Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis de datos en redes sociales: Caso de estudio aplicado en Twitter

Autor: David Márquez Mínguez

Tutor: Juan José Cuadrado Gallego

2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis de datos en redes sociales: Caso de estudio aplicado en Twitter

Autor: David Márquez Mínguez

Tutor: Juan José Cuadrado Gallego

Tribunal:

Presidente: Name of the tribunal president

Vocal 1º: Name of the first vocal

Vocal 2º: Name of the second vocal

Fecha de depósito: X de X de 2021

A nuestros alumnos pasados, presentes y futuros...

"Empieza haciendo lo necesario, luego haz lo posible y de pronto empezarás a hacer lo imposible."
Francisco de Asís

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Aqui va la parte de agradecimientos.....

Resumen

Resumen..... correo de contacto: David Márquez Mínguez <david.marquez@edu.uah.es>.

Palabras clave: Trabajo fin de /grado, LaTeX, soporte de español e inglés, hasta cinco....

Abstract

Abstract..... contact email: David Márquez Mínguez <david.marquez@edu.uah.es>.

Keywords: Bachelor final project , LaTeX, English/Spanish support, maximum of five....

Resumen extendido

Con un máximo de cuatro o cinco páginas. Se supone que sólo está definido como obligatorio para los TFGs y PFCs de UAH.

Índice general

Índice de figuras

Índice de tablas

Índice de listados de código fuente

Índice de algoritmos

Capítulo 1

Ejemplo práctico: Text Mining con R

En este capítulo se va a desarrollar un caso práctico sobre minería de textos, con el fin de corroborar su utilidad y su importancia en el mundo actual. Para dicho análisis necesitaremos una fuente de información de donde extraer los textos a analizar, dicha fuente bien podría ser un artículo de opinión, discursos transcritos... Durante este capítulo, se procederá a realizar el proceso de minería de textos sobre publicaciones en redes sociales, en este caso Twitter.

Twitter es actualmente una dinámica e ingente fuente de contenidos que, dada su popularidad e impacto, se ha convertido en la principal fuente de información para estudios de Social Media Analytics. Multitud de empresas emplean este tipo de técnica para obtener información muy valiosa de individuos o de corporaciones. Análisis de reputación de empresas, productos o personalidades, estudios de impacto relacionados con marketing, extracción de opiniones y predicción de tendencias son sólo algunos ejemplos de aplicaciones.

Además de R existen otros lenguajes de programación como Python, MatLab u Octave. Si bien Python es el lenguaje que domina en este ámbito, este análisis se realizará a través de la programación en R, pues contiene tanto librerías, como paquetes que facilitan y extienden sus capacidades como herramienta de análisis de texto.

1.1 Introducción

Tal y como ocurre en muchas redes sociales, Twitter otorga la posibilidad de compartir sus datos tanto con empresas como con desarrolladores y/o usuarios particulares. Aunque en la mayoría de casos se trata de web Services API, con frecuencia existen librerías que permiten interactuar con la API desde diversos lenguajes de programación.

La forma en la que Twitter permite acceder a su contenido es a través de lo que se conoce como Twitter App, al crear dicha Twitter App, se adquieren una serie de claves y tokens de identificación que permiten acceder a la aplicación y consultar la información necesaria.

Algo que se deberá tener en cuenta durante este análisis, es que Twitter tiene una normativa que regula la frecuencia máxima de peticiones, así como la cantidad máxima de tweets que se pueden extraer. Durante la fase de extracción de la información, se deberá tener en cuenta dichos limites con el objetivo de evitar ser sancionado por la organización.

1.2 Acceso a la API

Para acceder a la API de Twitter, como se indica en la documentación de la misma, existen dos métodos de acceso Oauth2 y Oauth1a. El acceso con cada uno de ellos dependerá del tipo de información que se desee extraer [1]. Como veremos a continuación, nosotros nos centraremos en Oauth1a.

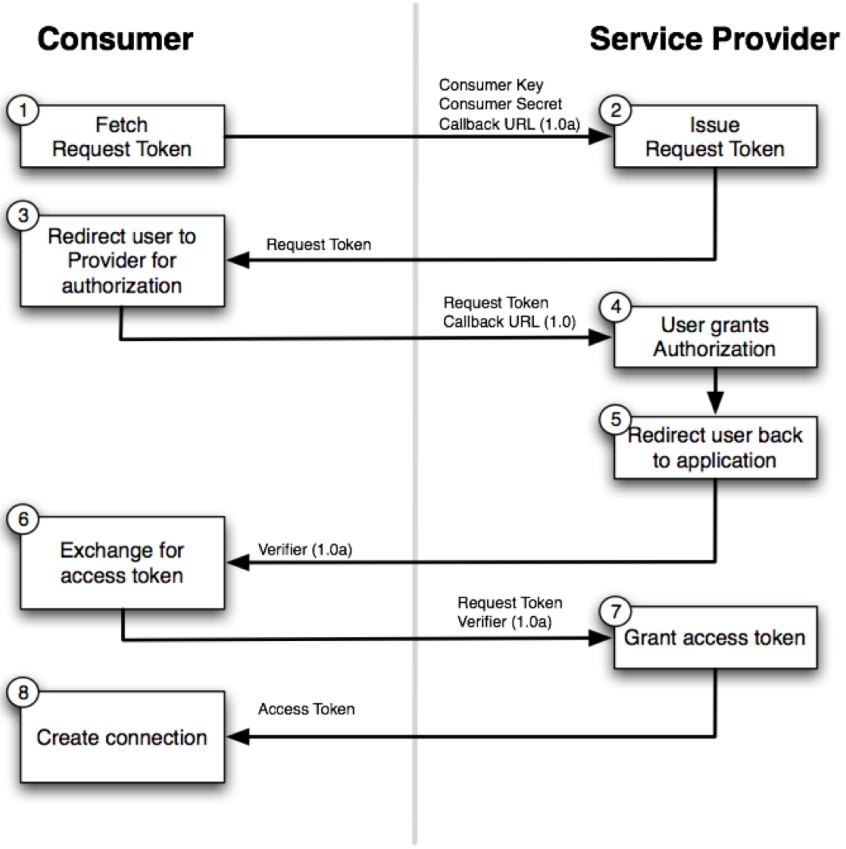

Figura 1.1: Funcionamiento de Oauth1a

Puesto que se desea extraer información específica de algunos usuarios de la plataforma, el metodo de acceso será a través de Oauth1a. Así pues, OAuth1a para funcionar requiere de cuatro elementos que provienen de la aplicación de Twitter creada por el usuario desarrollador, estos elementos son:

1. Clave del consumidor
2. Clave del consumidor secreta
3. Token de acceso
4. Token de acceso secreto

Todos estos recursos deberán ser proporcionados por la plataforma de Twitter para desarrolladores. Una vez se disponen de dichos recursos, realizamos el proceso de identificación y obtención de tokens [2], no sin antes cargar las librerías necesarias para el desarrollo de dicho caso práctico.

```
>   #En primer lugar se cargan las librerias necesarias.
>   library(rtweet)
>   library(tidyverse)
>   library(knitr)
```

Ahora que las librerías necesarias están cargadas, se procede a crear las variables que almacenarán los recursos de acceso. A conitnuación, solo debemos llamar a la función de acceso. Esta funcion, se encargará de enviar una solicitud para generar tokens OAuth 1.0, dichos tokens solo se podrán utilizar para solicitar información [3].

```
>   #Ahora se definen ciertas variables que almacenaran la informacion de acceso
>   nombre_app <- "MarquezDavidTFG"
>   clave <- "bsS2cbbZe7BDsxFPLYRM0GKJ8"
>   clave_secreta <- "WvUilwEZbgJhHiUgpjyboQcQHYSSCNKwImHF1TeINe9aWskkDo"
>   token_acceso <- "3147132436-kFG9XkuWsdI8n1KPVZQTOrf6rw45lrqPEPoUXPr"
>   token_acceso_secreto <- "zK8dyBAgB4sTrRhjVBXmNCAjX4QQvCUfQIXckQmTrcAix"

>   #Creamos nuestro twitter token
>   twitter_token <- create_token(app=nombre_app, consumer_key=clave,
>                                 consumer_secret=clave_secreta,
>                                 access_token=token_acceso,
>                                 access_secret=token_acceso_secreto)
```

1.3 Extracción y carga de datos

Una vez que se ha conseguido acceder a la API a través de la aplicación para desarrolladores proporcionada por Twitter, se procede con la fase de extracción de datos. Como se ha indicado en la sección 1.1, en Twitter existe una normativa que regula la frecuencia máxima de peticiones, así como la cantidad máxima de tweets que se pueden extraer [4]. Para el tipo de análisis que se quiere realizar en este trabajo, se pretenden obtener la mayor cantidad de tweets posible, puesto que el número máximo de publicaciones que se pueden extraer por consulta son 200, se sigue la siguiente estrategia:

1. Toda publicación tiene un identificador numérico global;, que sigue un orden temporal. Esto permitirá identificar y distinguir los tweets mas recientes de los más antiguos.
2. Es posible recuperar y trabajar solo con publicaciones antiguas, empelando como parámetro de la función el identificador de cada tweet.
3. Antes de cada consulta, se lee el fichero donde se almacenan las publicaciones y se identifica el ID del ultimo tweets recuperado. Si no existe dicho fichero de almacenamiento para el usuario en cuestión, se crea uno.
4. Se realiza una nueva consulta empleando como argumento maxId el ID recuperado en el paso anterior.
5. Se incorporan los nuevos datos al archivo de almacenamiento.

Capítulo 2

Presupuesto

Blah, blah, blah.

Bibliografía

[1] D. Spring, "Service provider 'connect' framework," https://docs.spring.io/spring-social/docs/1.0.0.RC1/reference/html/serviceprovider.html [Ultimo acceso 16/marzo/2021].

[2] S. Huppmann, https://github.com/ropensci/rtweet/issues/251 [Ultimo acceso 17/marzo/2021].

[3] M. W. Kearney, "Creación de token (s) de autorización de twitter." https://www.rdocumentation.org/packages/rtweet/versions/0.7.0/topics/create_token [Ultimo acceso 17/marzo/2021].

[4] Twitter, "Rate limits," https://developer.twitter.com/en/docs/rate-limits [Ultimo acceso 19/marzo/2021].

Apéndice A

Manual de usuario

A.1 Introducción

Blah, blah, blah...

A.2 Manual

Pues eso.

A.3 Ejemplos de inclusión de fragmentos de código fuente

Para la inclusión de código fuente se utiliza el paquete `listings`, para el que se han definido algunos estilos de ejemplo que pueden verse en el fichero `config/preamble.tex` y que se usan a continuación.

Así se inserta código fuente, usando el estilo `CppExample` que hemos definido en el preamble, escribiendo el código directamente :

```
#include <stdio.h>

// Esto es una funcion de prueba
void funcionPrueba(int argumento)
{
  int prueba = 1;

  printf("Esto es una prueba [%d][%d]\n", argumento, prueba);

}
```

O bien insertando directamente código de un fichero externo, como en el ejemplo A.1, usando `\lstinputlisting` y cambiando el estilo a `Cbluebox` (ademÃ¡s de usar el entorno `codefloat` para evitar pagebreaks, etc.).

Listado A.1: Ejemplo de código fuente con un `lstinputlisting` dentro de un `codefloat`

```
#include <stdio.h>

// Esto es una función de prueba
void funcionPrueba(int argumento)
{
        int prueba = 1;

  printf("Esto es una prueba [%d][%d]\n", argumento, prueba);

}
```

O por ejemplo en matlab, definiendo settings en lugar de usar estilos definidos:

```
%
% add_simple.m - Simple matlab script to run with condor
%
a = 9;
b = 10;

c = a+b;

fprintf(1, 'La suma de %d y %d es igual a %d\n', a, b, c);
```

O incluso como en el listado A.2, usando un layout mÃ¡s refinado (con los settings de http://www.rafalinux.com/?p=599 en un `lststyle Cnice`).

Listado A.2: Ejemplo de código fuente con estilo `Cnice`, de nuevo con un `lstinputlisting` dentro de un `codefloat`

```
1  #include <stdio.h>
2
3  #define LOOP_TIMES 5
4
5  int main(int argc, char* argv[])
6  {
7    int i;
8
9    for (i = 1; i < LOOP_TIMES; i++)
10     puts("Hola mundo!");
11 }
```

Y podemos reutilizar estilos cambiando algÃºn parÃ¡metro, como podemos ver en el listado A.3, en el que hemos vuelto a usar el estilo `Cnice` eliminando la numeración.

Listado A.3: Ejemplo de código fuente con estilo `Cnice`, modificado para que no aparezca la numeración.

```
#include <stdio.h>

#define LOOP_TIMES 5

int main(int argc, char* argv[])
{
  int i;

  for (i = 1; i < LOOP_TIMES; i++)
    puts("Hola mundo!");
}
```

Ahora compila usando `gcc`:

```
$ gcc  -o hello hello.c
```

Y tambiÃ©n podemos poner ejemplos de código *coloreado*, como se muestra en el A.4.

Listado A.4: Ejemplo con colores usando el estilo `Ccolor`

```
#include <stdio.h>

#define LOOP_TIMES 5

int main(int argc, char* argv[])
{
  int i;

  for (i = 1; i < LOOP_TIMES; i++)
    puts("Hola mundo!");
}
```

Finalmente aquÃ tenÃ©is un ejemplo de código shell, usando el estilo `BashInputStyle`:

```
#!/bin/sh

HOSTS_ALL="gc000 gc001 gc002 gc003 gc004 gc005 gc006 gc007"

for h in $HOSTS_ALL
do
  echo "Running [$*] in $h..."
  echo -n "  "
  ssh root@$h $*
done
```

A.4 Ejemplos de inclusión de algoritmos

En la versión actual (abril de 2014), empezamos a usar el paquete `algorithm2e` para incluir algoritmos, y hay ajustes especÃficos y dependientes de este paquete tanto en `config/preamble.tex` como en `cover/extralistings.tex` (editadlos segÃºn vuestras necesidades).

Hay otras opciones disponibles (por ejemplo las descritas en `http://en.wikibooks.org/wiki/LaTeX/Algorithm`), y podemos abordarlas, pero por el momento nos quedamos con `algorithm2e`.

Incluimos dos ejemplos directamente del manual: uno sencillo en el algoritmo A.1, y otro un poco mÃ¡s complicado en el algoritmo A.2.

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
    read current;
    if *understand* **then**
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;

Algoritmo A.1: How to write algorithms

Data: $G = (X, U)$ such that G^{tc} is an order.
Result: $G' = (X, V)$ with $V \subseteq U$ such that G'^{tc} is an interval order.
begin
    $V \longleftarrow U$
    $S \longleftarrow \emptyset$
    for $x \in X$ **do**
        $NbSuccInS(x) \longleftarrow 0$
        $NbPredInMin(x) \longleftarrow 0$
        $NbPredNotInMin(x) \longleftarrow |ImPred(x)|$
    for $x \in X$ **do**
        if $NbPredInMin(x) = 0$ **and** $NbPredNotInMin(x) = 0$ **then**
            $AppendToMin(x)$
1   **while** $S \neq \emptyset$ **do**
REM   remove x from the list of T of maximal index
2   **while** $|S \cap ImSucc(x)| \neq |S|$ **do**
            for $y \in S - ImSucc(x)$ **do**
                { remove from V all the arcs zy : }
                for $z \in ImPred(y) \cap Min$ **do**
                    remove the arc zy from V
                    $NbSuccInS(z) \longleftarrow NbSuccInS(z) - 1$
                    move z in T to the list preceding its present list
                    {i.e. If $z \in T[k]$, move z from $T[k]$ to $T[k-1]$}
                $NbPredInMin(y) \longleftarrow 0$
                $NbPredNotInMin(y) \longleftarrow 0$
                $S \longleftarrow S - \{y\}$
                $AppendToMin(y)$
        $RemoveFromMin(x)$

Algoritmo A.2: IntervalRestriction

Apéndice B

Herramientas y recursos

Las herramientas necesarias para la elaboración del proyecto han sido:

- PC compatible
- Sistema operativo GNU/Linux [**?**]
- Entorno de desarrollo Emacs [**?**]
- Entorno de desarrollo KDevelop [**?**]
- Procesador de textos LaTeX[**?**]
- Lenguaje de procesamiento matemático Octave [**?**]
- Control de versiones CVS [**?**]
- Compilador C/C++ gcc [**?**]
- Gestor de compilaciones make [**?**]

Apéndice C

Versiones

En este apartado incluyo el historial de cambios más relevantes de la plantilla a lo largo del tiempo.

No empecé este apéndice hasta principios de 2015, con lo que se ha perdido parte de la información de los cambios importantes que ha ido sufriendo esta plantilla.

- Mayo 2015:
  - Hay disponible un `make bare` para que deje los capítulos mondos y lirondos y se pueda escribir desde casi cero sin tener que andar borrando manualmente.
- Abril 2015:
  - Ahora manejamos masculino/femenino en algunos sitios (el/la, autor/autora, alumno/alumna, del/de la, ...). Hay que definir variable con el género del autor (todavía queda pendiente lo de los tutores y tal). NOT FINISHED!!
- Enero 2015:
  - Solucionado el problema (gordo) de compilación del `anteproyecto.tex` y el `book.tex`, debido al uso de paths distintos en la compilación de la bibliografía. El sistema se ha complicado un poco (ver `biblio\bibliography.tex`).
  - Añadido un (rudimentario) sistema para generar pdf con las diferencias entre el documento en su estado actual y lo último disponible en el repositorio (usando `latexdiff`).
- Diciembre 2015:
  - Separada la compilación del anteproyecto de la del documento principal. Para el primero se ha creado el directorio `anteproyecto` donde está todo lo necesario.

Universidad de Alcalá
Escuela Politécnica Superior

Universidad
de Alcalá