CSE313 BigData - Assessment 2

Name: Minglang.Tuo

ID NUMBER: 1508978

EMAIL: Minglang.Tuo15@STUDENT.XJTLU.EDU.CN

Program: Information and computer science

Deadline:Nov.18.2019

|  |
| --- |
| * + - 1. **My results files:** |
| File 1: top 10 bigram and occurrences  File 2: the single line contains “torture” (print the line number)  Read me:   1. Build the environment about hadoop (minimum installing)   (jdk 1.8 + ubuntu 12.04+hadoop2.7.7+eclipse 4.7.0)   1. Connect the eclipse (on the host machine) to the hadoop by eclipse-hadoop-plugin 2.7.7.0 2. Finish Task1. For example, “Eating egg is beneficial to healthy” can be divided to “ea”, “at”, “ti”, “in”,”ng”….. 3. Finish Task2. For example, the program output *the line number* contains the ”torture” |
| **2. My code with comments:** |
| 1. **package** com.annn.job1; 3. **import** java.io.IOException; 5. **import** org.apache.hadoop.fs.Path; 6. **import** org.apache.hadoop.io.IntWritable; 7. **import** org.apache.hadoop.io.Text; 8. **import** org.apache.hadoop.mapreduce.Job; 9. **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat; 10. **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; 11. **import** org.apache.hadoop.mapreduce.Reducer; 12. **import** org.apache.hadoop.mapreduce.Reducer.Context; 13. **import** org.apache.hadoop.mapreduce.lib.reduce.LongSumReducer; 14. **import** org.apache.hadoop.io.LongWritable; 15. **import** org.apache.hadoop.mapreduce.Mapper; 16. **import** org.apache.hadoop.mapreduce.Mapper.Context; 17. **import** org.apache.hadoop.yarn.util.Times;  20. **import** java.io.IOException; 21. **import** java.util.ArrayList; 22. **import** java.util.Collections; 23. **import** java.util.Comparator; 24. **import** java.util.List; 25. **import** java.net.InetAddress; 26. **import** java.text.SimpleDateFormat; 27. **import** java.util.Date;     33. **import** com.annn.test.Zmtime;    38. **public** **class** MaxTimes { 40. **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  43. Job job = **new** Job(); 44. job.setJarByClass(MaxTimes.**class**);   48. job.setJobName("output times");  51. FileInputFormat.addInputPath(job, **new** Path(args[0]));  54. FileOutputFormat.setOutputPath(job, **new** Path(args[1]));  57. job.setMapperClass(MaxTimesMapper.**class**);  60. job.setReducerClass(MaxTimesReducer.**class**);  63. job.setOutputKeyClass(Text.**class**); 65. job.setOutputValueClass(IntWritable.**class**);  68. job.waitForCompletion(**true**); 70. } 72. } 74. **class** MaxTimesReducer **extends** 75. Reducer<Text, IntWritable, Text, IntWritable>  { 77. **private** List<Zmtime> list1 = **new** ArrayList<Zmtime>(); 78. **private** List<Zmtime> list2 = **new** ArrayList<Zmtime>(); 80. **protected** **void** reduce(Text keyin, Iterable<IntWritable> valuein,Context context) 81. **throws** IOException, InterruptedException { 83. **if** (keyin.toString().length() == 2) { 84. /\* 85. \* task1 86. \*/ 88. **int** i = 0; 90. **for** (IntWritable value : valuein) { 91. i++; 92. } 93. Zmtime zmtime = **new** Zmtime(keyin.toString(),i); 94. list1.add(zmtime); 96. } **else** { 98. /\* 99. \* task2 100. \*/ 102. **for** (IntWritable value : valuein) { 103. list2.add(**new** Zmtime(keyin.toString(),Integer.valueOf(value.toString()))); 104. //context.write(keyin, value); 105. } 106. }      113. } 115. **protected** **void** cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) 116. **throws** IOException, InterruptedException { 117. //将list列表按照Zmtime对象的Times字段即当前字母对出现的次数降序排序 118. Collections.sort(list1,Comparator.comparing(Zmtime::getTimes).reversed()); 119. //将list中前十位写入context上下文，作为输出 120. **for** (**int** i = 0; i < 10; i++) { 121. Text text = **new** Text(list1.get(i).getName()); 122. context.write(text, **new** IntWritable(list1.get(i).getTimes())); 123. } 125. //context.write(new Text("cleanup部分  比较输出开始："), new IntWritable(list.size())); 126. Collections.sort(list2,Comparator.comparing(Zmtime::getTimes)); 127. **for** (Zmtime zmtime : list2) { 128. **if** (zmtime.getName().equals("torture")) { 129. context.write(**new** Text("torture所在行："), **new** IntWritable(list2.indexOf(zmtime) + 1)); 130. } 131. }   135. }  138. } 140. **class** MaxTimesMapper **extends** Mapper<LongWritable, Text, Text, IntWritable> { 142. //主机地址 143. InetAddress addr; 144. String ip ; 145. String hostname;  148. //map方法 149. //四个参数含义，按顺序分别是 输入键  输入值  输出键  输出值 150. **public** **void** map(LongWritable key, Text value, Context context) 151. **throws** IOException, InterruptedException { 153. /\* 154. \* 任务一 map 155. \*/ 156. //转换text类型到string类型 157. String teString = value.toString(); 158. //所有字母转小写 159. teString = teString.toLowerCase(); 160. //定义list 用来存放 Zmtime 对象 161. List<Zmtime> list = **new** ArrayList<Zmtime>(); 162. //列出字符串中所有相邻两字母 163. String demo = ""; 164. //循环获取该行紧挨两字符 165. **for**(**int** i = 0 ;i+2<=teString.length();i++) { 166. demo = teString.substring(i, i+2); 167. **char** c1 = demo.toCharArray()[0]; 168. //获取当前得到的两字母中第一个字母的ASCII码 169. Integer value1 = Integer.valueOf(c1); 170. //获取当前得到的两字母中第二个字母的ASCII码 171. **char** c2 = demo.toCharArray()[1]; 172. Integer value2 = Integer.valueOf(c2); 173. //判断这两个字符是否都为字母 174. **if** ((value1 > 96)&(value1 <123)&(value2 > 96)&(value2 <123)) { 175. //当前两字符都为字母，将其值存入Zmtime对象，name值为当前两字母  times 记录为1 176. Zmtime zmtime = **new** Zmtime(); 177. zmtime.setName(demo); 178. zmtime.setTimes(1); 179. //将Zmtime对象存入list 180. list.add(zmtime); 181. } 182. } 183. //遍历list 184. **for** (Zmtime zmtime : list) { 185. //将zmtime对象的name 和 Times 作为键值对 作为map的输出，reduce的输入 186. context.write(**new** Text(zmtime.getName()), **new** IntWritable(zmtime.getTimes())); 187. } 189. /\* 190. \* 任务二 191. \* 存储时，键值对为：  键 1 值为当前的偏移量 192. \* 若找到torture  键为  torture  值为 当前偏移量 193. \* 194. \*/ 196. String teString1 = value.toString(); 197. Text text1 =  **new** Text(); 198. //将当前行偏移量（输入键）作为输出值 199. IntWritable iWritable1 = **new** IntWritable(Integer.valueOf(key.toString())); 200. //列出字符串中所有相邻两字母 201. String demo1 = ""; 202. //默认输出键 为  1 203. text1 = **new** Text("1"); 204. //遍历查找 torture 205. **for**(**int** i = 0 ;i+7<=teString1.length();i++) { 206. demo1 = teString1.substring(i, i+7); 207. //当在该行找到torture时，键值改为torture，跳出循环 208. **if** (demo1.equals("torture")) { 209. text1 = **new** Text(demo1); 210. **break**; 211. } 212. } 213. //将键值对从map输出，作为reduce输入 214. context.write(text1,iWritable1); 215. }  218. /\*\* 219. \* 在map task 执行之前调用，只调用一次 220. \*/ 221. **protected** **void** setup(org.apache.hadoop.mapreduce.Mapper.Context context) 222. **throws** IOException, InterruptedException { 224. addr = InetAddress.getLocalHost(); 226. ip = addr.getHostAddress(); 228. hostname = addr.getHostName(); 229. SimpleDateFormat sdf = **new** SimpleDateFormat(); 230. sdf.applyPattern("yyyy-MM-dd hh:mm:ss"); 231. System.out.println(hostname+":"+ip + sdf.format(**new** Date()) + ":map()"); 233. } 235. }  238. **class** Zmtime { 240. **private** String name; 241. **private** **int** times; 243. **public** Zmtime() { 244. **super**(); 245. **this**.name = ""; 246. **this**.times = 0; 247. } 248. **public** Zmtime(String name, **int** times) { 249. **super**(); 250. **this**.name = name; 251. **this**.times = times; 252. } 253. **public** String getName() { 254. **return** name; 255. } 256. **public** **void** setName(String name) { 257. **this**.name = name; 258. } 259. **public** **int** getTimes() { 260. **return** times; 261. } 262. **public** **void** setTimes(**int** times) { 263. **this**.times = times; 264. }  267. } |
| 3.Hadoop command list and order you have used to run your job and get your results |
| Hadoop command list:  Master machine  Line: start-all.sh  Line: cd /mnt/hgfs/share/  Line: hadoop fs -ls -R  Line: hadoop fs -mkdir /user  Line:hadoop dfs -put pg100.txt /user/  Line:hadoop jar task\_all.jar /user/pg100.txt /user/out1  Line:hadoop fs -cat /user/out1/part-r-00000  Line:hadoop dfs -get /user/out1 /mnt/hgfs/Downloads  Results:  th 113718  he 83251  an 61317  er 59039  ou 55632  in 54012  re 44693  nd 44668  ha 44362  or 40855  1. torture所在行： 647  2. torture所在行： 2428  3. torture所在行： 3664  4. torture所在行： 5165  5. torture所在行： 8733  6. torture所在行： 9406  7. torture所在行： 22663  8. torture所在行： 22729  9. torture所在行： 23407  10. torture所在行： 23414  11. torture所在行： 23415  12. torture所在行： 23496  13. torture所在行： 33106  14. torture所在行： 42200  15. torture所在行： 43465  16. torture所在行： 44025  17. torture所在行： 44283  18. torture所在行： 44577  19. torture所在行： 44736  20. torture所在行：44769  21. torture所在行： 47634  22. torture所在行： 55229  23. torture所在行： 55808  24. torture所在行： 65580  25. torture所在行： 67932  26. torture所在行： 74098  27. torture所在行： 74102  28. torture所在行： 74158  29. torture所在行： 77114  30. torture所在行：83140  31. torture所在行： 86189  32. torture所在行： 87934  33. torture所在行： 89738  34. torture所在行： 94565  35. torture所在行： 95950  36. torture所在行： 97732  37. torture所在行： 97865  38. torture所在行： 97876  39. torture所在行： 121600  40. torture所在行：121895  41. torture所在行： 123229  You can compare the line number with the pg100.txt with the Notepad++.  For example: (647 line)    (2428 line)    (3664 line)    (5165 line)    ……. |