UNIVERSIDAD DE JAÉN

Computación Distribuida para la gestión de datos a gran escala

Práctica 4. Búsquedas eficientes con Spark.

Sergio Perea De La Casa

Máster Universitario en Ingeniería Informática

Ejercicio **3**

Considerando los ficheros texto1.txt, texto2.txt y texto3.txt vamos a obtener su índice invertido a nivel de registro pero considerando solo una serie de palabras clave que nos interesan. Las palabras claves están en el fichero claves.txt. Hay que tener en cuenta que la palabra puede aparecer tanto en mayúscula como en minúscula y se deben eliminar los signos de puntuación. Como nombre de archivo que se muestre solo el nombre sin la ruta y que no aparezca repetido el nombre de un archivo en una misma palabra, si ésta aparece más de una vez en él. En los ficheros se han eliminado las tildes para evitar problemas. 3

Ejercicio

Considerando los ficheros texto1.txt, texto2.txt y texto3.txt vamos a obtener su índice invertido a nivel de registro pero considerando solo una serie de palabras clave que nos interesan. Las palabras claves están en el fichero claves.txt. Hay que tener en cuenta que la palabra puede aparecer tanto en mayúscula como en minúscula y se deben eliminar los signos de puntuación. Como nombre de archivo que se muestre solo el nombre sin la ruta y que no aparezca repetido el nombre de un archivo en una misma palabra, si ésta aparece más de una vez en él. En los ficheros se han eliminado las tildes para evitar problemas.

Unset

```
//Iniciamos el archivo claves.txt

scala> val claves = sc.textFile("../claves.txt")
claves: org.apache.spark.rdd.RDD[String] = ../claves.txt
MapPartitionsRDD[26] at textFile at <console>:23



//broadcast para guardar las palabras claves

scala> val broadcoast = sc.broadcast(claves.collect)
broadcoast:
org.apache.spark.broadcast.Broadcast[Array[String]] =
Broadcast(18)

scala> broadcoast.value
res5: Array[String] = Array(diccionario, indice, palabra,
documento, tokenizacion)


// Se carga la carpeta con todos los 3 ficheros txt

scala> val rdd_t1 = sc.textFile("C:/spark/texto1.txt")
rdd_t1: org.apache.spark.rdd.RDD[String] = C:/spark/texto1.txt
MapPartitionsRDD[28] at textFile at <console>:23

scala> val rdd_t2 = sc.textFile("C:/spark/texto2.txt")
rdd_t2: org.apache.spark.rdd.RDD[String] = C:/spark/texto2.txt
```

```
MapPartitionsRDD[30] at textFile at <console>:23

scala> val rdd_t3 = sc.textFile("C:/spark/texto3.txt")
rdd_t3: org.apache.spark.rdd.RDD[String] = C:/spark/texto3.txt
MapPartitionsRDD[32] at textFile at <console>:23

scala> val rdd_textos =
sc.parallelize(Array(("C:/spark/texto1.txt",rdd_t1.collect),("
C:/spark/texto2.txt",rdd_t2.collect),("C:/spark/texto3.txt",rd
d_t3.collect)))
rdd_textos: org.apache.spark.rdd.RDD[(String, Array[String])]
= ParallelCollectionRDD[33] at parallelize at <console>:26

scala> rdd_textos.count
res6: Long = 3


//Se formatea los ficheros txt, para que no tengan puntuación,
espacios en blanco y pasamos todas para minúsculas.

scala> val formateo = rdd_textos.map(x => (x._2.mkString("
").replaceAll("""[\p{Punct}]""","").toLowerCase().split(" "),
x._1.substring(x._1.length-10, x._1.length)))
formateo: org.apache.spark.rdd.RDD[(Array[String], String)] =
MapPartitionsRDD[35] at map at <console>:23



// map para ligar las palabras claves con los documentos que
contiene las palabras

scala> val result = formateo.map(x =>
(x._1.filter(broadcoast.value.contains(_)), x._2))
result: org.apache.spark.rdd.RDD[(Array[String], String)] =
MapPartitionsRDD[36] at map at <console>:24



// RESULTADO

scala> result.collect()
res8: Array[(Array[String], String)] =
Array((Array(diccionario),texto1.txt), (Array(diccionario,
indice, indice),texto2.txt), (Array(tokenizacion, documento,
tokenizacion, indice),texto3.txt))
```

```
scala> val claves = sc.textFile("../claves.txt")
claves: org.apache.spark.rdd.RDD[String] = ../claves.txt MapPartitionsRDD[26] at textFile at <console>:23

scala> val broadcoast = sc.broadcast(claves.collect)
broadcoast: org.apache.spark.broadcast.Broadcast[Array[String]] = Broadcast(18)

scala> broadcoast.value
res5: Array[String] = Array(diccionario, indice, palabra, documento, tokenizacion)

scala> val rdd_t1 = sc.textFile("C:/spark/texto1.txt")
rdd_t1: org.apache.spark.rdd.RDD[String] = C:/spark/texto1.txt MapPartitionsRDD[28] at textFile at <console>:23

scala> val rdd_t2 = sc.textFile("C:/spark/texto2.txt")
rdd_t2: org.apache.spark.rdd.RDD[String] = C:/spark/texto2.txt MapPartitionsRDD[30] at textFile at <console>:23

scala> val rdd_t3 = sc.textFile("C:/spark/texto3.txt")
rdd_t3: org.apache.spark.rdd.RDD[String] = C:/spark/texto3.txt MapPartitionsRDD[32] at textFile at <console>:23

scala> val rdd_textos = sc.parallelize(Array(("C:/spark/texto1.txt",rdd_t1.collect),("C:/spark/texto2.txt",rdd_t2.collect),("C:/spark/texto3.txt",rdd_t3.collect)))
rdd_textos: org.apache.spark.rdd.RDD[(String, Array[String])] = ParallelCollectionRDD[33] at parallelize at <console>:26

scala> rdd_textos.count
res6: Long = 3
```

```
scala> val formateo = rdd_textos.map(x => (x._2.mkString(" ").replaceAll("""[\p{Punct}]""","").toLowerCase().split(" "), x._1.substring(x._1.length-10, x._1.length)))
formateo: org.apache.spark.rdd.RDD[(Array[String], String)] = MapPartitionsRDD[35] at map at <console>:23

scala> val result = formateo.map(x => (x._1.filter(broadcoast.value.contains(_)), x._2))
result: org.apache.spark.rdd.RDD[(Array[String], String)] = MapPartitionsRDD[36] at map at <console>:24

scala> result.collect()
res8: Array[(Array[String], String)] = Array((Array(diccionario),texto1.txt), (Array(diccionario, indice, indice),texto2.txt), (Array(tokenizacion, documento, tokenizacion, i
ndice),texto3.txt))
```