# Practica1 Sistemas distribuidos: Analisis de sentimientos con Hadoop

*Veronica Gomez Gomez, Pablo Olmos Martinez y Carlos Grande Nuñez*

*11/29/2019*

## Contents



## Descripcion del codigo

Nuestro codigo basado en mapreduce se divide en varios bloques importantes.

- Diccionario emocional AFINN-111
- Script Python usado para la obtencion de datos de Twitter llamado TwitterStream.py
- Fichero de datos obtenidos de Twitter. 7,0 GB (6.957.687.003 bytes) llamado bigTwitter.json
- Codigo en Python de la solucion al analisis de emociones de Twitter llamado mrJobSimple.py

#### Diccionario AFINN-111

Hemos decidido usar el diccionario AFINN-111, se opto por usar el diccionario ingles por mayor simpleza de lenguaje. Tambien debido a que a continuacion el filtrado como se explicara mas en detalle como se ha realizado, filtra los Tweets eligiendo solo aquellos que pertenecen a Estados Unidos.

#### TwitterStream

Se uso el script provisto por la practica para la captura de datos, para obtener un volumen de datos interesante para la realizacion de la practica se mantuvo ejecutando alrededor de 5h. Para la configuracion del script se uso una cuenta personal de Twitter.

#### bigTwitter

El nombre del fichero utilizado para nuestro estudio. Nos sirvio tambien para comprender que cada linea correspondia a un Tweet, ello junto con la informacion sobre developers para Twitter nos permitio posteriormente que apartados deberiamos observar para un filtrado efectivo.

#### mrJobSimple

Al final se opto por usar la libreria MrJob para la ejecucion y pruebas de nuestro Mapreduce en AWS y en local. Tanto el README de la propia libreria como los ejemplos disponibles en GitHub nos ayudo en la comprension y posterior ejecucion del script usando esta libreria.

El script tiene tres partes o bloques (sin contar con la parte de declaracion de variables)

### Mapper

Actualmente la funcion de mapper es la que contiene toda la logica de la practica. Primero carga el diccionario en un mapa. En el siguiente paso se leen todas las lineas del fichero obtenido gracias al script de TwitterStream y se realiza un parseado indicando que mantenga el formato de Json. Esto nos facilito mucho la realizacion de la tarea de filtrado y tambien la de analizar palabras, ya que solo analizaremos aquellas palabras que correspondan al cuerpo del tweet.

A continuacion el primer filtrado que se realiza se basa en el campo de pais. En esta comprobacion de pais se comprueban tres valores importantes. El primero que exista el campo de "place" ya que se trata de un campo opcional en Twitter. Despues que sea distinto de none, nos hemos encontrado con problemas durante la fase de pruebas ya que en ocasiones pese a que el campo existia no tenia contenido y generaba un fallo en la ejecucion. La ultima comprobacion que se realiza es que el pais al que pertenece el Tweet sea el de Estados Unidos (US).

Con este primer filtrado en la funcion de mapper, evitamos que sea el reduce quien se encargue de ello y con esto reducimos en gran parte el consumo de memoria, CPU y tiempo de uso.

Finalmente nos centramos en analizar solo aquellas palabras que correspondan al cuerpo del Tweet, dejando de lado el nombre de usuario y otros valores que no son de importancia en esta practica. El ultimo paso consiste en eliminar mayusculas y caracteres extraños para a continuacion asignar a cada palabra resultante su valor correspondiente del diccionario. En caso de que la palabra no exista en el diccionario se le asigna el valor de 0

### Combiner

Nuestro combiner se podria decir que es basico y no cumple ninguna funcion particularmente importante, ya que lo que hace actualmente podria hacerlo el reducer por completo. La idea del combiner es que mas adelante siguiese filtrando, en este caso eliminando aquellas palabras que no existen en el diccionario y tienen un valor de 0.

Este seria un paso mas en la optimizacion del tiempo de ejecucion y uso de CPU.

### Reducer

Nuestro reducer es el sumatorio de valores de cada palabra.

### Otros scripts

Previo al uso de la libreria de MrJob se ejecuto otro script para pruebas iniciales, este script solo podia ser ejecutado en local y fue utilizado por su velocidad y simpleza. Esto nos permitio entender como navegar por el Json de Twitter, como filtrar y que bucles nos serian necesarios para su ejecucion posterior en AWS.

## Formas de ejecucion

Para ejecutar en AWS se utilizo el siguiente comando:

```
time python mrJobSimple.py -r emr --file AFINN-111.txt bigTwitter.json --output-dir=s3://hadoopdatascie
```

y para las pruebas iniciales en local ya usando la libreria de MrJob la ejecucion es la siguiente:

```
time python mrJobSimple.py --file AFINN-111.txt bigTwitter.json
```

## Evaluacion escalabilidad

En las ejecuciones que usamos de prueba pudimos ver que una de las partes que mas tiempo ocupaba era la propia subida del fichero bigTwitter.json.

Para nuestra practica utilizamos el entorno que se genera por defecto, es decir, no utilizamos. Por lo que hemos podido ver al ejecutar nuestro MapReduce entendemos que AWS junto con Python ofrecen unos entornos realmente flexibles con los que se trabaja muy bien la escalabilidad.

Los tiempos y cargas de ejecucion fueron los siguientes:

```
No configs found; falling back on auto-configuration
No configs specified for emr runner
Auto-created temp S3 bucket mrjob-316e63eca3db4318
Using s3://mrjob-316e63eca3db4318/tmp/ as our temp dir on S3
Creating temp directory /tmp/mrJobSimple.evergom.20191209.171831.284355
writing master bootstrap script to /tmp/mrJobSimple.evergom.20191209.171831.284355/b.sh
uploading working dir files to s3://mrjob-316e63eca3db4318/tmp/mrJobSimple.evergom.20191209.171831.2843
Copying other local files to s3://mrjob-316e63eca3db4318/tmp/mrJobSimple.evergom.20191209.171831.284355/
Created new cluster j-36OFPAU8SW46A
Added EMR tags to cluster j-36OFPAU8SW46A: __mrjob_label=mrJobSimple, __mrjob_owner=evergom, __mrjob_ver
Waiting for Step 1 of 1 (s-3D2WQ8N4QRO5N) to complete...
  PENDING (cluster is STARTING)
  PENDING (cluster is STARTING)
  PENDING (cluster is BOOTSTRAPPING: Running bootstrap actions)
  PENDING (cluster is BOOTSTRAPPING: Running bootstrap actions)
  PENDING (cluster is BOOTSTRAPPING: Running bootstrap actions)
  master node is ec2-54-190-154-11.us-west-2.compute.amazonaws.com
  RUNNING for 0:00:21
  RUNNING for 0:00:52
  .
  .
  .
  RUNNING for 0:35:05
  COMPLETED
Attempting to fetch counters from logs...
Waiting for cluster (j-36OFPAU8SW46A) to terminate...
  TERMINATING
  TERMINATING
  TERMINATED
Looking for step log in s3://mrjob-316e63eca3db4318/tmp/logs/j-36OFPAU8SW46A/steps/s-3D2WQ8N4QRO5N...
  Parsing step log: s3://mrjob-316e63eca3db4318/tmp/logs/j-36OFPAU8SW46A/steps/s-3D2WQ8N4QRO5N/syslog.2
  Parsing step log: s3://mrjob-316e63eca3db4318/tmp/logs/j-36OFPAU8SW46A/steps/s-3D2WQ8N4QRO5N/syslog.gz
Counters: 55
    File Input Format Counters
        Bytes Read=6958986713
    File Output Format Counters
        Bytes Written=198081
    File System Counters
        FILE: Number of bytes read=151919
        FILE: Number of bytes written=18772537
        FILE: Number of large read operations=0
        FILE: Number of read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=15392
```

```
        HDFS: Number of bytes written=0
        HDFS: Number of large read operations=0
        HDFS: Number of read operations=104
        HDFS: Number of write operations=0
        S3: Number of bytes read=6958986713
        S3: Number of bytes written=198081
        S3: Number of large read operations=0
        S3: Number of read operations=0
        S3: Number of write operations=0
    Job Counters
        Data-local map tasks=104
        Killed map tasks=1
        Launched map tasks=104
        Launched reduce tasks=1
        Total megabyte-milliseconds taken by all map tasks=12460978176
        Total megabyte-milliseconds taken by all reduce tasks=34775040
        Total time spent by all map tasks (ms)=4056308
        Total time spent by all maps in occupied slots (ms)=389405568
        Total time spent by all reduce tasks (ms)=5660
        Total time spent by all reduces in occupied slots (ms)=1086720
        Total vcore-milliseconds taken by all map tasks=4056308
        Total vcore-milliseconds taken by all reduce tasks=5660
    Map-Reduce Framework
        CPU time spent (ms)=3700390
        Combine input records=55168
        Combine output records=36998
        Failed Shuffles=0
        GC time elapsed (ms)=19597
        Input split bytes=15392
        Map input records=1600519
        Map output bytes=551371
        Map output materialized bytes=321497
        Map output records=55168
        Merged Map outputs=104
        Physical memory (bytes) snapshot=73811386368
        Reduce input groups=14238
        Reduce input records=36998
        Reduce output records=14238
        Reduce shuffle bytes=321497
        Shuffled Maps =104
        Spilled Records=73996
        Total committed heap usage (bytes)=69532647424
        Virtual memory (bytes) snapshot=491668406272
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
job output is in s3://hadoopdatasciencetest/output/
Removing s3 temp directory s3://mrjob-316e63eca3db4318/tmp/mrJobSimple.evergom.20191209.171831.284355/.
Removing temp directory /tmp/mrJobSimple.evergom.20191209.171831.284355...
Removing log files in s3://mrjob-316e63eca3db4318/tmp/logs/j-36OFPAU8SW46A/...
```

```
Terminating cluster: j-36OFPAU8SW46A

real    61m35,664s
user    1m50,558s
sys 0m44,342s
```

Entendemos que con mas pruebas los tiempos de ejecucion serian menores hasta cierto punto. Ya que hay un momento en el que no compensa seguir escalando. Nuestro mayor cuello de botella como ya se comento al principio es la subida del fichero (que se puede solucionar subiendo el archivo a un bucket y posteriormente trabajando desde el)

Se observa una gran diferencia entre map tasks y reduce tasks. Esto se entiende ya que la mayor carga de trabajo recae sobre la funcion de map, filtra los tweets en funcion del pais y da a cada palabra un valor correspondiente.

## Comentarios personales

**Los mayores problemas que encontramos fueron los siguientes:**

Al principio se intento probar con la maquina virtual de Hadoop, posteriormente se decidio por rapidez y simpleza optar por AWS.

Aqui uno de los mayores problemas fue el desconocimiento de AWS, sumado al "respeto" a una posible factura debido al uso de sus componentes. Una vez solventada la primera curva de aprendizaje hizo muy sencillo la posterior ejecucion del script.

El aprendizaje de la libreria MrJob, como toda libreria lleva una complejidad entender su funcionalidad. La parte positiva es la cantidad de ejemplos que se pueden encontrar en el propio repositorio en GitHub.

### Critica

Se trata de una practica muy interesante, la posibilidad de usar AWS es una buenisima alternativa a cuando la maquina de Hadoop da problemas.

Nos ha permitido entender en mayor detalle como funciona un mapreduce en un entorno real y empezamos a comprender el potencial del uso de sistemas distribuidos.

Nuestro mayor hueso, por asi llamarlo fue la curva de aprendizaje de AWS y la libreria principal usada en el script. Entendemos tambien que es un proceso natural y genera momentos gratificantes cuando se comprobo que la ejecucion iba segun lo debido.

El script que hemos generado se podria mejorar en varios puntos, entre ellos quizas el mas importante seria la funcion de combiner. Actualmente esta funcion realiza un sumatorio, que deberia estar mas orientado a la parte del reduce. La mejora podria ser que combiner simplemente eliminase aquellas palabras que tienen un valor 0. Estos valores meten "ruido" tanto en la solucion volcada como en el siguiente reduce que se ejecuta a continuacion.

En esta practica al tratarse de un fichero pequeño, entendemos que no habria una gran diferencia en carga y tiempo entre usar el combiner o no. Hemos decidido mantenerlo ya que nos parece un paso interesante con el que se podria trabajar mas en detalle.

### Tiempo dedicado

Es una practica a la que hay que dedicarle tiempo, si bien al ser un equipo equilibrado hemos conseguido hacerlo en un tiempo prudencial, hay que tener en cuenta que aprender el uso correcto de las herramientas

lleva su tiempo. Se podria decir que al final lo que mas tiempo llevo fue la parte de Python correspondiente a la navegacion por Json.