  
第9章 大數據分析應用基礎



### 字數計算（word count）

|  |  |
| --- | --- |
| 指令 | 說明 |
| start-all.sh | 啟動hadoop |
| cd ~ | 回到家目錄 |
| wget http://www.gutenberg.org/files/74/74-0.txt | 下載湯姆歷險記英文版檔案 |
| hdfs dfs -mkdir /data | 在hdfs上建一目錄 |
| hdfs dfs -put ~/74-0.txt /data/Tom.txt | 將文章上傳hdfs |
| hdfs dfs -ls /data | 查看hdfs資料 |
| val rawData=sc.textFile("hdfs://master:9000/data/Tom.txt",1)  val novelText=rawData.zipWithIndex.filter(x=> x.\_2 >= 482 && x.\_2 <=8859).map(x=>x.\_1) | 讀入文本資料，移除不需要的段落 |
| val remove\_ph="[.,:()!?;\_\*$\\[\\]\n\"]"  val remove\_pp=Array("'s","--"," '","' ")  def doRemove(s:String)={  var rlt=s.replaceAll(remove\_ph, " ")  for(pp<-remove\_pp){  rlt=rlt.replaceAll(pp, " ")  }  rlt  } | 移除標點符號和雙位元資料的副程式和內容 |
| val tri\_words=Array("the","a","an","and","but","to","of","in","at","on","for","as","up","out","by","it","or","with","not")  val words=novelText.flatMap(line=>doRemove(line).split("\\s+"))  val words\_nt= words.map(x=>x.toLowerCase).filter(x=> x.length > 0 && !tri\_words.contains(x))  val result=words\_nt.map(word=>(word, 1)).reduceByKey(\_ + \_).sortBy(x=>x.\_2, false)  result.take(30).foreach(println) |  |

### 矩陣相乘

|  |  |
| --- | --- |
| 指令 | 說明 |
| val a=Array(Array(1,2))  val b=Array(Array(3,4,5),Array(6,7,8))  var v=sc.makeRDD(Array[((Int,Int,Int),Int)]())  val m=a.length  val n=a(0).length  val p=b(0).length | 設定基本資料 |
| for(i<-0 until m; k<-0 until p) {  for(j<-0 until n) {  val v1 = sc.makeRDD(Array(((i, j, k), a(i)(j))))  val v2 = sc.makeRDD(Array(((i, j, k), b(j)(k))))  v=v.union(v1).union(v2)  }  } | 設定計算用的資料格式 |
| val mulRDD=v.reduceByKey(\_\*\_)  val kvRDD=mulRDD.map{case ((i,j,k),v)=> ((i,k), v)}  val result=kvRDD.reduceByKey(\_+\_).collectAsMap | 相乘與相加計算 |
| println("矩陣相乘結果如下")  for(i<-0 until m; k<-0 until p) {  if(k==0) println  print (result((i,k))+" ")  } | 列印出結果 |

### 頻繁項（frequent items）

|  |  |
| --- | --- |
| 指令 | 說明 |
| val trans=sc.makeRDD(Array(  Array("牛奶","香蕉","可樂","麵包"),  Array("麵包","啤酒","尿布"),  Array("香蕉","牛奶","尿布","餅乾"),  Array("可樂","尿布","啤酒"),  Array("啤酒","小蘋果","尿布"),  Array("尿布","奇異果","啤酒"),  Array("可樂果","啤酒","冰淇淋","布丁","尿布"))  ) | 建立購買資料集 |
| val allComb=trans.map{t=>  for(i<-1 to t.length) yield {  val eleCom=t.combinations(i)  val kv=eleCom.map(ele=>"("+ele.sorted.mkString(",")+")")  kv  }  }.flatMap(x=>x).flatMap(x=>x)  allComb.collect | 產生購物清單排列組合 |
| allComb.map(x=>(x,1)).reduceByKey(\_+\_).sortBy(x=>x.\_2,false).take(10).foreach(println) | 得到前10大排行榜 |

### One-hot編碼

|  |  |
| --- | --- |
| 指令 | 說明 |
| val dataRDD=sc.makeRDD(Array(  Array("ID01","曾子","打籃球"),  Array("ID02","子路","看電影"),  Array("ID03","顏回","看書"),  Array("ID04","宰予","露營"),  Array("ID05","子貢","看書"),  Array("ID06","子騫","露營")  )) | 建立基本資料 |
| val habbyTypeMap=dataRDD.map(h=>h(2)).distinct.sortBy(x=>x).zipWithIndex.collectAsMap | 建立項目索引 |
| val dataWithOneHot=dataRDD.map{col=>  val hArray=Array.ofDim[Double](habbyTypeMap.size)  hArray(habbyTypeMap(col(2)).toInt)=1  col.slice(0,2) ++ hArray  }  dataWithOneHot.collect | 建立one-hot矩陣 |

### 相似性（similarity）計算

|  |  |
| --- | --- |
| 指令 | 說明 |
| val scoreRDD=sc.makeRDD(Array(  ("ID01", Array(1.0,3.0,2.5,1.0,5.0)),  ("ID02", Array(2.0,3.0,5.0,3.0,1.0)),  ("ID03", Array(1.0,2.5,3.0,1.0,4.5)),  ("ID04", Array(4.0,3.0,2.0,1.0,3.0))  ))  val scoMap=scoreRDD.collectAsMap | 建立評分矩陣，轉換成映射 |
| val ids=scoreRDD.map(x=>x.\_1)  val idsComb = ids.cartesian(ids).filter{ case (a,b) => a < b } | ids 與 ids 的笛卡爾乘積 |
| val simis=idsComb.map{ case (id1, id2) =>  val calcArr=scoMap(id1).zip(scoMap(id2))  val dis=calcArr.map(x=>math.pow((x.\_1-x.\_2),2)).reduce(\_+\_)  val simi=1/(1+math.sqrt(dis))  ((id1+"-"+id2), simi)  }.sortBy(x=>x.\_2,false)  simis.foreach(println) | 計算歐基里德距離dis，以及相似性simis |

### 文件資料反向排序

|  |  |
| --- | --- |
| 指令 | 說明 |
| cd ~ | 在Ubuntu的終端機 |
| echo "Old MACDONALD had a farm E-I-E-I-O And on his farm he had a cow E-I-E-I-O With a moo moo here And a moo moo there Here a moo there a moo Everywhere a moo moo Old MacDonald had a farm E-I-E-I-O" > doc01.txt  echo "Old MACDONALD had a farm E-I-E-I-O And on his farm he had a pig E-I-E-I-O With a oink oink here And a oink oink there Here a oink there a oink Everywhere a oink oink Old MacDonald had a farm E-I-E-I-O" > doc02.txt  echo "Old MACDONALD had a farm E-I-E-I-O And on his farm he had a duck E-I-E-I-O With a quack quack here And a quack quack there Here a quack there a quack Everywhere a quack quack Old MacDonald had a farm E-I-E-I-O" > doc03.txt  echo "Old MACDONALD had a farm E-I-E-I-O And on his farm he had a horse E-I-E-I-O With a neigh neigh here And a neigh neigh there Here a neigh there a neigh Everywhere a neigh neigh Old MacDonald had a farm E-I-E-I-O" > doc04.txt  echo "Old MACDONALD had a farm E-I-E-I-O And on his farm he had a lamb E-I-E-I-O With a baa baa here And a baa baa there Here a baa there a baa Everywhere a baa baa Old MacDonald had a farm E-I-E-I-O" > doc05.txt | 建立歌詞檔案doc01.txt～doc05.txt |
| hdfs dfs -mkdir /doc\_dir | 在hdfs上建一目錄 |
| hdfs dfs -put ~/doc\*.txt /doc\_dir | 將文章上傳hdfs |
| hdfs dfs -ls /doc\_dir | 查看hdfs資料 |
| val fileData=sc.wholeTextFiles("hdfs://master:9000/doc\_dir/") | 讀進整個目錄下的檔案資料 |
| val invIndex=fileData.map{case (fname,content) =>  val contSplited=content.toLowerCase.split("\\s+").groupBy(x=>x).mapValues(x=>x.length).toArray  val result=contSplited.map(x=>(x.\_1,(fname.takeRight(9),x.\_2)))  result // 返回結果  }.flatMap(x=>x).groupByKey().sortByKey(true)  invIndex.foreach(println) | 產生反向排序索引 |