2020春《大数据数据结构设计与实践》作业三

姓名: 崔津浩 学号: 1171000317 班号: 1710101

配置环境介绍

配置：X86-64 8-core processor; 16GB RAM; 1T ROM
系统：Ubuntu 19.10
开发框架：Hadoop 2.10
Java：OpenJDK version "11.0.5";

词频topk

1. 设计思路
   第一步Map将所有单词拆分出来传递给reduce，在ReduceTask中接受单词，计数，然后利用java的sort进行排序。
2. 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。分别对应四个文件（outp_.txt）中的内容。

共同粉丝

1. 设计思路
   需要两边mapreduce，首先求得某一个人是哪些人的粉丝。即从反向来寻找共同粉丝。第一步到此结束，将阶段性结果输入到文件中。第二步进行两两配对，即从第一步的计算结果中，将每一个人的集合中的元素进行两两配对。然后在reduce阶段进行合并。
2. 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（friends_output.txt）中。

源代码

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import org.apache.commons.io.FileUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
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.output.FileOutputFormat;

import com.alibaba.fastjson.JSON;


public class TopN1 {
	public static class MapTask extends Mapper<LongWritable, Text, Text, MovieBean>{
		@Override
		protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text
				throws IOException, InterruptedException {
			try {
				MovieBean movieBean = JSON.parseObject(value.toString(), MovieBe
				String movie = movieBean.getMovie();
				context.write(new Text(movie), movieBean);
			} catch (Exception e) {

			}
		}
	}

	public static class ReduceTask extends Reducer<Text, MovieBean, MovieBean, NullWritable>
		@Override
		protected void reduce(Text movieId, Iterable<MovieBean> movieBeans,
				Reducer<Text, MovieBean, MovieBean, NullWritable>.Context contex
				throws IOException, InterruptedException {
			List<MovieBean> list = new ArrayList<>();

			for (MovieBean movieBean : movieBeans) {
				MovieBean movieBean2 = new MovieBean();
				movieBean2.set(movieBean);
				list.add(movieBean2);
			}
			Collections.sort(list, new Comparator<MovieBean>() {
```

```
                        @Override
                        public int compare(MovieBean o1, MovieBean o2) {

                                return o2.getRate() - o1.getRate();
                        }
                });
                for (int i = 0; i < Math.min(20, list.size()); i++) {
                        context.write(list.get(i), NullWritable.get());
                }
        }
}

public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "avg");

        //设置map和reduce，以及提交的jar
        job.setMapperClass(MapTask.class);
        job.setReducerClass(ReduceTask.class);
        job.setJarByClass(TopN1.class);

        //设置输入输出类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(MovieBean.class);

        job.setOutputKeyClass(MovieBean.class);
        job.setOutputValueClass(NullWritable.class);

        //输入和输出目录
        FileInputFormat.addInputPath(job, new Path("D:\\Documents\\大数据创新实验\\第3次课
        FileOutputFormat.setOutputPath(job, new Path("D:\\Documents\\大数据创新实验\\第3次

        //判断文件是否存在
        File file = new File("D:\\Documents\\大数据创新实验\\第3次课\\第3次课\\wordcount\\
        if(file.exists()){
                FileUtils.deleteDirectory(file);
        }

        //提交任务
        boolean completion = job.waitForCompletion(true);
        System.out.println(completion?"你很优秀！！！":"滚去调bug！！");

}
}
```

```
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);
    }
}
```