金融大数据作业5

姓名：盛祺晨　　学号：191220093

观前提醒：github上请直接看README.pdf，因为有些图片加载不出来，pdf可以。

0.Interllij环境配置

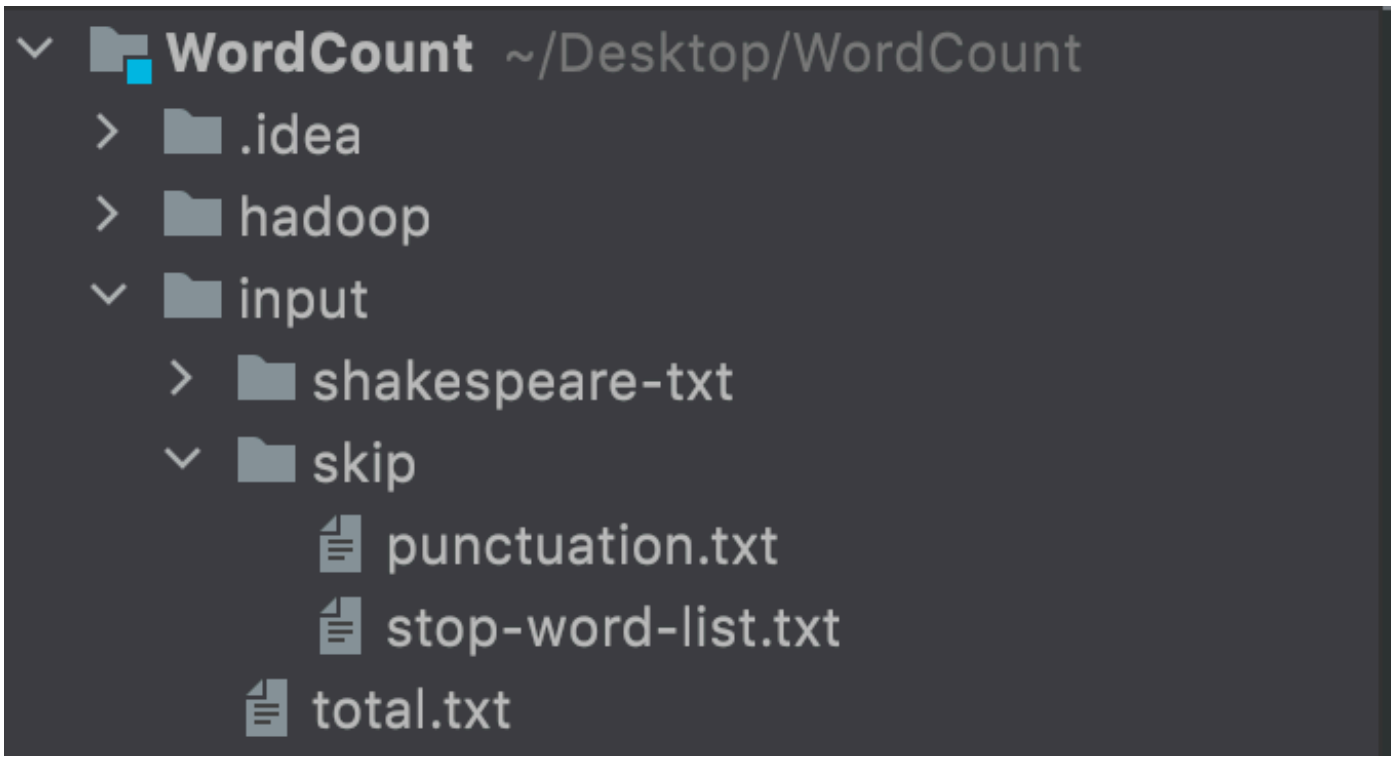
环境配置：参考https://zhuanlan.zhihu.com/p/285164803与https://blog.csdn.net/weixin_45774600/article/details/105289999

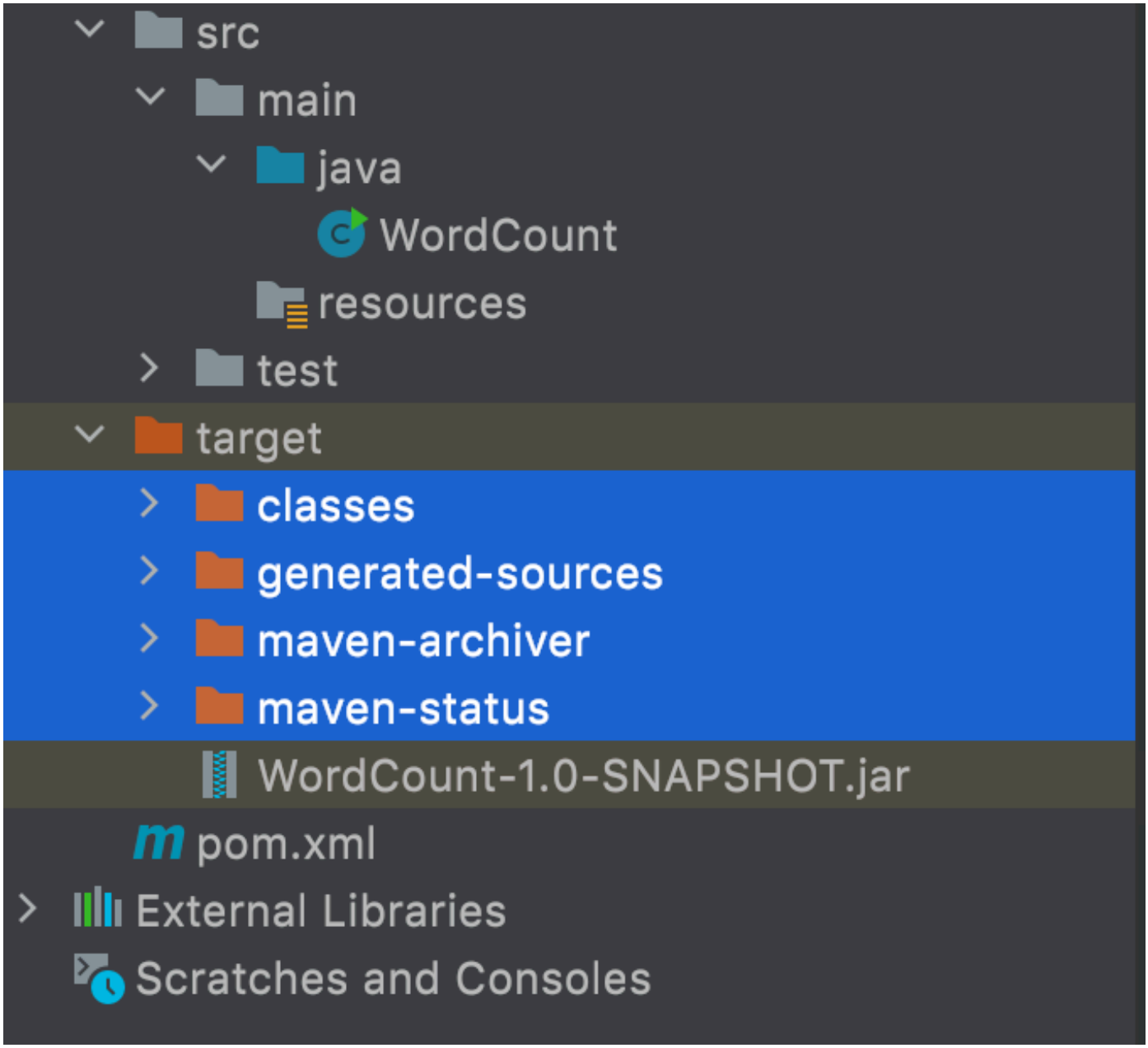
根据上述链接，几乎没有错误地配置完了环境。

1.设计思路

文件结构

文件结构如图：





其中，skip文件存放的是要跳过的标点和一些词语，total.txt存放的是经过合并的总文件（shakespeare-txt文件夹内所有文件内容合成的大文件，生成通过代码中的一个函数），因为这样，我们可以最大程度地复用代码。

设计思路

从命令行获取input和output的路径，input内存放的文件格式被固定在代码中，要求在skip子文件夹内存放一些用来跳过的txt文件，在shakespeare-txt文件夹下面存放一些用来计数的txt文件。

Map：

对于单个文件的计数，获取当前给出的所有的合规的txt文件，并在for循环中每次针对一个txt文件，进行mapreduce操作。map中将给出的要跳过的标点和给定词跳过，形成<单词, 1>这样的对，

```
1 public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
2     String line  = (caseSensitive) ? value.toString() :
value.toString().toLowerCase();// 全部变成小写
3     for (String pattern : punctuations) { // 将标点变成空格，来分开word
```

```
      line = line.replaceAll(pattern, " ");
    }
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      String curword = itr.nextToken();
      if (patternsToSkip.contains(curword) || curword.length()<3) { // 如果单词的长度<3或者有跳过词的话不计
        continue;
      }
      word.set(curword);
      context.write(word,one);// map
    }
  }
```

Reduce：

在SortReducer中，reduce函数先对收到的<单词，次数>进行总计数，获得每个单词的次数。结果放入定义TreeMap<Integer, String> treeMap来保持统计结果,由于treeMap是按key升序排列的,这里要人为指定Comparator以实现倒排，如果有相同次数的就用逗号","隔开，放在同一个key（Integer）下，这样在cleanup函数中，如果遍历的key对应的是一个有逗号的字符串，就以逗号为间隔，拆分（因为逗号已经在map中去除标点一步去除了，所以可以认为文字本身不引入逗号）。输出前100个即可。

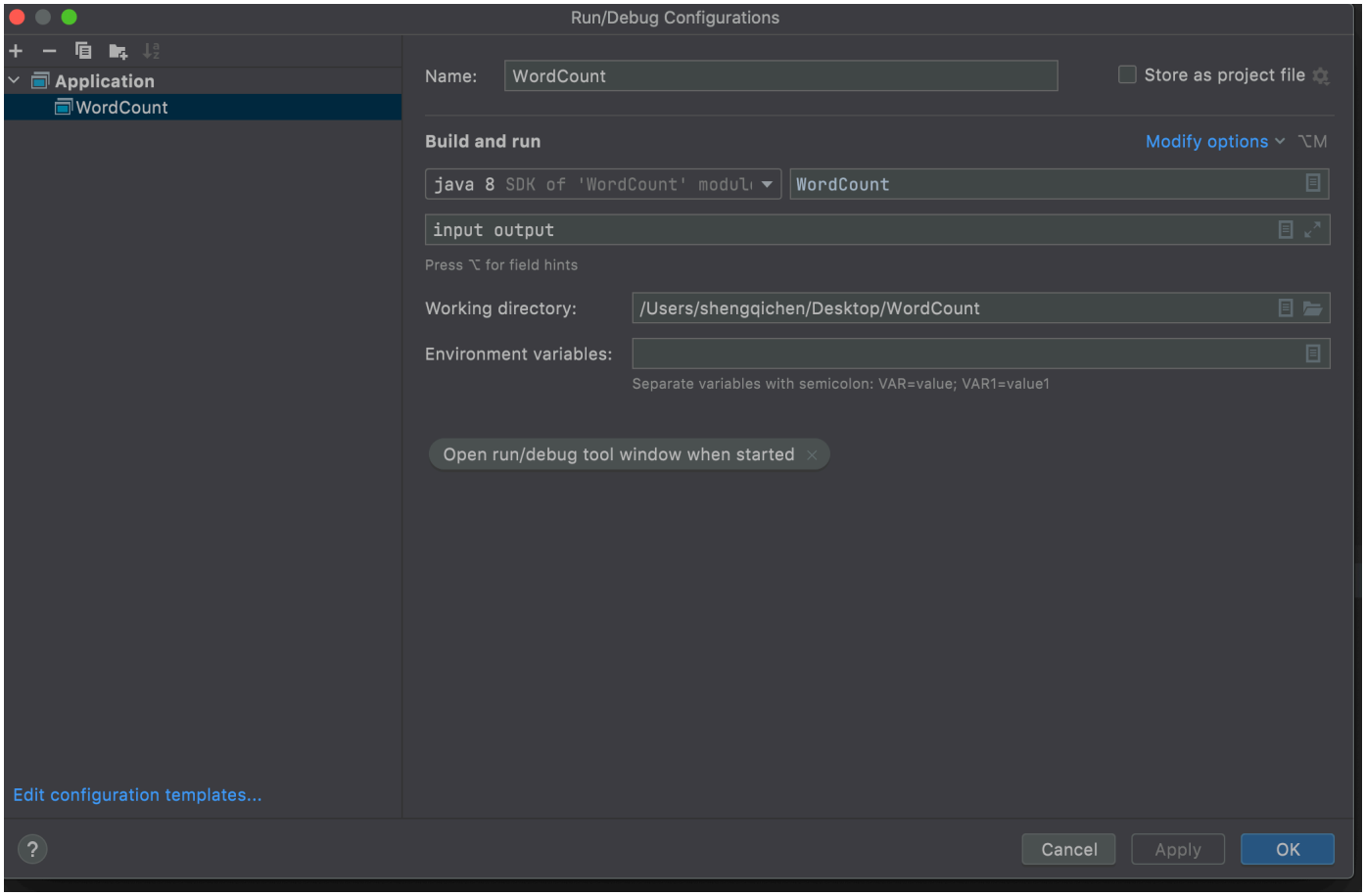
```
public static class SortReducer extends Reducer<Text, IntWritable,Text,IntWritable>{
    //定义treeMap来保持统计结果,由于treeMap是按key升序排列的,这里要人为指定Comparator以实现倒排
    //这里先使用统计数为key，被统计的单词为value
    private TreeMap<Integer, String> treeMap = new TreeMap<Integer, String>(new Comparator<Integer>() {
        @Override
        public int compare(Integer x, Integer y) {
            return y.compareTo(x);
        }
    });
   public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        //reduce后的结果放入treeMap,而不是向context中记入结果
       int sum = 0;
       for (IntWritable val : values) {
           sum += val.get();
       }
       if (treeMap.containsKey(sum)) {  //具有相同单词数的单词之间用逗号分隔
           String value = treeMap.get(sum) + "," + key.toString();
           treeMap.put(sum, value);
       } else {
           treeMap.put(sum, key.toString());
       }
   }
```

```
    protected void cleanup(Context context) throws IOException,
InterruptedException {
        //将treeMap中的结果,按value-key顺序写入context中
        int times = 1;
        for (Integer key : treeMap.keySet()) {
            if(times<=100){
                if (treeMap.get(key).toString().indexOf(",")!=-1) { // 说明有，有同样
个数的单词
                    String[] splitstr=treeMap.get(key).toString().split(",");
                    for (int i=0;i<splitstr.length;++i){
                        if(times<=100){
                            context.write(new Text(splitstr[i]), new
IntWritable(key));
                            times++;
                        }else{break;}
                    }
                }
                else{
                    String s = treeMap.get(key);
                    context.write(new Text(s),new IntWritable(key));
                    times++;
                }
            }
        }
    }
}
```

2.实验结果

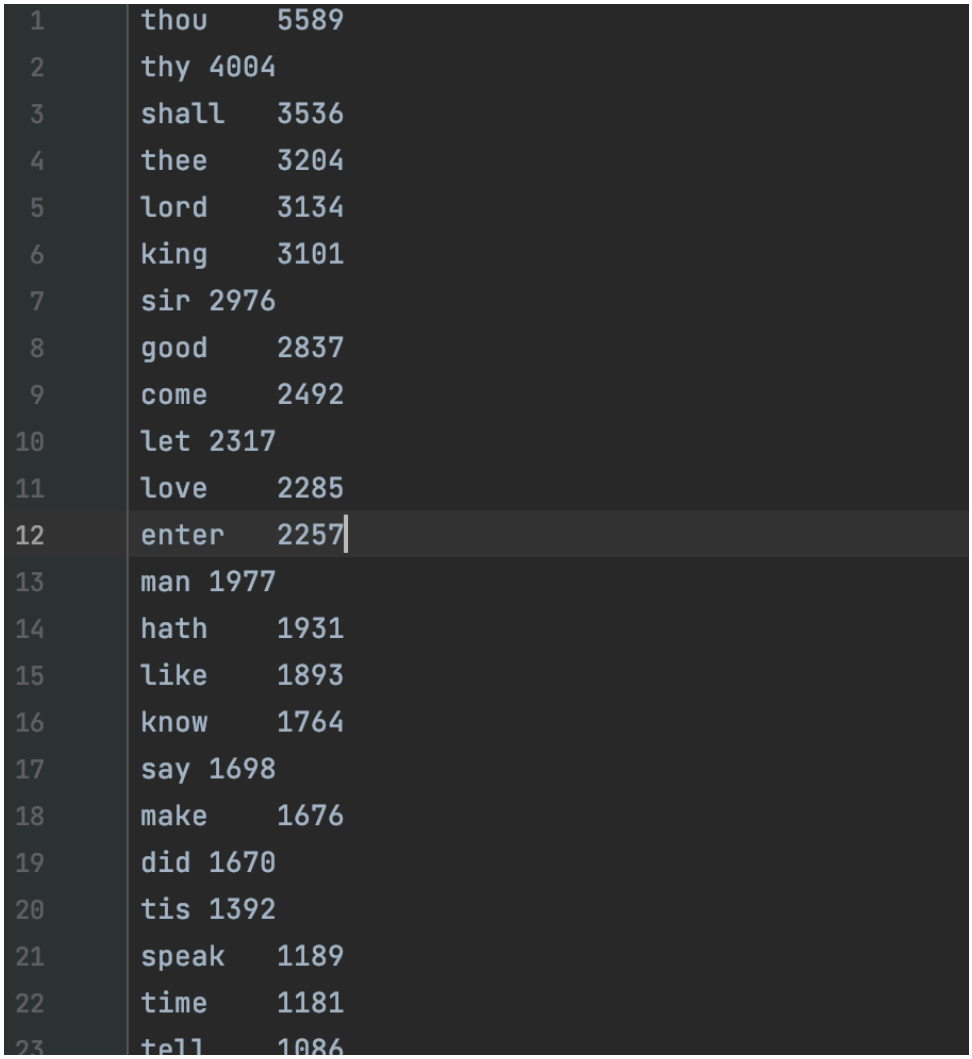
单机

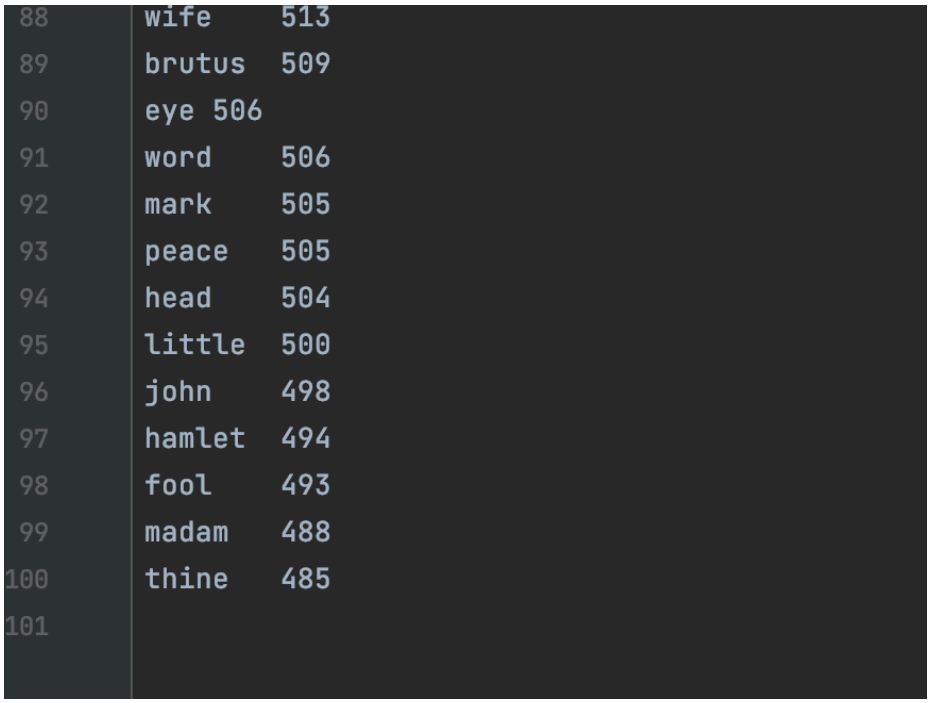
在电脑Intellij软件上直接点右上方的运行。对于其配置，参考如下。



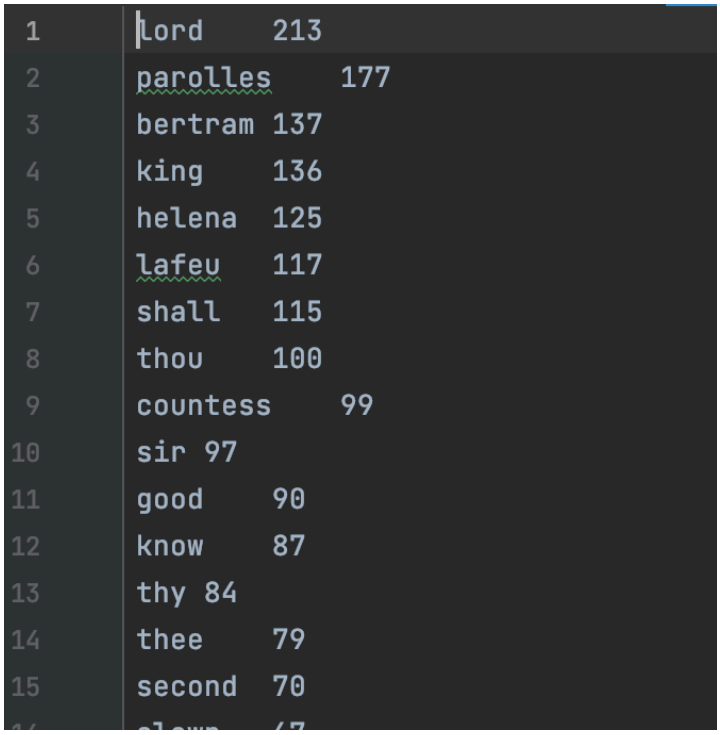
运行结果如下图所示：

全部词频：

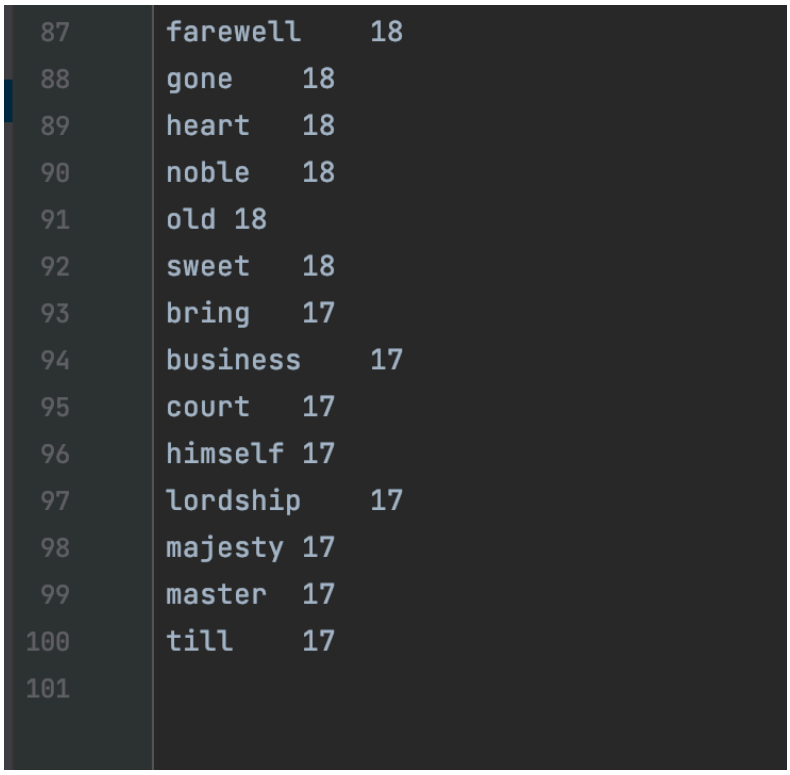




某个单文件的词频（取shakespeare-alls-11.txt）：



```
lord    213
parolles    177
bertram 137
king    136
helena  125
lafeu   117
shall   115
thou    100
countess    99
sir 97
good    90
know    87
thy 84
thee    79
second  70
```

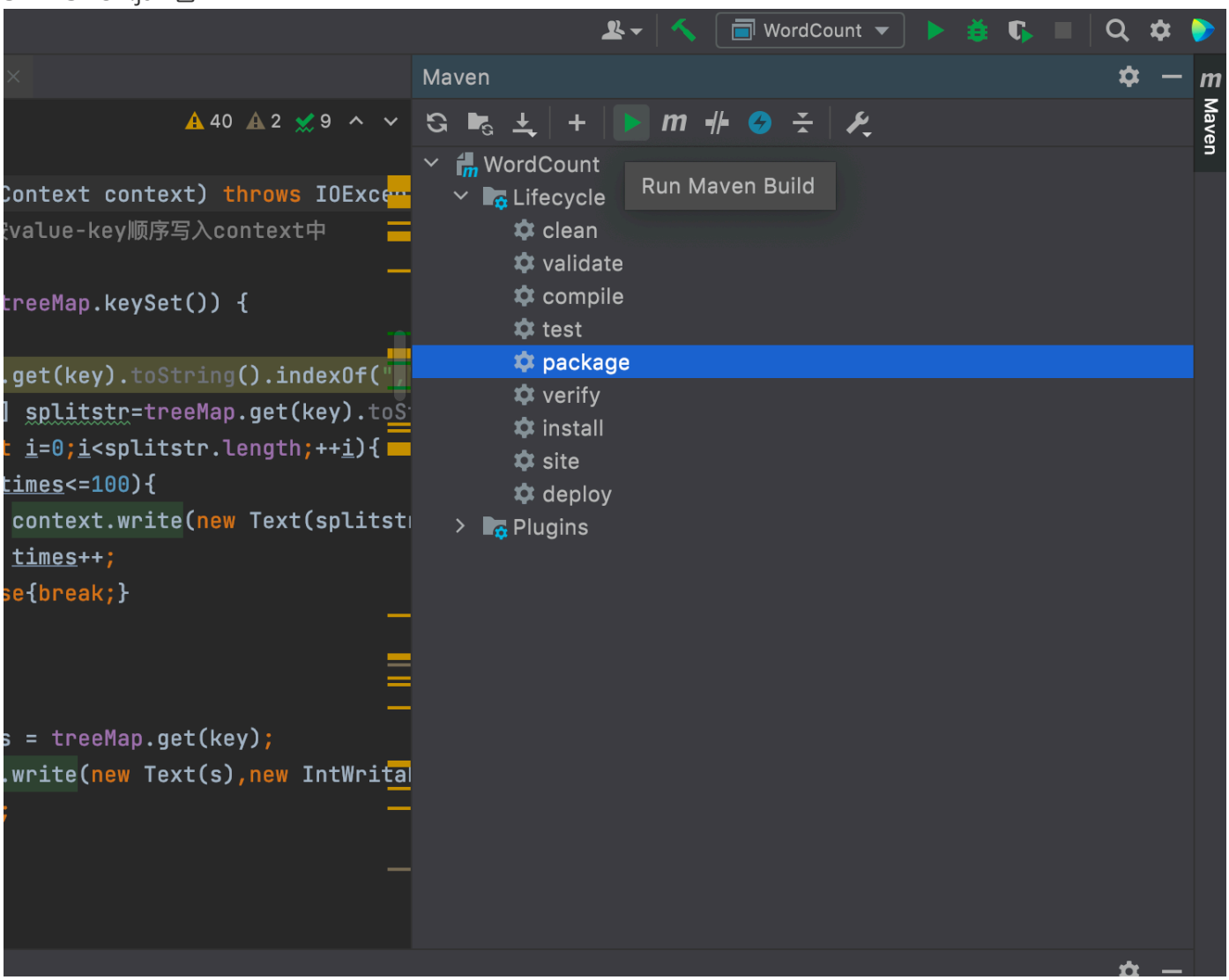


```
farewell    18
gone    18
heart   18
noble   18
old 18
sweet   18
bring   17
business    17
court   17
himself 17
lordship    17
majesty 17
master  17
till    17
```

备注：这个输出没有序号，原因在后面“问题与解决”中有叙述。

伪分布式

先在这里进行打包，点maven的package中的Run Maven Build，得到在taget文件下的WordCount-1.0-SNAPSHOT.jar 包




```
> start-all.sh # 打开
> hdfs dfs -put input input  # 将本地当前input文件整体传入hdfs上

> hadoop jar target/WordCount-1.0-SNAPSHOT.jar WordCount input output # 运行jar包，将input和output文件当成输入的args[]

> hdfs dfs -cat output/input/shakespeare-txt/shakespeare-alls-11.txt/part-r-00000 # 查看其中某文件的输出
```

```
(base) chengqichendeMacBook-Pro:WordCount shengqichen$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - shengqichen supergroup          0 2021-10-30 13:14 input
```

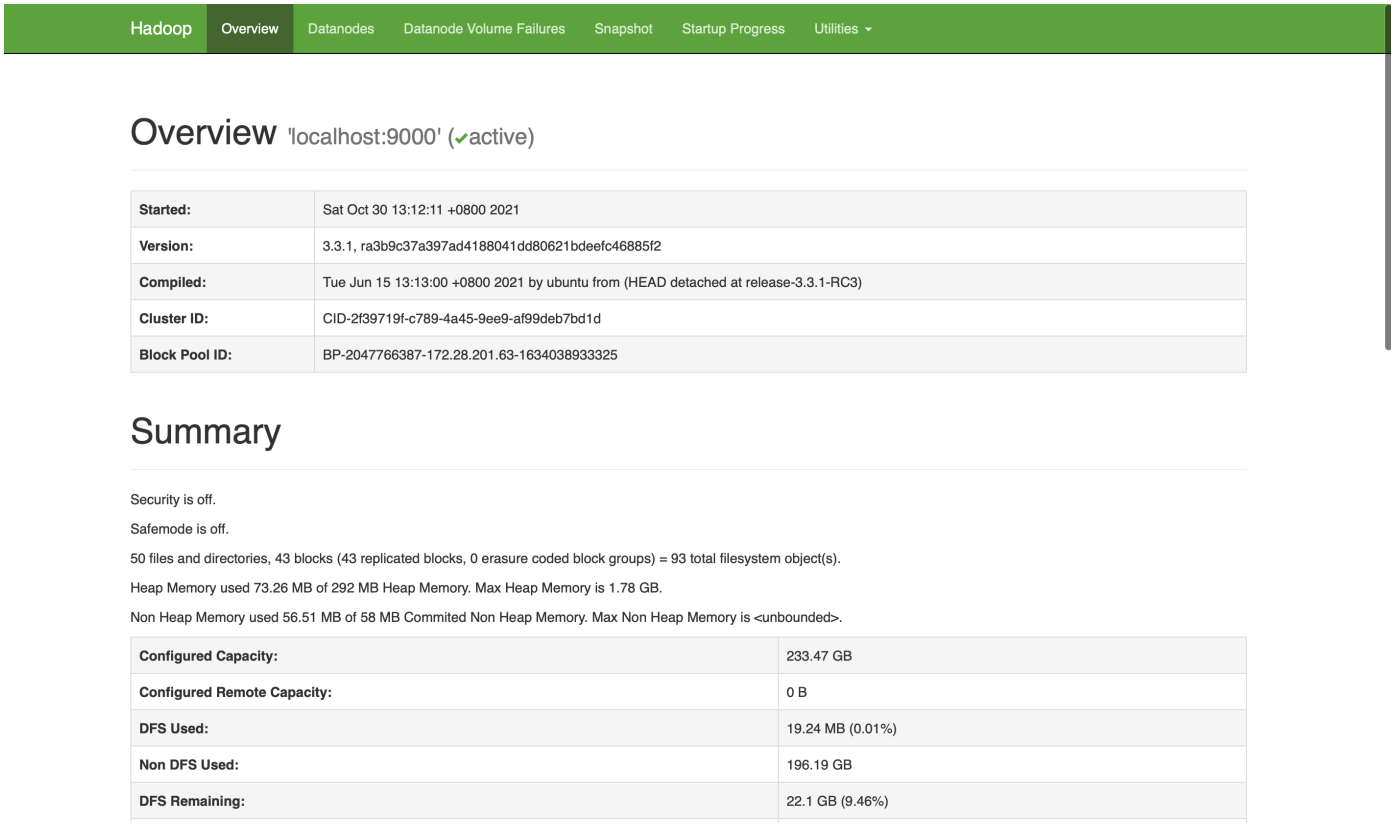
在hdfs dfs -put input input这样就有了input文件夹

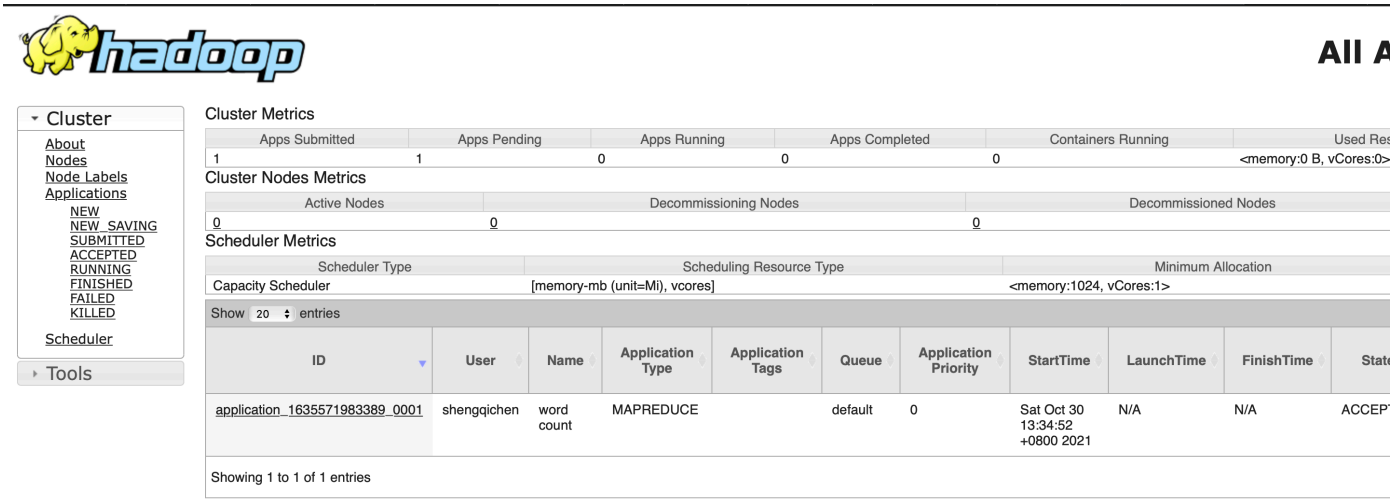
运行后“hadoop jar target/WordCount-1.0-SNAPSHOT.jar WordCount input output”，我们可以看到output文件输出。值得注意的是，我将每一个文件的输出对应一个文件夹，例如对于input/shakespeare-txt/shakespeare-alls-11.txt，我将其输出的词频文件放在output/input/shakespeare-txt/shakespeare-alls-11.txt//part-r-00000中，而对于全部词频文件total.txt,我将其output放在output/total/part-r-00000中，做到了分文件输出。

那么，当我们输入“hdfs dfs -cat output/input/shakespeare-txt/shakespeare-alls-11.txt/part-r-00000”时，我们可以看到输出。

```
(base) chengqichendeMacBook-Pro:WordCount shengqichen$ hdfs dfs -cat output/input/shakespeare-txt/shakespeare-alls-11.txt/part-r-00000
lord    213
parolles        177
bertram 137
king    136
helena  125
lafeu   117
shall   115
thou    100
countess        99
sir     97
good    90
know    87
thy     84
thee    79
second  70
clown   67
love    65
say     62
diana   58
```

是正常运行的。打开localhost查看情况。





展示结束。

3.问题及解决：

1.hdfs上有文件，但是yarn上并没有显示任务

查找资料发现是$HADOOP_HOME/conf/mapred-site.xml需要配置。参考链接https://www.cnblogs.com/zhanggl/p/3860796.html

2.未解决，但是不影响。发现在reduce中的cleanup函数中，我将reduce结果存入TreeMap中，用times记录已经打印了几次（100次为顶），拼接字符串的时候发现会出现time+“：”出现了两次的情况，但是源代码

```
context.write(new Text(times+": "+s),new IntWritable(key));
```

应该只打印了一次。于是，我将time+“：”去掉了。仍然还是输出100行。只是没有了序号。

```
1: 1: thou  5589
2: 2: thy   4004
3: 3: shall 3536
4: 4: thee  3204
5: 5: lord  3134
6: 6: king  3101
7: 7: sir   2976
8: 8: good  2837
9: 9: come  2492
10: 10: let 2317
11: 11: love    2285
12: 12: enter   2257
13: 13: man 1977
14: 14: hath    1931
15: 15: like    1893
16: 16: know    1764
17: 17: say 1698
18: 18: make    1676
19: 19: did 1670
20: 20: tis 1392
21: 21: speak   1189
```

4.性能及改进：

1.为了考虑代码的复用性，将所有文件组合成一个大文件，但是这样会造成很大的重复计算和空间浪费，接近两倍的开销。所以可以考虑重新开一个类，来对单个文件输出的结果，例如对输出的单个文件词频统计：

File1:king 1000\n thou 700\n

File2:king 500\n Jesus 100\n thou 20\n

新建一个map类来读取每一行，存为<word,次数>直接将word的次数相加，排序，取前100，这样会快很多。而且不用开一个文件来合并原始文件。