model*;  
 static long *index* = 1L;  
  
 public NBReducer() {  
 }  
  
 protected void reduce(LongWritable key, Iterable<Text> values, Reducer<LongWritable, Text, LongWritable, Text>.Context context) throws IOException, InterruptedException {  
 Iterator<Text> iterator = values.iterator();  
 int countGood = (Integer)*model*.get("好评");  
 int countBad = (Integer)*model*.get("差评");  
 System.out.println(countGood + "," + countBad);  
 int total = countBad + countGood;  
 double pGood = (double)countGood / (double)total;  
 double pBad = (double)countBad / (double)total;  
  
 double temp;  
 for(String var12 = ""; iterator.hasNext(); pGood /= temp) {  
 String line = ((Text)iterator.next()).toString();  
 String[] words = line.split(" ");  
 String[] var15 = words;  
 int var16 = words.length;  
  
 for(int var17 = 0; var17 < var16; ++var17) {  
 String word = var15[var17];  
 double pWordBad;  
 if (*model*.get("好评\_" + word) != null) {  
 pWordBad = (double)(Integer)*model*.get("好评\_" + word);  
 pWordBad /= (double)countGood;  
 pGood \*= pWordBad;  
 } else if (TrainMapper.*isChinese*(word)) {  
 pGood \*= 0.0D;  
 }  
  
 if (*model*.get("差评\_" + word) != null) {  
 pWordBad = (double)(Integer)*model*.get("差评\_" + word);  
 pWordBad /= (double)countBad;  
 pBad \*= pWordBad;  
 } else if (TrainMapper.*isChinese*(word)) {  
 pBad \*= 0.0D;  
 }  
 }  
  
 temp = pBad + pGood;  
 pBad /= temp;  
 }  
  
 context.write(new LongWritable(*index*), new Text(pGood > pBad ? "好评" : "差评"));  
 ++*index*;  
 }  
  
 protected void setup(Reducer<LongWritable, Text, LongWritable, Text>.Context context) throws IOException, InterruptedException {  
 *model* = *readModel2*(File.separator + "opt" + File.separator + "model1.txt");  
 }  
  
 public static void main(String[] args) throws IOException {  
 Map<String, Integer> map = *readModel2*("D://model1.txt");  
 System.out.println("好评".hashCode());  
 Set<String> keySet = map.keySet();  
 Iterator var3 = keySet.iterator();  
  
 while(var3.hasNext()) {  
 String k = (String)var3.next();  
 System.out.println(k + "," + k.hashCode());  
 }  
  
 System.out.println(map.get("好评"));  
 }  
  
 private static Map<String, Integer> readModel(String path) throws IOException {  
 Configuration conf = new Configuration();  
 conf.set("fs.defaultFS", "hdfs://192.168.154.128:9000");  
 FileSystem fs = FileSystem.get(conf);  
 FSDataInputStream inputStream = fs.open(new Path(path));  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream, "UTF-8"));  
 String line = "";  
 HashMap map = new HashMap();  
  
 while((line = reader.readLine()) != null) {  
 String[] contents = line.split("\t");  
 map.put(new String(contents[0].getBytes(), 0, contents[0].length(), "UTF-8"), Integer.valueOf(contents[1]));  
 }  
  
 reader.close();  
 inputStream.close();  
 return map;  
 }  
  
 private static Map<String, Integer> readModel2(String path) throws IOException {  
 File file = new File(path);  
 FileInputStream inputStream = new FileInputStream(file);  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream, "UTF-8"));  
 String line = "";  
 HashMap map = new HashMap();  
  
 while((line = reader.readLine()) != null) {  
 String[] contents = line.split("\t");  
 map.put(contents[0], Integer.valueOf(contents[1]));  
 }  
  
 reader.close();  
 inputStream.close();  
 return map;  
 }  
}

import java.io.IOException;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
  
public class NBMapper extends Mapper<LongWritable, Text, LongWritable, Text> {  
 public NBMapper() {  
 }  
  
 protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, LongWritable, Text>.Context context) throws IOException, InterruptedException {  
 String[] contents = value.toString().split("\t");  
 System.out.println(key.get());  
 context.write(key, new Text(contents[1]));  
 }  
}

2.4.2TrainMapper&TrainReduce文件

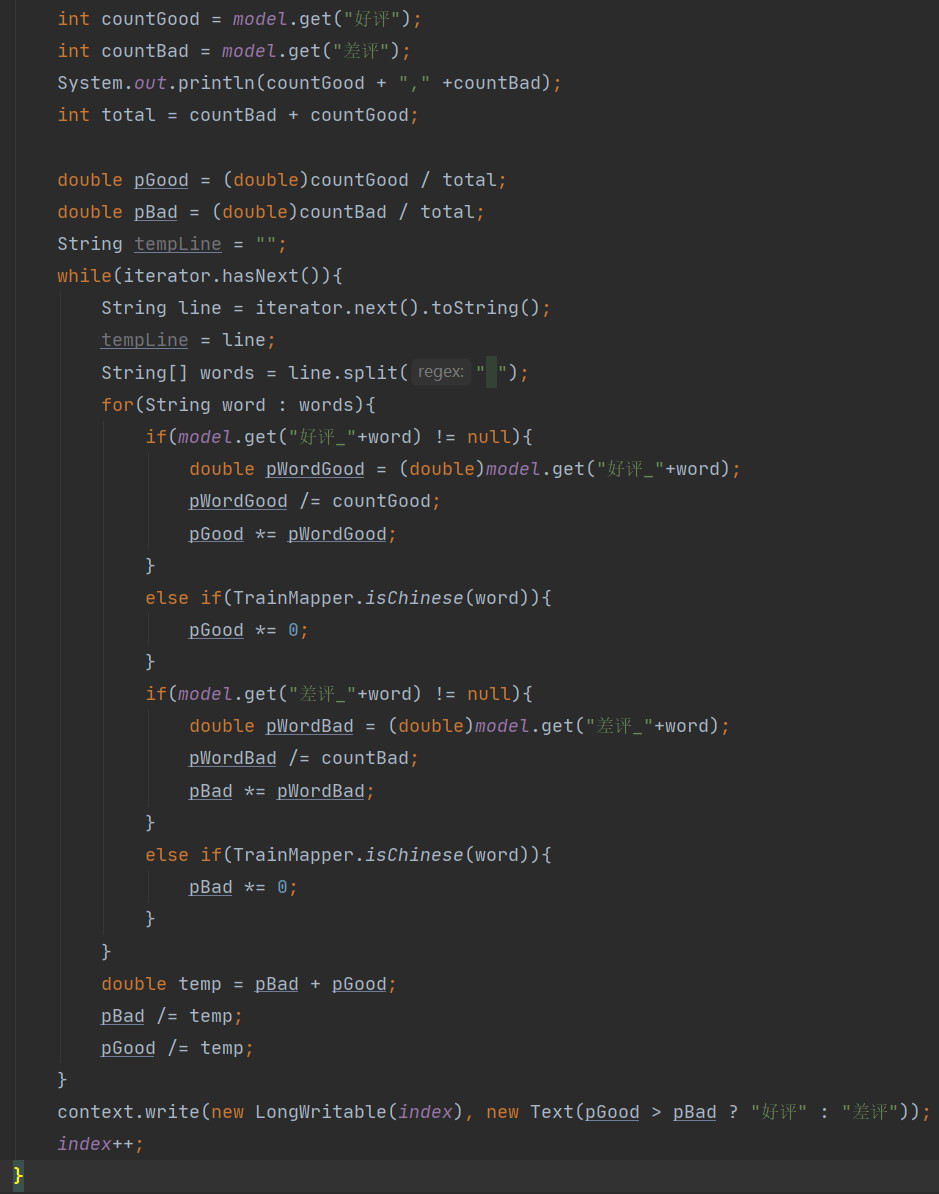
Trainreduce

package reducer;  
  
import java.io.IOException;  
import java.util.Iterator;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
public class TrainReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
 public TrainReducer() {  
 }  
  
 protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 int sum = 0;  
  
 for(Iterator iterator = values.iterator(); iterator.hasNext(); sum += ((IntWritable)iterator.next()).get()) {  
 }  
  
 context.write(key, new IntWritable(sum));  
 }  
}

TrainMapper

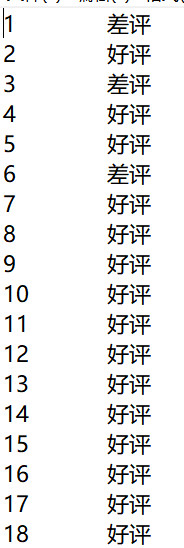
package mapper;  
  
import java.io.IOException;  
import java.lang.Character.UnicodeBlock;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
  
public class TrainMapper extends Mapper<Object, Text, Text, IntWritable> {  
 public TrainMapper() {  
 }  
  
 protected void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 String[] fields = value.toString().split("\t");  
 String label = fields[0];  
 context.write(new Text(label), new IntWritable(1));  
 if (fields.length == 2) {  
 String[] features = fields[1].split(" ");  
 String[] var7 = features;  
 int var8 = features.length;  
  
 for(int var9 = 0; var9 < var8; ++var9) {  
 String feature = var7[var9];  
 if (*isChinese*(feature)) {  
 feature = label + "\_" + feature;  
 context.write(new Text(feature), new IntWritable(1));  
 }  
 }  
 }  
  
 }  
  
 private static boolean isChinese(char c) {  
 UnicodeBlock ub = UnicodeBlock.of(c);  
 return ub == UnicodeBlock.CJK\_UNIFIED\_IDEOGRAPHS || ub == UnicodeBlock.CJK\_COMPATIBILITY\_IDEOGRAPHS || ub == UnicodeBlock.CJK\_UNIFIED\_IDEOGRAPHS\_EXTENSION\_A || ub == UnicodeBlock.CJK\_UNIFIED\_IDEOGRAPHS\_EXTENSION\_B || ub == UnicodeBlock.CJK\_SYMBOLS\_AND\_PUNCTUATION;  
 }  
  
 public static boolean isChinese(String strName) {  
 char[] ch = strName.toCharArray();  
  
 for(int i = 0; i < ch.length; ++i) {  
 char c = ch[i];  
 if (!*isChinese*(c)) {  
 return false;  
 }  
 }  
  
 return true;  
 }  
}

2.4.3 NativeBayes主类



1. **实验结论及结果**

训练模型：

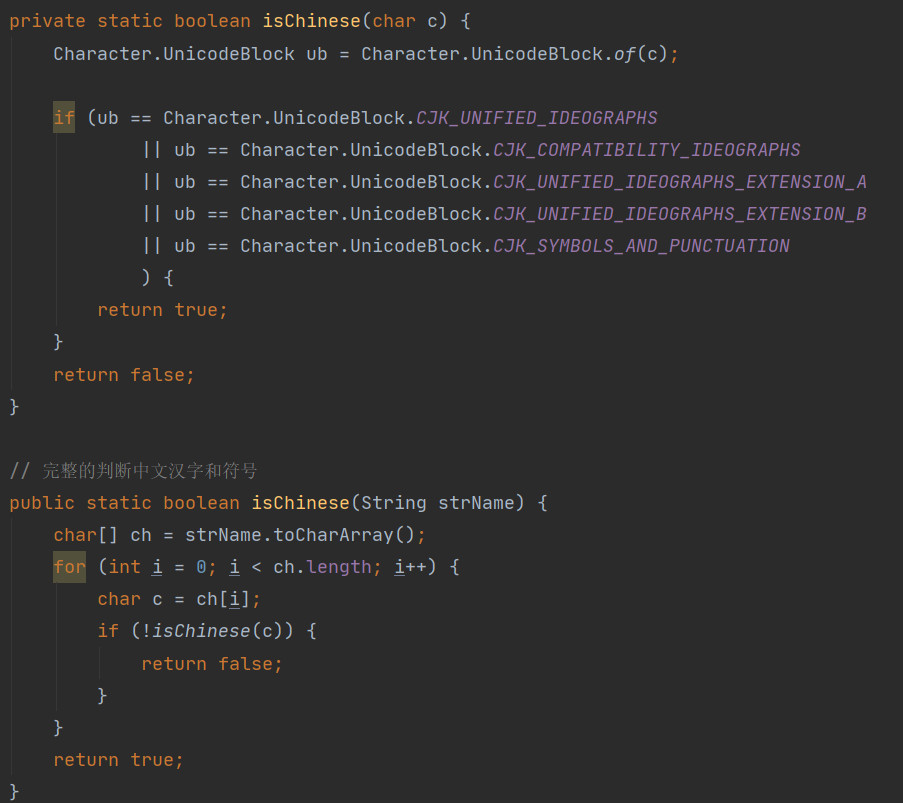
预测结果：

经过计算匹配率为93.15%

1. **感受心得**

代码关键部分

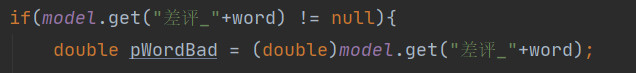
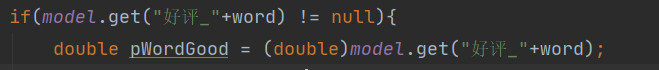
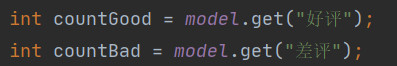
（1）在训练的过程中，过滤包含非中文字符或全部由非中文字符构成的词语。



1. 问题与解决

抛出空值异常。打印哈希码，输入的“好评”与从模型中取出的“好评”的哈希码不同。

解决：IDEA的编码格式为UTF-8，Maven生成jar包的编码格式为GBK，编码格式统一为UTF-8。



1. **答辩**





