原本想试试BDKit，后来被劝退还是用了伪分布，等一个大家都不用BDKit的时间再试试(#^.^#)

设计思路

原先有两个Map和Reduce类

第一个负责统计词频并进行筛选

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

        private Set<String> stopwords;
        private String localFiles;

        @Override
        public void setup(Context context) throws
IOException,InterruptedException{
            stopwords = new TreeSet<String>();
            // 获取在main函数中设置的conf配置文件
            Configuration conf = context.getConfiguration();
            // 获取停词表所在的hdfs路径
            localFiles  = conf.getStrings("stopwords")[0];
            FileSystem fs = FileSystem.get(URI.create(localFiles), conf);
            FSDataInputStream hdfsInStream = fs.open(new Path(localFiles));
            // 从hdfs中读取
            InputStreamReader isr = new InputStreamReader(hdfsInStream, "utf-
8");
            String line;
            BufferedReader br = new BufferedReader(isr);
            while ((line = br.readLine()) != null) {
                StringTokenizer itr = new StringTokenizer(line);
                while (itr.hasMoreTokens()) {
                    // 得到停词表
                    stopwords.add(itr.nextToken());
                }
            }
        }

        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
            // 非字母部分替换为空格，忽略标点和数字
            String pattern = "[^a-zA-Z]";
            String line = value.toString();
            line = line.replaceAll(pattern," ");

            StringTokenizer itr = new StringTokenizer(line);

            while (itr.hasMoreTokens()) {
                String strVal=itr.nextToken();
                // 每个单词大写转小写
                strVal=strVal.toLowerCase();
                // 单词长度须≥3且不在停词表内
                if(strVal.length()>=3 & !stopwords.contains(strVal)){
```

```
                    word.set(strVal);
                    context.write(word, one);
                }
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

        private TreeSet<Item> tree = new TreeSet<Item>();
        private IntWritable result = new IntWritable();

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

第二个负责先根据词频排序，在词频相同的情况下再根据字母序排序

```
public static class Map2 extends Mapper<LongWritable,Text,IntWritable,Text>
    {
        @Override
        public void map(LongWritable key,Text value,Context context) throws
IOException,InterruptedException
        {
            // 读取第一个mapreduce的结果，通过制表符将键和值分开
            String[] data = value.toString().split("\t");
            // 将词频作为键，单词作为值
            context.write(new IntWritable(Integer.parseInt(data[1])),new
Text(data[0]));
        }
    }

    static int no=1;
    public static class Reduce2 extends
Reducer<IntWritable,Text,Text,IntWritable>
    {
        IntWritable result = new IntWritable();
        @Override
        public void reduce(IntWritable key,Iterable<Text> values,Context
context) throws IOException,InterruptedException
        {
            // 相同词频的单词发送到一个reduce上，则只需要将相同词频的单词在第二个reduce中按
字母序排列即可
            List<String> sort = new ArrayList<String>();
            for(Text value : values){
                sort.add(value.toString());
            }
            String[] strings = new String[sort.size()];
```

```
            sort.toArray(strings);
            // 对单词按照字母序排序
            Arrays.sort(strings);
            for (int i = 0;i<strings.length;i++){
                context.write(new Text(no+": "+strings[i]+", "), key);
                no+=1;
            }
        }
    }

    // 对第二个mapreduce中map的key进行排序，实现降序排列

    public static class Sort extends IntWritable.Comparator{
        public int compare(Writable a, Writable b){
            return -super.compare(a, b);
        }
        public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2)
{
            return -super.compare(b1, s1, l1, b2, s2, l2);
        }
    }
```

前者的输出作为后者的输入（下面是main）

```
public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf1 = new Configuration();
        // job1,统计词频并筛选

conf1.setStrings("stopwords","hdfs://localhost:9000/user/sheepxi/input/stop-
word-list.txt");
        Job job1 = Job.getInstance(conf1, "word count 1.0");
        job1.setJarByClass(WordCount.class);
        job1.setMapperClass(TokenizerMapper.class);
        job1.setCombinerClass(IntSumReducer.class);
        job1.setReducerClass(IntSumReducer.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job1, new
Path("hdfs://localhost:9000/user/sheepxi/input"));
        FileOutputFormat.setOutputPath(job1, new
Path("hdfs://localhost:9000/user/sheepxi/output"));

        ControlledJob ctrlJob1 = new ControlledJob(conf1);
        ctrlJob1.setJob(job1);

        // job2,将词频按照降序排列，并且相同词频的单词按照字母序排列
        Configuration conf2 = new Configuration(true);
        Job job2= Job.getInstance(conf2,"sort");
        job2.setJarByClass(WordCount.class);
        job2.setMapperClass(Map2.class);
        job2.setReducerClass(Reduce2.class);
        FileInputFormat.addInputPath(job2,new
Path("hdfs://localhost:9000/user/sheepxi/output"));
        job2.setOutputKeyClass(IntWritable.class);
        job2.setOutputValueClass(Text.class);
        job2.setSortComparatorClass(Sort.class);
```

```
        FileOutputFormat.setOutputPath(job2,new Path
      ("hdfs://localhost:9000/user/sheepxi/output2"));
        ControlledJob ctrlJob2 = new ControlledJob(conf2);
        ctrlJob2.setJob(job2);
        ctrlJob2.addDependingJob(ctrlJob1);
        JobControl jobCtrl = new JobControl("myCtrl");
        jobCtrl.addJob(ctrlJob1);
        jobCtrl.addJob(ctrlJob2);

        Thread thread = new Thread(jobCtrl);
        thread.start();
        while (true) {
            if (jobCtrl.allFinished()) {
                System.out.println(jobCtrl.getSuccessfulJobList());
                jobCtrl.stop();
                break;
            }
        }
    }
}
```

后来发现第二个不方便加上输出前100个词的条件，就在第一个Reduce中用TreeSet排了序（见下），其他结构还是原样保留了，只在第二个Reduce里对输出格式做了修改

```
    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

        private TreeSet<Item> tree = new TreeSet<Item>();
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context
context)
                throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            Item item= new Item(key.toString(), (long)sum);
            if (tree.size()<100||tree.last().num<item.num){
                tree.add(item);
            }
            if (tree.size()>100){
                tree.pollLast();
            }
        }

        @Override
        protected void cleanup(Context context) throws IOException,
InterruptedException{
            while(!tree.isEmpty()){
                result.set((int)tree.first().num);
                context.write(new Text(tree.first().value), result);
                tree.pollFirst();
            }
        }
    }
```

实验结果（截图仅展示前几个）

完整结果见wc/WordCount/output2/part-r-00000文件

```
sheepxi@ubuntu:~/bdkit-demo/wc/WordCount$ hadoop fs -cat output2/part-r-00000
2020-10-28 06:22:58,724 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
2020-10-28 06:23:00,658 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false,
1: scene,       10241
2: thou,        9438
3: thy,         6592
4: shall,       6398
5: king,        6254
6: lord,        5702
7: sir,         5530
8: thee,        5381
9: good,        5123
10: come,       4473
11: enter,      4252
12: act,        4109
13: let,        4084
14: love,       3596
15: man,        3565
16: hath,       3379
17: like,       3346
18: henry,      3304
19: say,        3057
20: know,       3028
21: make,       2891
22: did,        2844
23: shakespeare,        2664
```

运行截图

```
sheepxi@ubuntu:~/bdkit-demo/wc/WordCount$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< nju:WordCount >----------------------------
[INFO] Building WordCount 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ WordCount ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/sheepxi/bdkit-demo/wc/WordCount/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ WordCount ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ WordCount ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/sheepxi/bdkit-demo/wc/WordCount/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ WordCount ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ WordCount ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running nju.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.081 s - in nju.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ WordCount ---
[INFO] Building jar: /home/sheepxi/bdkit-demo/wc/WordCount/target/WordCount-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.579 s
[INFO] Finished at: 2020-10-28T06:21:50-07:00
[INFO] ------------------------------------------------------------------------
```