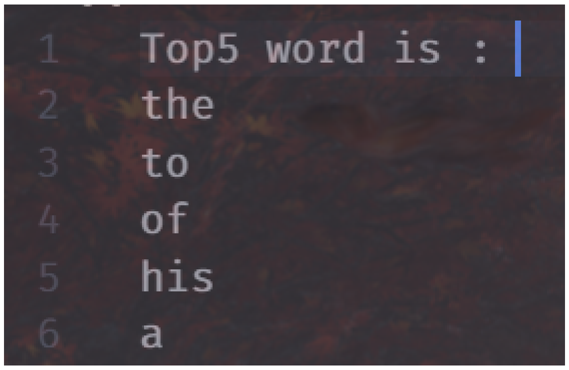
2020春《大数据数据结构设计与实践》作业三

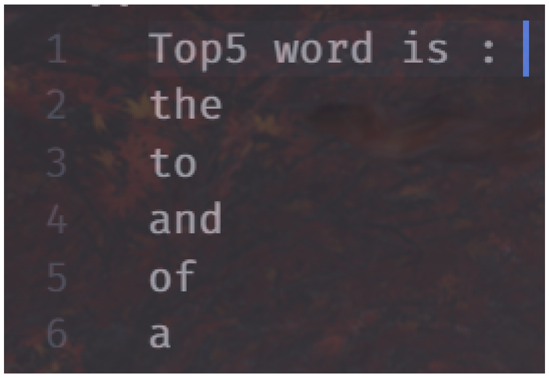
姓名: 许天骁 学号: 1171000405 班号: 1703109

词频topk

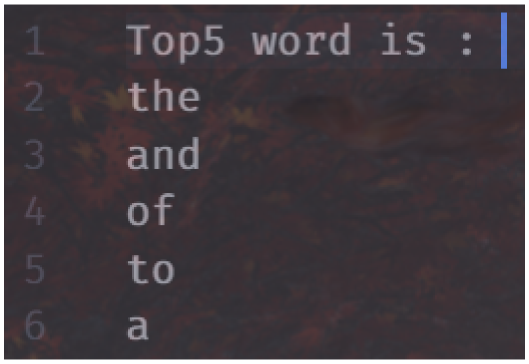
1. 程序使用环境
   配置：X86-64 8-core processor; 16GB RAM; 256GB ROM
   系统：Ubuntu 19.10
   开发框架：Hadoop 2.10（伪分布式模式）
   Java：OpenJDK version "11.0.5"; OpenJDK 64-Bit Server VM;
   IDE：InteliJ IDEA
2. 设计思路
   对于词频统计topk利用mapreduce算法进行计算，首先需要进行词频统计（wordcount），然后将wordcount的统计结果输出到文件之中。其中每一行两个元素，分别是单词（word）以及出现的次数（num）。之后运行另一个mapreduce程序，读取wordcount生成的文件。接下来分为两个阶段：
   map阶段：
   利用java自带的二叉搜索树，在map的过程中，将数据构造成大小小于K的树。即每读入一个单词，将其与其出现次数插入到树中，在每次map后判断树的大小和K的大小，当树的数据量大于K时，取出最小的数。在map方法结束后会执行cleanup方法，该方法将map任务中的前K个数据传入reduce任务中.
   reduce阶段：
   在reduce阶段中，依次将map方法中传入的K个数据放入java自带的二叉搜索树中，并依靠平衡特性来维持数据的有序性。从而将K个数据利用二叉搜索树的firstKey方法按从大到小或者利用二叉搜索树的lastKey方法按从小到大的顺序排列。从而求出前K个数。
3. 实现
   源代码均放在报告的结尾。
   首先是wordcount，然后将该文件作为topk程序的输入文件名，运行topk程序。得到topk结果，这里取k为5，得到四个文件中的top5单词。



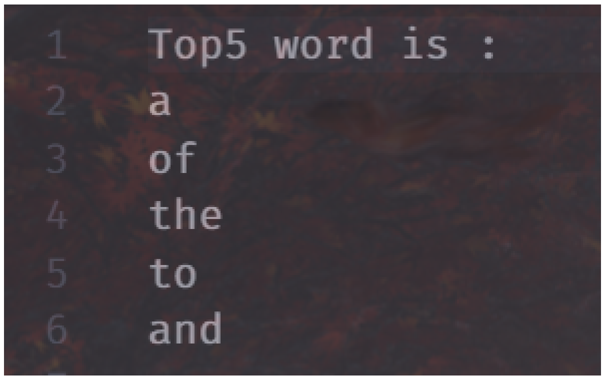
```
Top5 word is :
the
to
of
his
a
```



```
Top5 word is :
the
to
and
of
a
```



```
Top5 word is :
the
and
of
to
a
```



```
Top5 word is :
a
of
the
to
and
```

共同粉丝

1. 程序使用环境
   与词频统计topk相同
2. 设计思路
   同样需要两边mapreduce，首先求得某一个人是哪些人的粉丝，比如B是A,E,F,J的粉丝。这是第一步需要求的结果。第二步进行两两配对，即A，E的共同粉丝有B。A，F的共同粉丝有B。然后在reduce阶段进行合并。
3. 实现
   源代码放在报告的结尾
   首先求出某一人是那些人的粉丝，然后将该文件作为计算共同粉丝的输入文件，得到所有人的共同粉丝。文件放在另一个文档（friend.txt）中。

源代码

仅展示MapReduce关键代码

```
public static class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {
        IntWritable one = new IntWritable(1);
        Text word = new Text();

        public void map(Object key, Text value, Context context) throws IOException,InterruptedE
                StringTokenizer itr = new StringTokenizer(value.toString());
                while(itr.hasMoreTokens()) {
                        word.set(itr.nextToken());
                        context.write(word, one);
                }
        }
}

public static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOExc
                int sum = 0;
                for(IntWritable val:values) {
                        sum += val.get();
                }
                result.set(sum);
                context.write(key,result);
        }
}

public static class topkMapper extends Mapper<LongWritable, Text, NullWritable, LongWritable> {
   public static final int K = 5;
   private TreeMap<Long, Long> tm = new TreeMap<Long, Long>();

   protected void map( LongWritable key, Text value, Mapper<LongWritable, Text, NullWritable, Lo
         throws java.io.IOException, InterruptedException {
      try {
         long temp = Long.parseLong(value.toString().trim());
         tm.put(temp, temp);
         if (tm.size() > K) {
            tm.remove(tm.firstKey());
         }
      } catch (Exception e) {
         context.getCounter("TopK", "errorLog").increment(1L);
      }
   };

   protected void cleanup(
         org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, NullWritable, LongWritable>.C
         throws java.io.IOException, InterruptedException {
      for (Long num : tm.values()) {
         context.write(NullWritable.get(), new LongWritable(num));
      }
   };
}
```

```
public static class topkReducer extends Reducer<NullWritable, LongWritable, NullWritable, LongWr
   public static final int K = 5;
   private TreeMap<Long, Long> tm = new TreeMap<Long, Long>();

   protected void reduce(
            NullWritable key,
            java.lang.Iterable<LongWritable> values,
            Reducer<NullWritable, LongWritable, NullWritable, LongWritable>.Context context)
            throws java.io.IOException, InterruptedException {
      for (LongWritable num : values) {
            tm.put(num.get(), num.get());
            if (tm.size() > K) {
               tm.remove(tm.firstKey());
            }
      }
      for (Long value : tm.descendingKeySet()) {
            context.write(NullWritable.get(), new LongWritable(value));
      }
   };
}
```

```java
public static class Friends1Mapper extends Mapper<LongWritable, Text, Text, Text>{
   Text keyText = new Text();
   Text valueText = new Text();
   @Override
   protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
      String line = value.toString();
      String person = line.split(":")[0];
      String content = line.split(":")[1];
      String[] fans = content.split(",");
      valueText.set(person);
      for (int i = 0; i < fans.length; i++) {
            keyText.set(fans[i]);
            context.write(keyText, valueText);
      }
   }
}

public static class Friends1Reducer extends Reducer<Text, Text, Text, Text>{
   Text valueText = new Text();
   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
      StringBuffer sb = new StringBuffer();
      for (Text fan : values) {
            sb.append(fan).append(",");
      }
      String outFans = sb.substring(0, sb.length()-1);
      valueText.set(outFans);
      context.write(key, valueText);
   }
}

public static class Friends2Mapper extends Mapper<LongWritable, Text, Text, Text> {
   Text keyText = new Text();
   Text valueText = new Text();

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException, Interru
      String line = value.toString();
      String fan = line.split("\t")[0];
      String content = line.split("\t")[1];
      String[] persons = content.split(",");
      Arrays.sort(persons);
      valueText.set(fan);
      for (int i = 0; i < persons.length; i++) {
         for (int j = i + 1; j < persons.length; j++) {
            keyText.set(persons[i] + "," + persons[j]);
            context.write(keyText, valueText);
         }
      }
   }

public static class ShareFriendsStepTwoReducer extends Reducer<Text, Text, Text, Text> {
   Text valueText = new Text();

   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
      StringBuffer sb = new StringBuffer();
      sb.append("[");
      for (Text fan : values) {
            sb.append(fan).append(",");
      }
      sb.append("]");
      sb.deleteCharAt(sb.length()-2);
      valueText.set(sb.toString());
      context.write(key, valueText);
   }
}
```