

# Estudio sobre las interacciones en Twitter según gustos musicales

Guillermo López Leal

¿Hay interacción entre los usuarios de la red? ¿Qué hashtags utilizan? ¿Cuántos twits envían? ¿De qué hablan cuando escriben sobre su artista favorito? Código disponible en <https://github.com/willyaranda/pfm>

# Índice

---

1. Índice.....	pág. 2
2. Introducción .....	pág. 3
3. Arquitectura y despliegue.....	pág. 4
4. Tecnologías.....	pág. 10
5. Datos de entrada .....	pág. 20
6. Análisis .....	pág. 23
7. Conclusión .....	pág. 31
8. Bibliografía .....	pág. 33

# Introducción

---

Las redes sociales son cada vez más usadas por todo el mundo, en cualquier situación y por cualquier estrato social. A través de su análisis, podremos sacar conclusiones que nos permitirán conocer más en profundidad a la sociedad, y sobre todo, a los usuarios de las redes sociales y cómo se comportan cuando pueden interactuar con miles de posibles usuarios, quizás muchos de ellos a una gran distancia de ellos, que no conocerán en persona y que hablan en otro idioma que no es el nativo para ellos.

En ellas, podemos ver interacciones entre los usuarios, como conversaciones, patrocinios, incluyendo aquellas personas que no son muy amigas de la tecnología, pero que, sin embargo, encuentran en estas redes una manera distendida de conocer la tecnología a través de las relaciones sociales.

La necesidad de estudiar las redes sociales, las relaciones y las conversaciones no sólo es interesante desde el punto de vista tecnológico, puesto que podemos identificar patrones de conversaciones, pero a la vez necesitar crear unas respuestas tecnológicas desconocidas hasta ahora para poder procesar dichos datos: desde la ingesta, pasando por cómo se guardan hasta el análisis final sobre miles de millones de conversaciones e interacciones.

# Arquitectura y despliegue

---

## Amazon EC2

Todo el proyecto se ha basado en **máquinas en Amazon EC2**<sup>1</sup>, por la facilidad de **escalar**, **agregar** más máquinas, y el no necesitar contratar máquinas físicas o tener **costes de instalación** para su funcionamiento.



Amazon EC2 permite el arrendamiento de máquinas por un tiempo indefinido y sin costes iniciales, y es una solución muy interesante en **arquitecturas de Big Data**, porque permite iniciar los proyectos con poco hardware (virtual) e ir **expandiendo según crezcan las necesidades** del proyecto: más espacio en disco, más RAM, más capacidad de cómputo, más red...

La elección de las máquinas en EC2 es una cuestión compleja dentro del mundo de Big Data y de cualquier software que se quiera **distribuir en la nube**, porque requiere de antemano conocer cuáles van a ser las necesidades de nuestro software...

1. ¿Vamos a necesitar mucha **capacidad de cómputo**?
2. ¿Vamos a necesitar mucha **RAM** para mantener datos en memoria?
3. ¿Cuánto **disco** necesitaremos? ¿quizás unidades de **estado sólido**?

---

<sup>1</sup> <http://aws.amazon.com/es/ec2/>

<sup>2</sup> <https://api.twitter.com/1.1/statuses/show/490086133012115457.json>

4. ¿Cuándo vamos a realizar los cálculos de nuestro software? ¿es **tiempo real**? ¿es procesado **batch**?
5. ¿Necesitamos tener las **máquinas siempre** o somos **tolerantes** a que pueda haber máquinas que se **pierdan** y sustituirlas por otras?

Es necesario, por tanto, **responder a estas preguntas** de una forma clara y contundente:

1. **Sí**, puesto que vamos a estar ingiriendo **miles de twets por segundo** (en el caso de acceso al firehose completo de twitter, no con el API para desarrolladores, donde sólo se devuelve un 1%). Además, vamos a tener diferente software ejecutándose, y necesitamos **procesarlos en tiempo real** (para lo cual se usará Storm, como se comentará más adelante).
2. **No necesariamente**, puesto que en RAM sólo vamos a tener los **procesos**, y no los datos, porque el guardado principal se hace en **HBase**, que se guarda en **disco**.
3. **Mucho**, porque vamos a tener que guardar todos los **twits** en **HBase**. Un cálculo rápido indica que cada **twit** ocupa unos **5-6KB**. (Desde 2KB<sup>2</sup> hasta 9KB<sup>3</sup>). Si tenemos en cuenta que se han recibido prácticamente **100.000 tweets en un día**, y esto es un 1% de la capacidad de twitter, tendríamos unos **10M de twits diarios**, lo que llevaría a necesitar unos **47GB** diarios para guardar (o 470M en el caso del 1%). Además, el uso de discos SSD sería recomendado porque si se reciben unos 10M de twits diarios, esto quiere decir que se están recibiendo unos **115 tweets/s** solo de nuestro **hashtags**

---

<sup>2</sup> <https://api.twitter.com/1.1/statuses/show/490086133012115457.json>

<sup>3</sup> <https://api.twitter.com/1.1/statuses/show/490086978298585090.json>

(con **picos de 618,725 por minuto**, o **10000 por segundo**, en la final de la copa del mundo de la FIFA<sup>4</sup>)

4. **El proceso es en tiempo real**, porque con Storm lo que hacemos es **recoger los tweets e introducirlos en HBase**, por lo que las máquinas tienen que estar continuamente trabajando. Aunque también tendremos un proceso puntual (o **batch**) cuando se realicen los análisis de los datos.
5. Necesitaríamos **máquinas funcionando de forma constante** (ver punto 4), pero también podríamos tener algunas de forma puntual para los procesos batch con Hive.

### Máquinas elegidas

Una vez visto las necesidades del proyecto, se analizaron las diferentes máquinas que están disponibles en Amazon, y viendo el **coste** y el posible **rendimiento** necesario, se eligieron el uso de 5 máquinas con el formato **m3.xlarge**, que se componen de los siguientes grandes elementos:

- **4 CPUs virtuales**
- **15GB** de memoria **RAM**
- **2 discos SSD** de **40GB** cada uno
- Una **red** con un **rendimiento “alto”** (según la nomenclatura de Amazon).

Sin embargo, y en contra de lo que se haría en producción, al menos con una máquina que sea el nodo central, se han elegido **instancias spot**<sup>5</sup>, que son aquellas que oferta Amazon a un **precio bastante más bajo del estándar**, pero que los usuarios pueden **pujar** por ellas, llegando hasta un límite máximo de puja.

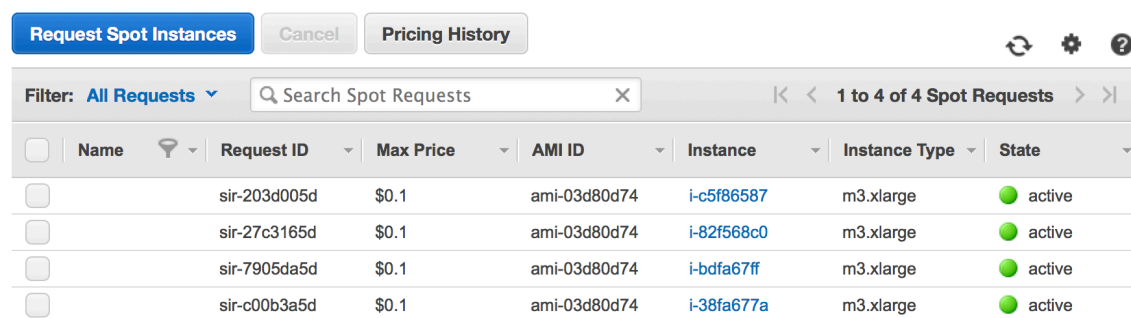
---

<sup>4</sup> <http://www.telegraph.co.uk/sport/football/world-cup/10965534/Germanys-victory-over-Argentina-in-World-Cup-final-breaks-Twitter-record.html>

<sup>5</sup> <http://aws.amazon.com/es/ec2/purchasing-options/spot-instances/>

Esto quiere decir que Amazon puede darnos máquinas por alrededor de un **70% más baratas** que si las “alquiláramos”, pero que en cualquier momento es posible que se nos **desconecte** de esa **máquina** y la **perdamos**.

De hecho, las instancias spot son un gran avance en un modelo computacional, porque somos capaces de **requerir máquinas sólo cuando las necesitamos**, pero además, como ocurre con la mayoría del software del ecosistema **Hadoop**, somos **tolerantes a fallos** (no nos importa que se pierda una máquina, porque podemos pasar los datos a una segunda, que realizará el mismo trabajo que la perdida), **el ahorro puede ser muy importante**, porque lo único que necesitaríamos es poner un precio máximo por máquina y monitorear de forma constante las necesidades de hardware, lanzando más o menos máquinas según necesitemos.



The screenshot shows the AWS Management Console interface for 'Request Spot Instances'. It includes buttons for 'Request Spot Instances', 'Cancel', and 'Pricing History'. Below the buttons is a filter section with 'All Requests' selected and a search bar. A table displays 4 spot requests, all with a 'Max Price' of '\$0.1' and 'State' of 'active'.

	Name	Request ID	Max Price	AMI ID	Instance	Instance Type	State
<input type="checkbox"/>		sir-203d005d	\$0.1	ami-03d80d74	<a href="#">i-c5f86587</a>	m3.xlarge	active
<input type="checkbox"/>		sir-27c3165d	\$0.1	ami-03d80d74	<a href="#">i-82f568c0</a>	m3.xlarge	active
<input type="checkbox"/>		sir-7905da5d	\$0.1	ami-03d80d74	<a href="#">i-bdfa67ff</a>	m3.xlarge	active
<input type="checkbox"/>		sir-c00b3a5d	\$0.1	ami-03d80d74	<a href="#">i-38fa677a</a>	m3.xlarge	active

## Cloudera Manager

Además, en estas máquinas se ha instalado la última versión de **Cloudera Manager**<sup>6</sup> (la versión 5), lo que permite tener un **control** mucho más **centralizado** y visual de todos los elementos necesarios del sistema, así como **despliegue automático** y **configuración** de los nodos de una forma muy **simple**, posibilitando la **adición** de nuevos **nodos** en un futuro para diferentes elementos del clúster.

<sup>6</sup> <http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html>



Cloudera Manager es un avanzado **sistema** de **administración** para el ecosistema de Hadoop que hace que los costes y el tiempo de administración y creación, así como del **mantenimiento** del clúster sea mucho **menor**, puesto que tiene una **gestión centralizada** y directa de los diferentes nodos, no ya solo del software que se está ejecutando, si no también de las **configuraciones** y **despliegues automáticos** de nuevos nodos por si necesitamos dimensionar el clúster en el que se está trabajando.

La **instalación** del **manager** en máquinas de **Amazon** se realiza de una forma muy **fácil**, siguiendo las instrucciones<sup>7</sup> que se pueden ver en su propia página web, donde hay que **configurar** diferentes **variables**, y realizar el **despliegue** (de nuevo, hecho de forma automática) para las diferentes **máquinas** con las que contamos.

Desde Cloudera Manager se han realizado, incluso, las **consultas** de **Hive**, a través de la interfaz de **Hue** que permite ver los resultados de una forma tabular, así como la exportación de estos a diferentes formatos de datos

<sup>7</sup> [http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM5/latest/Cloudera-Manager-Installation-Guide/cm5ig\\_install\\_on\\_ec2.html](http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM5/latest/Cloudera-Manager-Installation-Guide/cm5ig_install_on_ec2.html)



como CSV, JSON u otros, para ser **visualizado** de forma **sin conexión**, como el registro de las operaciones que se están realizando en el clúster, puesto **Hive** por debajo funciona realizando y **mapeando** a trabajos de **MapReduce** que al final son los que realizan el trabajo duro sobre los que la sintaxis de Hive es mucho más amigable y **rápida** de realizar.

The screenshot displays the Hive Editor web interface. The top navigation bar includes links for 'Editores de consultas', 'Data Browsers', 'Workflows', 'Buscar', and 'Explorador de archivos'. The main header shows 'Hive Editor' and 'Editor de consultas'. On the left, a 'Navigator' sidebar lists tables: 'sample\_07', 'sample\_08', 'tweet', and 'user'. The central area contains a SQL query editor with the text: `1 select count(*) from tweet where tweet.text like "%#%"`. Below the editor are buttons for 'Ejecutar', 'Guardar', 'Guardar como...', 'Explicar', 'o crear', and 'Nueva consulta'. At the bottom, a tabbed interface shows 'Consultas recientes', 'Consulta', 'Registro', 'Columnas', 'Resultados', and 'Gráfico'. The 'Resultados' tab is active, displaying a table with one row: 

	_c0
0	918752

# Tecnologías

---

Durante el curso se han estudiado diferentes tecnologías relacionadas con el mundo **Big Data**. Desde bases de datos, hasta software que ingiere datos en grandes cantidades, pasando por procesos **real-time** o **batch**, todos pensados para diferentes funcionamientos y paradigmas de datos y de programación.

Las diferentes partes del proyecto, que están explicadas en el siguiente apartado “Arquitectura”, requieren de diferentes tecnologías para funcionar, **ingerir los datos** y **procesarlos** adecuadamente, así como la necesidad de **analizarlos** cuidadosamente a la salida, separando la información no necesaria de aquella que es vital para que las conclusiones sean válidas.

## Ingesta de datos

El primer paso es tener un **dataset** (conjunto de datos para su estudio) suficientemente grande sobre el que realizar las queries o las **consultas** para posteriormente sacar las **conclusiones** necesarias.

En este primer paso se realiza la ingesta de datos desde **Twitter**.

Para realizar dicha ingesta, se procede a usar la librería **Twitter4j**, que permite interactuar con la API de **Twitter** de una manera muy transparente y sencilla, a través del lenguaje de programación **Java**, por lo que se puede unir a otros posibles proyectos, librerías o software para pasar dichos datos extraídos de Twitter hacia otras partes del sistema.

Para utilizar Twitter4j es necesario crear **una nueva aplicación**<sup>8</sup> dentro de Twitter, lo que permitirá conectarnos a los streamings públicos a través de su **API**<sup>9</sup>.

Sin embargo, sólo con Twitter4j no tenemos los **ingredientes** necesarios para poder recibir los twits y pasarlos a otros elementos del sistema, necesarios para **filtrarlos** (que se verán en el siguiente paso) o bien para **guardarlos** (que también se verá más adelante).

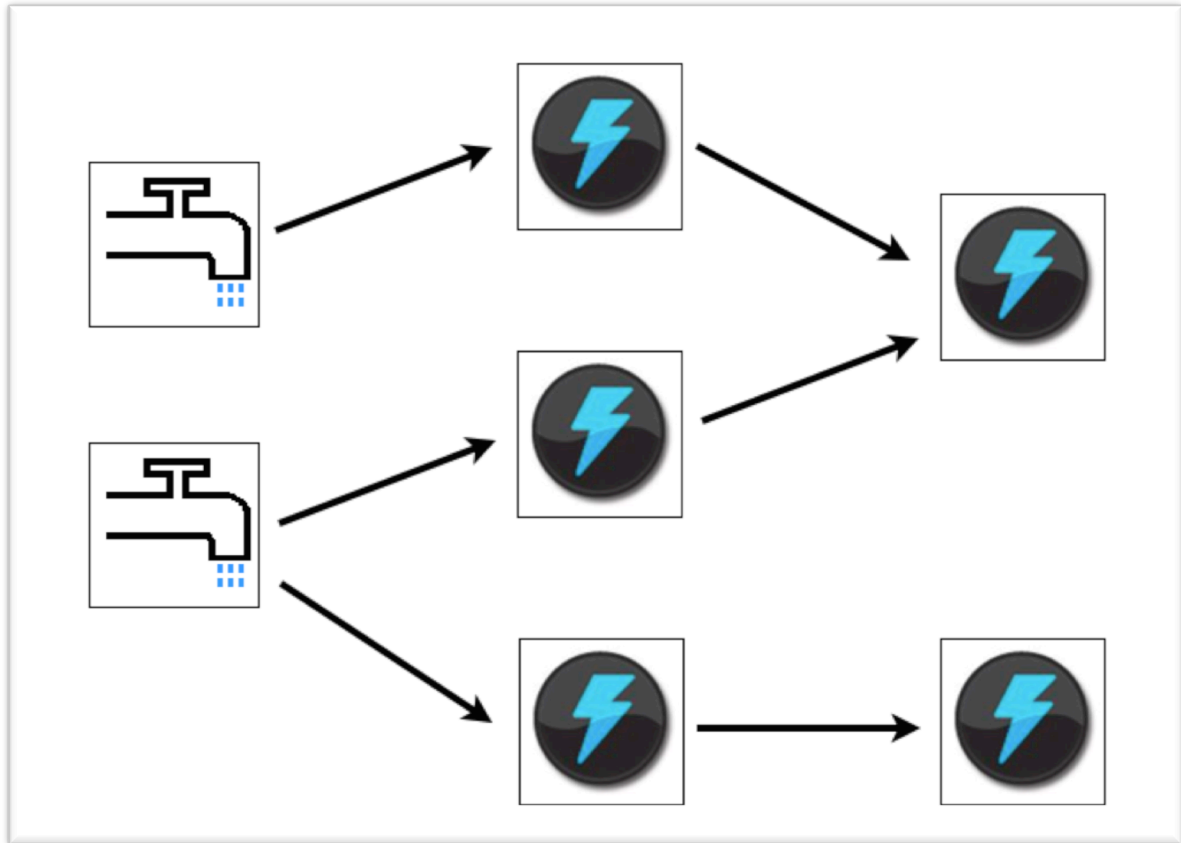
Para este paso, se utiliza **Storm**, que es un sistema de **computación distribuida en tiempo real**, que permite procesar flujos de datos de una manera continua, haciendo procesado (filtrado) de dichos datos antes de llegar a su destino. El funcionamiento de Storm es muy simple, puesto que se compone de dos elementos diferentes dentro de su topología:

- Fuentes, o **spouts**, que son las que **generan los datos**, a partir de eventos, ya sean entradas en ficheros de log, o twits, como es en nuestro caso.
- Procesadores, o **bolts**, que sirven para procesar los datos que le llegan a través de suscripciones a **colas de mensajes** (funcionando bajo **ZeroMQ**), y a tópicos.

---

<sup>8</sup> <https://dev.twitter.com/>

<sup>9</sup> <https://dev.twitter.com/docs/api/streaming>



(en la imagen, la “fuente” son los “spouts” y los “rayos” los “bolts”)

Así pues, podemos entender que **la ingesta se está realizando en los spouts**, posteriormente enviando información a los **bolts** que correspondan, que **realizan el análisis inicial**.

### Análisis inicial (filtrado)

Los datos que los spouts están generando (que a su vez vienen de los tweets que nos está devolviendo el API de twitter), son aquellos que necesitamos para analizar.

Como se ha comentado anteriormente, **Storm** permite realizar estos filtrados, o análisis, de todos los datos que nos llegan en tiempo real y con una capacidad de programación muy alta.

## Guardado de datos

El guardado de los datos se realiza en otra de las clases que implementan los **bolts**, puesto que nos permite realizar los guardados según nos va llegando las **tuplas** de datos desde los bolts anteriores, que son los que analizan los datos (en concreto, eligen sólo algunas partes de los tweets para guardar, y eliminan lo que no nos es interesante para nuestros análisis posteriores).

Sin embargo, aquí se introduce un nuevo elemento tecnológico para guardar los tweets, donde necesitamos las siguientes características:

1. **Rápido**: para guardar, puesto que vamos a estar recibiendo cientos o miles de tweets por segundo.
2. **Fiable**: que el perder una máquina no haga que perdamos todos los datos.
3. **API simple**
4. **Escalable**: puesto que si vamos a seguir ingiriendo datos, llegará un momento en el que tengamos que añadir más máquinas y que funcione sin complicaciones.
5. **Escrituras y lecturas consistentes**: no queremos perder datos, y que si alguien lee en un momento, sea lo mismo que en ese mismo momento lee otro.
6. Que **guarde** nuestros datos de forma **válida y lógica**.

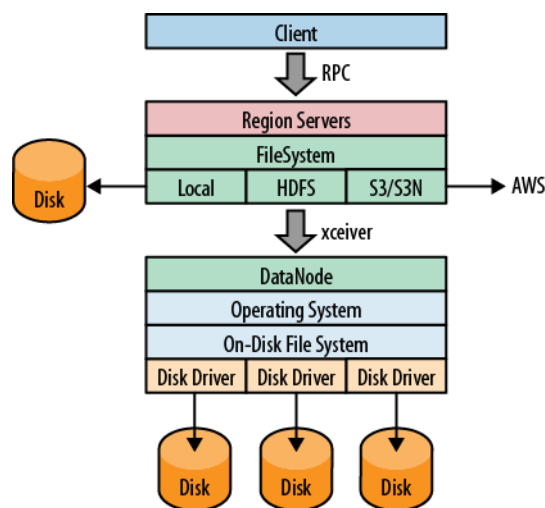
Así que la decisión fue escoger **HBase**, frente a otras tecnologías vistas en clase, como **MongoDB** u bases de datos **SQL** tradicionales, debido principalmente a que no escalan de forma linear y que la fiabilidad es inferior.

**HBase** es una base de datos pensada para realizar **accesos aleatorios** y lecturas y escrituras en **tiempo real** a los datos de Big Data que tengamos, a la vez siendo **distribuida** y muy **escalable**. Su objetivo es el de alojar tablas

muy grandes (donde muy grandes el del orden de billones de registros y millones de columnas) en un **hardware** “commodity” (que es el que podemos encontrar en las tiendas de informática o en las principales webs de venta en internet de venta al usuario final). Fue creada después de la publicación de Google sobre “**Bigtable**”<sup>10</sup> que funciona sobre el **Google File System**, o, en el mundo **Hadoop**, sobre **HDFS**.



La arquitectura de HBase se compone de dos elementos principales: un **HBase Master** y varios **HBase RegionServers**, y por supuesto, la configuración necesaria para que se ejecute **HDFS**, que es donde HBase guarda finalmente los datos (bajo la ruta **/hbase**).



<sup>10</sup> <http://research.google.com/archive/bigtable.html>

## Importación de los datos en una infraestructura de consultas

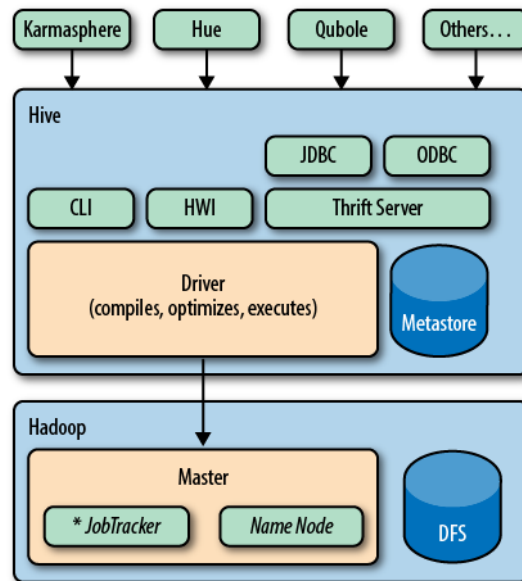
Para este paso, hay que elegir una **infraestructura robusta** que permita realizar consultas o queries a nuestros datos de una **forma sencilla y rápida**, debido a que es mucho más sencillo realizar consultas tipo **SELECT**, con modificadores como **WHERE, GROUP BY**, que escribir código Java para lanzar procesos de **MapReduce** que nos harían tener más lentitud en el desarrollo e iteración del proyecto.

Es por esto que se ha decidido usar **Hive**, que es un software para realizar consultas sobre datasets muy amplios (recordar que en el paso anterior se han guardado millones de twits en HBase), precisamente porque permite realizar **consultas simples** y porque se **integra** muy bien con **HBase** (no hace falta importar todos los datos, si no que puede consultar directamente los ficheros **HFiles** que se generan para guardar los datos en HBase).



Podemos contar, que Hive permite que muchos analistas que no tienen conocimientos de Java, o que no quieren pegarse con el mundo Hadoop (MapReduce) y las configuraciones que son necesarias, son capaces de realizar esos **mismos trabajos de una forma más rápida**, simplemente usando los conocimientos que poseen de

**lenguaje SQL**, por lo que podemos verlo como un lenguaje a alto nivel que compila a un bajo nivel “Java” para poder iterar más rápido.



Lo único necesario para que Hive funcione encima de Hadoop es un nuevo componente llamado “**Metastore**” que guarda los metadatos (como el **esquema de tablas** y la **información de las particiones** que son creadas cuando se realizan comandos como `create table` o `alter table`).

Para poder **enlazar las tablas de HBase con Hive**, es necesario ejecutar una serie de comandos que hará que se cree el vínculo necesario entre estas dos tecnologías, sin necesidad de que se dupliquen datos.

En primer lugar creamos la **tabla user** que está enlazada con la tabla en HBase del mismo nombre:

```
hive> create external table user (key string, username string) stored
by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH
SERDEPROPERTIES ("hbase.columns.mapping" = ":key,username:username")
TBLPROPERTIES ("hbase.table.name" = "User");
```

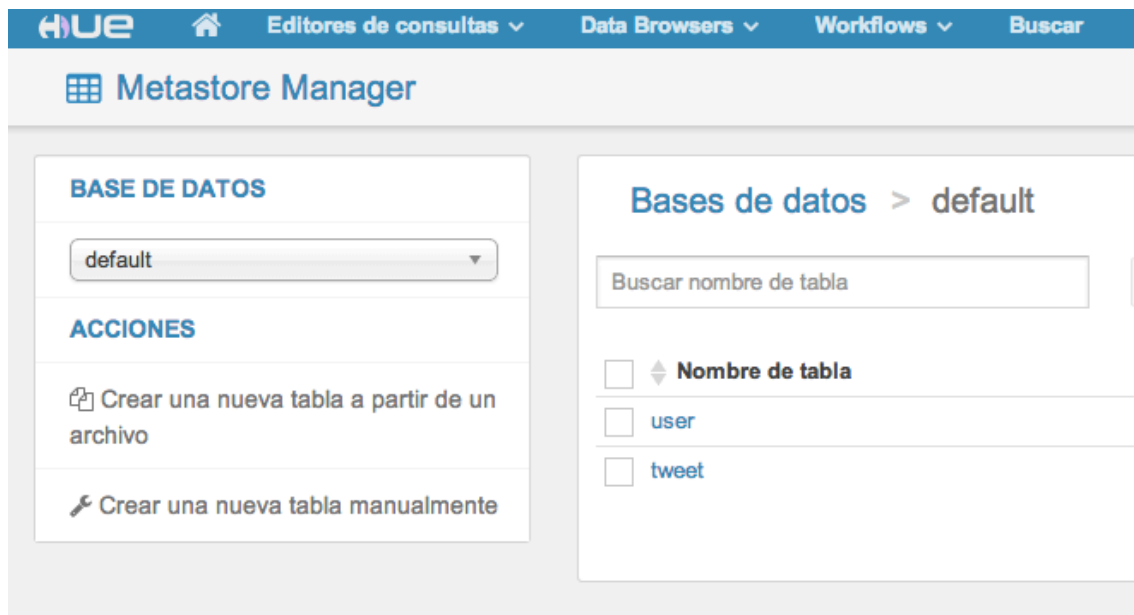
Y posteriormente ejecutamos la misma sentencia para la **tabla de tweets**:

```
hive> create external table tweet (key string, text string, userId
string, isRetweet string, userName string) stored by
```

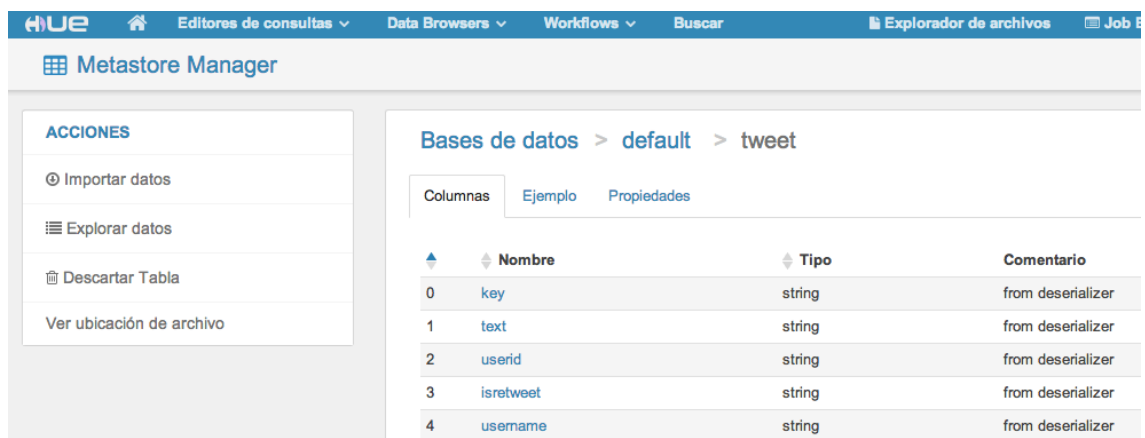


```
'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH  
SERDEPROPERTIES ("hbase.columns.mapping" =  
":key,text:text,userId:userId,isRetweet:isRetweet,userName:userName")  
TBLPROPERTIES ("hbase.table.name" = "Tweet");
```

Así que ya tenemos en el **metastore** de HBase ambas tablas, que podemos ver en el **inspector de Hue**:



En el cual además podemos ver el **formato de la tabla** según se ha introducido en los momentos anteriores:



Y también unos **datos de ejemplo** que se están mostrando automáticamente para ver si todo está correcto:

Bases de datos > default > tweet

Columnas

Ejemplo

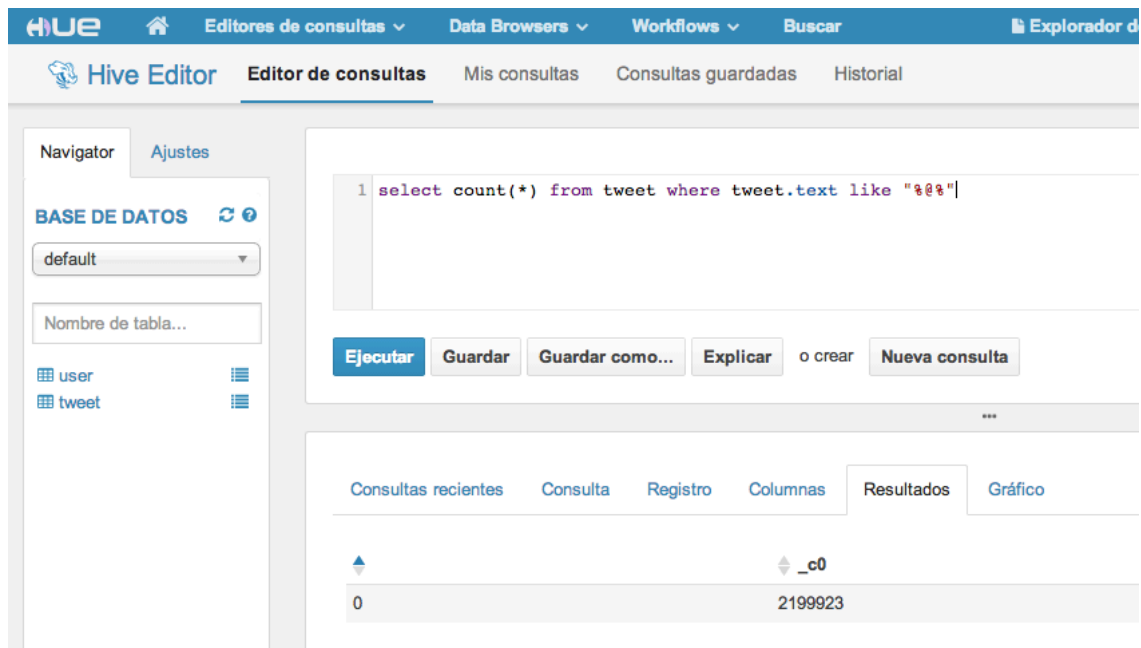
Propiedades

key	text
477724692246056961	@justinbieber I love you so much. you are my everything ,stay smiling x19
478592123806420992	@justinbieber EVERYONE IN HONDURAS DOWLOAD SHOTS!! Check out #selfie http://t.co/l2l3x9RQEg PLEA
478592124007370753	@mccannary OMG LOOK AT WHAT I FOUND http://t.co/lMwRJP69Qs
478592125018587136	@justinbieber hi Justin, I love you so, you're amazing, follow me please, you don't know how much i have tried
478592125081505793	RT @Much: Be proud, #Beliebers! ,Ã°All That Matters,Ã° is you voted to give @JustinBieber the #MMVA for Yc
478592125391896576	@justinbieber PLS FOLLOW ME, I LOVE YOU SO MUCH, @alfredoflores HAPPY BDAY MY LOVE, @sammy
478592126859890688	Nunca vi una pareja tan perfecta. Ella ten\#a unas Vans rayadas que dec\#an "Justin Bieber", \#l llevaba una p
478592127065391104	DOWNLOAD @SHOTS BY @john and check out @justinbieber's #selfie http://t.co/2SkpxMgJ1i I LOVE U SO M
478592128512450560	@justinbieber love ypu too
478592128843800576	@justinbieber beautiful http://t.co/NtXfuXev31
478592129162543105	@justinbieber i love you, follow me my love? ,õ• X 109 #MileylsOurWinner #ProudToBeSMILERS
478592129225072641	@justinbieber
478592129493925889	Congrats @justinbieber i~¥m so proud of you. Love you with my heart. Forever support you &lt;3 #belieberslovey
478592129573601281	RT @Disse_Rodrigo: ,Ãú@FloppGrande: O que dizer sobre isso: #ArianaGrandePTW http://t.co/mOTRYPYHBB

## Análisis

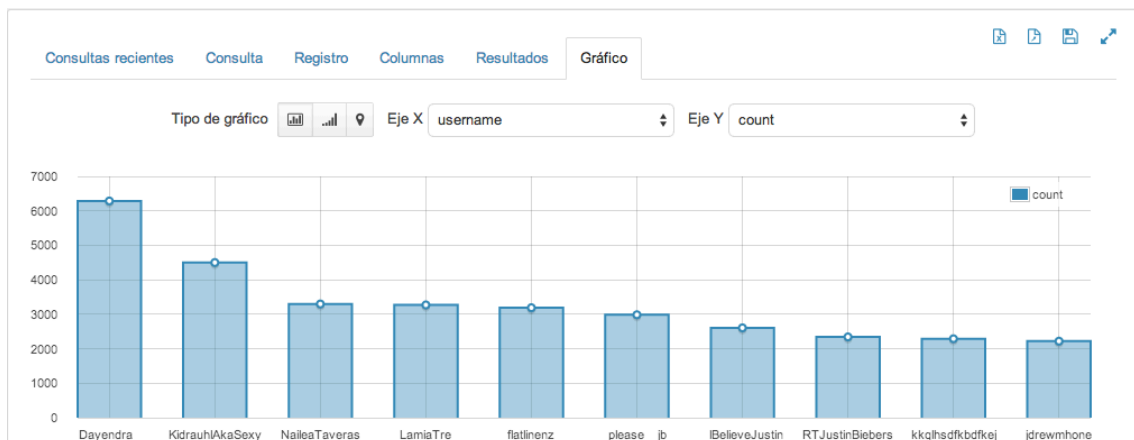
Una vez que tenemos los **datos introducidos** en HBase y **enlazados** a través de **Hive**, ya somos capaces de realizar **consultas** a dichas bases de datos a través de Hive, que como se ha explicado anteriormente es muy sencillo debido a su similitud con un lenguaje **SQL** tradicional, y que nos abstrae de muchos de los conocimientos necesarios que hacen falta tener en cuenta para usar un **MapReduce** de Hadoop escrito en Java.

A través de dicha interfaz, podemos escribir las diferentes consultas que necesitamos en esta etapa:



The screenshot shows the Hive Editor interface. At the top, there's a navigation bar with 'HUE' logo and links to 'Editores de consultas', 'Data Browsers', 'Workflows', 'Buscar', and 'Explorador d'. Below this, a sub-bar shows 'Hive Editor', 'Editor de consultas' (active), 'Mis consultas', 'Consultas guardadas', and 'Historial'. On the left, a 'Navigator' sidebar shows 'BASE DE DATOS' with a 'default' dropdown and a 'Nombre de tabla...' input. Below this, a list of tables includes 'user' and 'tweet'. The main area displays a SQL query: `1 select count(*) from tweet where tweet.text like "%%"`. Below the query are buttons: 'Ejecutar', 'Guardar', 'Guardar como...', 'Explicar', 'o crear', and 'Nueva consulta'. At the bottom, a tabbed interface shows 'Consultas recientes', 'Consulta', 'Registro', 'Columnas', 'Resultados' (active), and 'Gráfico'. The 'Resultados' tab displays a single row of results: a column labeled '\_c0' with the value '2199923'.

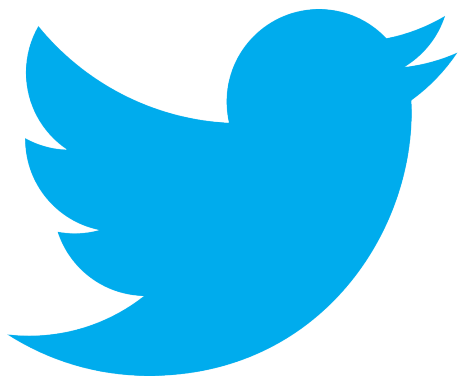
En la cual podemos ver diferentes **elementos** que podemos usar para interpretar nuestros datos, como la parte de **Resultados**, donde se nos muestra la salida de la consulta, pero a la vez podemos realizar un **gráfico** sobre los datos recibidos. Esto es interesante en el caso que queramos mostrar algo de forma muy visual.



## Datos de entrada

---

Como se ha comentado durante esta memoria, los **datos** de **entrada** que se están usando son los **twits públicos** que envían los **usuarios** a esta red social, ya sean simplemente **comentarios**, o **respuestas** o **retuiteos** a estados de otros usuarios de la red.



Esta **información** es completamente **pública** y puede ser accedida a través de las **APIs** que twitter<sup>11</sup> ofrece a los **desarrolladores**, por lo que somos capaces de poder recoger un amplio conjunto de las conversaciones que se están produciendo.

Sin embargo, y debido a restricciones de Twitter<sup>12</sup>, el API usado no nos da el **100%** de los twits que estamos esperando, ya que para poder recoger todo hay que ser un “**partner**”<sup>13</sup> **oficial** y, probablemente, pagar una alta suma de dinero.

Entonces, y según las informaciones de twitter, es **cerca del 1% del total**, lo que nos permite escalar a la baja nuestros sistemas y software, pero siempre teniendo en cuenta que si tuviéramos acceso completo al **firehose** de esta red social, necesitaríamos procesar **100 veces más información** que la que estamos jugando actualmente.

Estas restricciones del 1% se aplican a cualquier aplicación que esté usando el **API de streaming**, por lo que **no** es posible tener **más de una aplicación**

---

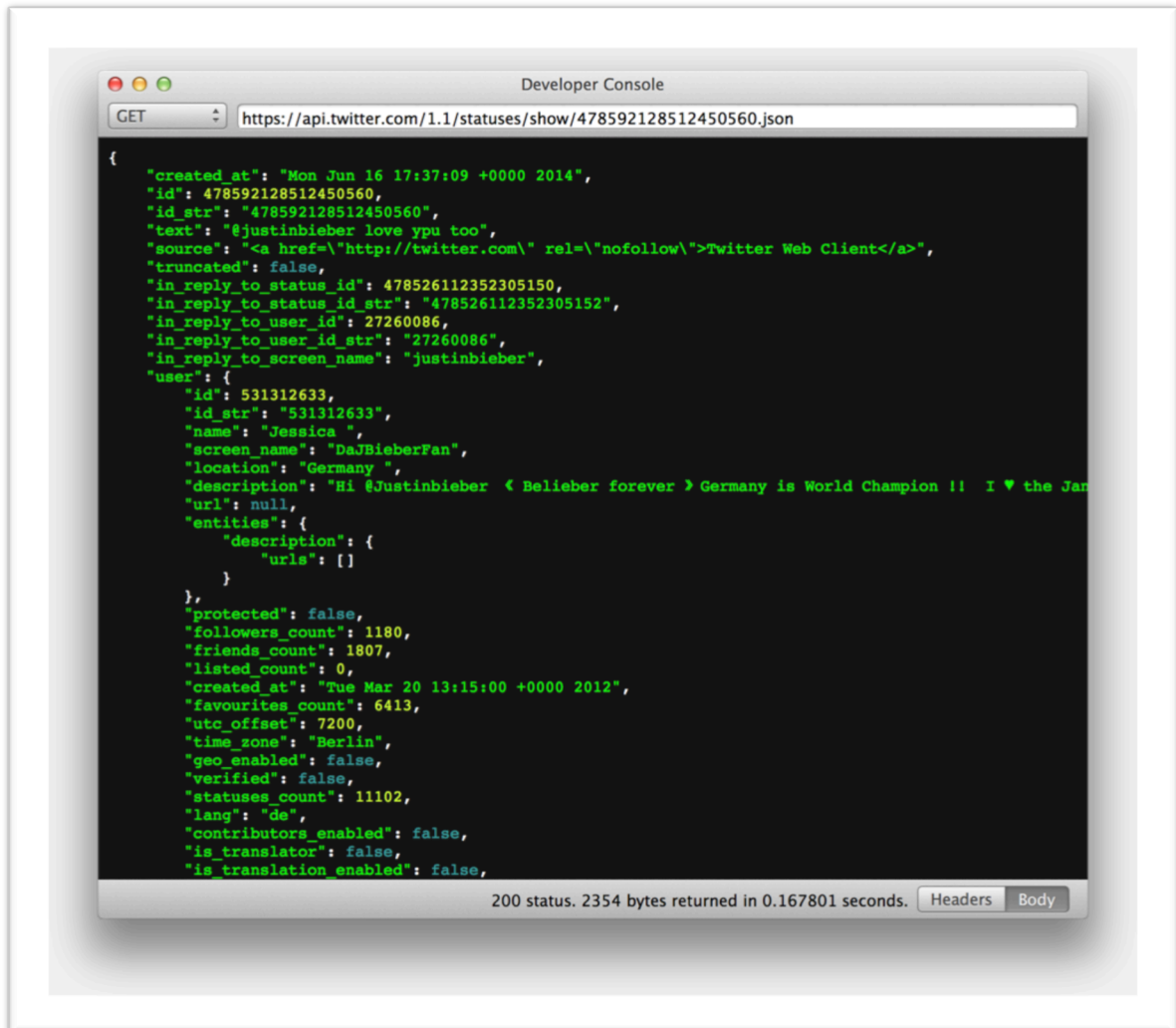
<sup>11</sup> <https://dev.twitter.com/docs/api/>

<sup>12</sup> <https://dev.twitter.com/docs/rate-limiting/1.1>

<sup>13</sup> <https://business.twitter.com/partners>

funcionando al mismo tiempo para recoger, en el mismo hashtag, diferentes tweets, porque **Twitter** los **filtra** en su origen, y no cuando van a ser entregados.

Además, es necesario recalcar que el **API** de streaming de twitter **obliga** a mantener **una única conexión** con sus servidores, pero que permiten un



límite mucho más alto de operaciones que las que se pueden hacer contra el API 1.1, por lo que no es posible utilizar dos o más conexiones con una misma tupla de claves privadas para recibir diferentes hashtags.

Estas **restricciones** pueden **superarse** también si se usan uno de los **servicios externos** a twitter pero que tienen acuerdos con ellos, y que

**permiten** interactuar con **todos** los **twits** que se han recibido, puesto que tienen una base de datos gigantesca en la cual almacenan todos los mensajes desde el inicio de la red social.

Algunos de estos servicios son provistos por empresas como **DataSift**<sup>14</sup> o **Gnip**<sup>15</sup> (esta comprada por Twitter en abril de 2014<sup>16</sup>), que **venden** la **información** recogida de **Twitter**, a veces incluso añadiendo más datos como conexiones con otras redes o noticias que han enlazado a dichos twits.

Así pues, la **ingesta** de los datos se ha realizado con el **API** de **streaming** de Twitter, juntándola con la tecnología **Storm** y **guardando** en **HBase** para mantener una **persistencia** de dichos datos para después realizar análisis sobre ellos como se mostrarán posteriormente, ya sea a través de **Hive** o de procesos **batch (MapReduce)** de **Hadoop**.

#### Información de los datos recopilados durante el periodo de recogida

- **Fecha:** desde el 24 de junio a las 14:34:12 hasta el 3 de julio a las 20:56:21
- **Keywords:** "Justin Bieber", "Bieber", "@justinbieber", "believers", "believer", "#beliebers"
- **Número total de twits:** 2534999
- **Total de usuarios** diferentes: 431184
- **Tamaño en MongoDB:** 28.74GB

---

<sup>14</sup> <http://datasift.com/>

<sup>15</sup> <http://gnip.com/>


<sup>16</sup> <https://blog.twitter.com/2014/twitter-welcomes-gnip-to-the-flock>


## Análisis

---

En primer lugar, simplemente vamos a ver cuántos twits tenemos, y cuántos usuarios.

### ¿Cuántos twits y usuarios tenemos?

SELECT COUNT(*) as Total FROM tweet;
 <b>total</b>
2534999

SELECT COUNT(*) as total_users FROM user;
 <b>total_users</b>
431184

Así podemos ver que los usuarios totales son mucho menores que los twits que se han guardado en HBase, esto, en una aproximación fácil, podemos ver como que de media podemos obtener que cada usuario ha escrito:

$$twits/usuario = \frac{2534999}{431184} \approx 5.88$$

Sin embargo, esta es una aproximación con un gran grado de fallo, puesto que suponemos que todos los usuarios que hablan sobre nuestro músico lo hacen en la misma proporción, así que la siguiente pregunta sería...

### ¿Cuántos twits manda cada usuario?

Podemos incluso filtrar por aquel top 20 de usuarios (y además, buscamos aquellos que hayan escrito más de 100 twits).

```
select tweet.username, count(*) as count from tweet group
by tweet.username having count (*) > 100 order by count
DESC LIMIT 20
```

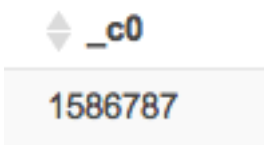
username	count
Dayendra_	6287
KidrauhlAkaSexy	4501
NaileaTaveras	3299
LamiaTre	3271
flatlinenz	3190
please__jb	2986
_IBelieveJustin	2607
RTJustinBiebers	2344
kkqlhsdfkbfkej	2291
jdrewmhone	2220
FANSJustinMUSIC	2204
JBizzleFollowMe	2177
JB_Bizzle	2103
kiki_du_bois	2076
AnnieAny1	1999
De_Dea_Vionita	1992
Myidolforever_	1894
jess_belieba	1841
Belieber_Girlxd	1776
JustinsNiall	1748

Así que podemos ver que hay muchos usuarios que son muy activos hablando de nuestro querido músico.




### ¿Cuántos twits mencionan a nuestro artista?

Durante el tiempo en el que se recogió la muestra (recordamos, un 1% de todo lo que se habla en Twitter sobre nuestros criterios de búsqueda), se han encontrado cerca de 1.6M de menciones a nuestro artista seleccionado.

<pre>select count(*) from tweet where tweet.text like "%@justinbieber%"</pre>


### ¿Qué hashtags utilizan? ¿Cuáles son?

Los hashtags son etiquetas, conformadas como cadenas de caracteres, comenzando por el símbolo #, que utilizan los usuarios para marcar sus comentarios acordes a una conversación o tema similar. Podemos contarla con la siguiente sentencia:

<pre>select count(*) from tweet where tweet.text like "%#%"</pre>


Aquí podemos ver la cantidad de twits que contienen hashtags, puesto que es prácticamente un 36% de ellos, esto quiere decir que el uso de hashtags en las redes sociales, y en concreto Twitter es muy alto, indicando una tendencia a la alza y que se puede ver todos los días en medios de comunicación de masas, donde para seguir una misma conversación invitan a sus oyentes a escribir el mismo hashtag, ya sea para generar una discusión entre ellos o para utilizarlos en los propios programas.

Mediante consultas más complejas de Hive, somos capaces de sacar cuántos hashtags diferentes encontramos entre todos los twits que se han recogido:


```
select tbl.hashtag, count(*) as count from
  (select split(lower(text), "[^a-zA-Z_0-9@#]+") as words
   from tweet) arr
  LATERAL VIEW explode(arr.words) tbl AS hashtag
  where  tbl.hashtag rlike  "^#[a-zA-Z]+$"  group  by
tbl.hashtag order by count DESC
```

hashtag	count
#mtvhottest	260823
#selfie	143701
#beliebers	97254
#mmva	29499
#selfies	27456
#artistadoano	22719
#mtvkickoff	17820
#belieberfandommemories	17312
#fandomdoseculo	15522
#rt	13682
#etalkmmvas	11922
#justinbieber	8287
#unbreakable	6488
#shots	5094
#gossip	4823
#celebrities	4612
#news	3677
#believe	3654
#justinbieberptw	3468

### ¿Cuántas menciones hay? ¿Cuál es el usuario más mencionado?

De nuevo, podemos usar Hive para ver cuántos twits tienen menciones, con una consulta similar a la anterior.

```
select count(*) from tweet where tweet.text like "%@";
```


 \_c0  
 2199923

Pero también necesitamos un MapReduce para ver cuáles son los usuarios más mencionados:

```
select tbl.mention, count(*) as count from
```

```
(select split(lower(text), "[^a-zA-Z_0-9@#]+") as
words
from tweet) arr
LATERAL VIEW explode(arr.words) tabl AS mention
where tabl.mention rlike "^@[a-zA-Z]+$" group by
tabl.mention order by count DESC
```

mention	count
@justinbieber	1708870
@shots	498197
@john	461997
@alfredoflores	74692
@mtv	21353
@madisonellebeer	21258
@much	17713
@biebersmaniabr	13640
@etalkctv	12264
@idoloscurosos	11877
@scooterbraun	11806
@jbcrowdotcom	11702
@selenagomez	10824
@quincy	8526
@thatrygood	8445
@kidrauhlakacute	6854
@maejorali	6653
@secutebelieber	6461
@youtube	5823

### ¿Cuáles son las palabras más repetidas? ¿Y los trigramas?

Podemos también ver cuáles son las palabras más repetidas en los twits que hemos analizado simplemente haciendo trigramas:

```
SELECT ngrams(sentences(lower(text)), 3, 10) FROM tweet;
```

```
▼ Array[10]
▼ 0: Object
  estfrequency: 392879
  ▼ ngram: Array[3]
    0: "i"
    1: "love"
    2: "you"
    length: 3
    ► __proto__: Array[0]
    ► __proto__: Object
  ► 1: Object
  ► 2: Object
  ► 3: Object
  ► 4: Object
  ▼ 5: Object
    estfrequency: 287501
    ▼ ngram: Array[3]
      0: "please"
      1: "follow"
      2: "me"
      length: 3
      ► __proto__: Array[0]
      ► __proto__: Object
    ► 6: Object
    ► 7: Object
  ▼ 8: Object
    estfrequency: 242105
    ▼ ngram: Array[3]
      0: "mtvhottest"
      1: "justin"
      2: "bieber"
      length: 3
      ► __proto__: Array[0]
      ► __proto__: Object
    ► 9: Object
      length: 10
    ► __proto__: Array[0]
```

Eliminando algunos de los twits que son publicidad, nos quedamos con los siguientes trigramas más repetidos:

- “I love you”, repetido 392879 veces
- “Please follow me”, repetido 287501
- “Mtvhottest justin bieber”<sup>17</sup>, repetido 242105 veces.

### ¿Hay alguna believer que le pida matrimonio a Justin Bieber?

Parece ser que algunas, viendo los resultados de la consulta...

select text, username from tweet where tweet.text like "%marry%"	
	text
0	Shit. The battery's low ugh. Let's pretend I'm Justin Bieber!!! I fucking love my #Beliebers. I wanna marry every single one of u :**
1	@8ball_ but am i marrying justin bieber
2	@LUXURIOUSMCCANN @justinbieber he can marry us two
3	will you marry me? ,ô @justinbieber http://t.co/BmEzqh2dOX
4	RT @SelinaDenmark: I'd like to marry, Zac Efron, Ed Westwick, Dave Franco, Chace Crawford, Justin Bieber, Ian Somerhalder, Paul Wesley, Cha,Ä¶
5	@bizzlestiger @justinbieber @DebbyRyan @Brittanysmooch can u tell him to follow me it would mean the weld to no lie but i want to marry him
6	Will you marry me @justinbieber http://t.co/CkTsndRqe6
7	@justinbieber f±n love you bizzle marry me
8	I your first like. now marry me lol @justinbieber http://t.co/87HHYolpzK
9	@justinbieber marry me?
10	@justinbieber Jiley ,ô*,ô*,ô*Justin please marry @MileyCyrus ,you two are perfect together ,ô* http://t.co/Svk7iRUT6D
11	,Äú@MentionTo: #MentionTo someone you want to marry like crazy.,Äú @justinbieber
12	@justinbieber maybe one day you'll marry me :)))
13	RT @Baileydance22: ,Äú@MentionTo: #MentionTo someone you want to marry like crazy.,Äú @justinbieber
14	@8ball_ Will @justinbieber marry @selenagomez ? *~*
15	@justinbieber i think i want to marry you.

Y contando el número de mensajes que se han recibido con la palabra “marry”...

select count(*) from tweet where tweet.text like "%marry%"	
	_c0
	1334

<sup>17</sup> <https://blog.twitter.com/2013/mtvhottest-generates-166-million-tweets-with-a-single-hashtag>

Y además, podemos ver que hay algún usuario (o usuaria) que se dedica a realizar declaraciones de amor de forma constante a nuestro artista estudiado:

```
select text, username from tweet where text like "%marry%" AND
username like "Catarina_Bizzle"
```

	text	username
0	@justinbieber Babe , will you marry me?	Catarina_Bizzle
1	@justinbieber Babe , will you marry me? x2	Catarina_Bizzle
2	@justinbieber Babe , will you marry me? x3	Catarina_Bizzle
3	@justinbieber Babe , will you marry me? x4	Catarina_Bizzle
4	@justinbieber Babe , will you marry me? x5	Catarina_Bizzle
5	@justinbieber Babe , will you marry me? x6	Catarina_Bizzle
6	@justinbieber Babe , will you marry me? x7	Catarina_Bizzle
7	@justinbieber Babe , will you marry me? x8	Catarina_Bizzle
8	@justinbieber Babe , will you marry me? x9	Catarina_Bizzle
9	@justinbieber Babe , will you marry me? x10	Catarina_Bizzle
10	@justinbieber Babe , will you marry me? x11	Catarina_Bizzle
11	@justinbieber Babe , will you marry me? x12	Catarina_Bizzle

Con una cuenta bastante alta:

```
select count(*) from tweet where text like "%marry%" AND
username like "Catarina Bizzle"
```

◆ \_c0  
213

## Conclusión

---

En primer lugar, mediante el estudio hemos podido ver que los usuarios que hablan sobre Justin Bieber (ya sea mencionándole, o hablando con #hashtags específicos), son muy activos, con varios de ellos publicando constantemente este artista. De hecho, mucho de los usuarios llegan al extremo de que en su propio nombre tienen algo relacionado con Justin Bieber (como “please\_jb” o “\_IBelieveJustin” o “RTJustinBiebers”).

Además, se puede ver que el uso de hashtags es muy alto, y que para seguir las conversaciones, los diferentes usuarios se guían entre ellos, generando nuevas informaciones y comentarios a través de ellos.

En tercer lugar vemos cuál es el sentimiento de las “believers”: quieren a Justin Bieber, mucho, tanto como para pedirle matrimonio de forma constante, y para que gane los concursos más televisivos, como el MTV Hottest Award.

Finalmente, se aprecia que el usuario @justinbieber es muy mencionado (1.7M de veces, frente a 2.5M de tuits descargados), pero también se puede ver que hay otros que son muy comentados, como @shots, una aplicación que parece ser que nuestro artista utiliza en innumerables ocasiones, y que seguramente sus seguidores hayan empezado a usar siguiendo los patrones de su querido artista. También @john, que es el creador de @shots, la aplicación de selfies comentada anteriormente.

Incluso tiene influencia en las compras de sus fieles seguidoras, como se puede ver en el siguiente tuit:



**ASKFORFOLLBACK**

@Sharaa25

I bought an android just for download @shots by @john so i can see @justinbieber's selfie!!  
#MTVHottest Justin Bieber

24/07/14 17:43

Para concluir, Twitter se ha demostrado que es una gran herramienta para crear conversaciones entre los diferentes usuarios de la red, sin tener que conocerse entre ellos, si no que simplemente ayudan a hablar de un mismo tema o idea, y que incluso algunas empresas utilizan para realizar encuestas rápidas debido a su facilidad y gratitud.



## Bibliografía

---

White, Tom. “Hadoop. The definitive Guide”, 2010, O'REILLY

Sammer, Eric. “Hadoop Operations”, 2012, O'REILLY

George, Lars. “HBase. The definitive Guide”, 2011, O'REILLY

Russell, Matthew A. “Mining the social web”, 2013, O'REILLY

Leibiusky, Jonathan; Eisbruch, Gabriel; Simonassi, Dario. “Getting Started with Storm”, 2013, O'REILLY

Chodorow, Christina. “MongoDB. The definitive Guide”, 2013. O'REILLY