## 案例一 Top K

## 实例描述

假设取Top3，才有如下输入和输出

输入：

|  |
| --- |
| Hello World Bye World  Hello Hadoop Bye Hadoop  Bye Hadoop Hello Hadoop  count count while iter  iter count count while iter  output input output put  put |

输出：

|  |
| --- |
| count : 4  Hadoop : 4  iter : 3 |

## 设计思路

第一步：统计词频，将数据转换成（词，词频）的数据对；

第二步：根据词频倒排序

第三步：取前3个值即可

## **代码**

|  |
| --- |
| **import** org.apache.spark.{SparkConf, SparkContext}  **object** TopK {  **def** main(args: Array[String]): Unit = {   **if**(args.length!=3){  System.*out*.println(**"Usage:TopK<input><master><appname>"**)  System.*exit*(1)  }  **val** input = args(0)  **val** master = args(1)  **val** appname = args(2)  **val** conf = **new** SparkConf().setMaster(master).setAppName(appname)  **val** sc = **new** SparkContext(conf)   **val** line = sc.textFile(input).flatMap(\_.split((**" "**))).map(x=>(x,1)).reduceByKey(\_+\_)  **val** topk = line.map(x=>(x.\_2,x.\_1)).sortByKey(**false**).top(3)   **val** result = topk.map(x=>x.\_2+**" : "**+ x.\_1)   result.foreach(*println*)   sc.stop()   } } |

## **配置**

|  |
| --- |
| D:\\甲骨文资料\\Spark师资培训\\data\\input\\\* local TopK |

## **案例二 倒排索引**

1. **实例描述**

输入为一批文档集合，以分隔符分隔

输入内容：

|  |
| --- |
| ID1 Spark HIVE  ID2 Spark HADOOP  ID3 hbase hive IMPALA  ID4 KUDU ZOOKEEPER |

输出内容：

|  |
| --- |
| Spark:ID1 ID2  hive:ID3  KUDU:ID4  HIVE:ID1  HADOOP:ID2  IMPALA:ID3  ZOOKEEPER:ID4  hbase:ID3 |

1. **设计思路**

首先进行预处理分词，转换数据项目为（文档ID,文档词集合）的RDD，然后将数据映射为（词，文档ID）的RDD,最后在reduceByKey阶段聚合每个词的文档ID.

1. **代码**

|  |
| --- |
| **import** org.apache.spark.{SparkConf, SparkContext} **import** scala.collection.mutable  **object** InvertedIndex {   **def** main(args: Array[String]): Unit = {   **val** conf = **new** SparkConf().setAppName(**"InvertedIndex"**).setMaster(**"local"**)  **val** sc = **new** SparkContext(conf)  **val** lines = sc.textFile(**"D:\\甲骨文资料\\Spark师资培训\\data\\input\\invertedindex.txt"**)  **val** line = lines.map(\_.split(**" "**, 2)).map(x => (x(0), x(1).split(**" "**).toList))  **val** line1 = line.flatMap(x=> {  **val** map = **new** mutable.LinkedHashMap[String,String]   **for**(a <- x.\_2)  {  map.put(a,x.\_1)   }  map  })  **val** result = line1.reduceByKey((a,b)=>a+**" "**+b).map(x=>x.\_1+**":"**+x.\_2)   *//println(result.map(x=>x.\_1+":"+x.\_2).collect().mkString(","))* result.foreach(*println*)   }  } |

## **案例三 分组取平均值、最大和最小**

1. **实例描述**

**输入数据**

|  |
| --- |
| (**"Fred"**, 88.0)  (**"Fred"**, 95.0)  (**"Fred"**, 91.0)  (**"Wilma"**, 93.0)  (**"Wilma"**, 95.0)  (**"Wilma"**, 98.0)) |

**输出数据**

|  |
| --- |
| **(Wilma,95.33333333333333,98.0,93.0)**  **(Fred,91.33333333333333,95.0,88.0)** |

1. **设计思路**

用combineByKey

1. 代码

|  |
| --- |
| **object** ColumnValueAvg {  **def** main(args: Array[String]): Unit = {   **val** initialScores = *Array*((**"Fred"**, 88.0), (**"Fred"**, 95.0), (**"Fred"**, 91.0), (**"Wilma"**, 93.0), (**"Wilma"**, 95.0), (**"Wilma"**, 98.0))  **val** conf = **new** SparkConf().setAppName(**this**.getClass.getName).setMaster(**"local"**)  **val** sc = **new** SparkContext(conf)  **val** d1 = sc.parallelize(initialScores)  **type** MVType = (Int, Double,Double,Double) *//定义一个元组类型(科目计数器,分数)* **val** result = d1.combineByKey(  score => (1, score,score,score),  (c1: MVType, newScore) => (c1.\_1 + 1, c1.\_2 + newScore,{**if** (c1.\_3>newScore) c1.\_3 **else** newScore},{**if** (c1.\_4<newScore) c1.\_4 **else** newScore}),  (c1: MVType, c2: MVType) => (c1.\_1 + c2.\_1, c1.\_2 + c2.\_2,{**if**(c1.\_3>c2.\_3) c1.\_3 **else** c2.\_3},{**if**(c1.\_4<c2.\_4) c1.\_4 **else** c2.\_4})  )  **val** result1=result.map { **case** (name, (num, socre,maxs,mins)) => (name, socre / num,maxs,mins) }.collect  *println*(result1.mkString(**","**))  } } |