Facultad de Ingeniería, UdelaR

Recuperación de Información y Recomendaciones en la Web

Curso 2019 - Grupo 18

Buscador de usuarios relevantes por temática en Twitter

Profesora:
Libertad Tansini

Estudiantes:

Verónica Bentancor	**CI: 5.099.465-3**
Nicolas Bermudez	**CI: 4.730.288-7**
Rodrigo Sastre	**CI: 4.672.828-2**
Fernando Outeda	**CI: 4.722.770-6**

Índice

1. Introducción

Hoy en día es cada vez más común y masivo el uso de redes sociales como Twitter para informarse sobre diversos temas de actualidad, tanto a nivel regional como mundial.

Tal es el alcance y la importancia de esta red social, que cuenta con 330 millones de usuarios activos por mes, de los cuales 139 millones lo hacen diariamente, y maneja un volumen de 500 millones de *tweets* por día. (*Twitter Statistics*, s.f.).

Al tener un flujo tan grande de información disponible, surge la necesidad por parte de los usuarios de poder seleccionar la fuente de dicha información en dicha plataforma, a través de cuentas confiables que dominen los temas de interés del usuario. Esta tarea resulta inalcanzable por un humano, debido a la gran cantidad de información y diversos factores a tener en cuenta al momento de elegir a quién escuchar.

Este trabajo tiene el objetivo de utilizar técnicas de recuperación de información en la web para ayudar a encontrar qué usuarios de *Twitter* tienen mayor influencia en lo que respecta a un determinado asunto de actualidad.

2. Descripción del problema

En esta red social hay una gran cantidad de usuarios escribiendo tweets con información cada minuto, esto hace que haya mucha información y de diversos temas. Como consecuencia, resulta difícil para un usuario informarse de una temática en particular (más aún si no está interiorizado en la misma), y saber a qué usuario leer.

La aplicación buscará acercar al usuario a la temática, retornando los usuarios más relevantes para el tema consultado por este. Para esto, se contarán con diversos criterios entre los que el usuario podrá elegir.

Que sea relevante un usuario implica que este hable mucho sobre el tema, que tenga tweets con una alta repercusión cuando traten del tema, y otros diversos parámetros que den a entender una mayor importancia del usuario frente a los demás.

3. Indicadores de influencia

Para poder determinar la relevancia de un usuario sobre otros en determinada temática, se decidió emplear algunos de los indicadores más utilizados para determinar el grado de influencia de cuentas de Twitter. La aplicación permitirá seleccionar uno o varios indicadores dentro de los listados a continuación, de forma de proporcionar al usuario el control de los criterios de relevancia.

- Cantidad de seguidores.
- Proporción de seguidores/seguidos (3 tipos de cuentas).
  - Spammer. Baja proporción, el de menor influencia.
  - Conversador. Proporción cercana a 1 seguidor/1 seguido, mayor influencia que spammer.
  - Celebridad. Proporción alta, influencia similar a conversador.
- Capacidad de generar acciones.
  - Respuestas a tweets.
  - Retweets. Una medida muy utilizada es la del promedio de retweets por tweet.
  - Menciones.
  - Likes.
- Potencia de seguidores. Si los seguidores de la cuenta tienen a su vez una gran cantidad de seguidores, potenciales retweets tendrán mayor relevancia.
- Actividad reciente. Las cuentas que actualizan su contenido con frecuencia se consideran más influyentes.
- Cantidad de listas a las que pertenece la cuenta.
- Si se trata de una cuenta verificada o no. En caso de ser así, puede considerarse de mayor influencia.

4. Integración con Twitter

4.1. API de Twitter

La API devuelve una colección de Tweets relevantes que coinciden con una consulta específica. Los datos que harán posible esta aplicación van a ser extraídos usando la capa gratis de la API de Twitter (Standard API). Esta API provee varias funcionalidades dentro de ellas existe la Standard Search API la cual permite buscar Tweets relevantes de hasta 7 días atrás que correspondan con algún criterio de búsqueda.

La url del endpoint es : https://api.twitter.com/1.1/search/tweets.json y acepta diversos parámetros entre ellos,

- q - La "query" de búsqueda de 500 caracteres máximo.
- geocode - Permite retornar tweets que sean de usuarios que se encuentran en algún radio dado de un punto latitud longitud deseado.
- result_type - Especifica el tipo de búsqueda que se desea realizar, las opciones son:
  - mixed - Se retornan los tweets más populares y recientes.
  - recent - Se retornan solamente los tweets más recientes en el resultado.
  - popular - Se retornan solamente los más populares por defecto la opción establecida es mixed

- include_entities - Permite filtrar resultados por metadata del contenido a retornar, por ejemplo, por uno o múltiples hashtags.

Como respuesta la API retorna los tweets junto con información de la cuenta del usuarios que realizó el tweet, incluyendo información necesaria (retweets, likes, seguidores de la cuenta, etc) para que nuestra app devuelva un resultado relevante.

4.2. Capas de desarrollo

Las tres capas principales de *Twitter*, *Standard*, *Premium* y *Enterprise* ofrecen distintas características y capacidades. La principal diferencia entre la capa *Standard* y las capas por suscripción (*Premium* y *Enterprise*), son la capacidad de obtener *tweets* de los últimos 30 días, y la posibilidad de obtener acceso al archivo completo de *Twitter* desde el primer *tweet* en 2006. (*Twitter API Reference - Premium Operators*, s.f.)

La figura 1 muestra a grandes rasgos las características de las diferentes capas disponibles.(*Twitter API Reference - Pricing*, s.f.):

Tweets	Standard (free)	Premium	Enterprise
Publish and engage	✓		
Search Tweets: 7-days	✓		
Search Tweets: 30-days		✓	✓
Search Tweets: Full-archive		✓	✓
Filter Tweets	✓		✓
Sample Tweets	✓		✓
Batch Tweets			✓
Direct Messages	✓		
Account and users	✓	✓	✓
Metrics			✓
Ads API	✓		
Publisher tools and SDKs	✓		

Figura 1: Comparación de capas de la *API*

5. Arquitectura

La arquitectura de la solución cuenta con múltiples componentes. Una aplicación web realizada en *Angular*, que se comunica con un *backend* implementado en *NodeJs*. El componente de back se encarga de la comunicación con el componente de persistencia (una base de datos en *Elasticsearch*) y con la *API* de *Twitter*.

Se decidió utilizar Elasticsearch ya que soporta el gran volumen de datos que representa la consulta de tweets, permite realizar consultas de forma eficiente sobre dicho volumen y resulta especialmente compatible con los documentos JSON (formato de las respuestas de Twitter API). NodeJS y Angular se utilizaron debido a su performance en tiempos de respuesta (en NodeJS se debe a su ejecución asíncrona) y a la experiencia con la que contaba parte del equipo en dichas tecnologías.

A continuación se muestra la comunicación entre los cuatro componentes.

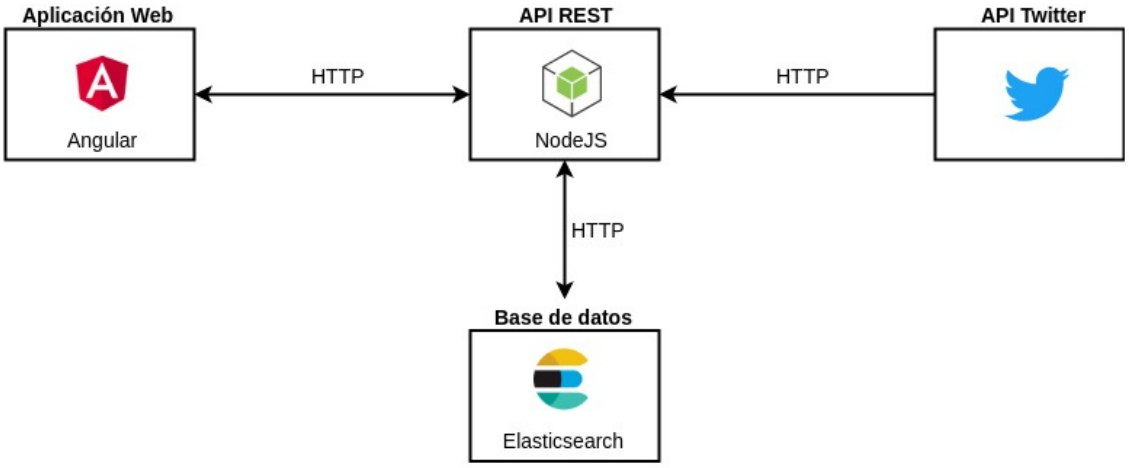

Figura 2: Arquitectura de la solución y tecnologías a utilizar.

6. Implementación

La solución fue implementada con las tecnologías que se pueden ver en el diagrama de la Figura 4. Se utilizó *Angular* en su versión 8.0 para realizar el *frontend*, *nodeJS* versión 10.16.3 para implementar el *backend* y *elasticsearch* para persistir los datos y realizar consultas sobre los mismos. El flujo de la aplicación es el siguiente:

- El frontend permite ingresar términos de búsqueda y seleccionar de acuerdo a qué indicador se quiere buscar.
- Se realiza un GET al endpoint del backend que corresponda según el indicador seleccionado.
- El backend consulta la API de Twitter para obtener todos los tweets que contienen los términos de búsqueda. Estos tweets se almacenan al recibirse en la base de datos de elasticsearch.
- Una vez almacenado los tweets de la búsqueda se utilizan consultas especificas sobre la base de datos, para obtener el ranking de los usuarios.
- Finalmente se retornan los usuarios encontrados al front con su información asociada, y se despliega la información en la página.

6.1. Autenticación con API Twitter

Una vez que el pedido de acceso a la API es admitido, se obtienen los siguientes datos de acceso:

- *Consumer key*
- *Consumer secret*
- *OAuth token*
- *OAuth token secret*

Para utilizar la *user-based authentication* con OAuth 1.0 provista por Twitter, es necesario agregar a cada request las credenciales obtenidas de Twitter, junto con algunos valores generados por la aplicación (nonce, timestamp, método de codificación y firma que depende de los valores previos).

Es por esto que se resolvió utilizar la librería de NodeJS *twitter*[1]. La misma permite realizar los distintos pedidos a la API de Twitter, facilitando la autenticación. De esta manera, toma los valores de *consumer key, consumer secret, oauth token* y *oauth token secret*, generando todos los cabezales de autenticación necesarios para realizar cada request. Por lo tanto, todos los llamados a la API en cuestión se hacen a través de este módulo.

6.2. Respuesta de la *API Twitter*

La respuesta de la *API* frente al pedido del *backend* consiste en un *array Json* con los campos relevantes al *Tweet*, *Listing 1* muestra un ejemplo de respuesta *Json* (*Twitter API Reference - Get Search Tweets*, s.f.), entre ellos se destacan:

- *created_ at*: Fecha de publicación del *Tweet*.
- *text*: Texto del *Tweet*.
- *hashtags*: Lista de *hashtags* del *Tweet*.
- *screen_ name*: Nombre de usuario del autor del *Tweet*.
- *followers_ count*: Cantidad de seguidores del autor.
- *friends_ count*: Cantidad de amigos del autor.
- *verified*: Si la cuenta del autor es verificada o no.

[1] https://www.npmjs.com/package/twitter

Listing 1: Ejemplo de respuesta *Json* de la *API Twiter*

```
{
  "statuses": [
    {
      "created_at": "Mon May 06 20:01:29 +0000 2019",
      ...
      "text": "Todays new update means that you can...",
      ....
      "entities": {
        "hashtags": [],
        ...
        "user_mentions": [],
        ....
      "user": {
        "id": 2244994945,
        "id_str": "2244994945",
        "name": "Twitter Dev",
        "screen_name": "TwitterDev",
        ...
        "followers_count": 501947,
        "friends_count": 1473,
        "listed_count": 1507,
        ...
        "verified": true,
        ...
  }
}
```

6.3. Indicadores implementados

En esta sección se desarrollarán algunos puntos relevantes en la implementación de los endpoints que conforman la API REST en NodeJS, correspondientes a cada indicador y criterio de búsqueda provistos por la aplicación.

6.4. Retweets

El endpoint /**influencersByRetweets** retorna los 10 usuarios de Twitter con mayor cantidad de retweets promedio por tweet, para el tópico consultado (en los últimos 7 días). El tema por el que se desea consultar se debe pasar como un parámetro en la URL, en el cual se espera una lista de hashtags separados por coma. Por ejemplo:

```
/influencersByRetweets?hashtags=uruguay,seguridad
```

Al igual que para el resto de los endpoints, para la implementación de influencersByRetweets se utilizaron operaciones de los controladores *Tweets* e *Influencers*. El primero encargado de realizar el pedido correspondiente a la API de Twitter, mientras que el segundo lleva a cabo la interacción con Elasticsearch y el armado de la respuesta al pedido que recibe la API REST.

De esta manera, debido a la capacidad de Elasticsearch de almacenar en sus índices grandes volúmenes de datos en formato JSON y de realizar consultas eficientes sobre estos, se decidió almacenar los tweets completos obtenidos. Asimismo, se simplifican las consultas a Twitter y el procesamiento de su respuesta, delegando la complejidad a las consultas ejecutadas sobre Elasticsearch luego de almacenada la totalidad de los tweets. Para este indicador en particular, se utilizó la siguiente consulta:

```
{
    "_source": ["user.screen_name", "user.profile_image_url"],
    "aggs": {
      "group_by_user": {
        "terms": {
          "field": "user.id",
          "order": {
            "total_retweets": "desc"
          }
        },
        "aggs": {
          "total_retweets": {
            "avg": {
              "field": "retweet_count"
            }
          }
        }
      }
    }
}
```

6.5. Seguidores

Análogamente se hizo lo mismo que lo explicado en el punto anterior pero en ves de utilizar el número de retweets se utilizó el número de seguidores del usuario que escribió el tweet.

6.6. Cuentas verificadas

Como se mencionó anteriormente, la *Standard Search* de *Twitter* no permite filtrar *tweets* por cuentas verificadas (*Twitter Premium Operators*, s.f.), sin embargo, se logró filtrar desde *ElasticSearch*.

Los *tweets* se consiguen a través de la *API* utilizando cualquiera de los indicadores anteriores y luego se filtran mediante *ElasticSearch*, utilizando *aggregate boolean filters* (*Search Aggregations*, s.f.).

Es posible filtrar manualmente los *tweets* ya que dentro de la respuesta de la *API* se encuentra un campo *verified*, el cual podemos filtrar desde *ElasticSearch* (*Twitter API Reference - Get Search Tweets*, s.f.), agregando los siguientes campos a cualquiera de los demás indicadores:

```
"query": { "match": { "user.verified": "true" } }
```

Por otro lado, con una cuenta *Premium* o *Enterprise*, se podrían filtrar los *tweets* directo desde una request a la *API* utilizando el operador *is:verified* como filtro de *tweet* (*Twitter Premium Operators*, s.f.)

7. Uso de la aplicación

El uso de la aplicación es muy simple. Al ingresar a la página se encuentra una barra de búsqueda donde el usuario puede ingresar los diferentes términos de búsqueda.

Luego se selecciona qué indicador se desea utilizar si por *retweets* o por cantidad de seguidores, y si se quiere solamente tener en cuenta usuarios verificados.

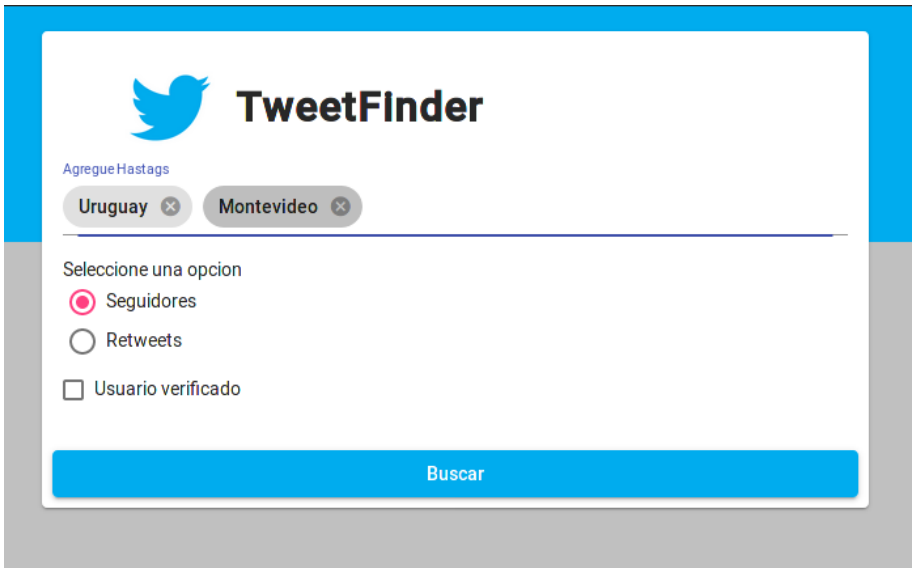

Figura 3: Página principal de la aplicación

Una vez presionado buscar se va a desplegar un listado con los diez usuarios más relevantes para la búsqueda realizada.

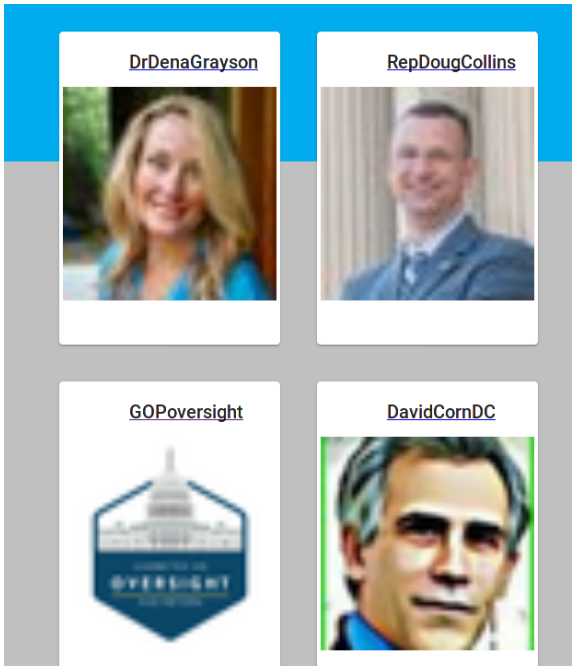

Figura 4: Resultado de busqueda para las palabras USA Impeachment

8. Evaluación y resultados

El resultado obtenido al realizar las búsquedas fue bueno, ya que utilizándose los términos adecuados se pueden encontrar usuarios realmente relevantes para ciertos temas de interés. Por otra parte, el hecho de que solamente se puedan encontrar *tweets* de los últimos 7 días presenta un problema ya que pueden aparecer usuarios que tuvieron un solo *tweet* con una gran repercusión sobre el tema en ese periodo de tiempo, donde la cuenta de este quizás no sea la más apropiada para informarse sobre ello.

En cuanto al estudio de las cuentas validadas, se encontró que en muchos casos, los usuarios más influyentes provenían de cuentas verificadas.

9. Conclusiones

En cuanto al estudio de las cuentas validadas, se mencionó en los resultados del estudio la influencia de las cuentas verificadas en *Twitter*. Se podría expandir esta hipótesis y estudiar a futuro qué tanto se mantiene esta tendencia, y comparar su influencia con las cuentas no verificadas.

En conclusión el trabajo presento resultados positivos. En primer lugar, en la mayoría de las pruebas realizadas, utilizando los términos de búsqueda apropiados, se obtuvieron usuarios que tenían relación con los temas buscados. Por otra parte, los indicadores elegidos, si bien se trata de criterios básicos, demostraron tener una buena capacidad de clasificación. Igualmente hay posibilidades de trabajar sobre este proyecto y extenderlo, como se menciona en la sección siguiente, para obtener mejores resultados haciendo uso de otros clasificadores más elaborados y de aquellos que están disponibles en las capas de por subscripción de *Twitter*.

10. Trabajo futuro

Se plantea como futuras líneas de investigación estudiar las capacidades de las capas *Premium* y *Enterprise* para lograr nuevos y mejores indicadores.

Haciendo uso de las capas no gratuitas de *Twitter* se obtiene un mayor potencial para poder obtener información más precisa y mejorar los resultados de la búsqueda. Además se podrían tener resultados de *tweets* más antiguos no solo los de los últimos siete días y eso permitiría obtener los usuarios relevantes de eventos no tan recientes de los que se halla *tweeteado* hace más tiempo.

Una opción interesante de las cuentas *premium* es poder filtrar los *tweets* por biografía de los usuarios. Por ejemplo, podría pedirse solamente se retornara información correspondiente a usuarios tengan la palabra *ingeniería de software* en su biografía (*Twitter API Reference - Premium Operators*, s.f.):

```
bio:software engineer
```

Otra extensión posible a lo realizado podría ser agregar la noción de ubicación en la cual se permita seleccionar un país, o un grupo de países, y se retornen solamente resultados de gente que vive en los lugares seleccionados.

Una característica ya mencionada que afecto la calidad de los resultados es la de poder filtrar usuarios verificados desde la *API*.

En lo implementado se utilizaron indicadores por sí solos para clasificar pero se podrían combinar estos por ejemplo ponderando cada indicador y realizando una suma para obtener un puntaje representativo de la relevancia de cada usuario.

Por ejemplo si se la asigna un peso de 0.6 a la cantidad de *retweets* un peso de 1 a la cantidad de seguidores y un peso de 0.5 a la cantidad de *likes* si tenemos:

- Usuario1 ->tiene 2500 seguidores, tiene un *tweet* relacionado con el tema buscado con 105 *retweets* y 200 *likes*.
- Usuario2 ->tiene 3000 seguidores, tiene un *tweet* relacionado con el tema buscado con 90 *retweets* y 350 *likes*.
- Usuario3 ->tiene 1000 seguidores, tiene un *tweet* relacionado con el tema buscado con 1000 *retweets* y 1800 *likes*.

Utilizando los valores anteriores se tendría que el Usuario1 tiene un puntaje de $2500 * 1 + 105 * 0{,}6 + 200 * 0{,}5 = 2663$, el Usuario2 $3000 * 1 + 90 * 0{,}6 + 350 * 0{,}5 = 3229$ y el Usuario3 $1000 * 1 + 1000 * 0{,}6 + 1800 * 0{,}5 = 2680$.

Con estos valores y dado que a mayor puntaje implica mayor relevancia el ranking de usuarios quedaría Usuario2 primero, luego el Usuario3 y por ultimo el Usuario1.

Cabe destacar que si se usan estos indicadores de forma independiente para clasificar estos usuarios, el resultado es diferente. Posiblemente ajustando con cierto criterio estos valores se podría refinar los resultados y la precisión de los mismos.

11. Referencias

Referencias

Search aggregations. (s.f.). Descargado 2019-11-13, de https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-filter-aggregation.htmls

Twitter api reference - get search tweets. (s.f.). Descargado 2019-11-13, de https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

Twitter api reference - premium operators. (s.f.). Descargado 2019-11-13, de https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators

Twitter api reference - pricing. (s.f.). Descargado 2019-11-13, de https://developer.twitter.com/en/pricing.html

Twitter premium operators. (s.f.). Descargado 2019-11-13, de https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators

Twitter statistics. (s.f.). Descargado 2019-11-13, de https://www.omnicoreagency.com/twitter-statistics/