

XI'AN JIAOTONG-LIVERPOOL UNIVERSITY

西 交 利 物 浦 大 学

YEAR 4

COURSE WORK SUBMISSION

Name	Lu	Kai-Yu
ID Number	1614649	
Programme	Computer Science and Technology	
Module Title	Big Data Analytics	
Module Code	CSE313	
Assignment Title	Lab 2: Big Data Analysis with Hadoop	
Submission Deadline	Monday, 18 November 2019, 5:00 PM	
Lecturer Responsible	Gangmin Li	

I certify that:

- I have read and understood the University's definitions of COLLUSION and PLAGIARISM (available in the Student Handbook of Xi'an Jiaotong-Liverpool University).

With reference to these definitions, I certify that:

- I have not colluded with any other student in the preparation and production of this work;
- this document has been written solely by me and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web);
- where appropriate, I have provided an honest statement of the contributions made to my work by other people including technical and other support staff.

I understand that unauthorised collusion and the incorporation of material from other works without acknowledgement (plagiarism) are serious disciplinary offences.

SignatureKai-Yu Lu.....

Date2019.11.16.....

For Academic Office use:	Date Received	Days Late	Penalty

1. Introduction

MapReduce is one of models in programming and used for computing large data set in parallel. Map and Reduce are the two cores in MapReduce algorithm. Programmers can run programs on distributed system very conveniently without distributed programming in parallel. For Map function, it will receive a set of data and transform it in to a list containing key-value. Regarding to Reduce function, it takes the output from mapping as input and convert these pairs into smaller ones and guarantees all the mapped key-value pairs share one identical key-value pair.

In reality, it is hard for traditional computers and certain work station to carry out data analysis because it will become insufficient when the data set is huge. Therefore, a professional software called Hadoop is very powerful and has high-speed computing performance. Hadoop Distributed File System is also called HDFS, which has highly fault tolerance and quite suitable for deploy on low-cost hardware. Consequently, the throughput of the hardware would be increased in order to deal with large data set.

This assignment intends to help students get start using Hadoop to do certain data analysis on a computer cluster and write MapReduce program based on Java. The first task asks students to output the top 10 most common bigram in the dataset “pg100.txt”. The second task that students need to output the lines contain the word ‘torture’. After compiling the Java programming, students will need to submit the job (Jar file) to a Hadoop cluster and use commands of Hadoop to run on the cluster. Lastly, students will need to observe and record the running results.

1. Result

2.1 The result files.

Task 1: Output top 10 most common bigram and occurrences.

Method 1: Revert key-value and it will sort by key which is number. Then use Hadoop command to output the last 10 lines which are the top 10 most common bigram and occurrences.

Value	Key
695	that I
811	to be
822	I do
1178	my lord
1298	to the
1438	of the
1514	I will
1530	in the
1580	I have
1832	I am

Table 1: The output file for top 10 bigram and occurrences with method 1.

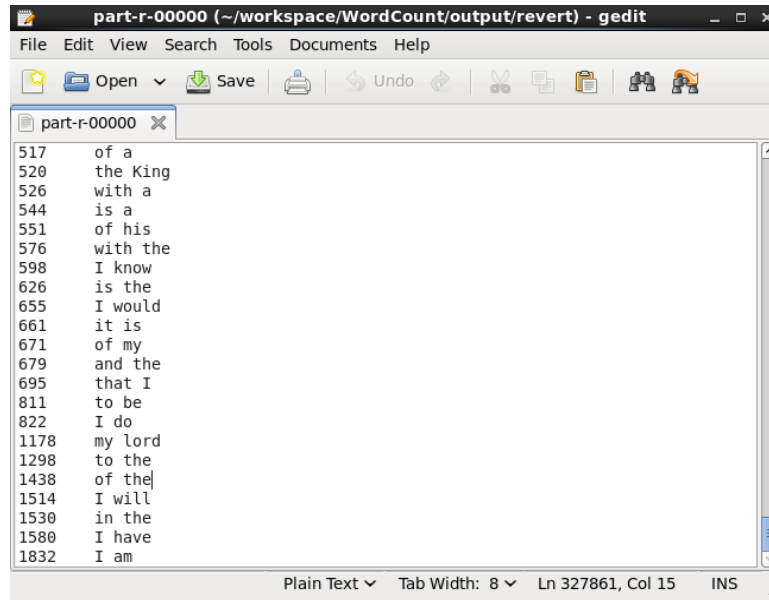


Figure 1: The screenshot for top 10 bigram and occurrences with method 1.

Method 2: Use TreeMap to save the top 10 most common bigram and occurrences.

Key	Value
that I	695
to be	811
I do	822
my lord	1178
to the	1298
of the	1438
I will	1514
in the	1530
I have	1580
I am	1832

Table 2: The output file for top 10 bigram and occurrences with method 2.

A screenshot of a gedit window titled "part-r-00000 (~/.workspace/WordCount/output/top10) - gedit". The window displays a list of bigrams and their occurrences, sorted by frequency. The bigrams are listed on the left, and their corresponding counts are on the right. The bigrams are: "that I", "to be", "I do", "my lord", "to the", "of the", "I will", "in the", "I have", and "I am". The counts are: 695, 811, 822, 1178, 1298, 1438, 1514, 1530, 1580, and 1832 respectively. The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 10, Col 13", and "INS".

```
that I 695
to be 811
I do 822
my lord 1178
to the 1298
of the 1438
I will 1514
in the 1530
I have 1580
I am 1832
```

Figure 2: The screenshot for top 10 bigram and occurrences in method 2.

Task 2: Output only the lines that contain the word ‘torture’.

A screenshot of a gedit window titled "part-r-00000 (~/.workspace/WordCount/output/findWord) - gedit". The window displays a list of lines from a text file, filtered to show only those containing the word "torture". Each line is followed by a count of 1. The lines are: "All length is torture. Since the torch is out,", "And torture him with grievous ling'ring death.", "But purgatory, torture, hell itself.", "By a sharp torture.", "Drawn on with torture.", "For torturers ingenious. It is I", "From thee to die were torture more than death.", "He may at pleasure whip or hang or torture,", "I play the torturer, by small and small", "In leads or oils? What old or newer torture", "Let hell want pains enough to torture me!", "O happy torment, when my torturer", "O, torture me no more! I will confess.", "On pain of torture, from those bloody hands", "On thy soul's peril and thy body's torture,", "Refuse me, hate me, torture me to death!", "Say he be taken, rack'd, and tortured;", "Strange tortures for offenders, never heard of,", "Than on the torture of the mind to lie", "That same Berowne I'll torture ere I go.", "That so her torture may be shortened.", "The time, the place, the torture. O, enforce it!", "This torture should be roar'd in dismal hell.", "To torture thee the more, being what thou art.", "Turning despiteous torture out of door!", "Which is our honour, bitter torture shall", "Which to be spoke would torture thee.", "With vilest torture let my life be ended.", "You go about to torture me in vain.", "curses he shall have, the tortures he shall feel, will break the", "hand, to th' infernal deep, with Erebus and tortures vile also.", "then torture my wife, pluck the borrowed veil of modesty", "Do in consent shake hands to torture me,", "FIRST SOLDIER. He calls for the tortures. What will you say without", "GEORGE. While we devise fell tortures for thy faults.", "IACHIMO. Thou'lt torture me to leave unspoken that", "Is't not enough to torture me alone,", "OTHELLO. If thou dost slander her and torture me,", "Rom. 'Tis torture, and not mercy. Heaven is here,", "SHYLOCK. I am very glad of it; I'll plague him, I'll torture him; I", "SHYLOCK. Out upon her! Thou torturest me, Tubal. It was my". The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 41, Col 66", and "INS".

```
All length is torture. Since the torch is out, 1
And torture him with grievous ling'ring death. 1
But purgatory, torture, hell itself. 1
By a sharp torture. 1
Drawn on with torture. 1
For torturers ingenious. It is I 1
From thee to die were torture more than death. 1
He may at pleasure whip or hang or torture, 1
I play the torturer, by small and small 1
In leads or oils? What old or newer torture 1
Let hell want pains enough to torture me! 1
O happy torment, when my torturer 1
O, torture me no more! I will confess. 1
On pain of torture, from those bloody hands 1
On thy soul's peril and thy body's torture, 1
Refuse me, hate me, torture me to death! 1
Say he be taken, rack'd, and tortured; 1
Strange tortures for offenders, never heard of, 1
Than on the torture of the mind to lie 1
That same Berowne I'll torture ere I go. 1
That so her torture may be shortened. 1
The time, the place, the torture. O, enforce it! 1
This torture should be roar'd in dismal hell. 1
To torture thee the more, being what thou art. 1
Turning despiteous torture out of door! 1
Which is our honour, bitter torture shall 1
Which to be spoke would torture thee. 1
With vilest torture let my life be ended. 1
You go about to torture me in vain. 1
curses he shall have, the tortures he shall feel, will break the 1
hand, to th' infernal deep, with Erebus and tortures vile also. 1
then torture my wife, pluck the borrowed veil of modesty 1
Do in consent shake hands to torture me, 1
FIRST SOLDIER. He calls for the tortures. What will you say without 1
GEORGE. While we devise fell tortures for thy faults. 1
IACHIMO. Thou'lt torture me to leave unspoken that 1
Is't not enough to torture me alone, 1
OTHELLO. If thou dost slander her and torture me, 1
Rom. 'Tis torture, and not mercy. Heaven is here, 1
SHYLOCK. I am very glad of it; I'll plague him, I'll torture him; I 1
SHYLOCK. Out upon her! Thou torturest me, Tubal. It was my 1
```

Figure 3: The screenshot for lines contain the word ‘torture’.

Figure 4 is screenshot of output files of this assignment. Inside each of them is corresponding text files.



Figure 4: The output files of this assignment

2. Code with comments

```
1.  /*This is assignment 2: Big Data Analysis with Hadoop.
2.   * Student: Kai-Yu Lu. ID: 1614649.
3.   * Task 1: Output 10 bigram and occurrences.
4.   * Task 2: Output the line contains "torture".
5.   *
6.   * -----Task 1-----
7.   * Method 1:
8.   * A. Use regular expression to filter out unnecessary punctuation.
9.   * B. Reverse the key and value in text file completed in Method A.
10.  * C. Hadoop will sort the top 10 bigrams from small to large according to the
11.  *     value.
12.  *     Then use the command of hadoop:
13.  *     $hadoop fs -cat kaiyu16/revert/part-r-00000 | tail -n 10
14.  *     to output top 10 bigrams and occurrences.
15.  *
16.  * Method 2:
17.  * A. Use regular expression to filter out unnecessary punctuation.
18.  * B. Reverse the key and value in text file completed in Method A and use
19.  *     TreeMap to filter top 10 bigrams and occurrences and then use Reducer to
20.  *     reverse value and key again.
21.  * C. Output the top 10 bigrams and occurrences by using command of Hadoop:
22.  *     $hadoop fs -cat kaiyu16/top10/part-r-00000
23.  *
24.  * -----Task 2-----
25.  * A. Read pg100.txt and record the lines having "torture".
26.  * B. Output the lines having "torture" by using command of Hadoop:
27.  *     $hadoop fs -cat kaiyu16/findWord/part-r-00000.
28.  */
29. package edu.xjtlu.cse313.wordcount;
30.
31. import java.io.IOException;
32. import java.util.Arrays;
33. import java.util.TreeMap;
34. import org.apache.hadoop.conf.Configuration;
35. import org.apache.hadoop.conf.Configured;
36. import org.apache.hadoop.fs.Path;
37. import org.apache.hadoop.io.IntWritable;
38. import org.apache.hadoop.io.LongWritable;
```

```
39. import org.apache.hadoop.io.Text;
40. import org.apache.hadoop.mapreduce.Job;
41. import org.apache.hadoop.mapreduce.Mapper;
42. import org.apache.hadoop.mapreduce.Reducer;
43. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
44. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
45. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
46. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
47. import org.apache.hadoop.util.Tool;
48. import org.apache.hadoop.util.ToolRunner;
49.
50. public class WordCount extends Configured implements Tool {
51.
52.     public static void main(String[] args) throws Exception {
53.         System.out.println(Arrays.toString(args));
54.         int res = ToolRunner.run(new Configuration(), new WordCount(), args);
55.
56.         System.exit(res);
57.     }
58.
59.     @Override
60.     public int run(String[] args) throws Exception {
61.         System.out.println(Arrays.toString(args));
62.         Job filterJob = new Job(getConf(), "WordCount"); //Initialize filterJob.
63.         Job revertJob = new Job(getConf(), "WordCount"); //Initialize revertJob.
64.         Job top10Job = new Job(getConf(), "WordCount"); //Initialize top10Job.
65.         Job wordJob = new Job(getConf(), "WordCount"); //Initialize wordJob.
66.
67.         //Setting drivers.
68.         filterJob.setJarByClass(WordCount.class);
69.         filterJob.setOutputKeyClass(Text.class);
70.         filterJob.setOutputValueClass(IntWritable.class);
71.         filterJob.setMapperClass(filterMap.class);
72.         filterJob.setReducerClass(filterReduce.class);
73.         filterJob.setInputFormatClass(TextInputFormat.class);
74.         filterJob.setOutputFormatClass(TextOutputFormat.class);
75.         //The locations of I/O files.
76.         FileInputFormat.addInputPath(filterJob, new Path(args[0]));
77.         FileOutputFormat.setOutputPath(filterJob, new Path(args[1]+"/filter"));
78.         filterJob.waitForCompletion(true);
79.
80.         //Setting drivers.
81.         revertJob.setJarByClass(WordCount.class);
82.         revertJob.setOutputKeyClass(IntWritable.class);
83.         revertJob.setOutputValueClass(Text.class);
84.         revertJob.setMapperClass(revertMap.class);
85.         revertJob.setInputFormatClass(TextInputFormat.class);
86.         revertJob.setOutputFormatClass(TextOutputFormat.class);
```

```

87.     //The locations of I/O files.
88.     FileInputFormat.addInputPath(revertJob, new Path(args[1]+"/filter"));
89.     FileOutputFormat.setOutputPath(revertJob, new Path(args[1]+"/revert"));
90.     revertJob.waitForCompletion(true);
91.
92.     //Setting drivers.
93.     top10Job.setJarByClass(WordCount.class);
94.     top10Job.setOutputKeyClass(IntWritable.class);
95.     top10Job.setOutputValueClass(Text.class);
96.     top10Job.setMapperClass(sortMap.class);
97.     top10Job.setReducerClass(sortReduce.class);
98.     top10Job.setInputFormatClass(TextInputFormat.class);
99.     top10Job.setOutputFormatClass(TextOutputFormat.class);
100.    //The locations of I/O files.
101.    FileInputFormat.addInputPath(top10Job, new Path(args[1]+"/filter"));
102.    FileOutputFormat.setOutputPath(top10Job, new Path(args[1]+"/top10"));
103.    top10Job.waitForCompletion(true);
104.
105.    //Setting drivers.
106.    wordJob.setJarByClass(WordCount.class);
107.    wordJob.setOutputKeyClass(Text.class);
108.    wordJob.setOutputValueClass(IntWritable.class);
109.    wordJob.setMapperClass(wordMap.class);
110.    wordJob.setReducerClass(wordReduce.class);
111.    wordJob.setInputFormatClass(TextInputFormat.class);
112.    wordJob.setOutputFormatClass(TextOutputFormat.class);
113.    //The locations of I/O files.
114.    FileInputFormat.addInputPath(wordJob, new Path(args[0]));
115.    FileOutputFormat.setOutputPath(wordJob, new Path(args[1]+"/findWord"));
116.    wordJob.waitForCompletion(true);
117.
118.    return 0;
119. }
120.
121. //This is for filtering out unnecessary punctuation.
122. public static class filterMap extends Mapper<LongWritable, Text, Text, IntWritable> {
123.     private final static IntWritable ONE = new IntWritable(1);
124.     //Initialize num for adding numbers of showing times.
125.     private Text word = new Text(); //Initialize word for storing key.
126.
127.     @Override
128.     public void map(LongWritable key, Text value, Context context)
129.         throws IOException, InterruptedException {
130.         //Filtering out unnecessary punctuation for every line.
131.         String lines = value.toString().replaceAll("[~!\"@#%&*\\]()<>_+\\[=,\\.\\'\"?{}/:;]", "")
132.             .replaceAll("-", " ").trim();
133.         String [] wdlist = lines.split("\\s+");//Split value and key if any space is between t
134.             hem.

```



```

133.         for(int i=0; i<wdlist.length-1;i++){
134.             word.set(wdlist[i]+" "+wdlist[i+1]);
135.             context.write(word, ONE); //Obtain bgrams and the numbers.
136.         }
137.     }
138. }
139.
140. public static class filterReduce extends Reducer<Text, IntWritable, Text, IntWritable> {
141.
142.     @Override
143.     public void reduce(Text key, Iterable<IntWritable> values, Context context)
144.         throws IOException, InterruptedException {
145.         int sum = 0;
146.         for (IntWritable val : values) {
147.             sum += val.get();
148.         }
149.         context.write(key, new IntWritable(sum));
150.     }
151. }
152.
153. // This is for the Task 1 by using Method 1.
154. public static class revertMap extends Mapper<LongWritable, Text, IntWritable,Text> {
155.
156.     private Text word = new Text(); //Initialize word for storing key.
157.     private static IntWritable num = new IntWritable();
158.     //Initialize num for storing value.
159.
160.     @Override
161.     public void map(LongWritable key, Text value, Context context)
162.         throws IOException, InterruptedException {
163.         //Obtain every line and the type is String.
164.         String lines = value.toString();
165.         //Split value and key if a "\t" is between them.
166.
167.         String [] wdlist = lines.split("\t");
168.         //Change type from String to Int.
169.         num.set(Integer.parseInt(wdlist[1]));
170.         word.set(wdlist[0]); //Obtain bigram.
171.         context.write(num,word); // Realize convert key and value.
172.     }
173. }
174.
175. // This is for the Task 1 by using Method 2.
176. public static class sortMap extends Mapper<LongWritable, Text, IntWritable,Text> {
177.     private TreeMap<Integer, String> map = new TreeMap<Integer, String>();
178.     public static int K = 10;
179.     private static IntWritable num = new IntWritable();
180.

```

```

181.         @Override
182.         protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, IntWritable,Text>.Context context)
183.             throws IOException, InterruptedException {
184.             //Obtain every line and the type is String.
185.             String lines = value.toString();
186.             //Split value and key if a "\t" is between them.
187.             String [] wdlist = lines.split("\t");
188.             String bigram = wdlist[0].trim(); //Obtain bigram.
189.             //Change type from String to Int.
190.             Integer num = Integer.parseInt(wdlist[1]);
191.             map.put(num,bigram);
192.             // Filter out the top 10.
193.             if(map.size() > K){
194.                 map.remove(map.firstKey());
195.             }
196.
197.         }
198.         @Override
199.         protected void cleanup(Mapper<LongWritable, Text, IntWritable,Text>.Context context) throws IOException, InterruptedException{
200.             //Save the top 10 bigrams and occurrences.
201.             for(Integer num : map.keySet()){
202.                 context.write(new IntWritable(num), new Text(map.get(num)));
203.             }
204.         }
205.     }
206.     public static class sortReduce extends Reducer<IntWritable,Text,Text,IntWritable> {
207.         private Text result = new Text();
208.
209.         @Override
210.         public void reduce(IntWritable key, Iterable<Text> values, Context context)
211.             throws IOException, InterruptedException {
212.
213.             for (Text val : values) {
214.                 result.set(val.toString());
215.                 context.write(result, key);
216.             }
217.         }
218.     }
219.
220.     // This is for the Task 2.
221.     public static class wordMap extends Mapper<LongWritable, Text, Text, IntWritable> {
222.         private final static IntWritable ONE = new IntWritable(1);
223.         private Text word = new Text();
224.
225.         @Override
226.         public void map(LongWritable key, Text value, Context context)

```

```

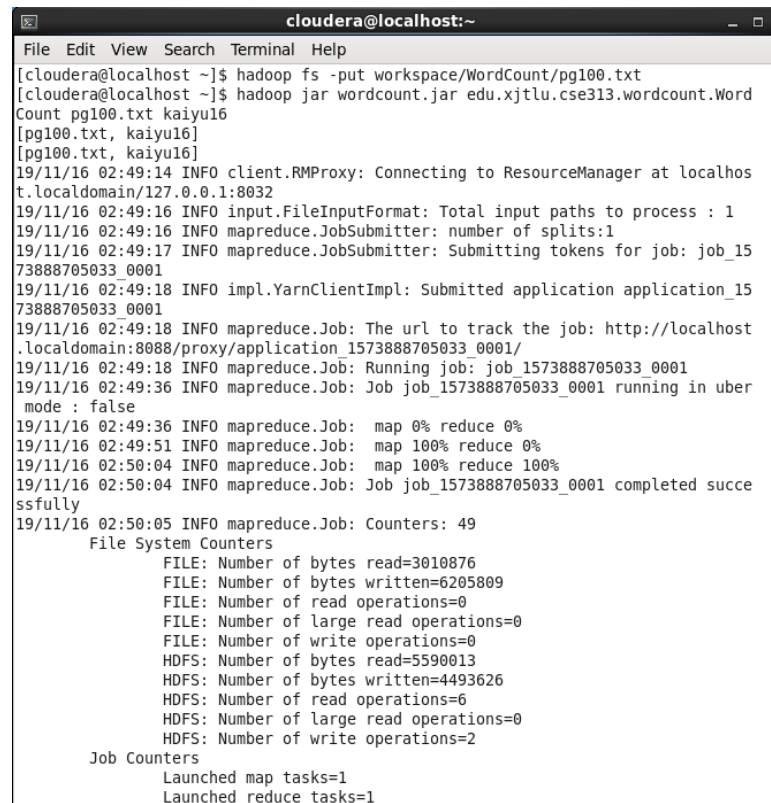
227.         throws IOException, InterruptedException {
228.
229.         String lines = value.toString(); //Obtain every line and the type is String.
230.
231.         //Find the lines containing "torture" and store the line.
232.         if(lines.contains("torture")){
233.             word.set(lines); //Obtain bigram.
234.             context.write(word, ONE);
235.         }
236.     }
237.
238.     //This part is optional and used to examine if there have identical lines.
239.     public static class wordReduce extends Reducer<Text, IntWritable, Text, IntWritable>
240.     {
241.         @Override
242.         public void reduce(Text key, Iterable<IntWritable> values, Context context)
243.         throws IOException, InterruptedException {
244.             for (IntWritable val: values) {
245.                 context.write(key, val);
246.             }
247.         }
248.     }

```

3. Hadoop command list and results of running job.

Command 1: `hadoop fs -put workspace/WordCount/pg100.txt`

Command 2: `hadoop jar wordcount.jar edu.xjtlu.cse313.wordcount.WordCount pg100.txt kaiyu16`



```

cloudera@localhost:~
File Edit View Search Terminal Help
[cloudera@localhost ~]$ hadoop fs -put workspace/WordCount/pg100.txt
[cloudera@localhost ~]$ hadoop jar wordcount.jar edu.xjtlu.cse313.wordcount.Word
Count pg100.txt kaiyu16
[pg100.txt, kaiyu16]
[pg100.txt, kaiyu16]
19/11/16 02:49:14 INFO client.RMProxy: Connecting to ResourceManager at localhos
t.localdomain/127.0.0.1:8032
19/11/16 02:49:16 INFO input.FileInputFormat: Total input paths to process : 1
19/11/16 02:49:16 INFO mapreduce.JobSubmitter: number of splits:1
19/11/16 02:49:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
73888705033_0001
19/11/16 02:49:18 INFO impl.YarnClientImpl: Submitted application application_15
73888705033_0001
19/11/16 02:49:18 INFO mapreduce.Job: The url to track the job: http://localhost
.localdomain:8088/proxy/application_1573888705033_0001/
19/11/16 02:49:18 INFO mapreduce.Job: Running job: job_1573888705033_0001
19/11/16 02:49:36 INFO mapreduce.Job: Job job_1573888705033_0001 running in uber
mode : false
19/11/16 02:49:36 INFO mapreduce.Job: map 0% reduce 0%
19/11/16 02:49:51 INFO mapreduce.Job: map 100% reduce 0%
19/11/16 02:50:04 INFO mapreduce.Job: map 100% reduce 100%
19/11/16 02:50:04 INFO mapreduce.Job: Job job_1573888705033_0001 completed succe
ssfully
19/11/16 02:50:05 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=3010876
        FILE: Number of bytes written=6205809
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=5590013
        HDFS: Number of bytes written=4493626
        HDFS: Number of read operations=6
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=1
        Launched reduce tasks=1

```

Figure 5: Result after doing command 1 and command 2.

Command 3: `hadoop fs -ls kaiyu16`

```
[cloudera@localhost ~]$ hadoop fs -ls kaiyu16
Found 4 items
drwxr-xr-x  - cloudera cloudera      0 2019-11-16 02:50 kaiyu16/filter
drwxr-xr-x  - cloudera cloudera      0 2019-11-16 02:52 kaiyu16/findWord
drwxr-xr-x  - cloudera cloudera      0 2019-11-16 02:50 kaiyu16/revert
drwxr-xr-x  - cloudera cloudera      0 2019-11-16 02:51 kaiyu16/top10
```

Figure 6: Result after doing command 3.

Task 1: Output top 10 most common bigram and occurrences.

Method 1 & Command 4: `hadoop fs -cat kaiyu16/revert/part-r-00000 | tail -n 10`

```
[cloudera@localhost ~]$ hadoop fs -cat kaiyu16/revert/part-r-00000 | tail -n 10
695      that I
811      to be
822      I do
1178     my lord
1298     to the
1438     of the
1514     I will
1530     in the
1580     I have
1832     I am
```

Figure 7: Result after doing command 4.

Method 2 & Command 5: `hadoop fs -cat kaiyu16/top10/part-r-00000`

```
[cloudera@localhost ~]$ hadoop fs -cat kaiyu16/top10/part-r-00000
that I 695
to be 811
I do 822
my lord 1178
to the 1298
of the 1438
I will 1514
in the 1530
I have 1580
I am 1832
```

Figure 8: Result after doing command 5.

Task 2: Output only the lines that contain the word ‘torture’.

Command 6: `hadoop fs -cat kaiyu16/findWord/part-r-00000`

```

[cloudera@localhost ~]$ hadoop fs -cat kaiyul6/findWord/part-r-00000
All length is torture. Since the torch is out,      1
And torture him with grievous ling'ring death.      1
But purgatory, torture, hell itself.                1
By a sharp torture. 1
Drawn on with torture.      1
For torturers ingenious. It is I      1
From thee to die were torture more than death.      1
He may at pleasure whip or hang or torture, 1
I play the torturer, by small and small      1
In leads or oils? What old or newer torture 1
Let hell want pains enough to torture me!  1
O happy torment, when my torturer      1
O, torture me no more! I will confess.      1
On pain of torture, from those bloody hands 1
On thy soul's peril and thy body's torture, 1
Refuse me, hate me, torture me to death!      1
Say he be taken, rack'd, and tortured;      1
Strange tortures for offenders, never heard of,    1
Than on the torture of the mind to lie      1
That same Berowne I'll torture ere I go.      1
That so her torture may be shortened.      1
The time, the place, the torture. O, enforce it!    1
This torture should be roar'd in dismal hell.      1
To torture thee the more, being what thou art.      1
Turning despiteous torture out of door!      1
Which is our honour, bitter torture shall      1
Which to be spoke would torture thee.      1
With vilest torture let my life be ended.      1
You go about to torture me in vain. 1
curses he shall have, the tortures he shall feel, will break the      1
hand, to th' infernal deep, with Erebus and tortures vile also.      1
then torture my wife, pluck the borrowed veil of modesty      1
Do in consent shake hands to torture me,      1
FIRST SOLDIER. He calls for the tortures. What will you say without      1
GEORGE. While we devise fell tortures for thy faults. 1
IACHIMO. Thou'lt torture me to leave unspoken that      1
Is't not enough to torture me alone,      1
OTHELLO. If thou dost slander her and torture me,      1
Rom. 'Tis torture, and not mercy. Heaven is here,      1
SHYLOCK. I am very glad of it; I'll plague him, I'll torture him; I      1
SHYLOCK. Out upon her! Thou torturest me, Tubal. It was my      1
[cloudera@localhost ~]$ █

```

Figure 9: Result after doing command 6.

4. Conclusion

In conclusion, this assignment could complete the two tasks without any bugs. It is worth mentioned that there were two methods for task 1. For the first method, it reverted the position of bigrams and values, then it would become number-bigram. Consequently, it will sort the pairs by number. Finally, Hadoop command to output last 10 lines could be used to print top 10 bigram and occurrences. This method combined the propertied of MapReduce and Hadoop, which is quite flexible. Another method was to use TreeMap to save the top 10 bigrams and occurrences without reverting the position of bigrams and values. Finally, the concepts of MapReduce and Hadoop have been understood further after this assignment.