Palabras más frecuentes de un texto con pyspark

El objetivo es crear un RDD cuyos elementos guardan cada una de las líneas del fichero

```
rdd=sc.textFile(“fichero”)
import re
words=rdd.flatMap(lambda linea:re.compile(“\W”).split(linea))
.filter(lambda word:word !=’’)
.map(lambda word:word.lower()))
```

Para contar el número de veces que aparece cada palabra se utiliza la función map,donde cada clave es la palabra.

Para agrupar todas las claves iguales se utiliza la función reduceByKey

```
histograma = (words.map(lambda word:(word,1)).reduceByKey(lambda x,y:x+y))
masFrecuentes = histograma.sortBy(lambda (k,v):-v).take(15)
for word in masFrecuentes:
print word[0]+ word[1]
```

Otra forma de extraer las palabras más frecuentes:

1. Leer el archivo de entrada y se paraleliza en un RDD con el método textFile.
2. Para cada línea, se extraen todas las palabras. Para esta operación, podemos utilizar el método flatMap y una expresión regular.
3. Cada palabra en el texto (es decir, cada elemento del RDD) ahora se asigna a un par clave-valor: la clave es la palabra en minúscula y el valor es siempre 1. Esto es una operación de mapeo.
4. Con una llamada reduceByKey, contamos cuántas veces cada palabra (clave) aparece en el texto (RDD). La salida es pares clave-valor, donde la clave es una palabra y el valor es el número de veces que la palabra aparece en el texto.
5. Lanzamos claves y valores, creando un nuevo RDD. Esta es una operación de mapeo
6. Ordenamos el RDD en orden descendente y extraemos el primer elemento. Esto es una acción y se puede hacer en una operación con el método takeOrdered.

```
import re
WORD_RE = re.compile(r"[\w']+")
print (sc.textFile("file.txt")
.flatMap(lambda line: WORD_RE.findall(line))
.map(lambda word: (word.lower(), 1))
.reduceByKey(lambda a,b: a+b)
.map(lambda (k,v): (v,k))
.takeOrdered(1, key = lambda x: -x[0]))
```

Contar el número de caracteres,palabras y líneas de un texto

1. Leer el archivo de entrada y se paraleliza en un RDD con el método textFile.
2. Para cada línea, se extraen todas las palabras. Para esta operación, podemos utilizar el método flatMap y una expresión regular.
3. Para cada clave, sumamos todos los valores. Esto se puede hacer con el método reduceByKey.
4. Finalmente, se recogen los resultados. En este caso, podemos usar el método collectAsMap que recoge los pares clave-valor en el RDD y devuelve un diccionario Python. Tenga en cuenta que se trata de una acción; Por lo tanto, la cadena RDD se ejecuta y se devuelve un resultado.

```
def file_stadistics(line):
return [("chars", len(line)), \ ("words", len(line.split())), \ ("lines", 1)]

print (sc.textFile("readme.txt")
.flatMap(file_stadistics)
.reduceByKey(lambda a,b: a+b)
.collectAsMap())

Out:{'chars': 1335, 'lines': 31, 'words': 179}
```