案例描述

该案例中我们假设某搜索引擎公司要统计过去一年搜索频率最高的 K 个科技关键词或词组，为了简化问题，我们假设关键词组已经被整理到一个或者多个文本文件中，并且文档具有以下格式。

**案例四测试数据格式预览**



我们可以看到一个关键词或者词组可能出现多次，并且大小写格式可能不一致。

b. 案例分析

要解决这个问题，首先我们需要对每个关键词出现的次数进行计算，在这个过程中需要识别不同大小写的相同单词或者词组，如”Spark”和“spark” 需要被认定为一个单词。对于出现次数统计的过程和 word count 案例类似；其次我们需要对关键词或者词组按照出现的次数进行降序排序，在排序前需要把 RDD 数据元素从 (k,v) 转化成 (v,k)；最后取排在最前面的 K 个单词或者词组。

对于第一步，我们需要使用 map 算子对源数据对应的 RDD 数据进行全小写转化并且给词组记一次数，然后调用 reduceByKey 算子计算相同词组的出现次数；第二步我们需要对第一步产生的 RDD 的数据元素用 sortByKey 算子进行降序排序；第三步再对排好序的 RDD 数据使用 take 算子获取前 K 个数据元素。

c. 编程实现

**import** org.apache.spark.{SparkConf, SparkContext}  
  
*/\*\*  
 \* Created by 2281444815 on 2016/11/15.  
 \*/***object** KeyWorldCount {  
 **def** main(args: Array[String]): Unit = {  
 **val** args = "D:\\keyworddata.txt";  
 **if** (args.length < 1) {  
 *println*("Please some useful words");  
 System.*exit*(1)  
 }  
 **val** conf = **new** SparkConf().setAppName("Key World Count").setMaster("local");  
 **val** sc = **new** SparkContext(conf);  
 **val** textFile = sc.textFile(args, 1);  
 //将句子分割为(key,value)对  
 **val** wordCounts = textFile.flatMap(line => line.split(" "))  
 .map(word => (word, 1)).reduceByKey((a, b) => (a + b));  
 //将(key,value)转化为(value,key),用value值进行排序，  
 // 此时集合已变为(value,key)，若要想变回原来的集合，可在后面加上  
 // .map(c => (c.\_2, c.\_1))   
 //其中map、sortByKey都是懒操作，如果没有触发计算的foreach（action），将不会对集合产生影响  
 **val** SortWord = wordCounts.map(c => (c.\_2, c.\_1))  
 .sortByKey(**false**).map(c => (c.\_2, c.\_1)).foreach(*println*);  
 }  
}

结果：

(you,6)

(like,5)

(if,4)

(i,4)

(will,4)

(me,3)

(Spark,2)

(all,2)

(my,2)

(study,2)

(money,2)

(learn,1)

(not,1)

(donot,1)

(Do,1)

(love,1)

(be,1)

(ask,1)

(stude,i,1)

(tell,1)

(something,1)

(why,1)

(take,1)

(stronger?,1)

(and,1)

(give,1)

(Hadoop,1)