Título

Clasificación supervisada para consulta temática en Twitter

Autores

Sixto Jansa Sanz
sjansa@ucm.es
Enrique Ortiz Torralba
enriqueo@ucm.es

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y PROGRAMACIÓN
Universidad Complutense de Madrid

TRABAJO FIN DE GRADO

Junio 2017

Director:
Rafael Caballero Roldán

Agradecimientos

Queremos agradecer a nuestro directo de proyecto, Rafael Caballero Roldán, su gran labor y dedicación durante todo el año en la elaboración de este TFG.

Índice

Resumen

Diariamente, la plataforma de *microblogging* Twitter genera una cantidad de información abrumadora. Es expresada en mensajes de 140 caracteres conocidos como *tweets*. Twitter permite buscar tanto tweets de usuarios concretos como tweets que contengan términos específicos, pero no permite la búsqueda temática, es decir no permite agrupar los mensajes bajo epígrafes generales.

Con este TFG hemos querido plantear una solución que facilite la búsqueda temática en Twitter. Nuestra propuesta es capaz de descargar tweets y agruparlos bajo diferentes categorías dándole una utilidad a los tweets más allá de su propio contenido, pudiendo realizar búsquedas por temáticas, facilitando la experiencia al usuario.

Como caso de uso, nos centramos en el caso de búsqueda temática sobre una ciudad.

Para la clasificación en categorías, es necesario realizar primero una categorización de forma manual para que posteriormente mediante técnicas de aprendizaje automático se realice una clasificación de forma automática.

Para la clasificación automática, hemos utilizado la técnica de aprendizaje supervisado basada en el Teorema de Bayes, que deduce mediante entrenamiento y validación a qué categoría puede pertenecer un tweet.

El usuario finalmente podrá filtrar toda esta información en función de las categorías creadas, fechas y palabras clave que contenga el tweet.

Palabras clave:

Twitter, Teorema de Bayes, clasificación automática, aprendizaje supervisado, análisis de sentimiento.

Abstract

Daily, the Twitter microblogging platform generates an overwhelming amount of information. It is expressed in messages of 140 characters known as tweets. Twitter allows you to search for specific tweets of users as well as tweets that contain specific terms, but does not allow thematic search, as it does not allow grouping of messages under general headings.

With this project we wanted to propose a solution that facilitates thematic search on Twitter.

Our proposal is to enable tweets to be downloaded and to group them under different categories giving a utility to the tweets beyond their own content, being able to perform searches by themes, this facilitating the user experience.

As a case study, we focus on the thematic search about a city.

For the classification, it is necessary to previously carry out a manual categorization, so that with machine learning techniques the classification is performed automatically.

For automatic classification, we used the supervised learning technique based on the Bayes Theorem, which deducts through training and validation to which category a tweet may belong.

The user will finally be able to filter all this information according to the created categories, dates and keywords that the tweet contains.

Keywords:

Machine learning, Twitter, Bayes theorem, automatic classification, supervised learning, sentiment analysis.

1. Introducción

Twitter es una plataforma de *microblogging* creada en 2006 por Jack Dorsey, quien la define de la siguiente forma:

> "Twitter es una forma de comunicarte con la gente muy accesible y portátil, cualquiera con un teléfono puede participar en mi conversación […] No considero que Twitter sea una red social, sino una herramienta de comunicación." (Dorsey, 2009)

Problema

Siendo Twitter una herramienta de comunicación accesible a todo el mundo, la cantidad de información que se genera diariamente es abrumadora. Se estima que actualmente tiene alrededor de 328 millones de usuarios activos, que generan 65 millones de tweets y 800.000 peticiones de búsqueda diarias. (Smith, 2016) En los tweets se representan opiniones, experiencias, datos, tendencias…, significativos tanto en tiempo real como de forma acumulada.

Toda esta información se encuentra sin clasificar ni ordenar de ninguna forma, por lo que más allá del contenido de cada tweet no puede aprovecharse ni generar utilidad alguna, como por ejemplo agrupados por temáticas, provocando que cuando el usuario accede a la plataforma se le muestre una gran cantidad de información sin ninguna relación entre sí.

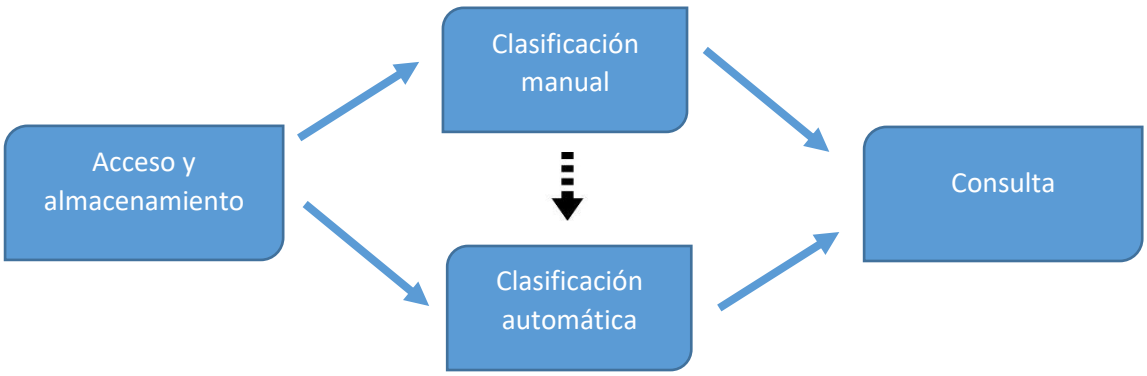
Nuestra propuesta

El objetivo de nuestro trabajo de fin de grado es crear una aplicación que permita clasificar la abundante cantidad de tweets que se generan diariamente en temáticas en relación a una ciudad, ya que esta funcionalidad no la permite Twitter. Solamente desde la versión de escritorio Twitter permite aplicar una serie de filtros a la búsqueda, que son de utilidad si se quieren encontrar tweets conteniendo unas palabras concretas. Nuestra aplicación va más allá, pudiendo categorizar tweets que no

contengan palabras concretas, pero que puedan agruparse temáticamente bajo epígrafes como teatro, deporte…

Un buen ejemplo de esto sería un tweet que contenga "Opera en el Real". Mediante los filtros propios de Twitter este tweet seguramente no lo encontraríamos buscando temas de teatro, pero nuestra aplicación si podría categorizarlo como tal gracias a las técnicas de aprendizaje automático.

Nuestro proyecto consta de 4 módulos relacionados entre sí que se detallarán a lo largo de esta memoria:



- Acceso y almacenamiento: esta aplicación es la encargada de conectarse a Twitter y descargar los tweets que serán utilizados en fases posteriores.
- Clasificación manual: parte de los tweets descargados en el punto anterior y permite al usuario clasificarlos de forma manual. Estos tweets clasificados sirven de base para la clasificación automática.
- Clasificación automática: clasifica automáticamente los tweets descargados en el primer punto en función de los clasificados manualmente. Esta clasificación automática solo se podrá llevar a cabo si previamente se ha clasificado manualmente la muestra que le sirva como entrenamiento.
- Consulta: esta aplicación es la encargada de mostrar los resultados de las anteriores clasificaciones en función de unos filtros aplicados.

Trabajos relacionados

Hemos realizado un estudio de mercado de aplicaciones que se asemejen a nuestro proyecto, y hemos encontrado algunos estudios o aplicaciones que guardan alguna similitud:

Un estudio realizado por (Sriram, Fuhry, Demir, Ferhatosmanoglu, & Demirbas, 2010) que habla de un servicio que categoriza los tweets en función de características extraídas del perfil del usuario que los publica y el contenido de sus tweets. Las características se evalúan y se clasifican en un conjunto de categorías predefinidas (noticias, eventos, opiniones…). Con esto, lo que buscan conseguir es mostrar los tweets agrupados por categorías y que el usuario no se sienta abrumado con tantos tweets sin ninguna relación aparente entre ellos. Nosotros no nos basamos en el usuario, sino en la objetividad para clasificar del administrador de la aplicación.

Otro estudio muy interesante realizado con base en Twitter que busca la clasificación, pero en este caso se trata de categorizar a los usuarios. Esto lo consiguen estudiando como tuitean, que publicaciones les gustan, su lenguaje… (Pennacchiotti & Popescu, 2011). Examinando estos comportamientos pueden deducirse aspectos personales como: orientaciones políticas, etnia o religión.

La página web que más se asemeja es la búsqueda avanzada del propio Twitter (Ilustración 1). Esta búsqueda sólo está disponible en la versión web y no en la aplicación móvil.

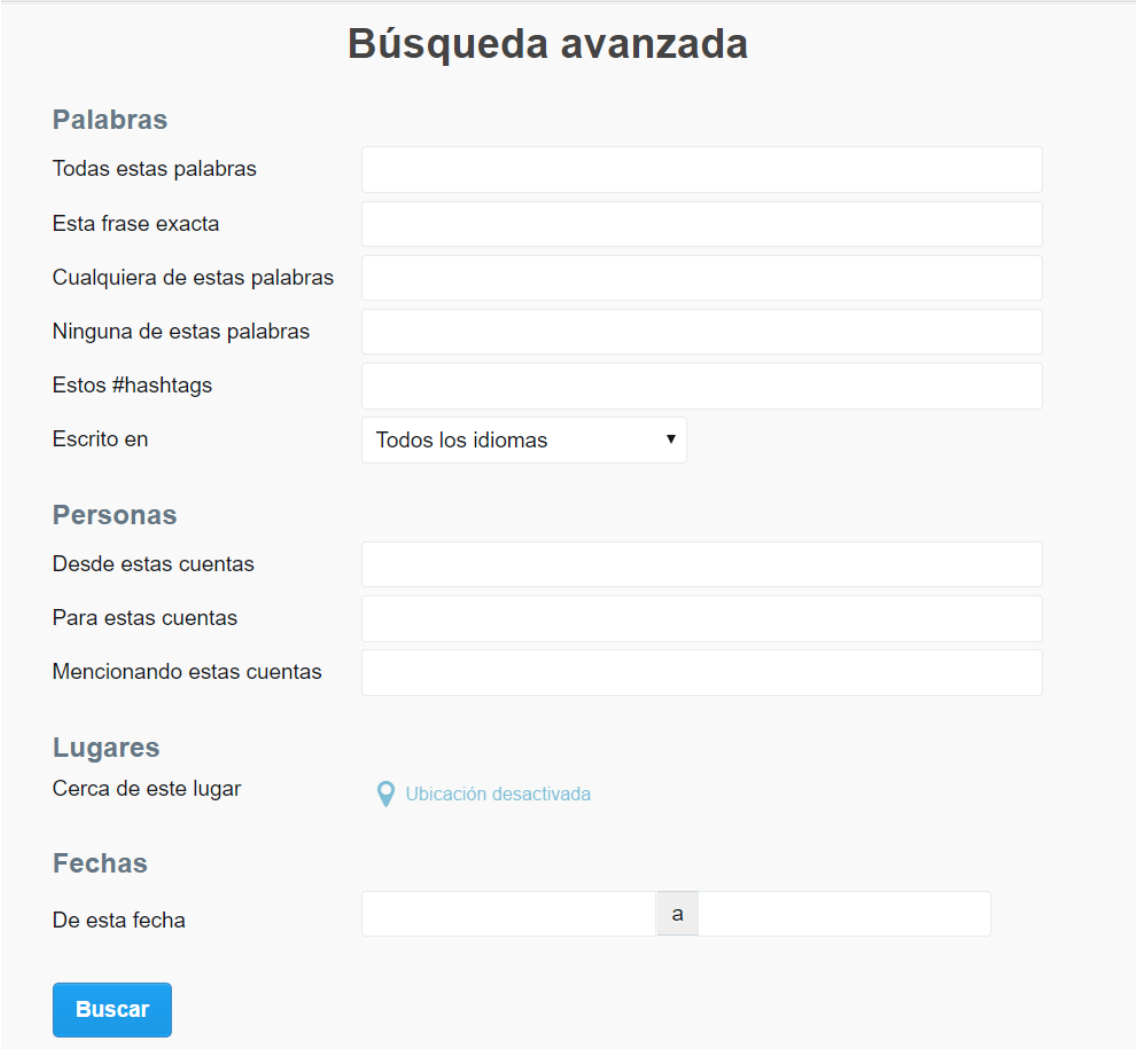


Ilustración 1

Esta búsqueda permite aplicar un gran número de filtros, más de los aplicables en nuestro proyecto. La gran diferencia entre las dos, es el punto fuerte de nuestra aplicación, la búsqueda por temáticas, que en Twitter no se puede aplicar. En la red social sólo se mostrarán tweets que contengan la palabra o palabras que se hayan indicado en la búsqueda, o que dicha búsqueda tenga alguna similitud con el nombre de alguna cuenta.

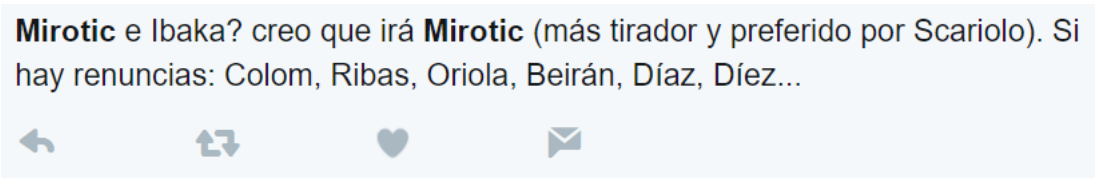


Ilustración 2

Si buscásemos por ejemplo el término “baloncesto” en Twitter, no se nos mostrará el tweet de la Ilustración 2, habría que buscar concretamente cualquiera de las palabras contenidas en el tweet, mientras que con nuestra aplicación, gracias a la clasificación automática, si tuviésemos una categoría llamada baloncesto, este tweet estaría contenida en ella y se mostraría.

2. Acceso y almacenamiento

En este capítulo, se hablará de cómo acceder a Twitter mediante su API para la descarga de contenido y su almacenamiento. El contenido descargado son un gran número de tweets que nos facilitan información sobre lo que se está hablando. Esta información descargada nos servirá como base para ver qué temas son más relevantes y poder crear categorías y clasificar el contenido almacenado.

Twitter

Desde la creación de Twitter, su popularidad no ha hecho más que aumentar. Este servicio funciona en forma de red social, permitiendo enviar mensajes de texto plano de un máximo de 140 caracteres llamados tweets. Además de texto se pueden incluir imágenes, videos, urls… También existen los retweets, que consiste en publicar un tweet de otra persona en tú perfil, pero manteniendo su autoría. El número de retweets que obtenga una publicación, son un indicativo de la repercusión que ha tenido un tweet. Los tweets son la pieza principal de la red social y nuestra base para esta aplicación.

Los tweets son generados por usuarios dados de alta en la aplicación. Estos usuarios pueden seguirse unos a otros e interactuar. El número de cuentas activas en esta red social asciende a 328 millones en el primer trimestre de 2017. (Statista, 2017)

La popularidad de Twitter es tal que entre los usuarios se encuentran políticos, personalidades influentes e incluso medios de comunicación e información como pueden ser Antena3 o El Mundo. Algunas cuentas de Twitter están verificadas, esto quiere decir que son aquellas que Twitter ha comprobado que son auténticas y legalmente en poder de la persona o entidad correcta. "*Una cuenta se puede verificar si se determina que es de interés público. En general, son las cuentas de usuarios del ámbito de la música, la actuación, la moda, el gobierno, la política, la religión, el periodismo, los medios de comunicación, el deporte, los negocios y otras áreas de interés.*" (TwitterSupport, 2017)

APIs

"Una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Las siglas API vienen del inglés *Application Programming Interface*. Una API nos permite implementar las funciones y procedimientos que engloba en nuestro proyecto sin la necesidad de programarlas de nuevo. En términos de programación, es una capa de abstracción." (R. Andrea, 2014).

Para acceder a toda la información que posee Twitter y poder tratarla, esta red social facilita dos APIs:

- Streaming APIs: Esta API permite la descarga y monitorización de tweets en tiempo real.
- REST APIs: Esta API por el contrario permite la descarga de tweets que hayan ocurrido atrás en el tiempo, además permite al usuario crear tweets, leer información del usuario, sus seguidores, etc.

En nuestro proyecto, hemos utilizado REST APIs, porque es más completa y nos permite trabajar de forma más sencilla, descargando mucha información de una sola ejecución, en lugar de tener que monitorizar Twitter constantemente.

Si queremos hacer uso de estas APIs, el primer paso para que Twitter te permita utilizarlas, es tener una cuenta en dicha red social, y a su vez con dicha cuenta, crear una nueva aplicación en su página de gestión de aplicaciones, donde se nos facilitan las siguientes contraseñas que se utilizarán para que nuestra aplicación se conecte con Twitter.

- Consumer Key (API Key)
- Consumer Secret (API Secret)
- Access Token
- Access Token Secret

Las dos primeras claves nos permiten autenticar las peticiones que realicemos a Twitter, y las dos últimas permiten realizar solicitudes a la API en nombre de la cuenta. En nuestro caso sólo vamos a necesitar las dos primeras claves.

Autenticación y limitaciones.

Para que Twitter permita acceder a su contenido, hay que hacer uso de las claves mencionadas anteriormente y utilizarlas junto a un protocolo de autenticación.
El protocolo de autenticación que utiliza Twitter se llama OAuth. Es un protocolo de carácter general, que permite autenticar una aplicación para actuar en tú nombre de forma segura. (Cook & Messina, 2006).
En el protocolo OAuth existen dos funciones que permiten nuestra autenticación en función de la API que se haya escogido:

- OAuth signed
- Application-only authentication

La función OAuth signed se puede utilizar con ambas APIs, mientras que Application-only authentication sólo puede usarse con la API que hemos escogido. Ambas funciones tienen comportamientos similares, pero Application-only authentication limita a la API REST frente a la Streaming reduciendo las acciones que se pueden hacer con ella, provocando que sólo se pueda utilizar la API para leer información. Pese a esta limitación, y a que en las primeras versiones de este proyecto utilizamos el autenticador OAuth signed, decidimos decantarnos por el autenticador Application-only authentication debido a las limitaciones que presenta Twitter.
Si se hace uso de OAuth signed se aplica una restricción de 180 peticiones cada 15 minutos, o 18000 tweets cada 15 minutos, ya que por petición podemos descargar un máximo de 100 tweets. Esta restricción podemos ampliarla (ver blog (Karambelkar´s, 2015)) a 450 peticiones cada 15 minutos, o 45000 tweets si utilizamos Application-only authentication, lo que supone una mejora del 250% frente al anterior, razón principal por la que finalmente nos decantamos por esta función de autenticación.
A su vez esta versión de la API nos aplica un retroceso de dos horas en el momento de realizar la búsqueda de tweets, y recopilará información hasta 7 días anteriores al momento en el que se realiza la petición.

Búsqueda y almacenamiento.

Con el registro, las claves, y conociendo las limitaciones de las API's, tenemos que escribir un programa que permita recolectar la información de Twitter que queramos. Para ello hemos escrito un programa en el lenguaje Python, versión 2.7.13, y hecho uso de la biblioteca Tweepy (Roesslein, 2009), que nos facilita las funciones para poder acceder al contenido de Twitter.

Este programa pedirá al usuario una palabra o conjunto de palabras que se buscarán en Twitter obteniendo tweets relacionados, así como el número de tweets que se desean obtener. Recogidos estos parámetros y mientras haya conexión a internet y las claves de autenticación estén bien, el programa iniciará la búsqueda.

Para realizar la búsqueda de tweets, la documentación de la biblioteca Tweepy nos indica que se utilice una función api.cursor(), pero por experiencia y como también se dice en el blog anteriormente citado, cuando se le requiere una gran carga de trabajo a esta función, consume mucha memoria RAM, llegando a saturar el sistema. Por esta razón es recomendable utilizar la función api.search(), que resuelve el problema de consumo de memoria pero ofreciendo los mismos resultados de búsqueda.

Se invoca de la siguiente manera:

api.search(topicName, tweetsConsulta, maxId, sinceId , lang)

- topicName: Palabra o conjunto de palabras que se van a buscar.
- tweetsConsulta: Número de tweets que se van a descargar por iteración, cómo se ha mencionado antes el máximo es 100.
- maxId: El ID del último tweet leído, para proseguir la búsqueda desde ese ID, evitando duplicados.
- sinceId: Si se ha hecho una búsqueda anterior, esta variable contiene el ID del último tweet escrito, siendo este el tope de la búsqueda evitando duplicados.
- lang: Para establecer el idioma de los tweets a descargar.

Esta función está recogida dentro de un bucle, finalizando cuando el número de tweets descargados sea igual a los pedidos al iniciar el programa, o cuando no pueda encontrar más tweets con nuestro *topicName*. Si se llega al límite de consultas

establecido por Twitter, el programa esperará el tiempo necesario y después proseguirá con la búsqueda donde la dejó. Esto se puede realizar de forma automática si establecemos la variable *wait_on_rate_limit* a True, de esta forma la API, se encarga de gestionar el tiempo necesario de espera. Por el contrario, si no hacemos esto, tendremos que gestionar el tiempo nosotros mismo, por ejemplo, mediante la función *time.sleep()*.

En nuestro caso, en cada iteración del bucle se tratan 100 tweets, de los cuales se filtran y se desechan los retweets, para evitar lo máximo posible el duplicado de información. De la misma forma, en este proyecto sólo seleccionamos los tweets pertenecientes a cuentas verificadas, permitiendo almacenar información lo más fidedigna posible.

Toda esta información ya filtrada se va almacenando en formato JSON, en un archivo llamado *porClasificar.json*. Este archivo será la base que utilizaremos en otros programas para clasificar. En él se almacenan un conjunto de variables que van asociadas a cada tweet, tales como: texto, fecha, ID, Seguidores, Nombre de Usuario…, guardándose en el archivo con las siguientes claves:

{"Usuario verificado": , "Seguidores":, "manual": , "Nombre usuario": , "Texto": , "Creado": , "Zona horaria": , "ID": , "Siguiendo": }

- “manual” es la única variable que no facilita Twitter. La hemos añadido nosotros para que mejore el comportamiento de la aplicación, indicando si el tweet se ha clasificado de forma manual o automática.

Este programa crea una carpeta por cada búsqueda que realicemos, y guarda los tweets en el archivo JSON anteriormente mencionado. Si se ha realizado previamente una búsqueda similar, se añadirá la nueva descarga de tweets al archivo JSON.

En el caso de que exista una búsqueda previa, el programa antes de comenzar la búsqueda y descarga, recorre el archivo JSON para encontrar el tweet con la variable ID más alta (a mayor ID más reciente es el Tweet), almacenándolo en el parámetro *sinceId*. Así evitamos que en todas las posibles búsquedas que se realicen sobre un tema, se solapen los tweets repitiéndose.

En el caso de este proyecto, como se busca filtrar el contenido por ciudades, se han descargado tweets usando como término de búsqueda "Madrid".

Integración con el proyecto.

El programa principal del proyecto está desarrollado en Java y el acceso a Twitter y la descarga de tweets está programado en Python. Para no hacer dos aplicaciones separadas e independientes y poder conectarse con el código desarrollado en Python, hemos utilizado en Java el método Runtime exec de la siguiente forma:

Runtime.getRuntime().exec("python continuo.py "+texto+" "+tweets);

- Si disponemos de una versión antigua de Python, en lugar del comando "python" lo sustituiremos por el comando "py", o por el comando que tengamos asociado en la variable del sistema que ejecute Python.
- *continuo.py*: es el código de acceso, búsqueda y almacenamiento de tweets.
- *texto*: es el nombre de la búsqueda que se va a realizar.
- *tweets*: es el número de tweets que se van a descargar.
- Estas dos últimas variables se recogen en una interfaz codificada en Java (ver Ilustración 3).

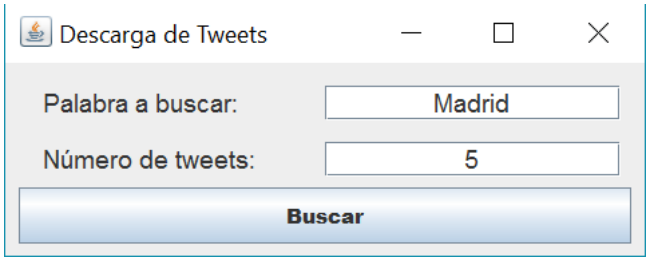


Ilustración 3

Cuando pulsemos en el botón *Buscar* se ejecutará el método *Runtime()* y cuando el programa haya descargado el número de tweets pedido terminará informando de ello.

3. Clasificación Manual

En este capítulo vamos describir la estructura de carpetas generada a partir de la sección de *Acceso y almacenamiento* anteriormente descrita. Partiremos de una carpeta llamada *Madrid* con su correspondiente archivo JSON que contiene los tweets descargados. A partir de dicha categoría inicial, se irán creando nuevas categorías que deriven de la mencionada anteriormente, distribuyendo así los diferentes tweets y creando una estructura de carpetas anidadas.

Estructura de carpetas

Un ejemplo de la estructura de carpetas que podríamos generar sería la siguiente (ver Ilustración 4).

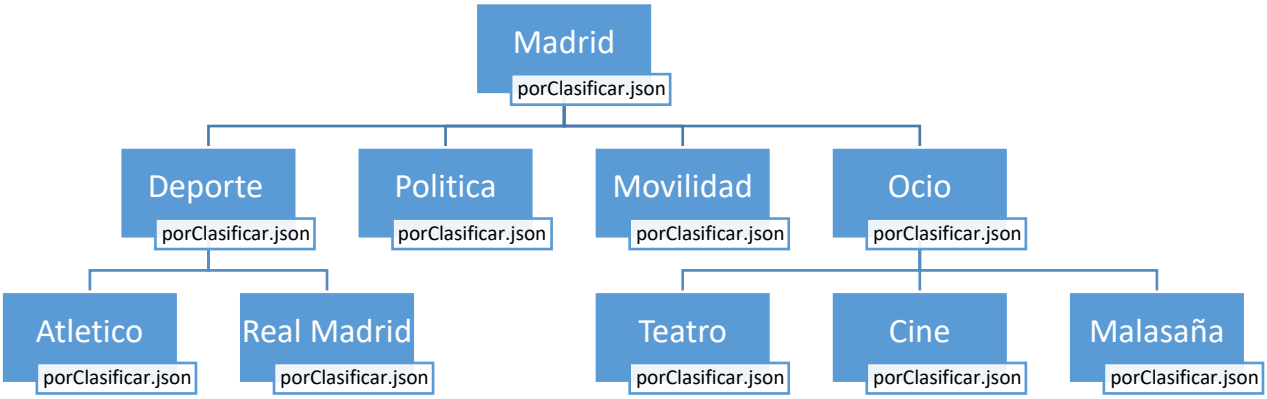


Ilustración 4

Cada carpeta (categoría) contiene un archivo llamado "porClasificar" que contiene los tweets. Estos archivos, inicialmente, estarán vacíos. Una vez que se empiecen a clasificar en dichas categorías se irán rellenando con los diferentes tweets descargados.

Este archivo "porClasificar" de cada categoría contiene los tweets clasificados de cada uno. Si, por ejemplo, se trata de una categoría que contenga subcarpetas (véase el caso de Deporte en la Ilustración 4), la categoría *Deporte* contendrá los tweets clasificados como *Deporte* pero que no se hayan clasificado en sus subcarpetas. Por ejemplo, un tweet que hable del club de futbol Barcelona quedará alojado en la carpeta Deporte. En el caso de la categoría *Real Madrid* contendrá solo los tweets relacionados con dicha categoría. Este ejemplo es aplicable a todas las categorías que se creen.

Traspaso de Tweets

Para clasificar los tweets manualmente, el usuario accederá mediante la estructura de carpetas generada en el apartado anterior, al archivo "porClasificar" correspondiente. Este archivo es de tipo JSON, el cual contiene una serie de atributos tales como el propio texto del tweet, el ID, fecha de creación. Una vez listados los tweets de dicho archivo, el usuario podrá seleccionarlos y clasificarlos en la categoría que crea más oportuna.
La primera vez que el usuario inicie la aplicación y descargue los tweets, partiremos de una única categoría, en nuestro caso *Madrid*. A partir de ahí, el usuario deberá crear las nuevas categorías que crea oportuno para posteriormente clasificar los tweets.

Esta clasificación en subcategorías permite al usuario que, si dispone de una gran cantidad de tweets de una categoría, repartirlos en subcategorías más concretas permitiendo en un futuro hacer búsquedas más precisas. Esta parte de búsquedas se explicará en el apartado *5 Consultas y funcionalidad general* de este documento.
Cada nueva categoría genera una nueva carpeta con su nombre conteniendo un archivo "porClasificar" vacío, donde se irán guardando los tweets clasificados.
Cualquier archivo "porClasificar" de cada categoría se puede listar para su posterior clasificación, permitiendo reclasificar los tweets si se desea. Por ejemplo, los tweets de la categoría *Atlético* se podrán volver a clasificar si se crean nuevas categorías por debajo de ésta, o si simplemente si se quiere traspasar algún tweet a otra categoría.

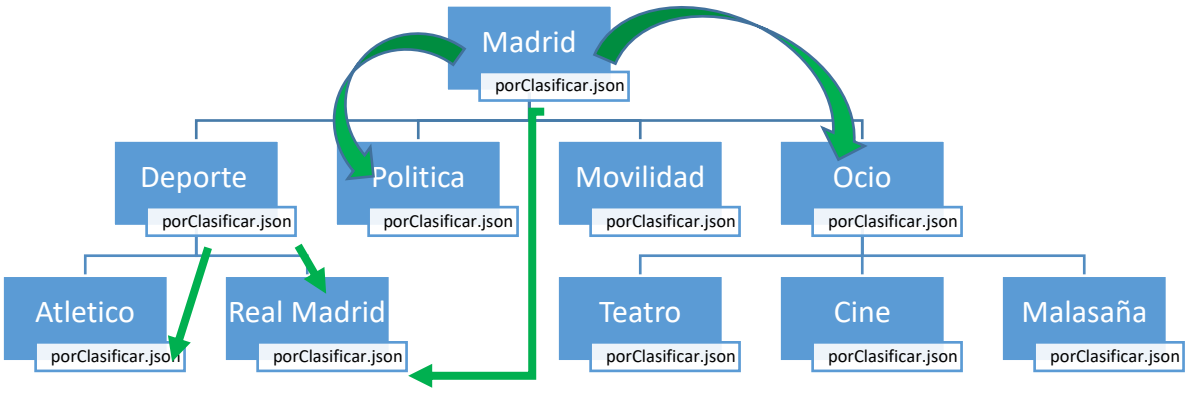


Ilustración 5

Como se muestra en la Ilustración *5*, podríamos traspasar tweets directamente de la categoría Madrid a la categoría Real Madrid, sin necesidad de pasar por la categoría Deportes. Del mismo modo podríamos realizarlo a la inversa, de Real Madrid a Madrid. El traspaso de tweets se puede realizar indistintamente entre todas las categorías y subcategorías.

Implementación

La implementación de esta parte del proyecto se ha elaborado en Java.

Para la elaboración de la interfaz se ha utilizado la librería gráfica de java llamada Swing. Con ella se ha realizado la parte de los paneles e incluido los botones necesarios para la aplicación.

Los componentes fundamentales que se han utilizado han sido los siguientes:

- Para mostrar la estructura de carpetas se ha utilizado el componente visual llamado *JTree*, el cual nos permite visualizar un árbol con las diferentes categorías. Para seleccionar la ruta donde se encuentra la estructura de carpetas creadas debemos pulsar el botón de *Buscar carpeta* que se muestra en la Ilustración 7. Tras pulsar en dicho botón, se nos abrirá un buscador donde tendremos que seleccionar la carpeta principal, en nuestro caso *Madrid*. Una vez realizado esto, el programa creará un archivo path.txt donde guardará la ruta de la estructura de carpetas. Esto nos permite al acceder en posteriores

ocasiones al módulo no tener que volver a seleccionar la ruta puesto que la leerá del *path.txt*.

A continuación, se muestra un ejemplo de una estructura generada en la aplicación.

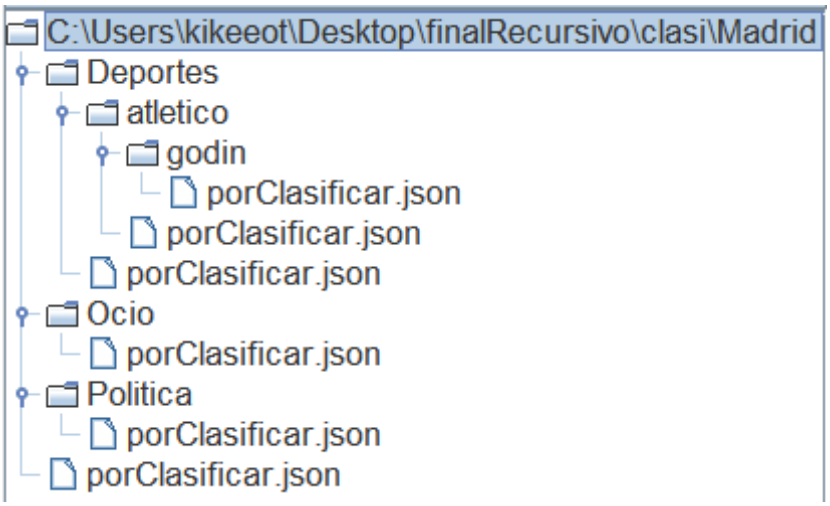


Ilustración 6

Como vemos en la Ilustración 6, se compone de 3 categorías principales (Deportes, Ocio y Política) y en el caso de Deportes una serie de subcategorías (Atlético) y dentro de esta otra más (Godin).

Si el usuario quiere añadir una nueva categoría al árbol, deberá pulsar el botón derecho del ratón sobre la categoría donde desee crearla. Tras ello, aparecerá un *frame* donde el usuario deberá introducir el nombre como se puede ver en la Ilustración 7.

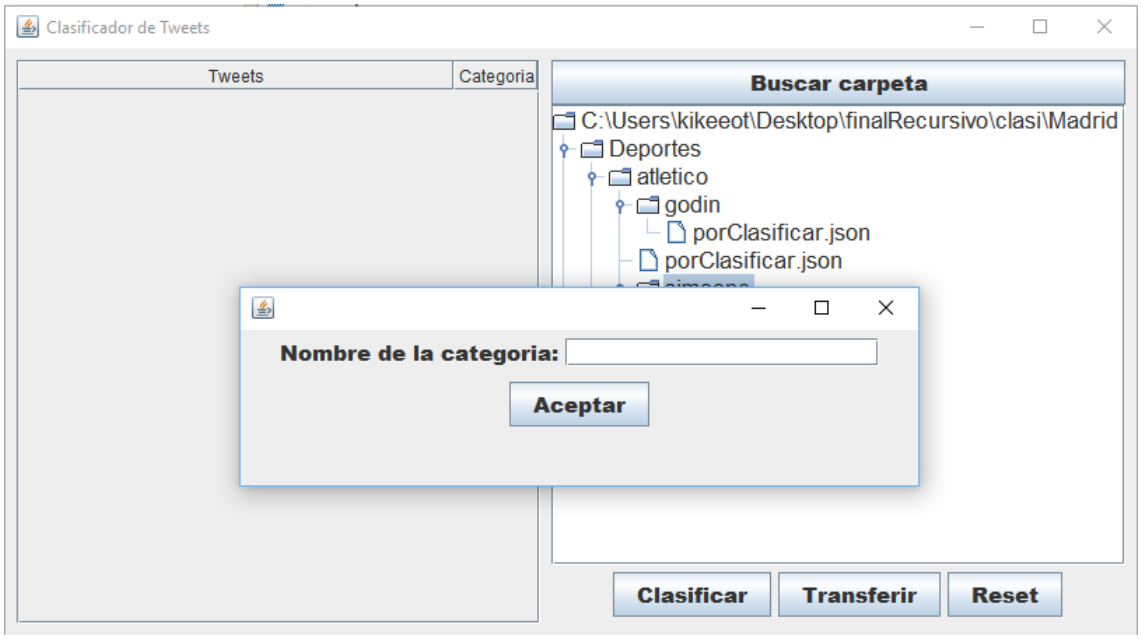


Ilustración 7

Una vez añadido el nombre, el usuario pulsará el botón *Aceptar* y la nueva categoría pasará a formar parte del árbol como se muestra en la Ilustración 8, pudiéndose así clasificar los tweets en ella si el usuario así lo desea.

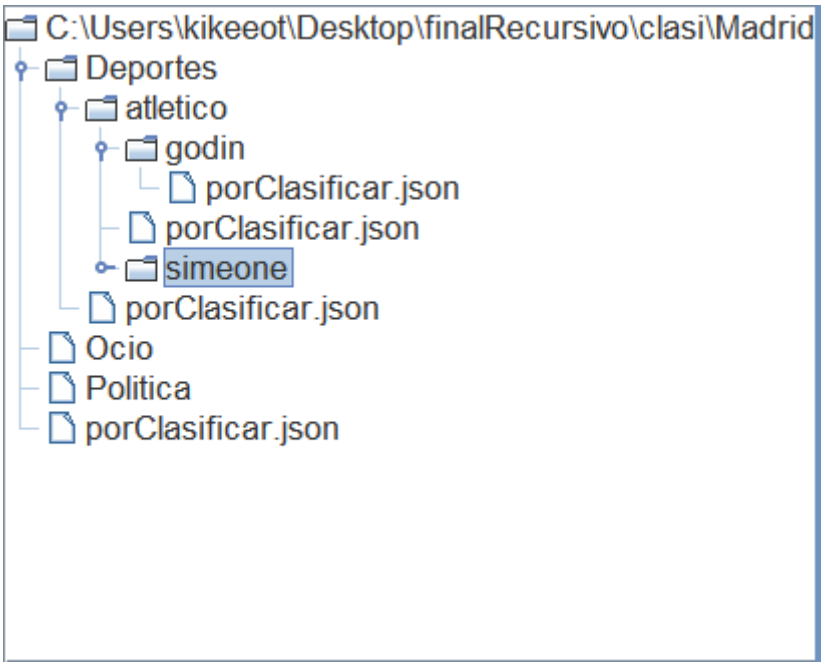
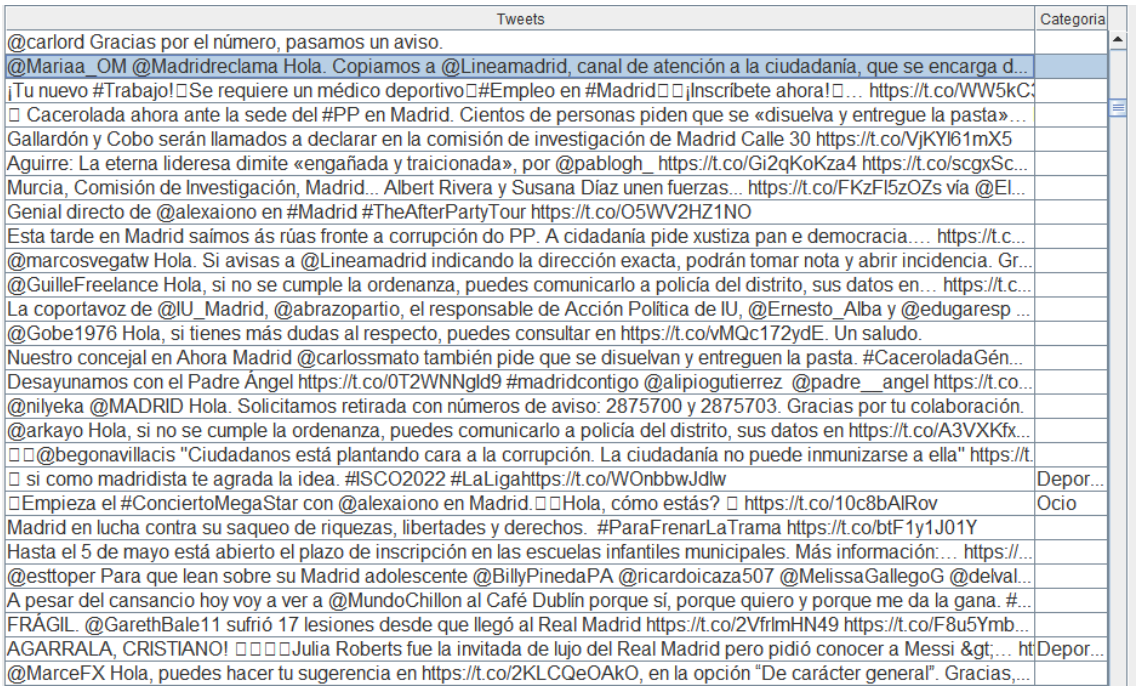


Ilustración 8

- Para la visualización de los tweets se ha utilizado el componente visual llamado *JTable*, el cual permite generar una tabla con celdas en las cuales se irán mostrando los tweets. También habrá una columna en la tabla que contendrá

la categoría a la cual se quiere transferir el tweet. Si no se ha seleccionado categoría, esta celda estará vacía. La Ilustración 9 muestra un ejemplo de cómo se visualiza un archivo JSON correspondiente a la categoría *Madrid*, preparado para la clasificación de los diferentes tweets.



Tweets	Categoria
@carlord Gracias por el número, pasamos un aviso.	
@Mariaa_OM @Madridreclama Hola. Copiamos a @Lineamadrid, canal de atención a la ciudadanía, que se encarga d...	
¡Tu nuevo #Trabajo!□Se requiere un médico deportivo□#Empleo en #Madrid□□¡Inscríbete ahora!□… https://t.co/WW5kC	
□ Cacerolada ahora ante la sede del #PP en Madrid. Cientos de personas piden que se «disuelva y entregue la pasta»…	
Gallardón y Cobo serán llamados a declarar en la comisión de investigación de Madrid Calle 30 https://t.co/VjKYl61mX5	
Aguirre: La eterna lideresa dimite «engañada y traicionada», por @pablogh_ https://t.co/Gi2qKoKza4 https://t.co/scgxSc...	
Murcia, Comisión de Investigación, Madrid... Albert Rivera y Susana Díaz unen fuerzas... https://t.co/FKzFl5zOZs vía @El...	
Genial directo de @alexaiono en #Madrid #TheAfterPartyTour https://t.co/O5WV2HZ1NO	
Esta tarde en Madrid saímos ás rúas fronte a corrupción do PP. A cidadanía pide xustiza pan e democracia.… https://t.c...	
@marcosvegatw Hola. Si avisas a @Lineamadrid indicando la dirección exacta, podrán tomar nota y abrir incidencia. Gr...	
@GuilleFreelance Hola, si no se cumple la ordenanza, puedes comunicarlo a policía del distrito, sus datos en… https://t.c...	
La coportavoz de @IU_Madrid, @abrazopartio, el responsable de Acción Política de IU, @Ernesto_Alba y @edugaresp ...	
@Gobe1976 Hola, si tienes más dudas al respecto, puedes consultar en https://t.co/vMQc172ydE. Un saludo.	
Nuestro concejal en Ahora Madrid @carlossmato también pide que se disuelvan y entreguen la pasta. #CaceroladaGén...	
Desayunamos con el Padre Ángel https://t.co/0T2WNNgld9 #madridcontigo @alipiogutierrez @padre__angel https://t.co...	
@nilyeka @MADRID Hola. Solicitamos retirada con números de aviso: 2875700 y 2875703. Gracias por tu colaboración.	
@arkayo Hola, si no se cumple la ordenanza, puedes comunicarlo a policía del distrito, sus datos en https://t.co/A3VXKfx...	
□□@begonavillacis "Ciudadanos está plantando cara a la corrupción. La ciudadanía no puede inmunizarse a ella" https://t.	
□ si como madridista te agrada la idea. #ISCO2022 #LaLigahttps://t.co/WOnbbwJdlw	Depor...
□Empieza el #ConciertoMegaStar con @alexaiono en Madrid.□□Hola, cómo estás? □ https://t.co/10c8bAlRov	Ocio
Madrid en lucha contra su saqueo de riquezas, libertades y derechos. #ParaFrenarLaTrama https://t.co/btF1y1J01Y	
Hasta el 5 de mayo está abierto el plazo de inscripción en las escuelas infantiles municipales. Más información:… https://...	
@esttoper Para que lean sobre su Madrid adolescente @BillyPinedaPA @ricardoicaza507 @MelissaGallegoG @delval...	
A pesar del cansancio hoy voy a ver a @MundoChillon al Café Dublín porque sí, porque quiero y porque me da la gana. #...	
FRÁGIL. @GarethBale11 sufrió 17 lesiones desde que llegó al Real Madrid https://t.co/2VfrlmHN49 https://t.co/F8u5Ymb...	
AGARRALA, CRISTIANO! □□□□Julia Roberts fue la invitada de lujo del Real Madrid pero pidió conocer a Messi >… ht	Depor...
@MarceFX Hola, puedes hacer tu sugerencia en https://t.co/2KLCQeOAkO, en la opción "De carácter general". Gracias,...	

Ilustración 9

A continuación, se muestra una imagen con el aspecto general de la aplicación:

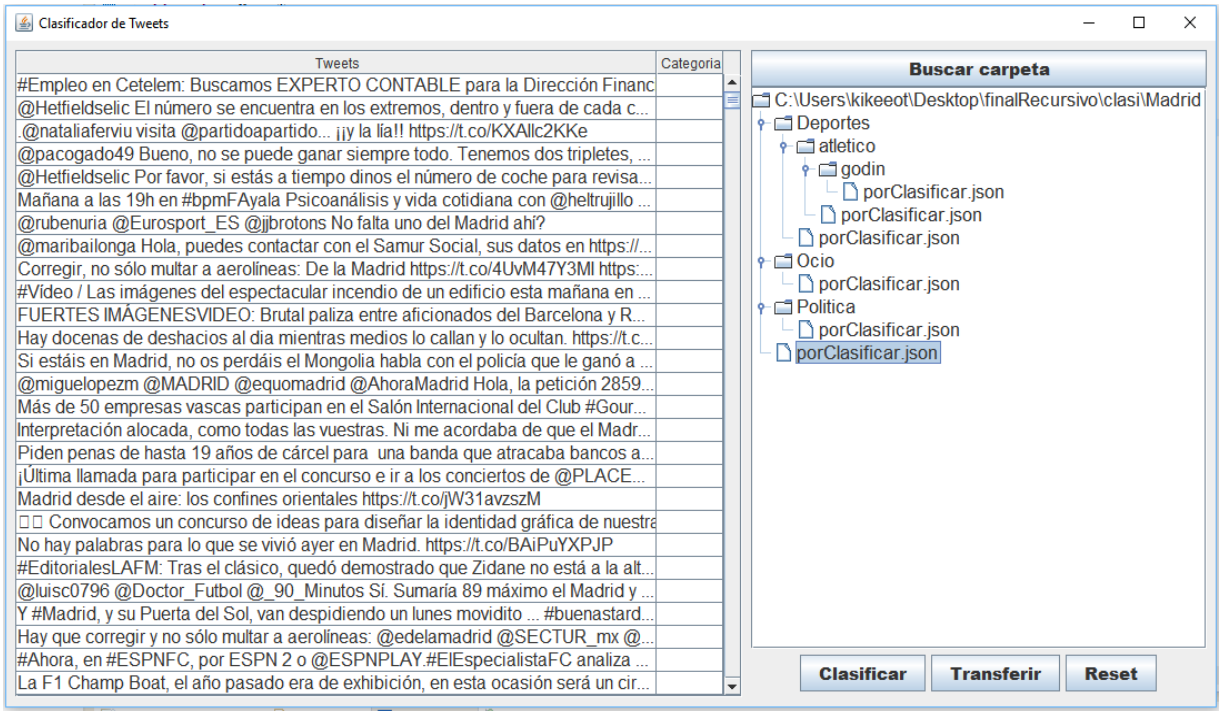


Ilustración 10

4. Clasificación automática

Vistos los capítulos anteriores, en este capítulo se explicará cómo partiendo de los tweets descargados en *Acceso y almacenamiento,* y utilizando como base para el entrenamiento los tweets clasificados en *Clasificación Manual*, se clasificarán automáticamente los tweets en las categorías ya creadas.

Teorema de Bayes

Para la clasificación automática hemos utilizado el *Teorema de Bayes*, el cual, fue planteado por el filósofo inglés Thomas Bayes en el año 1763.
"*Este teorema parte de una situación en la que es posible conocer las probabilidades de que ocurran una serie de sucesos Ai. A esta se le añade un suceso B cuya ocurrencia proporciona cierta información, porque las probabilidades de ocurrencia de B son distintas según el suceso Ai que haya ocurrido. Conociendo que ha ocurrido el suceso B, la fórmula del teorema de Bayes nos indica como modifica esta información las probabilidades de los sucesos Ai*." (Salinas, 2017)

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Veamos cómo se aplica el Teorema de Bayes en nuestro proyecto:

- *B:* Conjunto de palabras que componen el tweet.
- *Ai:* Categoría a la que queremos que pertenezca el tweet.
- *P(B):* Probabilidad de encontrar las características B en la muestra. Probabilidad de que las palabras que componen el tweet se encuentren en el contenido de todas las categorías.
- *P(Ai):* Probabilidad de que un elemento esté en el grupo Ai. Probabilidad que tiene la categoría frente al total.

- *P(B|Ai):* Probabilidad de que un objeto del grupo Ai tenga las características de B. Probabilidad de que el conjunto de palabras que componen el tweet se encuentren dentro de una categoría Ai.
- *P(Ai\B):* Probabilidad de que un objeto con las características B esté en el grupo Ai. Probabilidad de que un tweet pertenezca a una categoría. Este valor es el que deseamos calcular.

Para la implementación del Teorema de Bayes en nuestro proyecto hemos utilizado la API para Java de la biblioteca Apache Spark escrita en Scala, puesto que casi todo nuestro proyecto está desarrollado en Java. Apache Spark se utiliza fundamentalmente para aprendizaje automático y tratamiento masivo de datos.

Gracias a él hemos podido crear un programa, en el que se lee el contenido de cada categoría y establece qué palabras son más relevantes (a esta parte nos referiremos como entrenamiento), para posteriormente leer un tweet no clasificado y asignarle una categoría en función de los resultados del entrenamiento.

Entrenamiento

En el entrenamiento se lee el archivo JSON de cada categoría y se utiliza el 60% del contenido para entrenamiento, y el restante 40% se utiliza para validación.
Para establecer que palabras son más relevantes, los tweets se dividen en monogramas de palabras. A su vez, en el entrenamiento también se pueden reconocer bigramas de dos palabras, permitiendo diferenciar, por ejemplo, la palabra real, entre Real Madrid y Teatro Real.
Los procesos de entrenamiento y clasificación automática, explicado a continuación, solo se realizan con la variable "texto" del formato JSON, visto en el capítulo *Acceso y Almacenamiento*, que es lo realmente relevante. De esta forma quitamos todo el ruido que nos puedan ocasionar el resto de variables y caracteres contenidos en el archivo. A su vez, si el texto contiene alguna url, esta se quita, debido a que son enlaces acortados para que ocupen el menor espacio posible en un tweet, y que no se repiten en otros tweets aunque la url sea la misma. También se utiliza la variable "manual"

indicando que sólo aquellos tweets que se hayan clasificado en la clasificación manual, son válidos para el proceso de entrenamiento.
Si mediante la clasificación manual hemos creado subcategorías, el proceso de entrenamiento se encargará de recoger en un archivo temporal, toda la información fragmentada en distintas categorías. Evitando así la pérdida de información.
En el proceso de entrenamiento, es inevitable que en las distintas categorías existan archivos JSON con un número de tweets muy dispar entre ellas. Por ello, al entrenar se recoge el mínimo número de tweets contenido en una categoría, y se entrena con ese número de tweets en cada una de las categorías. De este modo, las palabras de los tweets evaluados tienen la misma probabilidad de encontrarse en todas las categorías.

Clasificación automática

Después de realizar el entrenamiento, el programa comienza el proceso de clasificación automática. Aquí se recorre el archivo JSON de la categoría padre y uno a uno se va clasificando siguiendo el Teorema de Bayes.
El tweet, al igual que en el proceso de entrenamiento, se divide en monogramas y bigramas de palabras, y se predice una a una la probabilidad de que estén contenidas en la categoría evaluada. Finalmente, se hace el productorio de los porcentajes de todas las palabras, resultando en la probabilidad de que un tweet pertenezca a una categoría $P(Ai\backslash B)$.
Esta probabilidad, si supera un umbral de similitud establecido por nosotros, permite a un tweet ser transferido a la categoría con la que guarde similitud, o por el contrario quedarse en el archivo de la categoría padre que se esté clasificando.
Cuando termina de clasificar un nivel del árbol de categorías, desciende en busca de subcategorías. Si existen, el programa vuelve a entrenar con las nuevas categorías y a clasificar con un nuevo archivo JSON. Este proceso se repite hasta que se recorre todo el árbol de categorías. Cuando finalice la clasificación, mostrará un mensaje informando de ello.

Resultados

Después de la clasificación automática, vamos a analizar si la utilización del Teorema de Bayes en nuestra aplicación, ha sido efectiva o no. Dentro de la clasificación supervisada, cómo es el caso del Teorema de Bayes, se utilizan dos términos para evaluar la efectividad del programa: *precision* y *recall*. (Powers, 2011)

- Precision: es la fracción de instancias recuperadas que son relevantes.
- Recall o exhaustividad: es la fracción de instancias relevantes que han sido recuperadas.

Veamos un ejemplo basándonos en la Ilustración 11. (Wikipedia, 2017)

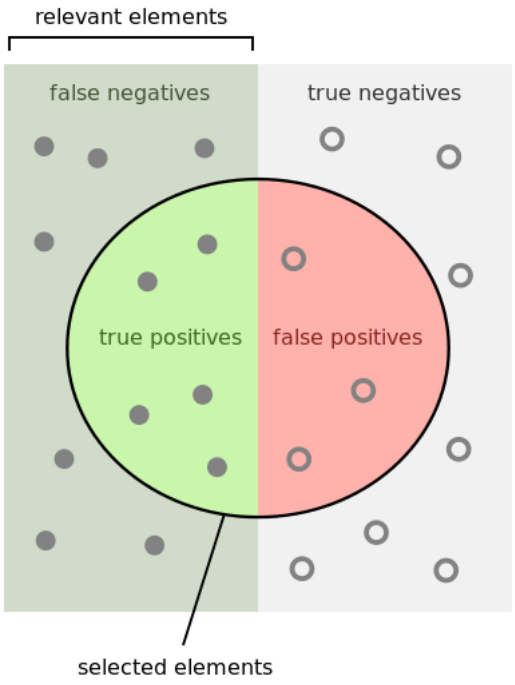
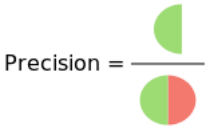


Ilustración 11

Partiendo de un programa de clasificación de tweets con 12 tweets de Deportes y 10 de Política, se han categorizado 8 tweets en deporte, de los cuales, sólo 5 realmente pertenecen a dicha categoría, según esto expresamos lo siguiente:



Precision: Son los tweets bien clasificados entre el total de clasificados en dicha categoría. 5/8

Recall: El número de tweets bien clasificados entre el total de tweets que realmente pertenecen a esa categoría. 5/12

Existe una medida de precisión que combina los valores explicados anteriormente. Esta medida, llamada Valor-F, se emplea en la determinación de un valor único ponderado de la precisión y el recall. Se calcula de la siguiente forma:

$$F_1 = 2 \cdot \frac{\mathrm{Precisión} \cdot \mathrm{Exhaustividad}}{\mathrm{Precisión} + \mathrm{Exhaustividad}}.$$

Para hacer pruebas hemos creado dos categorías, en las que previamente se han transferido tweets de forma manual. Estas categorías son Deporte y Política, pero para el estudio no debemos olvidar que hay un umbral de similitud que el tweet debe superar para ser transferido. Si no lo supera se queda en la categoría padre, por ello esta categoría también debe tenerse en cuenta.
El umbral de similitud lo hemos establecido en un 92%.

Se han seleccionado 800 tweets, los cuales se descomponen en las siguientes categorías: 300 tweets de Deportes, 300 de Política y 200 los cuales no pertenecen a ninguna de estas categorías. Se han obtenido los siguientes resultados tras realizar la clasificación automática.

- Deportes: ha clasificado 384 tweets como Deportes y realmente lo son 287.
  - Precision: 287/384 = 0,7473
  - Recall: 287/300 = 0,9566
  - Valor-F: 0,839

- Política: ha clasificado 361 tweets como Política y realmente lo son 275.
    - Precision: 275/361 = 0,7617
    - Recall: 275/300 = 0,9166
    - Valor-F: 0,832
- Sin clasificar: no se han conseguido clasificar 55 tweets y realmente lo son 34.
    - Precision: 34/55 = 0,6181
    - Recall: 34/200 = 0,17
    - Valor-F: 0,266

Ahora hemos realizado otro estudio, pero añadiendo otra categoría. Esta nueva categoría se llama Ocio.
Se han seleccionado 860 tweets los cuales se descomponen en las siguientes categorías: 300 Deportes, 300 Política, 160 Ocio y 100 tweets los cuales no pertenecen a ninguna de estas categorías. Se han obtenido los siguientes resultados tras realizar la clasificación automática.

- Deporte: ha clasificado 318 tweets como deporte y realmente lo son 274.
    - Precision: 274/318 = 0,8616
    - Recall: 274/300 = 0,9133
    - Valor-F: 0,8866
- Política: ha clasificado 277 tweets como deporte y realmente lo son 254.
    - Precision: 254/277 = 0,917
    - Recall: 254/300 = 0,8466
    - Valor-F: 0,8803
- Ocio: ha clasificado 197 tweets como deporte y realmente lo son 122.
    - Precision: 122/197 = 0,62
    - Recall: 122/160 = 0,76
    - Valor-F: 0,6829
- Sin clasificar: no se han conseguido clasificar 68 tweets y realmente lo son 23.
    - Precision: 23/68 = 0,338
    - Recall: 23/100 = 0,23

- Valor-F: 0,273

Interpretación de los resultados

Según los resultados obtenidos en las pruebas anteriores, sacamos las siguientes conclusiones:

- A mayor número de categorías se obtienen mejores resultados.
- La categoría sin clasificar siempre obtendrá inferiores resultados debido a que no tiene una base con la que entrenar y validar.
- Como podemos ver en el estudio con las 3 categorías, en la categoría Ocio obtenemos peores resultados debido a que se tratan temas más diversos que en las otras categorías.
- Con lo mencionado en el punto anterior, a mayor número de tweets que sirvan de base para el entrenamiento, más probabilidad hay de que los resultados sean más precisos.

5. Consultas y funcionalidad general

En este apartado vamos a mostrar una síntesis de la funcionalidad de la aplicación desarrollada, así como explicar el funcionamiento del módulo de consulta de tweets.

Como hemos comentado anteriormente, en Twitter hay una cantidad muy elevada de información en forma de tweets sin clasificar. Gracias al aprendizaje automático y a la clasificación manual hemos podido clasificar en categorías esta información.

Como resultado de ello, hemos desarrollado un módulo que nos permite tratar toda esa cantidad de información y categorizarla para posteriormente poder filtrarla.

Una vez hecho esto inicialmente de forma manual, el programa será capaz de hacerlo automáticamente, pasando el usuario a utilizar la aplicación como una aplicación de consulta.

Este módulo de consulta ha sido desarrollado también en Java. Lo que tratamos con este módulo es crear una interfaz con la que el usuario pueda acceder a las categorías creadas y mediante una serie de filtros realizar búsquedas de tweets en función de la información deseada.

Los filtros que se pueden aplicar para la búsqueda de tweets son los siguientes:

- Árbol con las categorías disponibles, en el cual el usuario tendrá que seleccionar la categoría en la cual desea buscar.
- Rango de fechas, donde el usuario podrá seleccionar unas fechas específicas de búsqueda de los tweets.
- Texto a buscar, con las palabras que el usuario quiera que aparezcan en el tweet. Si este campo se deja vacío no habrá filtro de palabras para los tweets.

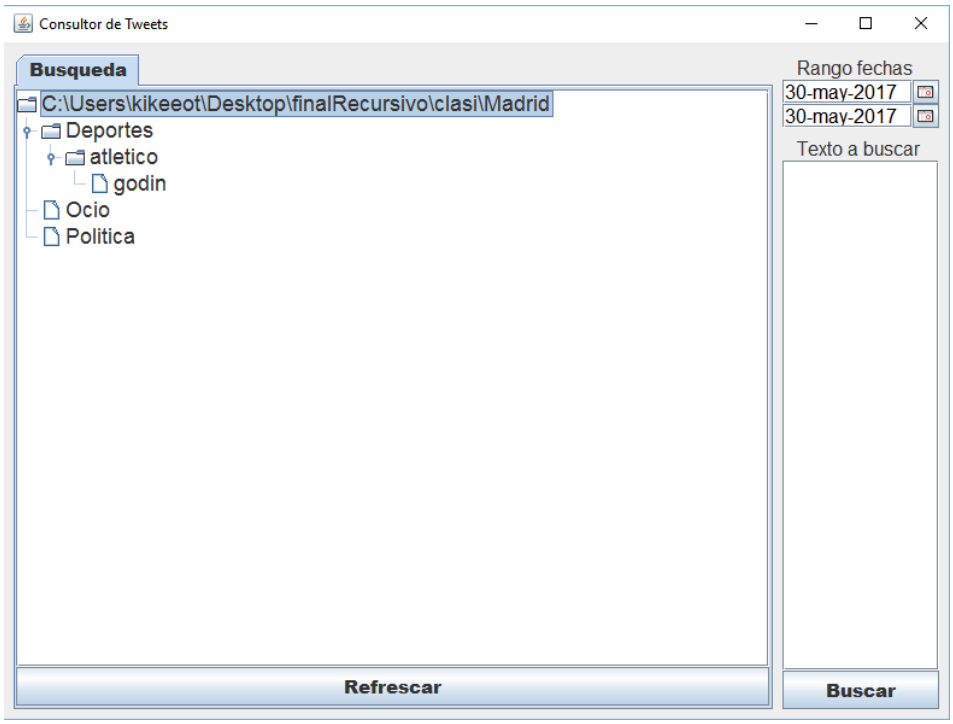


Ilustración 12

Una vez elegido la categoría deseada, rango de fechas y texto a buscar se mostrará en una nueva pestaña los resultados obtenidos como se muestra en la siguiente imagen.

Para mostrar los resultados de la búsqueda en pestañas, hemos utilizado el componente visual *JTabbedPane* así como el componente *JTable*. Se crearán tantas pestañas como búsquedas realice el usuario, pudiéndose también eliminarlas una vez consultadas.

Si el usuario está utilizando el módulo de consulta de tweets y el de clasificación manual al mismo tiempo y añade una nueva categoría al árbol de categorías, para que se refleje este cambio en el módulo de consulta el usuario deberá pulsar sobre el botón *Refrescar* que se puede ver en la Ilustración 12. Este paso será necesario solo cuando se estén utilizando ambos módulos a la vez.

Ilustración 13

Como se puede observar en la Ilustración 13, en una celda del *JTable* se muestra el texto del tweet y en la otra una url *clickable* que nos lleva o bien a la noticia del tweet abriendo el navegador por defecto del ordenador, o al propio tweet. Si el tweet no dispone de url no se mostrará nada. Mostramos esta url a parte debido a que esto nos permite acceder a la noticia y ver más información de ella, algo importante debido a la limitación de caracteres que tiene Twitter. Hoy en día la mayoría de tweets contienen dicha url para ampliar la noticia, por lo que nos ha parecido conveniente añadir esta funcionalidad a la aplicación, facilitando al usuario acceder a ella.

Para implementar dicha funcionalidad, hemos utilizado el método "Desktop.getDesktop().browse(url)", pasándole como parámetro la url seleccionada. Este método abrirá la dirección especificada con el navegador que tengamos predeterminado.

Acceso a los diferentes módulos

Una vez que el usurario acceda a la aplicación, inicialmente se abrirá el módulo de consultas. Para poder acceder al resto de módulos, el usuario tendrá que poner en el cuadro de texto, remarcado en rojo en la Ilustración 14, la siguiente serie de comandos dependiendo del módulo al que desee acceder:

- Si el usuario desea acceder al módulo de clasificación manual, deberá introducir "admin123" y pulsar el botón *buscar*.
- Si por el contrario desea acceder al módulo de almacenamiento de tweets deberá introducir en dicho cuadro de texto "buscar123" y pulsar el botón *buscar*.
- Del mismo modo, si se desea ejecutar el módulo de clasificación automática deberá introducir "automatico123". En este caso el programa empezará a clasificar automáticamente y cuando termine avisará de ello. Este proceso no dispone de interfaz gráfica.

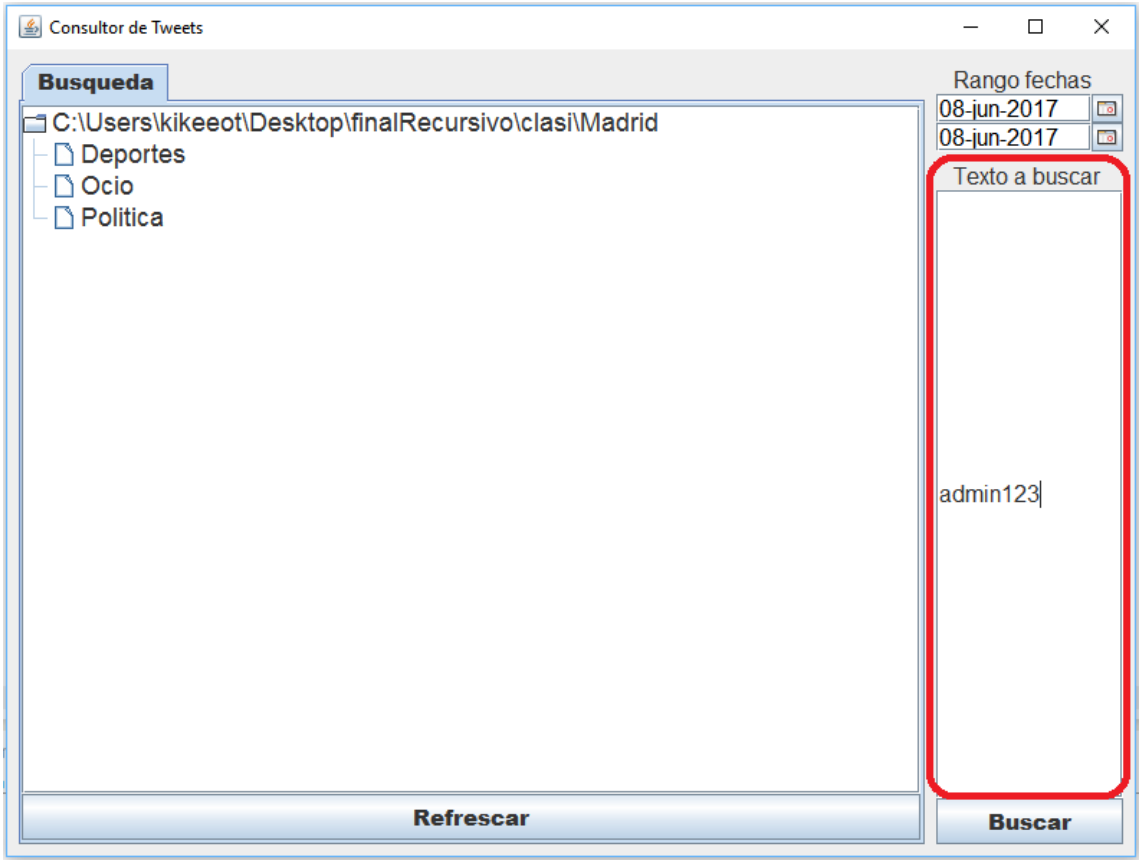


Ilustración 14

6. Conclusiones y trabajo futuro

El presente proyecto plantea una nueva forma de interacción con Twitter, de una manera sencilla para el usuario y nada costosa tanto en tiempo como en dinero.

Se ha comprobado estadísticamente que el programa funciona con un alto porcentaje de acierto en la clasificación automática de los tweets, sin olvidarse que para ello previamente se debe hacer una gran y precisa clasificación manual de tweets.

Podemos concluir que este programa podría ser de utilidad para un gran número de usuarios de Twitter, tanto particulares como para empresas.

Sin embargo, esta aplicación se podría mejorar en un futuro buscando mejores porcentajes de acierto. Para ello, se podría hacer una actualización en la que se combinasen tanto la API REST, que es la que actualmente se utiliza, con la Streaming API para así poder monitorizar eventos en tiempo real.

Disponer de una base de datos con tweets ya categorizados manualmente por otros usuarios sería una mejora de mucha utilidad, debido a que se podría ahorrar el trabajo de clasificación manual, o por lo menos facilitarlo.

Poder consultar la información en forma de gráficos o mapas de calor. Por ejemplo, mediante R o Python se podría monitorizar el contenido de la cuenta de Twitter de la DGT de Madrid, en la que anuncian las retenciones y que se vea reflejado en un mapa de la ciudad que estemos analizando.

Otra mejora que se podría implementar sería darle peso a los tweets en función de los seguidores que tenga el usuario o el número de retweets o respuestas que haya recibido. También se podría añadir un filtro de cuentas de Twitter y realizar búsquedas en función de una categoría y una cuenta determinada.

Finalmente, otra función que se podría añadir sería la de realizar un análisis sintáctico exhaustivo e identificar eventos futuros para poder añadirlos a un calendario.

Conclusions and future work

This project proposes a new way of interacting with Twitter, in a convenient non-costly way for the user in terms of time and money.

It's being statistically verified that this program works with a high percentage of success in the tweet's automatic classification, without forgetting that for this it is necessary to make a large and precise manual classification of tweets.

We can conclude that this program could be useful for a big number of Twitter users, both private and business.

However, this application could be improved in the future by looking for better percentage rates. To do this, we could combine both API REST, which is currently used, with the Streaming API and monitor events in real time.

Providing a data base with tweets already categorized manually by other users, would be a very useful improvement. It could ease the effort of the manual classification process of.

Being able to read the information in the form of graphs or heat maps. For example, via R or Python we could monitor the content of the Twitter account of the DGT of Madrid, in which they advertise the retentions and that is reflected in a map of the city we are analyzing.

Another improvement that could be implemented would be to give more weight to the tweets depending on the followers they have, retweets or answers. Also adding a Twitter account filter and performing searches based on a specific category and account, could help improve it.

Finally, another function that could be added, would be to perform a comprehensive syntactic analysis and identify future events that could be added to a calendar.

7. Contribuciones

El tema de este TFG lo acordamos con el director de proyecto antes de finalizar el curso 2015/16. Ese mismo verano, realizamos una investigación basada en un artículo que exponía que analizando la información de la red social Twitter se podían realizar análisis de sentimientos (Giachanou & Crestani, 2016).
Durante el verano comprobamos que la mayoría de usuarios de Twitter ya no lo utilizaban para expresar sentimiento u opiniones, se centraban más en noticias y en mensajes irónicos. Debido a esto concluimos que por cada tweet que expresará una opinión había miles de tweets desperdiciados que contenían información valiosa aunque no fuese un sentimiento u opinión. Por esta razón, una vez comenzó el curso decidimos enfocarlo en la categorización de tweets por temáticas.

Sixto Jansa Sanz

Una vez terminado el curso 2015/16 comencé con la primera tarea necesaria para este proyecto. La conexión con Twitter y descarga de tweets.
Nuestro director de proyecto nos recomendó utilizar el lenguaje Python con la biblioteca Tweepy. Hasta la fecha no había trabajado con dicho lenguaje, pero gracias a la documentación de la biblioteca Tweepy y a diversos tutoriales de Python en internet, conseguí crear una primera versión para la descarga de tweets. Esta primera versión fue mejorando gracias a los conocimientos sobre Python adquiridos en la asignatura Gestión de la información web y al blog (Karambelkar´s, 2015). Este blog ha sido de gran ayuda para conseguir la versión definitiva del programa de descarga de tweets. Gracias a él aprendimos como hacer mejor uso de los protocolos de autenticación de las API´s de Twitter permitiéndonos descargar 250% más tweets que antes. Este hecho nos facilitó la recolección de información en una sola ejecución.
Una vez conseguimos descargar un gran número de tweets y comenzar a clasificarlos de forma manual, había que comenzar con la clasificación automática.

Para hacer la clasificación automática, hemos hecho uso de técnicas de aprendizaje automático. Nuestro director de proyecto me facilitó un código en Java que hacía uso de Apache Spark. Este código siguiendo el Teorema de Bayes, se encargaba de leer

categorías y deducir a cuál de ellas podía pertenecer una frase dada. Mi labor consistió en adaptar este código a nuestras necesidades. Estas necesidades consistían principalmente en obtener toda la información contenida dentro de una categoría para crear un archivo con el que se pudiese entrenar. Después del entrenamiento se leen todos los tweets contenidos en el archivo JSON de la carpeta padre, clasificándolos automáticamente siempre que superen un umbral de similitud establecido por nosotros.

Una vez se implementaron todos los módulos que daban forma a la aplicación, decidimos que había que incluir una opción para descargar tweets, ya que este programa lo veníamos ejecutando desde el terminal, por lo que si querías descargar tweets y luego clasificarlos, tenías que utilizar dos aplicaciones diferentes. Elaboré una clase “conexión” en Java que recogía los atributos de la búsqueda y ejecutaba el programa de descarga de tweets gracias al método *Runtime.getRuntime().exec()*. De este modo simplificaba el uso de la aplicación, agrupando la ejecución de todos los módulos.

En la fase final, cuando todos los módulos encaraban su versión final era hora de estudiar la efectividad de la clasificación automática. Para ello tanto mi compañero como yo dedicamos mucho tiempo a la descarga de tweets y su posterior clasificación manual. Con una buena base para el entrenamiento inicié el estudio que se puede ver en el apartado de *Resultados.* Para ello había que llevar un conteo de los tweets que se clasificaban en una categoría y cuales realmente pertenecían a dicha categoría. De esta forma pude obtener los resultados precisión, *recall* o exhaustividad y valor-F con el porcentaje de efectividad.

Por último, también he desempeñado la labor de testing buscando errores en los módulos que elaboraba mi compañero, ya que siempre es más fácil encontrar errores si no has desarrollado tú la aplicación.

Enrique Ortiz Torralba

Mi aportación principal del proyecto ha sido la de diseñar e implementar los módulos de clasificación manual y consultas y funcionalidad general, así como la estructura de carpetas y el traspaso de tweets.

Para realizar esta parte creímos que lo más conveniente sería realizarlo en java, puesto que durante la carrera hemos visto este lenguaje y con el cual nos sentimos cómodos a la hora de trabajar. A su vez, en internet podemos encontrar multitud de foros y explicaciones que podíamos consultar en caso de surgirnos algún problema durante la implementación.

Realizamos unos primeros bocetos de cómo sería la aplicación para ir viendo que elementos íbamos a necesitar y como lo íbamos a integrar. Una vez decidimos como sería la aplicación, empecé a investigar que componentes serían los más oportunos para desarrollarla.

Puesto que la mejor opción para mostrar la estructura de carpetas sería algo en forma de árbol, encontré el componente *JTree* que permitía crearlos de una forma sencilla. Estuve investigando cómo funcionaba y la manera de utilizar este componente para la creación de la estructura de carpetas, algo que era esencial en nuestro proyecto. El componente *JTree* se basa en nodos, donde el nodo principal (*root*) sería la carpeta inicial, en nuestro caso Madrid. En el *JTree* también es posible añadir nuevos nodos si el usuario lo considera oportuno.

Para poder mostrar los tweets utilicé el componente *JTable*, el cual permite crear una tabla con el número de filas y columnas que se vayan necesitando.

Tanto el componente *JTree* como el componente *JTable* se han utilizado tanto en el módulo de clasificación manual como en el módulo consultor.

En el caso del módulo de clasificación, al *JTree* se le implementó un *listener* que recogía los eventos del ratón. Este *listener* era el encargado de saber si se había pulsado una categoría con el botón derecho, y de ser así crearse un nuevo *frame* donde introducir el nombre de la categoría. Después de esto, se añadía la nueva categoría al *JTree*. Si por el contrario se pulsa el botón izquierdo, el *listener* recoge la categoría pulsada. En el caso de pulsar sobre el archivo *porClasificar.json* este se mostrará en el *JTable*.

Una vez seleccionado el archivo que queremos clasificar, se pulsa sobre el botón Clasificar pasándose así a la categorización de tweets.

Para el *JTable* también se implementó un *listener*, pero en este caso lo que recogía era la fila seleccionada para llevar una contabilización de los tweets pulsados y que se querían clasificar. Una vez seleccionado un tweet, se deberá pulsar sobre la categoría en el *JTree* a la que se desee transferir. Tras clasificar los tweets deseados se deberá pulsar el botón Transferir, el cual tiene implementado un *listener* que llama a un método para transferir los tweets a las categorías correspondientes.

Para el caso del módulo consultor el *JTree* presenta unos pequeños cambios. Por un lado no permite añadir categorías ni mostrar los archivos JSON. Solo permite seleccionar una categoría para realizar la búsqueda en ella. También se han implementado una serie de filtros que se pueden aplicar al realizar la búsqueda. Se ha añadido un *JCalendar* para realizar un filtro de fechas, pudiéndose añadir un rango para la búsqueda. También se ha añadido un cuadro de texto para filtrar tweets en función de las palabras contenidas.

Tras aplicar los filtros deseados se ha utilizado el componente *JTabbedPane* para mostrar las búsquedas. Este componente se encarga de crear una pestaña por cada búsqueda que se realice.

Dentro de cada *JTabbedPane* se ha añadido un *JTable* al igual que en el módulo de clasificación manual para mostrar los tweets. Si existe una URL contenida en cada tweets, esta se mostrará en una nueva columna reservada para ellas en el *JTable*. Si pulsamos sobre estas URL se nos abrirá el navegador por defecto que tengamos en el ordenador con dicha dirección.

Bibliografía

Cook, B., & Messina, C. (Noviembre de 2006). Obtenido de https://oauth.net/

Dorsey, J. (25 de Marzo de 2009). *tecnologia.elpais.com*. Obtenido de http://tecnologia.elpais.com/tecnologia/2009/03/25/actualidad/1237973279_850215.html

Giachanou, A., & Crestani, F. (2016). Like It or Not: A Survey of Twitter Sentiment Analysis Methods. En *ACM Comput. Surv. Vol. 49, Issue 2, Article 28* (págs. 1-41). ACM. Obtenido de https://doi.org/10.1145/2938640

Karambelkar´s, B. (2015). *https://www.karambelkar.info.* Obtenido de https://www.karambelkar.info/2015/01/how-to-use-twitters-search-rest-api-most-effectively./

Pennacchiotti, M., & Popescu, A.-M. (2011). A Machine Learning Approach to Twitter User Classification. En *Icwsm, 11* (págs. 281-288).

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. En *International Journal of Machine Learning Technology, Vol. 2, No.1.* (págs. 37-63).

R. Andrea. (15 de Mayo de 2014). *https://hipertextual.com*. Obtenido de https://hipertextual.com/archivo/2014/05/que-es-api/

Roesslein, J. (2009). *http://www.tweepy.org/*. Obtenido de http://www.tweepy.org/

Salinas, J. M. (2017). *http://www.ugr.es.* Obtenido de http://www.ugr.es/~jsalinas/bayes.htm

Smith, K. (7 de Junio de 2016). *https://www.brandwatch.com*. Obtenido de https://www.brandwatch.com/es/2016/06/44-estadisticas-twitter-2016/

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (19 - 23 de Julio de 2010). Short text classification in twitter to improve information filtering. *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10)* (págs. 841-842). New York, NY, USA: ACM. Obtenido de http://dx.doi.org/10.1145/1835449.1835643

Statista. (Junio de 2017). *https://www.statista.com*. Obtenido de https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

TwitterSupport. (Junio de 2017). *https://support.twitter.com*. Obtenido de https://support.twitter.com/articles/247670?lang=es

Wikipedia. (Junio de 2017). *https://en.wikipedia.org*. Obtenido de Precision and recall: https://en.wikipedia.org/wiki/Precision_and_recall