

---

# T.E.F.G

## Generador de frecuencias de eventos temporales

---



Trabajo Fin de Grado de Ingeniería Informática

Esther Ávila Benito  
Guillermo Romero Alonso

*Dirigido por*  
Rafael Caballero Roldán

Departamento de Sistemas Informáticos y Computación  
Facultad de Informática  
Universidad Complutense de Madrid

Madrid, 28 de mayo de 2018



**T.E.F.G**

**Generador de frecuencias de eventos temporales**

*Trabajo de Fin de Grado*

**Esther Ávila Benito**  
**Guillermo Romero Alonso**

*Dirigido por*

**Rafael Caballero Roldán**

**Facultad de Informática**  
**Universidad Complutense de Madrid**

**Madrid, 28 de mayo de 2018**



# Autorización

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines academicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

*Esther Ávila Benito*

*Guillermo Romero Alonso*



# Agradecimientos

Principalmente, quiero dar las gracias a mi madre y a mi hermano. Ellos son los que me han acompañado en mi viaje universitario animándome en los malos momentos y compartiendo conmigo las alegrías. Gracias por haber confiado en mí en todo momento y haberme empujado en esta aventura.

Por último, quiero expresar mi más profundo agradecimiento a mi abuela. Sin saber nada sobre informática, me ha dado muchas de las lecciones que no olvidaré en toda mi vida. Desde mi más profunda admiración, este trabajo está dedicado a ella.

*Esther*

Quiero darle las gracias a mis padres por el apoyo que han puesto en mí, motivándome a lo largo de este proyecto y de toda mi carrera universitaria, sobretodo en los momentos más difíciles.

También quiero dedicar todo el esfuerzo a mis hermanos, Ana y Julio, que han mostrado un especial interés y han alimentado mis ganas de seguir adelante.

*Guillermo*

También queremos agradecer a nuestro director Rafael Caballero Roldán su implicación, ayuda y esfuerzo para que este proyecto llegara a buen puerto. Gracias a sus conocimientos y, sobre todo, a su pasión en el ámbito de este trabajo, ha ido creciendo nuestra motivación para sacar lo mejor de nosotros.

Tampoco nos queremos olvidar de nuestros compañeros, que nos han acompañado a lo largo de nuestra aventura universitaria. Ellos han hecho que esta etapa haya sido única, agradable y llena de momentos inolvidables.



# Resumen

Las redes sociales han supuesto un cambio en la forma de conectar con otras personas y compartir información: lo que se hace, comentarios sobre los que se ve en otros medios, pensamientos y opiniones, etc. A escala global, esto genera una ingente cantidad de datos que puede proporcionarnos información muy útil. Además, se ha generado un gran interés por parte de grandes empresas de sectores como el análisis de opinión o el marketing, que utilizan los gustos y preferencias expresados a través de estas redes para orientar la toma de decisiones en la empresa.

En este proyecto proponemos analizar eventos con duración limitada, como partidos de fútbol o debates electorales, con objeto de extraer la información que mejor representa estos eventos desde el punto de vista de los usuarios de redes sociales. Para ello, comenzamos almacenando los datos generados en estos eventos para su posterior análisis. Posteriormente, realizamos un tratamiento exhaustivo buscando la información más relevante a partir de la frecuencia de mensajes en cada instante del evento. Para visualizar esta información recurrimos a gráficos que muestran la frecuencia de mensajes y los momentos clave, así como los mensajes que mejor representan estos momentos. Posteriormente, el usuario final puede interactuar con la representación, por ejemplo, haciendo *zoom* de un intervalo de especial interés.

Además de mostrar la evolución del evento de forma gráfica, nuestra herramienta también proporciona un resumen cronológico textual de los acontecimientos más importantes del evento, mostrándonos de esta forma cómo se ha percibido el evento bajo el punto de vista de los usuarios de redes sociales.



# Palabras clave

- Redes sociales
- Twitter
- Evento
- Frecuencia
- Gráfica
- Timeline
- Pymongo



# Abstract

Social networks have supposed a change in the way people connect each other and share information: what they do, comments about what you see in other media, thoughts and opinions, etc. On a global scale, this generates a huge amount of data that can provide very useful information. Moreover, there is a great interest on the part of big companies in sectors such as opinion analysis or marketing, which make use of tastes and preferences stated through networks to guide the business policy-making.

In this project we suggest to analyze time-bound events, such as football matches or election debates, in order to retrieve the information which best represents these events under the social networks user's point of view. Later, we perform a comprehensive treatment to look for the most relevant information from the messages frequency at every moment of the event. To visualize this information, we make use of graphs that show the messages frequency and the key moments, as well as the messages that best represent the moment. After that, the final user can interact with the graph, for example, by zooming into an interval of great interest.

In addition to show the evolution of the event in a graphic way, our program also provides a textual timeline of the most important occurrences of the event, showing how the event has been perceived under the point of view of social network users.



# Keywords

- Social networks
- Twitter
- Event
- Frequency
- Graph
- Timeline
- Pymongo





# Índice general

<b>Autorización</b>	<b>v</b>
<b>Agradecimientos</b>	<b>vii</b>
<b>Resumen</b>	<b>ix</b>
<b>Palabras clave</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Keywords</b>	<b>xv</b>
<b>1 Introducción</b>	<b>5</b>
1.1 Motivación . . . . .	5
1.2 Objetivos . . . . .	7
1.3 Tecnologías utilizadas . . . . .	8
1.3.1 Twitter . . . . .	9
1.3.2 Python . . . . .	9
1.3.3 MongoDB . . . . .	10
1.4 Estructura de la memoria . . . . .	11
<b>2 Introduction</b>	<b>13</b>
2.1 Motivation . . . . .	13
2.2 Objectives . . . . .	15
2.3 Used technologies . . . . .	16
2.3.1 Twitter . . . . .	16
2.3.2 Python . . . . .	16
2.3.3 MongoDB . . . . .	18

2.4	Memory structure . . . . .	19
<b>3</b>	<b>Estado del arte</b>	<b>21</b>
3.1	Hotstream . . . . .	22
3.2	TwitterMonitor . . . . .	23
3.3	Buzzer . . . . .	24
3.4	Twical . . . . .	25
<b>4</b>	<b>Captura de los datos</b>	<b>27</b>
<b>5</b>	<b>Importación de los datos</b>	<b>29</b>
5.1	Requisitos previos . . . . .	29
5.2	Pantalla de importación . . . . .	30
5.2.1	Importación a la base de datos . . . . .	30
5.2.2	Preparación de fechas . . . . .	31
<b>6</b>	<b>Cálculo de frecuencias</b>	<b>35</b>
6.1	Colección de frecuencias . . . . .	35
<b>7</b>	<b>Representación gráfica</b>	<b>37</b>
7.1	Datos de entrada . . . . .	37
7.2	Funcionalidades de la gráfica . . . . .	39
7.2.1	Tweets informativos . . . . .	39
7.2.1.1	Tweet en un intervalo . . . . .	40
7.2.1.2	Tweet en un punto . . . . .	40
7.2.2	Información complementaria . . . . .	41
7.2.3	Zoom . . . . .	42
<b>8</b>	<b>TF-IDF</b>	<b>45</b>
8.1	TF-IDF en Python . . . . .	47
<b>9</b>	<b>Detección automática de picos y Timeline</b>	<b>49</b>
9.1	Definición y obtención de los picos . . . . .	49
9.2	Generación del Timeline . . . . .	52
<b>10</b>	<b>Contribuciones al proyecto</b>	<b>55</b>
10.1	Esther Ávila Benito . . . . .	56

10.2	Guillermo Romero Alonso . . . . .	56
10.3	Trabajo en común . . . . .	57
<b>11</b>	<b>Conclusiones y trabajo futuro</b>	<b>59</b>
11.1	Conclusiones . . . . .	59
11.2	Líneas de trabajo futuro . . . . .	61
<b>12</b>	<b>Conclusions and future work</b>	<b>63</b>
12.1	Conclusions . . . . .	63
12.2	Future work lines . . . . .	64

# Índice de figuras

1.1	Logo de Python . . . . .	10
1.2	Logo de MongoDB . . . . .	11
2.1	Python logo . . . . .	17
2.2	MongoDB logo . . . . .	19
3.1	Interfaz de Hotstream . . . . .	23
3.2	Interfaz de TwitterMonitor . . . . .	24
3.3	Interfaz de Buzzer . . . . .	25
3.4	Ejemplo de tupla extraída por TwiCal . . . . .	26
4.1	Diferencias entre los formatos JSON y CSV con los mismos campos	28
5.1	Pantalla inicial de T.E.F.G . . . . .	30
5.2	Ejecución de <i>mongoimport</i> con la creación de un subproceso . . . . .	31
5.3	Fragmento de la colección importada . . . . .	33
6.1	Fragmento de la colección de frecuencias . . . . .	36
7.1	Datos de entrada para la gráfica de una colección importada . . . . .	38
7.2	Tweet representativo del evento . . . . .	40
7.3	Tweet representativo de un punto de la gráfica . . . . .	41
7.4	Zoom aplicado sobre una gráfica y nueva gráfica generada . . . . .	43
8.1	Fórmula para el cálculo del factor IDF . . . . .	46
9.1	Interfaz inicial de T.E.F.G . . . . .	50
9.2	Tipos de picos en la gráfica de un evento . . . . .	51
9.3	Gráfica con tres picos . . . . .	53
9.4	Timeline de la gráfica anterior . . . . .	53

# Capítulo 1

## Introducción

### 1.1. Motivación

Cada vez son más las personas que tienen acceso a internet a través de un ordenador, smartphone, tablet, etc. Esto conlleva el acceso a las redes sociales. Se estima que el 37 % de la población mundial tiene cuenta de usuario en al menos una red social, siendo Facebook la más utilizada (EST, 2017).

Las redes sociales tienen sus inicios en el campo de la sociología (Barnes, 2007), aunque el concepto ha adquirido mayor notoriedad con el nacimiento de las redes sociales basadas en internet. Según (Boyd and Ellison, 2007), una red social basada en la web, que llamaremos simplemente red social en el resto del trabajo, es un servicio web que permite a los individuos publicar información bajo un perfil público o semi-público, gestionar la lista de usuarios con los que se conectan y tener acceso a la información generada por otros usuarios.

En este sentido se considera que la primera red social basada en web se desarrolló en 1995, cuando Randy Conrads creó el sitio web “classmates.com” (ORI, 2017). El objetivo de este portal era que las personas pudieran recuperar y mantener el contacto con antiguos compañeros de colegio, instituto, universidad o trabajo. En 1996 surgió sixdegrees.com, que permitió a los usuarios crear perfiles, listas de amigos y enviar mensajes. Sin embargo, no fue hasta el siglo XXI cuando aparecieron los primeros sitios web que promocionaban las redes de círculos de amigos en línea. En 2003 se produjo la aparición de importantes redes que produjeron

un incremento exponencial de la gente que estaba sumergida en este mundo. Nos referimos a Friendster, Ecademy, Soflow, Tribe.net, MySpace y LinkedIn, además de otros sitios más desconocidos que elevaron a 200 el número de redes sociales de ese año.

Si hay una fecha importante en el mundo de las redes sociales, es el año 2004, cuando se creó Facebook (FAC, 2009), que nació como una versión en línea de los “facebook” de las universidades americanas. Los “facebook” son publicaciones que las universidades hacen a comienzo de curso con las fotografías y nombres de los estudiantes. Tienen como objetivo ayudar a conocerse a los estudiantes de primer curso.

Facebook nació de la mano de Mark Zuckerberg, estudiante en aquel momento de Harvard. En su primer mes de funcionamiento contaba con la suscripción de más de la mitad de los estudiantes de Harvard. Un año después, Facebook tenía más de un millón de usuarios e incorporó a los alumnos de más de 25.000 escuelas secundarias y 2.000 universidades de Estados Unidos y del extranjero, logrando más de 11 millones de usuarios. En 2007 llegó a tener la mayor cantidad de usuarios registrados en comparación con otros sitios web dedicados a estudiantes de nivel superior, teniendo más de 19 millones de usuarios en todo el mundo.

Sin embargo, la red social más utilizada para compartir información y opiniones durante la realización de eventos es Twitter (TWI, 2009), un servicio de micro-blogging que permite a sus usuarios enviar micro-entradas de texto denominadas “tweets”. Twitter nació en el año 2006 de la mano de un grupo de jóvenes emprendedores que trabajaban para la compañía de Podcasts Odeo, en San Francisco. En el marco de una de las reuniones de la empresa, Jack Dorsey propuso una idea en la que se podrían usar SMS para decirle a un grupo pequeño de personas lo que estaba haciendo uno de ellos. Una vez iniciado el proyecto, propusieron varios nombres. El nombre original durante un tiempo fue “Status”, pasando por “Twitch” a causa del tipo de vibraciones de los móviles, quedándose finalmente con Twitter.

Twitter fue lanzado al público en julio de 2006. En el primer año de vida sufrió una evolución nada despreciable, pasando de 20.000 tweets al día hasta los 60.000 (EVO, 2010). En noviembre de 2009 apareció la versión de Twitter en español y un año después, el microblogging publicó una aplicación para que los usuarios de forma no lucrativa lo tradujeran al español, francés e italiano. De hecho, muchas de las características que Twitter ha añadido a lo largo de su vida, han sido inicia-

tivas de los usuarios. Un ejemplo son los “hashtags”, etiquetas conformadas por palabras clave elegidas por los usuarios y precedidas por el símbolo “#”. También los “trending topics” o temas del momento están establecidos como los términos más utilizados por los usuarios, según la región o país donde se encuentren.

Twitter ha estado entre los diez sitios más visitados del mundo durante la mayor parte de su existencia. En febrero de 2009 fue la tercera red social más visitada con 55 millones de visitas anuales (USU, 2010) y un crecimiento anual estimado de 1382 %. En 2013, ya contaba con 200 millones de usuarios, llegando a los 330 millones en la actualidad.

Por todo el impacto que ha tenido Twitter en los últimos años en la sociedad, hemos creado una herramienta que gestiona la masiva cantidad de datos generados en cualquier evento temporal que tenga cierto impacto social. A través de técnicas de visualización, ofrecemos al usuario una visión global de la actividad social durante el evento. Además, empleamos algoritmos de recuperación de información para mostrar al usuario la información más relevante.

Para cualquier evento, el usuario es capaz de ver el volumen de tweets por segundo, identificar los momentos clave (picos de la gráfica) e, interactuando con la gráfica, conocer el tema predominante en esos momentos.

## 1.2. Objetivos

El objetivo del entorno que proponemos y al que hemos llamado T.E.F.G (Temporary Event Frequency Generator), es aplicar de forma semi-automática técnicas de análisis de datos, que puedan ser utilizadas por usuarios no necesariamente expertos en la materia. Además, queremos que este entorno sea capaz de gestionar rápidamente una gran cantidad de información, generando representaciones visuales e informes breves que permitan captar de un solo vistazo el resumen de un evento que a menudo consta de centenares de miles de mensajes.

T.E.F.G también puede ser de gran utilidad para las empresas, pues no es ninguna novedad que en la actualidad la información se puede transformar en beneficio económico. Con nuestra herramienta, las empresas pueden conocer mejor los intereses de sus clientes y ofrecer sus productos en base a estos.

En particular, los objetivos de T.E.F.G. son:

- Resumir de forma gráfica un evento en Twitter. La gráfica debe mostrar la frecuencia de tweets en cada instante, ya que asumimos que los momentos más relevantes corresponderán con los picos de mayor frecuencia.
- Además, para cada pico de frecuencia, o en general, para cada intervalo que el usuario seleccione, deseamos ser capaces de ofrecer los tweets más significativos que resumen el contenido informativo del intervalo.
- Aplicando el punto anterior al evento completo deseamos ser capaces de generar un *timeline* con los tweets más significativos que resumen el evento.

Además, desde un punto de vista de nuestra formación, nos pareció interesante llevar a cabo un proyecto que nos permitiera:

- Trabajar con algo tan cercano a nosotros como son las redes sociales, en especial Twitter.
- Ser conscientes de la utilidad que tiene la información disponible en las redes sociales.
- Adentrarnos en las tecnologías Big Data, más en concreto la base de datos MongoDB.
- Ampliar nuestro conocimiento en lenguajes de programación, en nuestro caso Python 3, desconocido por nosotros antes de comenzar el proyecto.
- Familiarizarnos con las técnicas de visualización de datos, más en concreto la librería *matplotlib* de Python.
- Aprender a utilizar técnicas de recuperación de información. En nuestro caso hemos utilizado TF-IDF.

### 1.3. Tecnologías utilizadas

En este apartado haremos una descripción de las tecnologías que nos han permitido desarrollar nuestra aplicación.



### 1.3.1. Twitter

Además de ser una de las redes sociales más utilizadas, Twitter nos ofrece la posibilidad de descargar la información de forma gratuita, lo que nos permitirá obtener los tweets ocurridos durante un evento determinado.

### 1.3.2. Python

Es un lenguaje de programación interpretado diseñado por Guido van Rossum a principios del año 1991 (PYT, 2015). Este lenguaje surgió de las limitaciones y problemas que encontró su diseñador al trabajar con el lenguaje ABC en algunos proyectos. En la actualidad, Python ofrece bastante versatilidad y es utilizado para el desarrollo de interfaces gráficas, desarrollo web, gestión de bases de datos, etc. Python se encuentra bajo una licencia propia denominada Python Software Foundation Licence, que se diferencia de la licencia GPL al ser una licencia no-copyleft. Este hecho implica que es posible modificar el código fuente y desarrollar código derivado sin la necesidad de hacerlo *open source*.

Python es un lenguaje de programación multiparadigma. Esto significa que permite a los programadores la adopción de varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Usa tipado dinámico y conteo de referencias para la administración de memoria.

El código de este proyecto se ha desarrollado en su totalidad con la versión de Python 3.0, publicada en diciembre de 2008.

El otro lenguaje alternativo para el desarrollo de este proyecto era Java, pero fue descartado porque tal y como se describe a continuación, MongoDB ha sido la base de datos con la que hemos trabajado y Python dispone de un módulo llamado Pymongo. A pesar de que MongoDB disponga de módulos y drivers para la mayoría de lenguajes de programación, este módulo es el más intuitivo y no requiere instalación. Además permite conectar el programa con una base de datos MongoDB y utilizar en Python todas las consultas y operaciones disponibles.



Figura 1.1: Logo de Python

Además del código propio del “núcleo” de Python, hemos utilizado varias bibliotecas que proporcionan funcionalidades adicionales. Algunas de las que hemos empleado son las siguientes:

- **matplotlib:** es una biblioteca que permite la visualización de gráficos a partir de datos contenidos en listas.
- **tkinter:** biblioteca de Python empleada para la creación de interfaces gráficas de usuario.
- **pymongo:** módulo que permite la conexión de cualquier aplicación Python con una base de datos MongoDB.
- **csv:** se emplea para leer el contenido de un archivo en formato CSV. En nuestro caso, lo usamos para saber el tamaño del archivo que se importará y así poder actualizar la barra de progreso coherentemente.
- **subprocess:** es un módulo que permite usar directamente órdenes propias del sistema operativo mediante la creación de un proceso concurrente al que lo ha invocado. En este proyecto se ha empleado para la importación del archivo de datos mediante la ejecución de la herramienta *mongoimport*.

### 1.3.3. MongoDB

Es un sistema de base de datos NoSQL orientado a documentos y desarrollado bajo el concepto de código abierto (MON, 2014). Está disponible para los sistemas operativos Windows, Linux, OS X y Solaris. Fue desarrollado por la empresa

MongoDB Inc. y lanzado al mercado en 2007. MongoDB está disponible de forma gratuita bajo la licencia AGPL de GNU.

En MongoDB la información se almacena de manera interna en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON. MongoDB está escrito en C++ y viene de serie con una consola para ejecutar los comandos. Esta consola está construida sobre JavaScript, por lo que las consultas se realizan utilizando este lenguaje. Además, MongoDB ofrece drivers oficiales para una gran cantidad de lenguajes de programación como C#, Java, Node.js, PHP, Python, Ruby, C, C++ o Scala. Nosotros hemos utilizado el driver que permite su conexión con Python.

Este sistema de base de datos está disponible en todo nuestro proyecto, ya que es el único que empleamos para el almacenamiento de toda la información.



Figura 1.2: Logo de MongoDB

## 1.4. Estructura de la memoria

La estructura en la que hemos redactado esta memoria sigue un orden natural similar al que seguirá el usuario final cuando utilicé nuestra aplicación.

En el capítulo en el que nos encontramos, explicamos la motivación que nos ha llevado a realizar este proyecto, así como los objetivos propuestos al inicio de éste. Además, describimos las tecnologías que hemos utilizado para el desarrollo de nuestro trabajo. Este capítulo está redactado en español e inglés.

En el capítulo 3 hemos realizado un estudio sobre herramientas punteras dentro del mismo área que la nuestra. Además, hemos explicado características de estas aplicaciones que son relevantes y nos han servido de inspiración en nuestro

proyecto.

En el capítulo 4 detallamos cómo hemos obtenido los datos de Twitter que constituirán la base de nuestro posterior análisis.

En el capítulo 5 describimos los pasos seguidos para importar correctamente los datos obtenidos de un evento y crear la colección asociada. También explicamos cómo estos datos deben ser preparados para las etapas posteriores.

En el capítulo 6 detallamos el proceso que hemos seguido para la creación de la colección de frecuencias una vez que los datos han sido importados y preparados.

En el capítulo 7 explicamos los datos de entrada necesarios para generar la gráfica. Además, se describen detalladamente las funcionalidades que ésta ofrece al usuario final (tweets representativos, información adicional y *zoom*).

En el capítulo 8 se presenta el algoritmo TF-IDF utilizado, en nuestro caso, para la obtención del tweet más representativo de un intervalo y de un punto en un evento.

En el capítulo 9 se explican los cálculos llevados a cabo para la detección automática de picos en una colección de frecuencias y su posterior visualización en la gráfica. Además, se presenta la herramienta del *timeline*, relacionada estrechamente con el algoritmo TF-IDF y los picos de la gráfica.

En el capítulo 10 se redactan las aportaciones de cada uno de los integrantes del equipo al proyecto. También se detalla la metodología y organización del trabajo a lo largo del curso.

Por último, en el capítulo 11 se explican las conclusiones finales del proyecto y las posibles mejoras que se podrían incorporar a nuestro trabajo en un futuro.

# Capítulo 2

## Introduction

### 2.1. Motivation

A growing number of people are getting access into the Internet through a computer, smartphone, tablet, etc. This conveys the access to social networks. Is estimated that 37% of the world population has a user account in at least one social network, being Facebook the most used (EST, 2017).

Social networks have their roots in the field of sociology (Barnes, 2007) although the concept has acquired greater notoriety with the birth of social networks based on the Internet. According to (Boyd and Ellison, 2007), social network based on the web, that we simple call “social network” in this thesis, is a service that allows users to post information under a public or semipublic profile, manage the users list whose they connect with and get access to the information generated by other users.

In this way it is considered that the first social network based on the Internet was developed in 1995, when Randy Conrads created the website called “classmates.com” (ORI, 2017). The goal of this web was for people to recover and keep in touch with old schoolmates, high school, university or job partners. In 1996 appeared sixdegrees.com, that allowed users to create profiles, lists of Friends and share messages. However, it was not until the 21th century when the first websites promoting ring of friends networks. In 2003 it appeared significant social networks that produced an exponentially increase of people interested in this world. We

refer to Friendster, Ecademy, Soflow, Tribe.net, MySpace y LinkedIn, in addition to other websites less known that increased to 200 the number of social networks that year.

If there is an important date in the social networks world, that is 2004, when Facebook (FAC, 2009) was created, born as an online release of American universities “facebook”. The “facebook” are publications that universities post during the opening of the school year with pictures of students and their names.

Facebook was created by Mark Zuckerberg, who was studying at Harvard at that moment. During Facebook’s first month running, it already got the subscription of more than a half of Harvard students. A year later, it had more than a million users and brought in the alumni of more than 25.000 high schools and 2.000 EEUU and foreign universities, reaching more than 11 million users. In 2007 Facebook had a greater number of registered users in comparison with other websites for high level students, having more than 19 million users all over the world.

However, the most used social network for sharing information and opinions during an event is Twitter (TWI, 2009), a microblogging service that gives its users the possibility to send small text entries called “tweets”. Twitter was born in 2006, founded by a group of young entrepreneurs that worked for Podcasts Odeo company in San Francisco. During a company meeting, Jack Dorsey introduced an idea in which SMS could be used to tell a small group of Friends what one of them was doing. After being initiated the project, there were proposed multiple names. The original name for some time was “Status”, going through “Twitch”, due to mobile vibrations, and ending up with “Twitter”.

Twitter public release was in July of 2006. Along its first year it suffered a not inconsiderable evolution, going from 20.000 tweets a day to 60.000 (EVO, 2010). In November of 2009 appeared the Spanish version of Twitter and a year later, microblogging released an app for users to translate it to Spanish, French and Italian in a non-profit way. In fact, numerous features that Twitter has introduce along its live, have been user’s initiatives. An example of them are “hashtags”, which are tags composed of key words chosen by users and preceded by “#” symbol. Also “trending topics” or the topics at the moment are set as the most used terms by users, according to the region or country where they are.

Twitter has been positioned among the world top-ten most visited websites for

most of its existence. In February of 2009 it was the third most visited social network with 55 million visits per year (USU, 2010) and an estimated growth of 1382%. In 2013, it already had 200 million users, reaching 330 million today.

Due to the impact Twitter has had on the society in the last years, we have developed a tool that manages the huge amount of data generated at any moment in any temporary event with kind of social impact. With the use of visualization techniques, we offer the user a global look over the social activity during an event. Furthermore, we work with retrieval information algorithms to show the user information that may be more relevant.

For any event the user is able to see the volume of tweets per second, identify the key moments (peaks on the graph) and, interacting with the graph, know the dominant topic at those moments.

## 2.2. Objectives

The goal of the project we propose and which we have called T.E.F.G (Temporary Event Frequency Generator), is to apply data analysis techniques in a semi-automatic way, so they can be used by users who are not necessarily experts on the matter. Moreover, we want this project to be able to quickly deal with a huge amount of information, generating visual representations and brief reports that allow to catch at a glance an event resume that used to be composed of thousands of messages.

T.E.F.G can also be useful for companies, because it is not new that nowadays information can be translated into economic profits. With our tool, companies can know better their client's interests and offer their products based on those.

In particular, T.E.F.G goals are:

- Summarize in a graphic way an event on Twitter. The graphic must show the tweets frequencies at every moment since we assume that the most relevant moments correspond to the highest frequency peaks.
- Furthermore, for each frequency peak, or in general, for every interval selected by the user, we want to be capable to offer the most significant tweets that summarize the information within the interval.

- Making use of the previous point and applying it to an event, we want to build a timeline with the most significant tweets that best summarize the event.

Moreover, under our point of view and regarding our academic training, we found interesting to perform a project that could allow us to:

- Work with a technology close to us as social networks are, specially Twitter.
- Be aware of the use that information on social networks can give us.
- Delve into Big Data technologies, especially in MongoDB data bases.
- Extend our knowledge in programming languages, Python 3 in our case, unknown by us before starting this project.
- Get used to data visualization techniques, especially with the Python's library matplotlib.
- Learn how to use retrieval information techniques. In our case, we have used TF-IDF.

## **2.3. Used technologies**

In this chapter we will describe those technologies that have allowed us to develop our application.

### **2.3.1. Twitter**

Besides being one of the most used social networks, Twitter offer us the possibility to download information without any costs, which will allow us to get the tweets generated during an event.

### **2.3.2. Python**

Is a programming language designed by Guido van Rossum at the beginning of 1991 (PYT, 2015). This language app emerged from the limitations and problems



that his creator found while working on some projects using ABC language. Nowadays, Python offers a great versatility and it is used for developing graphical interfaces, web developing, database managing, etc. Python has its own licensed denominated Python Software Foundation License, which differs from GPL license in that Python Software Foundation License is a no-copyleft license. This makes possible to modify the source code and develop derivate without create it from scratch. *open source*.

Python is a multi-paradigm programming language. This means that allows programmers to adopt different styles: object-orient programming language, imperative programming and functional programming. It uses dynamic typing and references count for memory management.

The code used in this project has been developed in its entirety with the version of Python 3.0, published in December 2008.

The other alternative language for the development of this project was Java, but it was discarded since, as described below, MongoDB has been the database with which we have worked and Python has a module called Pymongo. Although MongoDB has modules and drivers for most programming languages, this module is the most intuitive and does not require installation. It also allows you to connect the program with a MongoDB database and use all available queries and operations in Python.



Figura 2.1: Python logo

In addition to Python's own corecode, we have used several libraries that provide additional functionalities. Some of the ones we have used are the following:

- **matplotlib:** is a library that allows the visualization of graphs from data contained in lists.
- **tkinter:** Python library used to create graphical user interfaces.
- **pymongo:** module that allows the connection of any Python application with a MongoDB database.
- **csv:** is used to read the contents of a file with csv extension. In our case, we use it to know the size of the file that will be imported and thus be able to update the progress bar coherently.
- **subprocess:** Graph is a module that allows you to directly use commands from the operating system by creating a concurrent process to which you invoked it. In this project it has been used to import the data file by running the mongoimport tool.

### 2.3.3. MongoDB

It is a document-oriented NoSQL database system developed under the concept of open source (MON, 2014). It is available for Windows, Linux, OS X and Solaris operating systems. It was developed by the company MongoDB Inc. and launched on the market in 2007. MongoDB is available for free under the GNU AGPL license.

In MongoDB the information is stored internally in documents. These documents are stored in BSON, which is a binary representation of JSON. MongoDB is written in C++ and comes standard with a console to execute the commands. This console is built on JavaScript, so queries are made using this language. In addition, MongoDB offers official drivers for a large number of programming languages such as C#, Java, Node.js, PHP, Python, Ruby, C, C++ or Scala. We have used the driver that allows its connection with Python.

This data base system is present in all our project since it is the only one we use for the storage of all information.



Figura 2.2: MongoDB logo

## 2.4. Memory structure

The structure in which we have drawn up this memory follows a natural order similar to the one the user will follow when using our application.

In the chapter in which we are, we explain the motivation that has carried out this project, as well as the goals proposed at the beginning of this. In addition, it describes the technologies that we have used for the development of our work. This chapter is written in Spanish and English.

In chapter 3 we have studied the leading tools in the same area as ours. In addition, we have explained features of these applications that are relevant for us and have inspired us in our project.

In chapter 4 we detail how we have obtained the Twitter data that will form the basis of our subsequent analysis.

In chapter 5 we describe the steps followed to correctly import the data obtained from an event and create the associated collection. We also explain how this data should be prepared for the later stages.

In chapter 6 we detail the process that we have followed for the creation of the collection of frequencies once the data has been imported and prepared.

In chapter 7 we explain the input data needed to generate the graph. In addition, the functionalities that it offers to the final user are described in detail (representative tweets, additional information and zoom).

In chapter 8 the TF-IDF algorithm used is presented, in our case, to obtain the most representative tweet of an interval and a point in an event.

In chapter 9 we explain the calculations carried out for the automatic detection of peaks in a collection of frequencies and their subsequent visualization in the graph. In addition, the Timeline tool is presented, closely related to the TF-IDF algorithm and the peaks of the graph.

In chapter 10 the contributions of each one of the members of the team to the project are written. It also details the work methodology and work organization throughout the course.

Finally, in chapter 12 we explain the final conclusions of the project and the possible improvements that could be integrated to our work in the future.

# Capítulo 3

## Estado del arte

Hoy en día, la popularidad de las redes sociales es tan grande, que cada vez más gente recurre a éstas para saber lo que está pasando en el mundo, dejando atrás medios tradicionales como la televisión (Hull and Lewis, 2014). En gran medida, esto sucede gracias a que las redes sociales nos ofrecen la información actualizada en cualquier momento sin esperas. Por todo esto, la extracción o la generación de resúmenes de la información disponible en las redes sociales se ha convertido en un tema de gran interés. Sobre ello se han realizado numerosas propuestas en cuyo ámbito se encuadra nuestro proyecto. En este capítulo vamos a repasar algunos de estos trabajos.

Una de las técnicas que nos ha servido de ayuda para elegir la forma en la que realizamos el tratamiento de los datos en nuestro proyecto ha sido NED (New Event Detection) (Kumaran and Allan, 2004). Esta técnica pretende diferenciar las últimas noticias de las que surgieron momentos justo después de un evento a partir de un umbral predefinido. Tomando como ejemplo un accidente aéreo, las primeras noticias hablarán del número de víctimas o del lugar del suceso sin entrar en demasiados detalles. Días después surgirán noticias que tratarán sobre las posibles causas del accidente o responsabilidades, dando una información mucho más detallada.

NED utiliza la métrica de similitud del coseno, que se aplica a los vectores de términos obtenidos tras calcular el TF-IDF de varios documentos. Esta métrica es utilizada en nuestro proyecto para la obtención de términos relevantes y se

explicará en capítulos posteriores. A continuación, haremos una exposición de los proyectos que, como el nuestro, realizan detección de eventos, análisis de noticias y procesamiento de la información en redes sociales.

### 3.1. Hotstream

Hotstream (Phuvipadawat and Murata, 2010) es una aplicación que genera a tiempo real noticias de última hora mediante el análisis de un conjunto de tweets bajo un mismo *hashtag*.

Este análisis se realiza a través de tres etapas:

- **Recolección de mensajes:** en esta etapa se utiliza la API de Twitter mencionada en el capítulo de “Captura de los datos”.
- **Indexación:** Hotstream utiliza Apache Lucene para construir un índice sobre los mensajes.
- **Agrupación:** agrupa mensajes similares para construir la noticia. Para comprobar a qué grupo está destinado cada mensaje utiliza, al igual que nosotros, el algoritmo TF-IDF que filtra los mensajes con unos *hashtags* predefinidos. Sin embargo, introduce una modificación a este algoritmo para darle mayor importancia a los sustantivos, ya que son los que aportan la mayor parte de información en una noticia. En nuestro caso, usamos una variante del algoritmo que incluye un diccionario con las *stop-words* de cada idioma, entendiendo como *stop-words* las palabras sin significado como artículos, pronombres, proposiciones, etc.

Una vez conformados los grupos, construye una noticia por cada grupo que engloba toda la información contenida en los mensajes de dicho grupo.

Algo que nos ha llamado la atención de esta aplicación es que todo lo descrito anteriormente se realiza en tiempo real teniendo que lidiar con los tiempos de procesamiento de las tres etapas descritas, especialmente con el cálculo de similitudes.

Una de las grandes diferencias con nuestro trabajo es que T.E.F.G trabaja en modo *offline* para poder soportar un mayor nivel de procesamiento. Además, los

resultados de T.E.F.G se obtienen a partir de un análisis de frecuencias, un medio sencillo para detectar momentos de mayor interés en un evento.

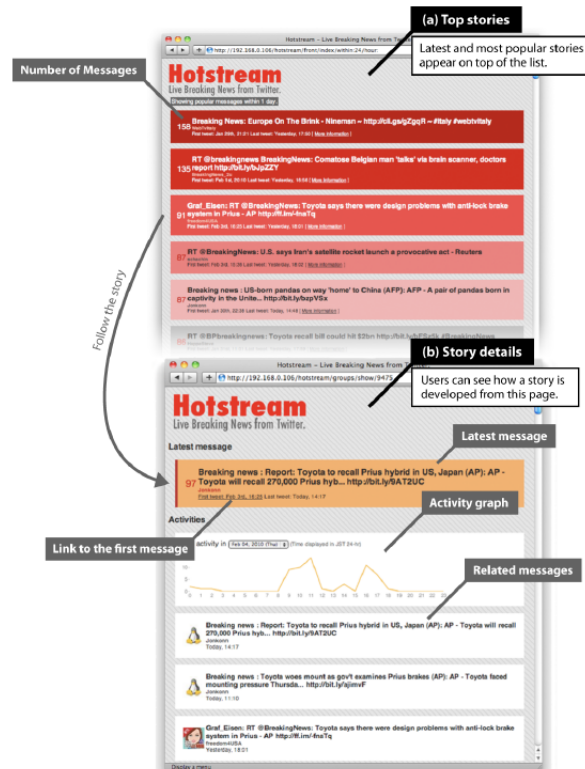


Figura 3.1: Interfaz de Hotstream

## 3.2. TwitterMonitor

Esta herramienta (Mathioudakis and Koudas, 2010) muestra a tiempo real los temas de los que hay un mayor tráfico de tweets para que el usuario pueda opinar sobre ellos. Además, presenta cada tema con varios tweets para proporcionar una información más detallada.

Cada tweet lleva asociada la hora de publicación. Gracias a este campo la herramienta detecta automáticamente los temas más hablados. T.E.F.G utiliza la hora de publicación para mostrar en una gráfica la evolución del tráfico de tweets de un evento en el tiempo.

Además, esta aplicación tiene en cuenta el origen geográfico de cada oleada de

términos para realizar la agrupación. En nuestro caso, en lugar de la localización geográfica, tenemos en cuenta el idioma de cada tweet. Por ejemplo, si en un partido España-Alemania hay una gran oleada de tweets en español, podremos deducir que probablemente en ese momento ha habido un gol de España.

Por último, uno de los aspectos más llamativos es que TwitterMonitor es capaz de detectar oleadas de palabras y asignarles un mismo tema para no considerarlo varias veces. Por ejemplo, las palabras portería, gol, falta o penalti se pueden incluir en un mismo evento como es un partido de fútbol. Todo esto lo hace agrupando las palabras del mismo tema en conjuntos disjuntos.

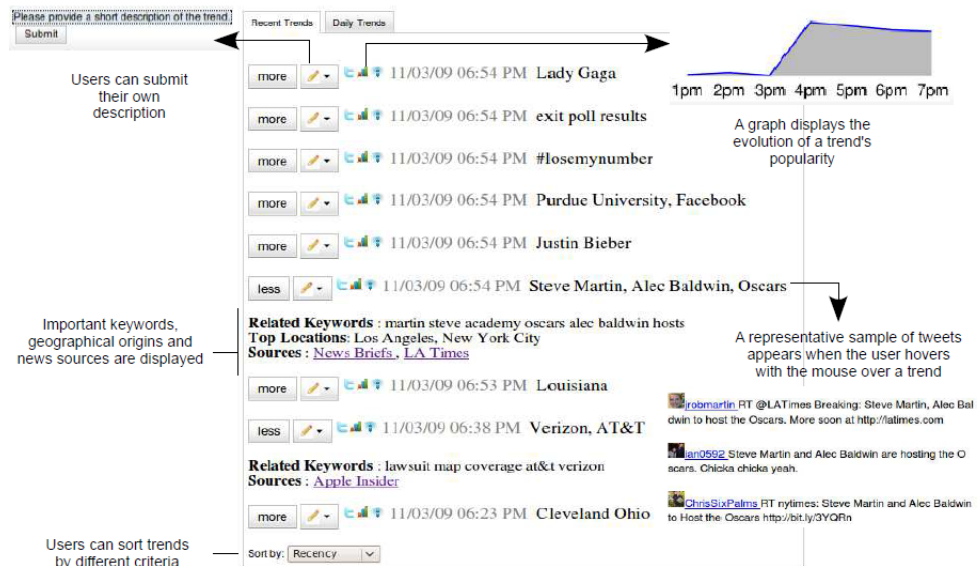


Figura 3.2: Interfaz de TwitterMonitor

### 3.3. Buzzer

Buzzer (Phelan et al., 2009) es una aplicación de recomendación de temas según los gustos indicados previamente por el usuario.

Esta herramienta, combina la información del RSS dado por el usuario junto con los tweets relacionados con el tema. RSS es un sistema de difusión de información actualizada de la web que llega a los usuarios suscritos a un tema. Para llevar esto a cabo, Buzzer se vale de tres componentes:



- La aplicación web para el proceso de registro, donde el usuario proporciona información de su cuenta de Twitter y una lista de RSS de su interés.
- El indexador Lucene, responsable de minar e indexar la información de Twitter y los RSS según la configuración del usuario.
- El recomendador genera una lista de puntuaciones de temas de RSS basados en las apariciones de términos dentro de los índices del Twitter y RSS del usuario.

Para seleccionar el artículo del RSS con mayor relevancia para el usuario, calcula la suma de las puntuaciones obtenidas del TF-IDF de todos los términos asociados al artículo. El sistema selecciona los artículos que contienen un mayor número de términos con un TF-IDF elevado. Este proceso es muy similar al que nosotros seguimos, de nuevo con la diferencia de que no se basa en la frecuencia de ocurrencia de tweets, que constituye nuestro primer filtro para la detección de noticias.

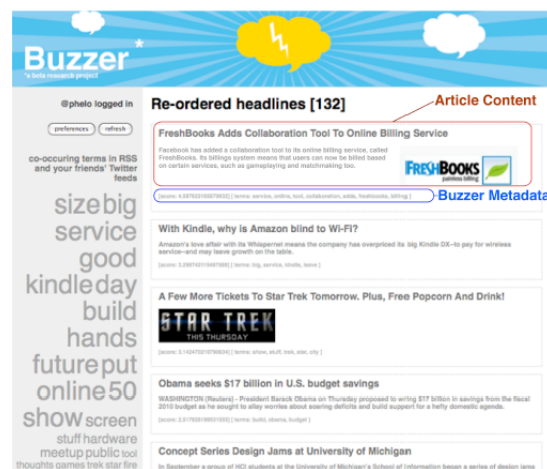


Figura 3.3: Interfaz de Buzzer

## 3.4. TwiCal

TwiCal (Ritter et al., 2012) pretende solventar el problema de la gran cantidad de datos generados en Twitter intentando extraer la información más valiosa y categorizándola para definir eventos. Para ello, extrae una tupla de cuatro columnas

por evento: entidad, evento, fecha y tipo. Para construir esta tupla, los tweets son etiquetados para extraer la entidad, definir el evento y clasificarlo. Cuando ocurre un evento importante, el tráfico de tweets sobre el mismo aumenta considerablemente. TwiCal relaciona este tráfico con la entidad para comprobar si se trata de un evento significativo.

Entity	Event Phrase	Date	Type
Steve Jobs	died	10/6/11	DEATH
iPhone	announcement	10/4/11	PRODUCTLAUNCH
GOP	debate	9/7/11	POLITICALEVENT
Amanda Knox	verdict	10/3/11	TRIAL

Figura 3.4: Ejemplo de tupla extraída por TwiCal

Para pulir el proceso de extracción y conseguir el menor ruido posible, TwiCal realiza un entrenamiento antes del etiquetado de las entidades. En nuestro caso, también realizamos un entrenamiento con el conjunto de tweets del evento del que se extraerá el más representativo.

A la hora de identificar un evento importante, TwiCal descarta los eventos sobre los que, aunque se hable durante un largo periodo de tiempo, no experimentan ninguna crecida sustancial. Por ejemplo, durante todo el año, miles de personas acuden a restaurantes de comida rápida publicándolo en las redes sociales. Sin embargo, esto no implica que haya sucedido un acontecimiento relevante sobre los restaurantes de comida rápida.

# Capítulo 4

## Captura de los datos

El primer paso en cualquier trabajo de análisis de datos es la recolección de los mismos. En este capítulo explicaremos cómo realizamos la captura de los datos de Twitter a tiempo real para tratarlos posteriormente en modo *offline*.

Para recolectar los tweets asociados a un evento concreto y generar la correspondiente base de datos, hemos empleado una herramienta de escucha a tiempo real proporcionada por el profesor Enrique Martín Martín. Con este programa, podemos descargar tweets a tiempo real filtrándolos con una serie de palabras. En nuestro caso, para este filtrado utilizaremos *hashtags* indicados antes de la ejecución. Un hashtag no es más que un término precedido por el símbolo ‘#’ en un tweet que utilizan los usuarios de esta red social para agrupar los tweets de un mismo tema.

Para que el programa de descarga pueda acceder a la información generada en Twitter, ha sido necesaria la creación de una cuenta de desarrollador en Twitter Apps utilizando nuestra cuenta personal de Twitter. Se trata de una plataforma para desarrolladores que ofrece herramientas para acceder a más información que con una cuenta de usuario corriente. Una vez creada la cuenta, tenemos disponibles unas claves que debemos facilitar a la herramienta de descarga. Estas claves son las siguientes:

- **CONSUMER TOKEN:** identifica al cliente, siendo éste un servicio/web que intenta acceder a los recursos del usuario final.

- **CONSUMER SECRET:** es la clave del cliente utilizada para autenticarse en los servidores. Un servidor de Twitter es el que identifica al cliente.
- **ACCESS TOKEN:** se emite al cliente una vez que se identifica correctamente para definir sus privilegios y así saber a qué datos puede acceder.
- **ACCESS TOKEN SECRET:** cuando el cliente desea acceder a los datos del usuario final, esta clave es generada y se envía junto con la anterior.

Es importante destacar que en Twitter podemos encontrar dos tipos de cuentas. Por defecto, los tweets publicados en una cuenta son visibles a todos los usuarios. Esto es lo que se conoce como una cuenta pública. No obstante, todo el contenido se puede privatizar y limitar su alcance a un grupo de personas determinadas.

La herramienta de descarga sólo tiene acceso a los tweets de las cuentas públicas y a los de los seguidores de la cuenta con la que se haya creado la aplicación en Twitter Apps.

Antes de pasar a explicar el preprocesamiento de los datos, cabe destacar que nuestra herramienta no incluye la escucha, es decir, parte de un fichero de entrada ya descargado. Por ello, el fichero puede haber sido obtenido con la herramienta de escucha aquí explicada o con cualquier otra.

Independientemente de la herramienta de escucha utilizada, T.E.F.G puede procesar cualquier archivo que esté en formato CSV o JSON.

JSON
<pre>{ "user": 793418853489152672, "created_at": "Tue Mar 06 19:44:52 +0000 2018", "lang": "en", "text": "Paris Saint-Germain v Real Madrid: Champions League \u2013 \u00a0\u00a0live https://t.co/AZFoC5srSf" }</pre>
CSV
<pre>user,created_at,lang,text 793418854462590977,2016-11-01T11:46:10.000Z,en,"RT @nia4-trump: Putin isn't 'Rigging' US elections, the #DNC is! @HillaryClinton knows this, we ALL do. But blame The Russians &amp; st\u00e4f!"</pre>

Figura 4.1: Diferencias entre los formatos JSON y CSV con los mismos campos

Estas son todas las operaciones que se realizan sobre los datos antes del uso de nuestra herramienta. En capítulos posteriores se explicará el resto del procesamiento de datos tras la importación.

# Capítulo 5

## Importación de los datos

Cuando ya disponemos de un fichero en un formato admitido por nuestra herramienta (JSON o CSV), se puede proceder a la importación de los datos. En este capítulo vamos a ver de forma detallada la importación y las operaciones que realizamos inmediatamente después.

### 5.1. Requisitos previos

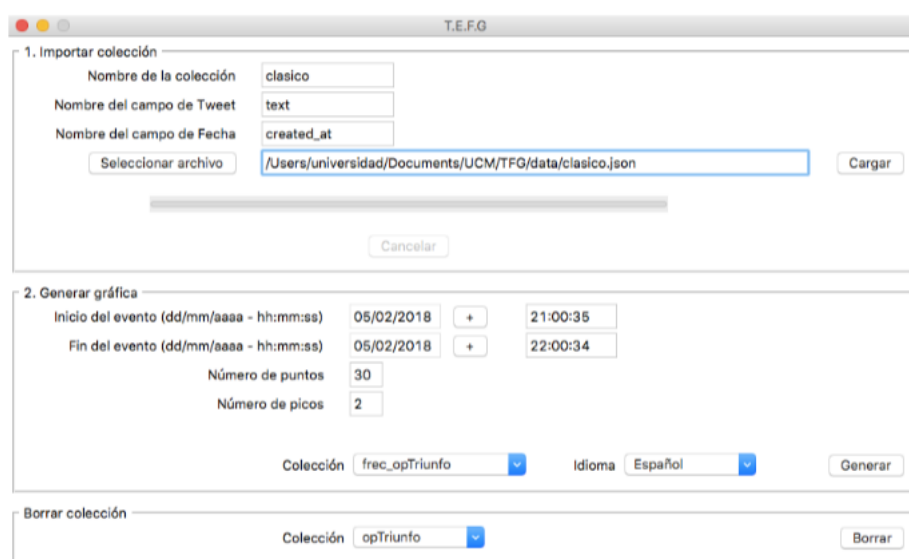
Como nuestra aplicación trabaja en modo *offline*, no es necesario disponer de conexión a internet.

El primer requisito es tener MongoDB instalado en nuestro sistema. En la instalación de MongoDB vienen incluidas numerosas herramientas que proporcionan funcionalidades extra. La que utilizamos en nuestro proyecto es *mongoimport*, como explicamos más adelante.

Por último, cuando MongoDB ya está instalado, es necesario tener iniciado un servidor Mongo. Al iniciarlo, hay que indicar el directorio donde se almacenarán las bases de datos que se crearán posteriormente.

## 5.2. Pantalla de importación

Lo primero que verá el usuario al iniciar nuestro programa es la interfaz mostrada en la imagen. La parte correspondiente al proceso de importación es la marcada como “1. Importar colección”. Antes de cargar un nuevo archivo, el usuario debe indicar el nombre de la colección que se creará en MongoDB y los campos asociados al texto del tweet y la fecha de publicación. El uso de estos campos lo explicaremos en capítulos posteriores.



The screenshot shows the T.E.F.G. application window. It is divided into three main sections:

- 1. Importar colección:** This section contains three input fields: "Nombre de la colección" (set to "clasico"), "Nombre del campo de Tweet" (set to "text"), and "Nombre del campo de Fecha" (set to "created\_at"). Below these is a "Seleccionar archivo" button and a text field containing the file path "/Users/universidad/Documents/UCM/TFG/data/clasico.json". To the right is a "Cargar" button. A progress bar is located below the file path.
- 2. Generar gráfica:** This section contains two rows of date and time pickers. The first row is for "Inicio del evento" (05/02/2018, 21:00:35) and the second for "Fin del evento" (05/02/2018, 22:00:34). Below these are input fields for "Número de puntos" (30) and "Número de picos" (2). At the bottom are dropdown menus for "Colección" (set to "frec\_opTriunfo") and "Idioma" (set to "Español"), followed by a "Generar" button.
- Borrar colección:** This section at the bottom has a dropdown menu for "Colección" (set to "opTriunfo") and a "Borrar" button.

Figura 5.1: Pantalla inicial de T.E.F.G

Para importar un nuevo documento, se deberá seleccionar el archivo y pulsar sobre “Cargar”. Junto a la barra de progreso, se indica la fase por la que está pasando la importación. Existen varias fases que se explican a continuación.

### 5.2.1. Importación a la base de datos

En una primera versión de nuestro programa, únicamente contemplábamos la importación de ficheros en formato CSV. Por ello empleamos la función *reader()* del módulo CSV de Python (CSV, 2012). Posteriormente descartamos su uso, ya que generaba problemas para importar archivos de gran tamaño o archivos cuyas filas tenían campos distintos, algo que sí permite MongoDB.

Debido a estos problemas, y a que pensamos en incluir también los archivos en formato JSON, decidimos utilizar la herramienta *mongoimport* incluida en la instalación de MongoDB. Cabe señalar que el usuario no hace uso directamente de esta herramienta. Simplemente debe pulsar el botón de “Cargar” en la interfaz. Cuando sucede esto, creamos un subproceso que ejecuta *mongoimport*. De esta forma, la carga de información se realiza de forma más rápida e independiente del proceso principal. Una vez concluida esta fase, se habrá creado una colección en MongoDB con el nombre indicado por el usuario.

En la siguiente imagen podemos ver la sentencia que ejecuta la herramienta citada:

```
subprocess.Popen(['mongoimport', '-d', 'tfg', '-c', collName.get(), '--file', pathField.get()])
```

Figura 5.2: Ejecución de *mongoimport* con la creación de un subproceso

Los campos incluidos son los siguientes:

- *tfg*: base de datos donde se creará la colección.
- *collName.get()*: nombre de la colección introducido por el usuario.
- *pathField.get()*: ruta donde se encuentra el archivo que va a ser importado. Este dato también es indicado por el usuario.

### 5.2.2. Preparación de fechas

En la segunda fase de la importación, realizamos dos operaciones en relación con la fecha de cada tweet. En primer lugar, se traduce cada fecha a formato ISO. Este paso únicamente se realiza en los archivos con formato CSV, ya que los archivos JSON ya disponen de la fecha en formato ISO.

El formato ISO DATE (ISO 8601) (ISO, 2018) es un formato internacional definido por la Organización Internacional de Normalización (ISO) que define un sistema de fechas aceptado mundialmente. Con este formato se puede representar una fecha y hora concreta e incluye información de la zona horaria.

Esta transformación nos permitirá en el resto del programa tratar la fecha de forma cómoda y rápida, ya que tanto Python como Mongo disponen de funciones específicas para tratar el tipo ISO DATE, permitiendo acceder al año, mes, día, hora, minuto, segundo y zona horaria. También permiten realizar operaciones aritméticas sobre las fechas. Por otro lado, disponer de un formato único es importante a la hora de mostrar información coherente al usuario.

En segundo lugar, realizamos otra operación que consiste en añadir un campo a la colección que hemos denominado “epoch”. Este campo, contiene la anterior fecha ISO convertida a formato Unix.

El tiempo Unix (EPO, 2017) es un sistema de representación de instantes de tiempo que se define como la cantidad de segundos transcurridos desde la medianoche del 1 de enero de 1970.

En nuestro caso, y para facilitar los cálculos, hemos tomado como referencia el 1 de enero de 2000. Más adelante veremos la gran importancia de disponer de la fecha en un tipo entero para tratar la información, ya que muchos cálculos han empleado este campo, aunque el usuario vea la fecha en un formato legible.

Tras estas fases, los elementos de la colección importada incluyen los campos *text* (texto del tweet), *created\_at* (fecha de publicación en formato ISO) y *epoch* (fecha en tiempo Unix), además de información sobre el usuario que publicó el tweet.



```

"_id": ObjectId("5ab2b668caac62bea08c6doc"),
"created_at": ISODate("2017-12-23T13:07:13Z"),
"id": NumberLong("944554991745945600"),
"text": "Really woke up at 7 am just to watch the game, Hala Madrid! #ElClasico #HalaMadrid #HastaElFinal",
"user": {
  "id": NumberLong("822848451268448256"),
  "name": "Brown Alpaca",
  "screen_name": "_BrownAlpaca_",
  "location": "Parts Unknown",
  "description": "I make silly sounds with my bass",
  "verified": false,
  "followers_count": 21,
  "friends_count": 96,
  "favourites_count": 1254,
  "created_at": "Sat Jan 21 16:48:53 +0000 2017",
  "time_zone": "Eastern Time (US & Canada)",
  "geo_enabled": true,
},
"retweet_count": 0,
"favorite_count": 0,
"entities": {
  "hashtags": [
    {
      "text": "ElClasico",
      "indices": [
        60,
        70
      ]
    }
  ],
},
"favorited": false,
"retweeted": false,
"lang": "en",
"epoch": 64674450

```

Figura 5.3: Fragmento de la colección importada



# Capítulo 6

## Cálculo de frecuencias

En este capítulo explicamos cómo generamos las frecuencias de tweets en un evento cuando ya tenemos la colección importada y cómo organizamos toda la información en nuestra base de datos.

### 6.1. Colección de frecuencias

En primer lugar, mediante una consulta de Mongo en la que se realiza una agrupación por el campo de fecha de la colección anterior, creamos una nueva colección que almacena, para cada segundo del intervalo del evento, el número de tweets que ha habido junto con la fecha completa en un formato legible. Cabe mencionar, que hemos definido como fecha legible aquella que representa con dos dígitos el día, el mes, la hora, el minuto y el segundo; y con cuatro dígitos el año. De esta forma, conseguimos una representación genérica en el entorno. Además, los campos de la fecha se almacenan en la nueva colección por separado, de forma que podemos acceder a cada componente sin necesidad de utilizar de nuevo las funciones de descomposición de fechas.

Por último, calculamos el total de tweets acumulados hasta cada segundo desde el inicio del evento (campo "acumulado"). El objetivo de añadir este campo es poder calcular fácilmente datos sobre intervalos. En particular, dado un segundo de inicio y de fin del intervalo, basta con restar los valores de este campo para conocer el total de tweets del intervalo y con este dato, dividiendo por el número

de segundos, obtenemos la media de tweets, algo que empleamos de forma intensiva para la realización de las gráficas.

En versiones iniciales de nuestra aplicación, decidimos contabilizar los tweets por minuto, pero esto podía suponer una pérdida de información. Nuestra herramienta está pensada para eventos con un alto volumen de tweets en un tiempo muy reducido, por ejemplo, un gol en un partido de futbol, el anuncio del ganador en un concurso televisivo, el presidente elegido en las elecciones, etc.

Tras estos pasos, ya está creada la nueva colección, que tendrá tantos documentos como segundos haya durado el evento. En la siguiente imagen se puede apreciar el formato y los campos de dicha colección:

```
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 12 }, "total": 20, "epoch": 64676609, "acumulado": 68421 }  
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 13 }, "total": 29, "epoch": 64676610, "acumulado": 68450 }  
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 14 }, "total": 20, "epoch": 64676611, "acumulado": 68470 }  
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 15 }, "total": 23, "epoch": 64676612, "acumulado": 68493 }  
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 16 }, "total": 25, "epoch": 64676613, "acumulado": 68518 }  
{ "_id": { "dia": 23, "mes": 12, "año": 2017, "hora": 13, "minuto": 43, "segundo": 17 }, "total": 32, "epoch": 64676614, "acumulado": 68550 }
```

Figura 6.1: Fragmento de la colección de frecuencias

Por otro lado, creamos otra colección de carácter informativo. En ella almacenamos los tres idiomas más hablados durante el evento para, más adelante, aplicar el TF-IDF según el idioma elegido por el usuario. También almacenamos la fecha y hora iniciales y finales del evento en un formato legible para definir los límites de la gráfica que se mostrará.

# Capítulo 7

## Representación gráfica

En este capítulo vamos a hablar sobre cómo hemos pasado de la información almacenada en la base de datos a la visualización de la misma utilizando principalmente las bibliotecas de Python *matplotlib* y *tkinter*. Además, vamos a detallar el proceso mediante el cual el usuario final es capaz de interactuar con los datos mostrados y cuáles son las opciones que tiene para visualizarlos.

### 7.1. Datos de entrada

Desde la interfaz, se puede generar la gráfica de frecuencias de dos formas. La primera es importando una nueva colección que, tras completarse, mostrará automáticamente la gráfica de toda la duración del evento. La segunda permite al usuario seleccionar una de las colecciones importada previamente. Obviamente, esta segunda posibilidad es mucho más rápida, ya que la importación se realizó la primera vez que se utilizó el conjunto de datos. En este caso, no es necesario visualizar todo el evento. El usuario tiene la posibilidad de seleccionar parte del mismo para representar en la gráfica.

Si el usuario opta por la segunda opción, debe indicar el intervalo de fecha y hora en la parte de la pantalla inicial marcada como "2. Generar gráfica", tal y como se puede ver en la siguiente imagen:

The image shows a web application interface for generating a graph. It consists of several sections:

- Fecha inicial:** A date picker showing May 2018.
- T.E.F.G:** A header or title.
- File Upload:** A text input field containing a file path, a 'Cargar' button, and a progress bar.
- 2. Generar gráfica:**
  - Inicio del evento (dd/mm/aaaa - hh:mm:ss):** 23/12/2017, 13:03:59
  - Fin del evento (dd/mm/aaaa - hh:mm:ss):** 23/12/2017, 13:51:32
  - Número de puntos:** 30
  - Número de picos:** 2
  - Colección:** frec\_clasico2
  - Idioma:** Inglés
  - Generar:** Button
- Borrar colección:**
  - Colección:** miniClasico
  - Borrar:** Button

Figura 7.1: Datos de entrada para la gráfica de una colección importada

En cualquier caso, los datos de entrada comunes para las dos formas de generación de la gráfica son: número de puntos de la gráfica y número de picos a resaltar en la misma.

De aquí en adelante, definiremos como intervalo la porción de tiempo del evento comprendido en la gráfica mostrada, y como subintervalo cada uno de los tramos de tiempo comprendidos entre dos puntos de la gráfica.

Con los datos iniciales definidos y para obtener una representación gráfica con el número de puntos deseado, calculamos la duración de los subintervalos en los que se puede dividir el intervalo de tiempo introducido por el usuario a través de la siguiente fórmula:

$$\text{duración subintervalos} = (\text{instante final} - \text{instante inicial}) / \text{nº de puntos}$$

Después, calculamos el valor del segundo que corresponde a la mitad del subintervalo, es decir, el instante que deja la mitad de los tweets del subintervalo a cada lado. Este segundo marcará el valor  $x$  de un punto de la gráfica. Su valor en el eje  $y$  corresponderá a la media del número de tweets del subintervalo.

El tema de la representación gráfica de una función compleja ha recibido un estudio muy amplio (Last et al., 2004) (capítulo 1). Nosotros nos limitamos a tomar los valores medios por subintervalos para asegurar una gráfica legible que, sin embargo, no pierda información relevante. De todas formas, y para paliar una posible pérdida de información, hemos incluido la posibilidad de hacer *zoom* en la gráfica, como veremos en seguida.

## 7.2. Funcionalidades de la gráfica

En esta sección vamos a detallar las posibilidades que tiene el usuario de interactuar con la gráfica del evento y la información que se mostrará en la misma.

### 7.2.1. Tweets informativos

Una de las funcionalidades más importantes de nuestra herramienta son los tweets representativos. Entendemos por tweet representativo aquel que mejor resume lo que ha sucedido en un determinado momento. Por ejemplo, en el ámbito de un partido de fútbol, un tweet representativo del momento en el que se ha marcado un gol será aquel que informe sobre el autor del gol, el equipo al que pertenece y el resultado del marcador.

Nuestra herramienta muestra los tweets informativos en dos ocasiones: cuando se obtiene la gráfica de un intervalo concreto y cuando el usuario hace *click* en algún punto de la gráfica. Estos se obtienen mediante un análisis empleando el algoritmo TF-IDF que detallaremos más adelante.

En ambas ocasiones, únicamente se analizarán los tweets escritos en el idioma que ha indicado el usuario al generar la gráfica. Si el usuario acaba de importar la colección, por defecto se analizarán los tweets escritos en el idioma más hablado. Además, seleccionando la opción "Todos" en el menú desplegable, se analizarán todos los tweets independientemente de su idioma.

### 7.2.1.1. Tweet en un intervalo

Cada vez que se muestra una gráfica, se mostrará también el tweet representativo de todo el intervalo de dicha gráfica. Para obtener este tweet, escogemos 500 tweets aleatorios de todo el intervalo. No es posible contemplar todos los tweets, ya que esto supondría un tiempo de cálculo no asumible por el usuario. La selección aleatoria se realiza para contemplar los tweets de todo el evento y disminuir la posibilidad de perder información de algún momento clave.

En la siguiente figura, podemos ver en el cuadro de color amarillo el tweet representativo del evento. El resto de información mostrada se explicará más adelante.

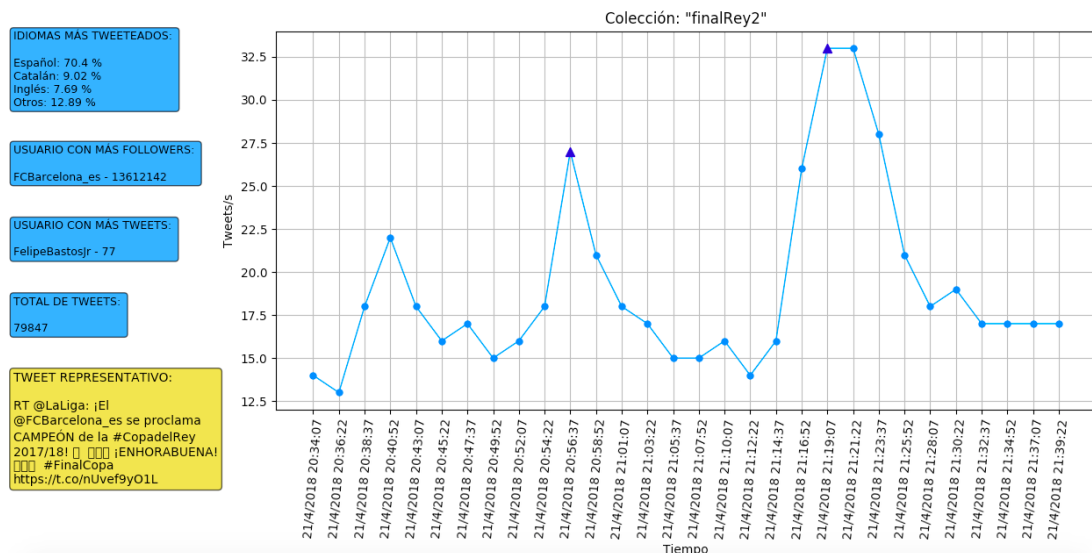


Figura 7.2: Tweet representativo del evento

### 7.2.1.2. Tweet en un punto

Es posible que el usuario desee saber qué ha ocurrido en un determinado momento del evento, ya sea por la existencia de un máximo o un mínimo local en la gráfica.

Para obtener un tweet representativo en un punto de la gráfica, es necesaria la



interacción del usuario con ésta. Para ello basta con que el usuario haga *click* en el punto deseado.

Para este caso no nos limitamos a mostrar el tweet del punto donde el usuario ha hecho *click*, sino que tenemos en cuenta 200 tweets alrededor de ese instante (los 100 anteriores y los 100 posteriores). Hemos decidido realizarlo de esta forma para, no sólo tener en cuenta los tweets en un segundo, sino tener un abanico más amplio de tweets que analizar y así mostrar aquel que ofrezca más información sobre el suceso.

En la siguiente figura, si el usuario hace *click* en el punto marcado en rojo, se actualiza el tweet representativo anterior por el del punto en el que se ha hecho *click*. Este caso pertenece a un partido de fútbol y es un buen ejemplo porque se puede ver cómo en un instante en el que ha subido el tráfico de tweets ha habido un gol.

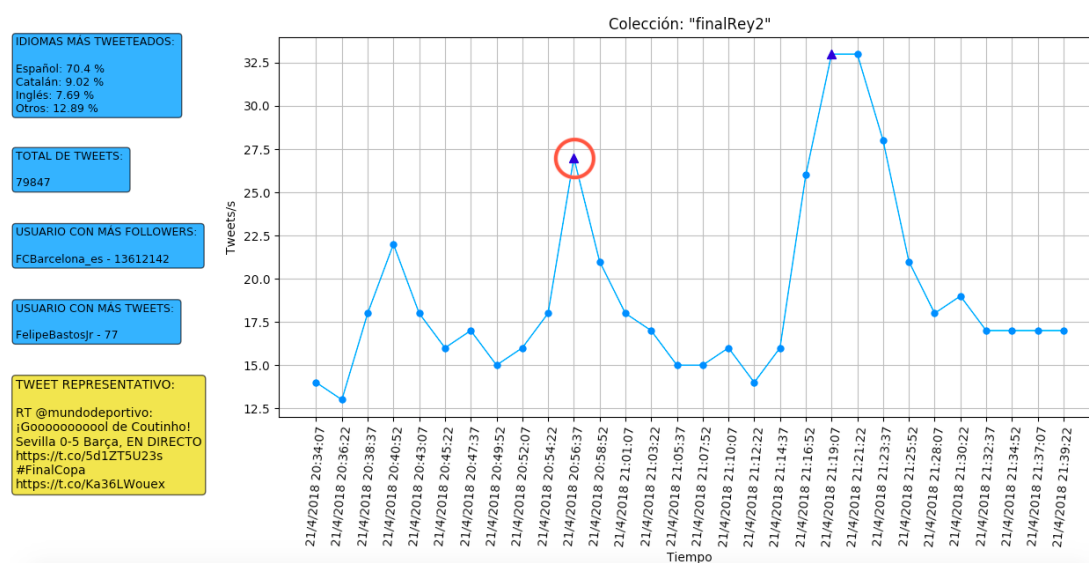


Figura 7.3: Tweet representativo de un punto de la gráfica

## 7.2.2. Información complementaria

Como se ha visto en las Figuras 7.2 y 7.3, se muestra más información a parte del tweet representativo.

En el primer cuadro azul se informa sobre los idiomas más utilizados en los tweets del intervalo. Hemos podido conocer estos idiomas gracias al campo "lang" de los tweets descargados. Cuando la colección ya está importada, ejecutamos una consulta de MongoDB para obtener los tres idiomas más usados. Este dato puede aportar mucha información. Por ejemplo, en un evento internacional como el festival de Eurovisión, si en un determinado intervalo hay una gran porcentaje de tweets en polaco, puede deberse a que haya actuado Polonia en ese momento, que haya recibido una buena puntuación o incluso que se haya proclamado ganadora.

En los dos siguientes cuadros se indica, respectivamente, el usuario con más seguidores y el que más tweets ha publicado, siempre contemplando el intervalo de la gráfica.

En el último cuadro azul se muestra el total de tweets publicados en el intervalo. Este dato aporta información sobre la repercusión que puede haber tenido el evento en las redes sociales.

### 7.2.3. Zoom

El *zoom* que hemos implementado en T.E.F.G no consiste simplemente en un *zoom* gráfico que aumente o disminuya la resolución de la gráfica, sino que supone un recálculo de todos los parámetros: subintervalos, máximos, mínimos, tweets representativos, etc. Por ejemplo, si en un momento del evento hay un elevado tráfico de tweets, puede ser útil saber qué ha pasado en ese suceso segundo a segundo.

Para que el usuario haga *zoom*, simplemente tiene que seleccionar un área dentro de la gráfica. Entonces, se creará una nueva gráfica del intervalo seleccionado volviéndose a calcular todos los datos explicados en los puntos 7.2.1 y 7.2.2.

Como la nueva gráfica tiene un intervalo distinto, también se recalcula la duración de los subintervalos en función del número de puntos que el usuario ha indicado al inicio.

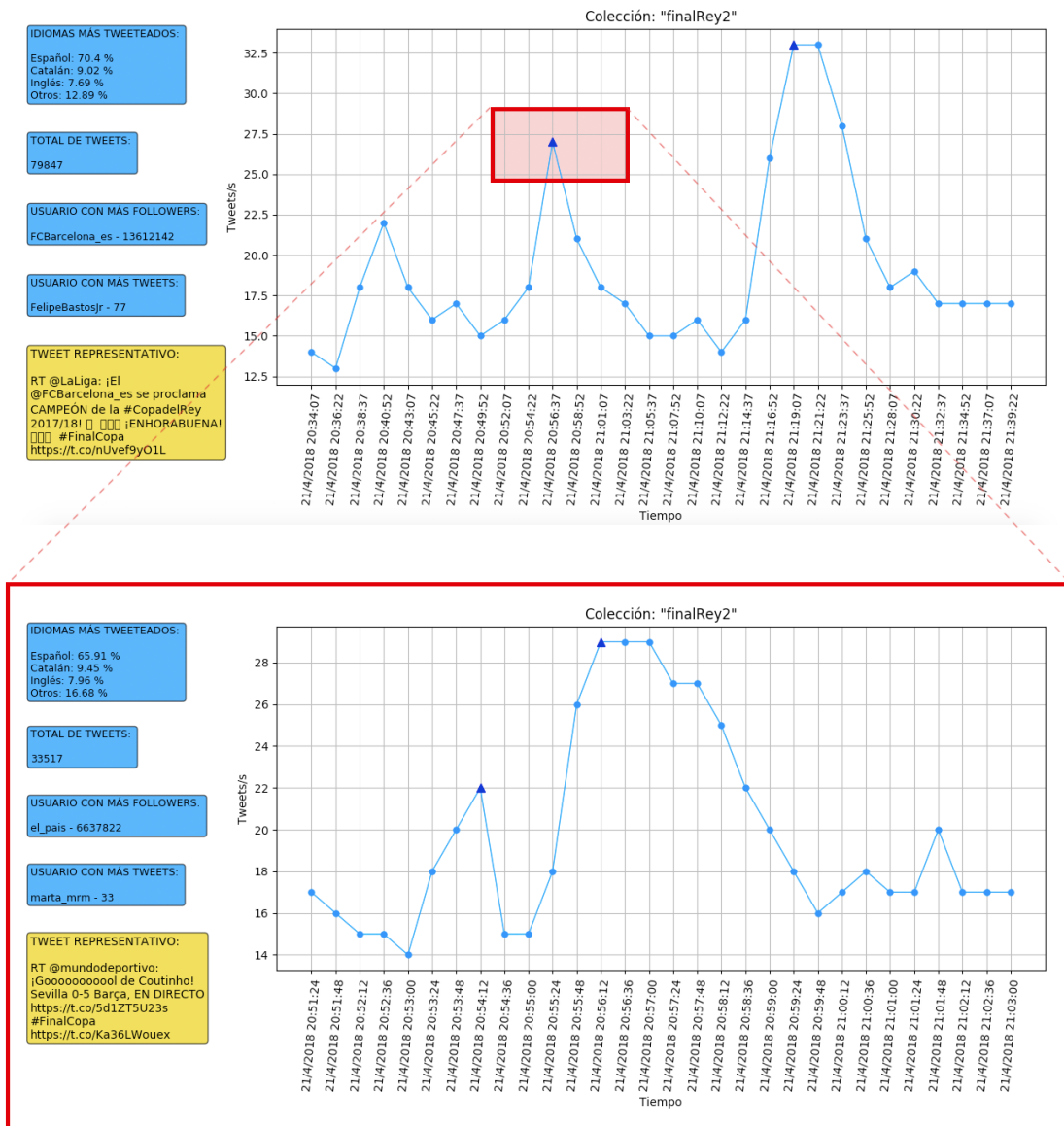


Figura 7.4: Zoom aplicado sobre una gráfica y nueva gráfica generada

La imagen anterior es un ejemplo de aplicación de *zoom*. En ella podemos observar que, al tener un intervalo menor, la duración de los subintervalos en la nueva gráfica también disminuye. En la primera gráfica los subintervalos son de 135 segundos mientras que en la segunda se reducen a 24 segundos.



# Capítulo 8

## TF-IDF

En este capítulo explicaremos el uso del algoritmo TF-IDF para obtener el tweet más representativo dentro de un conjunto, una vez que ya tenemos seleccionados los tweets que deseamos analizar.

La ponderación de términos es el proceso mediante el que se puede conocer la importancia de un término dentro de un documento. En nuestro caso, lo aplicaremos para conocer la importancia de un tweet dentro de un conjunto de tweets. TF-IDF es una técnica de recuperación de información que se basa en esta ponderación para obtener un documento que ofrezca la mayor información posible sobre todo el conjunto de documentos.

La técnica de TF-IDF está formada por dos fases (TFI, 2012): el cálculo de la frecuencia de aparición de un término (TF) y de la frecuencia inversa del documento para un término (IDF).

### TF: Frecuencia de aparición de un término

El factor TF es el número de veces que un término se repite en un documento, lo que permite determinar su capacidad de representación. Antes del cálculo es preciso realizar un filtrado del texto para eliminar las *stop-words* correspondientes al idioma elegido por el usuario para mostrar el tweet. Las *stop-words* son palabras sin significado como artículos, pronombres, preposiciones, etc. El nacimiento de

este término se atribuye a Hans Peter Luhn (STO, 2017), uno de los pioneros en recuperación de la información.

Como resultado de esta fase, se obtiene un listado con cada uno de los términos de un documento y su frecuencia de aparición en el mismo.

## IDF: Frecuencia inversa del documento para un término

El factor IDF de un término es el coeficiente que determina la capacidad discriminatoria del término de un documento con respecto a la colección. Este factor es inversamente proporcional al número de documentos en los que aparece el término. Por ejemplo, si disponemos de tres documentos cuyo contenido es "Mar Cantábrico", "Mar Mediterráneo" y "Mar Negro", el término "Mar" es tan común, que provocará que se destaquen incorrectamente documentos que utilizan ese término con más frecuencia, sin conceder suficiente peso a los demás términos. Por lo tanto, su IDF será menor.

En definitiva, este factor atenúa el peso de los términos que ocurren con mucha frecuencia en la colección de documentos e incrementa el peso de los términos que aparecen pocas veces (IDF, 2017).

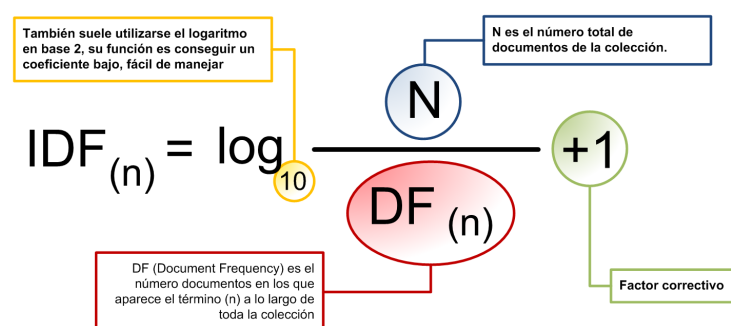


Figura 8.1: Fórmula para el cálculo del factor IDF

## 8.1. TF-IDF en Python

A continuación, procederemos a explicar cómo hemos aplicado el algoritmo de TF-IDF en Python para obtener el tweet más representativo de un conjunto. Tomaremos como ejemplo cuatro documentos cuyo contenido es: "hola", "adios", "barco" y "hola que tal estas". En primer lugar, descartamos las *stop words* que, en este caso, son "que" y "tal". Tras un proceso de entrenamiento, obtenemos una matriz como la siguiente con el valor del TF-IDF de cada palabra en cada documento:

	<b>adios</b>	<b>barco</b>	<b>hola</b>	<b>estas</b>
<b>hola</b>	0	0	1	0
<b>adios</b>	1	0	0	0
<b>barco</b>	0	1	0	0
<b>hola que tal estas</b>	0	0	0.619	0.785

Este es un ejemplo sencillo en el que cada fila de la tabla corresponde a un tweet. Cada columna es una palabra distinta de todas las que existen en el conjunto de tweets. Podemos observar que la palabra "hola" es más significativa en el primer documento que en el último. En el primer documento no hay más palabras, por lo que "hola" representa a la totalidad del documento, mientras que en el cuarto documento, al haber más palabras, "hola" pierde relevancia.

El TF-IDF de un tweet es el conjunto de los valores de TF-IDF de cada una de las palabras que lo forman. Por ejemplo, el TF-IDF del tweet "hola que tal estas" es el conjunto  $\{0, 0, 0.619, 0.785\}$ .

Ahora creamos una segunda matriz con los valores del TF-IDF calculados para cada posible pareja de tweets. La matriz obtenida es la siguiente:

	<b>hola</b>	<b>adios</b>	<b>barco</b>	<b>hola que tal estas</b>	<b>SUMA</b>
<b>hola</b>	1	0	0	0.785	1.785
<b>adios</b>	0	1	0	0	1
<b>barco</b>	0	0	1	0	1
<b>hola que tal estas</b>	0.785	0	0	1	1.785

Este cálculo consiste en realizar la raíz cuadrada del producto escalar de los

valores TF-IDF de dos tweets. Por ejemplo, el valor 0.785 de la primera columna es el resultado de aplicar el cálculo anterior a los tweets "hola" y "hola que tal estas" como se muestra a continuación:

$$\sqrt{(0 \cdot 0) + (0 \cdot 0) + (1 \cdot 0,619) + (0 \cdot 0,785)}$$

Finalmente sumamos los resultados de cada fila obteniendo los valores de la columna "SUMA". Cada uno de estos valores corresponde al TF-IDF global de cada tweet, eligiendo como tweet representativo aquel que mayor puntuación obtenga.



## Capítulo 9

# Detección automática de picos y Timeline

En este capítulo vamos a definir lo que entendemos como pico y explicaremos cómo detectar automáticamente picos para mostrarlos en la gráfica.

Además, detallaremos el proceso de generación de un Timeline para cada evento y la relación que guarda con el número de picos y con el capítulo anterior.

### 9.1. Definición y obtención de los picos

Para establecer el número de picos que se desea mostrar en la gráfica es necesaria la interacción del usuario con la interfaz tal y como se ve en la Figura 9.1 en el cuadro de texto marcado en azul. Es importante resaltar que el número de picos introducido por el usuario no modifica la gráfica original.

The screenshot shows the T.E.F.G. application window. It has a title bar with standard macOS window controls and the text 'T.E.F.G.'. The interface is divided into two main sections:

**1. Importar colección**

- Nombre de la colección:** A text input field containing 'Eurovision'.
- Nombre del campo de Tweet:** A text input field containing 'text'.
- Nombre del campo de Fecha:** A text input field containing 'created\_at'.
- Selecionar archivo:** A button next to a text input field containing the file path '/Users/universidad/Desktop/Eurovision/euro.json'.
- Cargar:** A button to load the collection.
- Cancelar:** A button at the bottom of the section.

**2. Generar gráfica**

- Inicio del evento (dd/mm/aaaa - hh:mm:ss):** Two input fields: '23/12/2017' and '13:03:59', separated by a '+' button.
- Fin del evento (dd/mm/aaaa - hh:mm:ss):** Two input fields: '23/12/2017' and '13:51:32', separated by a '+' button.
- Número de puntos:** A text input field containing '30'.
- Número de picos:** A text input field containing '6'.
- Colección:** A dropdown menu showing 'frec\_clasico2'.
- Idioma:** A dropdown menu showing 'Inglés'.
- Generar:** A button to generate the graph.

**Borrar colección**

- Colección:** A dropdown menu showing 'clasico2'.
- Borrar:** A button to delete the collection.

Figura 9.1: Interfaz inicial de T.E.F.G

Entendemos como pico cualquier máximo local de la gráfica que, en nuestro contexto, corresponderá con el segundo en el que más tweets se han publicado. Sin embargo, además de ser un máximo local, exigimos otra condición para considerar un punto como un pico. Siendo  $a$  un punto de la gráfica, para definir  $a$  como pico debe cumplir una de las siguientes condiciones:

- Los puntos  $a-1$  y  $a+1$  deben tener un valor en el eje  $y$  menor que  $a$ .
- Si el punto  $a+1$  tiene el mismo valor que  $a$  en el eje  $y$ , entonces el punto  $a-1$  debe tener un valor menor en el eje  $y$ .

En la siguiente imagen (Figura 9.2) podemos ver un ejemplo de estas dos condiciones, donde el pico dentro del círculo rojo corresponde con la primera y el pico dentro del círculo verde con la segunda.

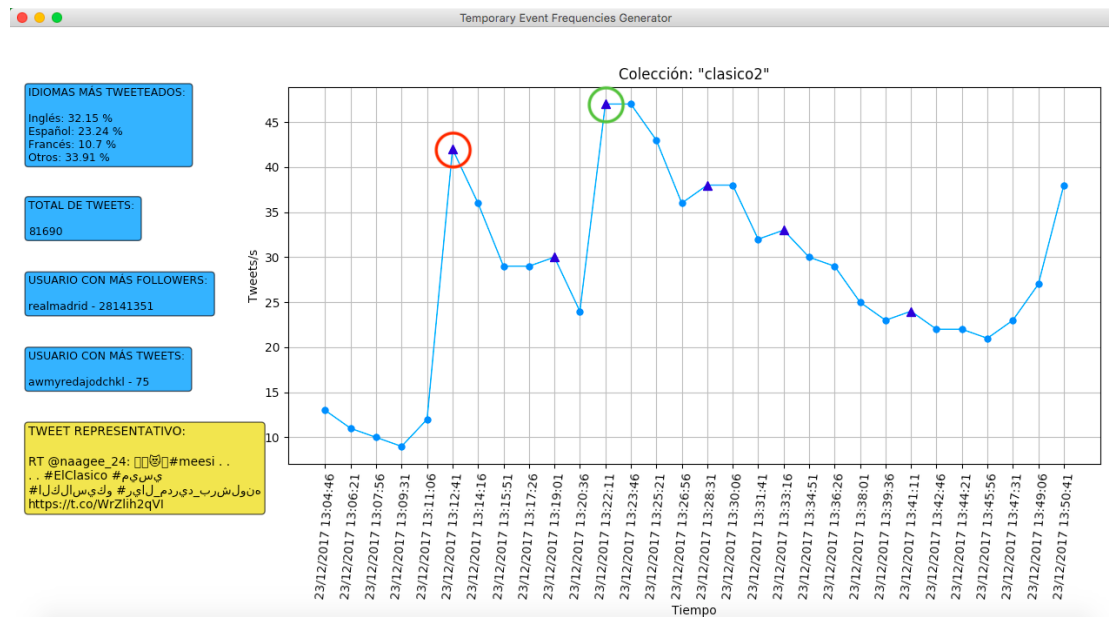


Figura 9.2: Tipos de picos en la gráfica de un evento

Tomando de nuevo como ejemplo la Figura 9.2 y siendo diez el número de picos elegido por el usuario, se resaltarán en la gráfica los diez puntos con mayor número de tweets y que cumplan alguna de las condiciones explicadas previamente.

Para conseguir esto, ordenamos en orden decreciente la colección de frecuencias por el campo del número de tweets y nos quedamos con la información de los diez primeros elementos. Nos interesa saber el número de tweets, el momento en el que se han publicado esos tweets (campo *epoch* de la colección) y también el tweet representativo en un punto (ver sección 7.2.1.2), para después mostrar el Timeline que explicaremos en la siguiente sección. Teniendo los diez picos sólo falta resaltarlos en la gráfica, para ello utilizamos la forma del triángulo en color azul oscuro.

Es importante mencionar que, a priori, puede parecer que algunos de los puntos considerados como picos no sean muy pronunciados. Sin embargo, si aplicamos la herramienta de zoom (ver sección 7.2.3) sobre ellos, probablemente veremos un perfil de frecuencias mucho más interesante con altibajos más pronunciados.

## 9.2. Generación del Timeline

El siguiente paso después de mostrar los picos es generar el Timeline correspondiente y mostrarlo al usuario.

Para ello, como hemos explicado en el apartado anterior, guardamos información de los picos. Para generar el Timeline necesitaremos el tweet representativo de cada pico, el usuario que ha publicado el tweet y el campo *epoch* o momento de publicación. Una vez tenemos esta información, ordenamos los tweets por su hora de publicación para que aparezcan en orden cronológico y los mostramos por pantalla como se puede ver en imágenes posteriores.

Disponer de un Timeline es útil porque nos proporciona, a través de unos pocos tweets, un resumen preciso de todo el evento sin tener que leer miles de tweets. Por ejemplo, para un partido de fútbol, el Timeline nos informará de los momentos clave como los goles, los autores de los mismos, las faltas, las tarjetas mostradas por el árbitro, el resultado final, etc.

Es importante mencionar que, siguiendo este ejemplo, nuestra herramienta no detecta automáticamente estos momentos clave. Es decir, cuando sucede algo importante en el evento aumenta el tráfico de tweets, aparecerá un pico en la gráfica y, en consecuencia, se mostrará un tweet en el Timeline sobre lo sucedido. Por ejemplo, si en un partido de fútbol los aficionados golpean a un jugador lanzándole algún objeto, se producirá un incremento del número de tweets en ese instante y en el Timeline aparecerá un tweet relacionado con el suceso (a pesar de que sea algo que no ocurre con frecuencia en los partidos de fútbol).

Continuando con la temática de los partidos de fútbol, a continuación se muestra una gráfica en la que se pueden ver tres picos resaltados que corresponden con sucesos que han tenido un gran impacto en el evento (Figura: 9.3). El tweet representativo de cada uno de los picos se mostrará en el Timeline (Figura 9.4). Podemos observar que en éste aparecen los tweets de sucesos significativos como dos goles y el ganador final del campeonato.

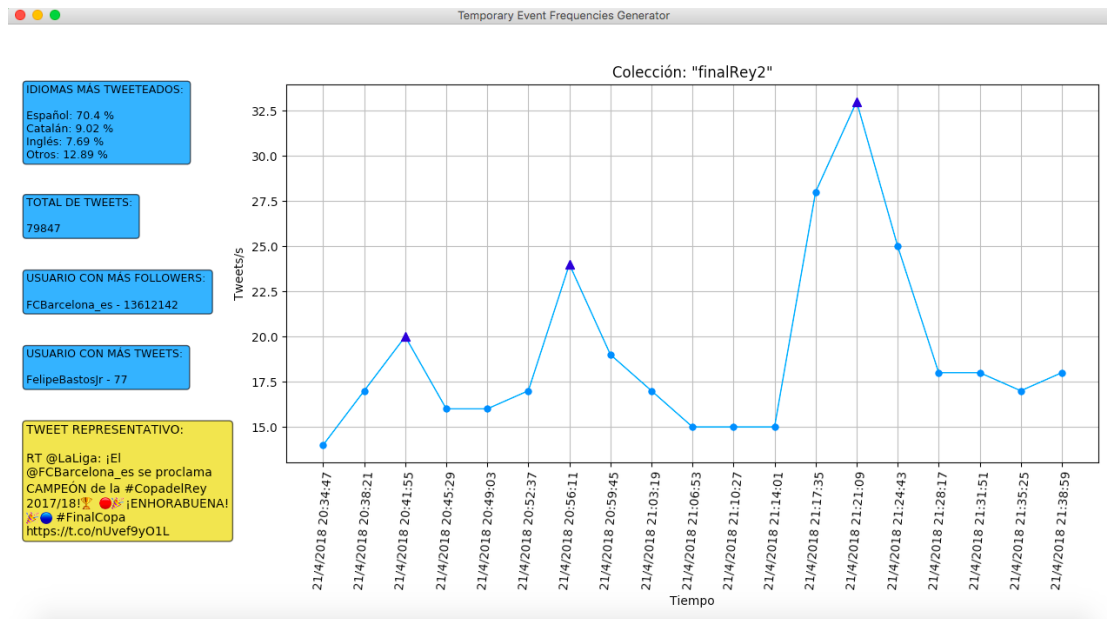


Figura 9.3: Gráfica con tres picos



Figura 9.4: Timeline de la gráfica anterior



# Capítulo 10

## Contribuciones al proyecto

En este capítulo vamos a explicar cómo nos hemos organizado a lo largo de este proyecto y las aportaciones de cada uno de los integrantes del equipo.

Durante las primeras semanas de proyecto, decidimos trabajar conjuntamente en la búsqueda de información sobre las tecnologías más convenientes para nuestro propósito. A medida que fuimos entrando en materia, con las ideas y objetivos iniciales ya definidos, decidimos realizar un reparto de tareas e intentar establecer una metodología de trabajo.

La metodología de trabajo que hemos mantenido es la siguiente:

- A principios de cada semana definíamos las tareas que iba a realizar cada uno, estableciendo una fecha límite (normalmente una semana).
- Llegada la fecha límite, poníamos en común el trabajo realizado por cada uno.
- Si llegada esta fecha, alguna tarea estaba incompleta debido a su dificultad, la realizábamos entre los dos.
- Si aun así, no éramos capaces de solventar los problemas, nos comunicábamos con nuestro director de proyecto solicitándole una reunión.
- Cuando las tareas de cada integrante estaban completadas con éxito, uníamos todo el trabajo y definíamos nuevas tareas.

Esta metodología, ha sido mantenida durante los tres primeros meses de proyecto. Más adelante continuamos trabajando de manera conjunta. A continuación, describimos las tareas que hemos realizado cada uno de los integrantes por separado:

## 10.1. Esther Ávila Benito

- En primer lugar me he encargado de dar los primeros pasos con MongoDB. Esto ha supuesto la creación de la estructura de la base de datos sobre la que está construido T.E.F.G, así como la ejecución de las primeras consultas.
- Además, tras importar las primeras colecciones, he realizado una preparación y limpieza de la información eliminando los datos innecesarios y añadiendo algunos campos que facilitarían el tratamiento posterior.
- Tras el preprocesamiento de los datos, tomé un contacto inicial con el módulo *matplotlib* de Python. Éste fue el elegido para el diseño de la gráfica de nuestra aplicación porque incluye todas las funcionalidades que posteriormente íbamos a utilizar. Tras una serie de pruebas, conseguí mostrar los datos en una gráfica inicial.

## 10.2. Guillermo Romero Alonso

- Tras estudiar las posibles librerías de Python para el diseño de interfaces gráficas, decidí optar por *tkinter*, ya que me pareció muy completa. Tras realizar varios *mockup*, desarrollé la interfaz inicial de T.E.F.G.
- Tras la importación de las colecciones, trabajé en la generación de las frecuencias correspondientes. Para esto, llevé a cabo una serie de cálculos sobre ciertos campos de una colección original obteniendo su colección de frecuencias.
- Con la gráfica inicial dibujada, implementé la detección de picos realizando una serie de comprobaciones sobre ciertos campos de las colecciones de frecuencias para después resaltar estos puntos en la gráfica.



## 10.3. Trabajo en común

Como hemos mencionado anteriormente, la mayor parte del trabajo se ha realizado de manera conjunta. Estas han sido las tareas llevadas a cabo por los dos miembros del equipo:

- Con la gráfica y la interfaz inicial dibujadas, incluimos la posibilidad de que el usuario eligiera el número de puntos de la gráfica. Para ello, fue necesario calcular la duración de los intervalos comprendidos entre cada uno de los puntos.
- Después, añadimos a la gráfica la información sobre los idiomas más utilizados en los tweets del intervalo representado. Posteriormente, además de mostrar esta información al usuario final, la utilizamos en el algoritmo TF-IDF para obtener el tweet más representativo.
- Con todas las funcionalidades de la gráfica completadas, añadimos la herramienta de *zoom*. Detectando el área de la gráfica en la que el usuario ha hecho *zoom*, mostramos una nueva gráfica con las mismas herramientas que la anterior y la información actualizada.
- A través del algoritmo TF-IDF hemos sido capaces de calcular el tweet más representativo de un intervalo o de un punto en un evento para mostrárselo al usuario.
- Como ya estaba realizada la detección de picos en la gráfica, utilizando el algoritmo de TF-IDF sobre estos picos, incluimos un Timeline que proporciona al usuario un resumen riguroso del evento.



# Capítulo 11

## Conclusiones y trabajo futuro

A lo largo de este trabajo hemos pretendido emplear conocimientos aprendidos a lo largo de la carrera además de adquirir otros nuevos. Desde el principio nos llamó la atención trabajar en un área puntera en la actualidad como es el *Big Data*. A continuación, explicaremos los objetivos que nos propusimos al comienzo de este trabajo, cuáles de ellos hemos alcanzado y las posibles mejoras que se podrían incluir en nuestra aplicación.

### 11.1. Conclusiones

Cuando comenzamos con este proyecto, nuestra idea era trabajar con algo cercano a nosotros como las redes sociales. Queríamos comprobar por nosotros mismos cómo era posible analizar millones de datos escritos por personas de todo el mundo. Nuestro principal interés era desarrollar una herramienta sencilla de usar y que fuera de gran utilidad para los expertos en el área.

Ahora que hemos concluido con este proyecto, podemos mirar atrás y afirmar que hemos conseguido los objetivos que nos propusimos:

- Hemos conseguido diseñar una interfaz sencilla con la que el usuario es capaz de visualizar una gran cantidad de datos gráficamente. En pocos pasos, obtiene toda la información de un evento que puede durar hasta semanas.

- Ofrecemos una serie de herramientas para que la información mostrada tenga cierto valor para el usuario y éste sea capaz de sacar sus propias conclusiones. Además, de toda la información de un evento, resaltamos aquella que tiene mayor importancia. Por ejemplo, de un evento en el que se han escrito medio millón de tweets, se le muestra al usuario un breve resumen de los momentos más importantes.
- Desde el punto de vista personal, hemos aprendido Python. Un lenguaje desconocido hasta ahora por nosotros y sobre el que teníamos gran curiosidad por lo utilizado que es en las empresas de hoy en día.
- Además, hemos reforzado conocimientos adquiridos en la carrera sobre las bases de datos *NoSQL* y más en concreto MongoDB.
- Hemos investigado sobre todas las posibles técnicas de visualización de datos, dándonos cuenta de la gran envergadura de este área. Finalmente, optamos por manejar la librería *matplotlib* de Python después de comprobar su versatilidad.
- Hemos trabajado con gráficas de frecuencias y hemos conseguido analizar y entender la visualización de los datos aplicando distintas fórmulas estadísticas.
- También hemos investigado sobre distintos algoritmos de recuperación de la información, eligiendo el TF-IDF para extraer la información más relevante de un conjunto de tweets.
- Hemos llegado a ser conscientes de la gran cantidad de datos que manejan las redes sociales sobre sus usuarios, más en concreto Twitter. Esto ha supuesto en nosotros un cambio de actitud a la hora de compartir información en la red.

## 11.2. Líneas de trabajo futuro

Pensamos que en el futuro las redes sociales seguirán teniendo la misma importancia en la sociedad que actualmente. Aunque también es posible que sufran cambios para adaptarse a los nuevos hábitos y comportamientos de la sociedad. Por eso, es necesario que nuestra aplicación permanezca actualizada y se adapte a estos cambios. Por otra parte, nuestra aplicación podría incluir una serie de mejoras que describiremos a continuación:

- Una mejora interesante sería añadir varias gráficas, una por idioma utilizado, y compararlas. Empleando esa información, se podrían sacar conclusiones útiles sobre el evento. Por ejemplo, se podrían detectar picos de los castellano-parlantes, pero que no lo fueran para los angloparlantes. Así, en un partido internacional, puede que el tweet más significativo de un idioma sea "Vaya golazo" mientras que en el otro idioma sea "Estaba en fuera de juego, vaya robo!".
- Otra mejora podría ser que el programa trabajara en modo *online*. Serviría para poner alertas o alarmas. Por ejemplo, una empresa pública de transportes podría detectar que algo funciona mal en sus instalaciones al detectar un incremento de los tweets sobre ello. Esta mejora supondría introducir otros cambios. Para localizar picos no bastaría con detectar una bajada. En un evento de larga duración, habría que utilizar datos históricos para descubrir los picos. Además, al hacer esto habría que tener en cuenta la "estacionalidad". En Twitter no hay la misma frecuencia de tweets en una hora punta del día que por la noche. Es decir, habría que detectar incrementos inesperados con respecto a la frecuencia media en circunstancias similares.
- Otro campo que admite mejoras es la detección del tweet más representativo para un intervalo o un pico. Nosotros usamos el algoritmo TF-IDF, pero se pueden añadir otros criterios. Por ejemplo, se puede suponer que un tweet será más representativo cuanto más "importante" sea el usuario que lo ha escrito. Hay muchos aspectos para considerar a un usuario como importante. Uno de ellos sería priorizar los mensajes de usuarios con más seguidores. También se podría tener en cuenta la importancia del usuario dentro del evento concreto: mensajes escritos, mensajes "retuiteados", menciones recibidas, etc.

- También se podría incorporar a nuestra aplicación cualquier herramienta de escucha de Twitter que permitiera al usuario la descarga de tweets con unos *hashtags* seleccionados con sólo pulsar un botón. Tras el proceso de descarga, se crearía automáticamente la colección correspondiente.
- Otra posibilidad sería almacenar las gráficas generadas e incluir una herramienta de comparación. Se podrían comparar dos gráficas en todo su dominio y comprobar si, por ejemplo, cuando en una de ellas hay un pico en la otra se produce una bajada de tweets. Esto podría suceder, por ejemplo, en el festival de Eurovisión. Si un país ha hecho una mala actuación, puede que no se escriban muchos tweets mientras que otro país "rival" la critique produciéndose un pico en este último.
- También se podría cambiar el Timeline para que se mostraran los tweets más representativos de todo el evento o de los instantes elegidos por el usuario, independientemente de si en esos momentos hay un pico o no.

Por último, queremos indicar que todo el código desarrollado en este trabajo se encuentra disponible en los siguientes links:

<https://github.com/estherab/TEFG>  
<https://github.com/gradog/TEFG>

# Capítulo 12

## Conclusions and future work

Throughout this work we have intend to apply the knowledge learned throughout the career in addition to acquiring new ones. From the beginning we were struck by the fact that we are working in a cutting-edge area such as *Big Data*. Next, we will explain the goals that we proposed at the beginning of this work, which of them we have reached and the possible improvements that could be included in our application.

### 12.1. Conclusions

When we started with this project, our idea was to work with something close to us like social networks are. We wanted to see for ourselves how it was possible to analyze millions of data written by people from all over the world. Our main interest was to develop a simple tool to use and that would be very useful for experts in the area.

Now that we have concluded with this project, we can look back in the past and affirm that we have achieved the goals we proposed:

- We have managed to design a simple interface with which the user is able to visualize a large amount of data in a graphic way. In a few steps, you get all the information of an event that can last up to weeks.

- We offer a series of tools so that the information shown has a certain value for the user and he is able to draw his/her own conclusions. In addition of all the information of an event, we highlight the one that has the most importance. For example, given an event in which half a million tweets have been written, the user is shown a brief summary of the most important moments.
- From a personal point of view, we have learned Python. A language unknown until now by us and about which we were very curious about how it is used in today's companies.
- In addition, we have reinforced knowledge acquired on the databases *NoSQL* and more specifically MongoDB.
- We have investigated all the possible techniques of data visualization, realizing also the great importance of this area. Finally, we decided to use Python's *matplotlib* library after checking its versatility.
- We have worked with frequency charts and we have managed to analyze and understand the visualization of the data by applying different statistical formulas.
- We have also investigated different information retrieval algorithms, choosing the TF-IDF to extract the most relevant information from a set of tweets.
- We have become aware of the large amount of data that social networks manage about their users, more specifically Twitter. This has given us a change of attitude when sharing information on the net.

## 12.2. Future work lines

We think that, in the future, social networks will stand having the same importance in society as today. Although it is also possible that they suffer changes to adapt to the new habits and behaviors of society. Therefore, it is necessary that our application remains updated and adapts to these changes. On the other hand, our application could include a series of improvements that we will describe below:

- An interesting improvement would be to add several graphs, one per language used, and compare them. Using that information, you could draw useful



conclusions about the event. For example, peaks of Spanish speakers could be detected, but not for English-speakers. Thus, in an international match, the most significant tweet of a language may be "Vaya golazo" while in the other language it is "It was offside, go steal!".

- Another improvement could be that the program worked in *on line* mode. It would serve to put alerts or alarms. For example, a public transport company could detect that something is wrong in its facilities by detecting an increase in tweets about it. This improvement would imply introducing other changes. To detect peaks, it would not be enough to detect a descent. In a long-term event, you would have to use historical data to discover the peaks. In addition in doing this, "seasonality" should be taken into account. On Twitter there is not the same frequency of tweets in a rush hour of the day as at night. That is, unexpected increases should be detected with respect to the average frequency in similar circumstances.
- Another field that supports improvements is the detection of the most representative tweet for an interval or a peak. We use the TF-IDF algorithm, but other criteria can be added. For example, it can be assumed that a tweet will be more representative the more important the user who wrote it is. There are many aspects to consider a user as important. One of them would be to prioritize messages from users with more followers. The importance of the user within the specific event could also be taken into account: written messages, "retweet" messages, received mentions, etc.
- It could be also incorporated into our application any Twitter listening tool that would allow the user to download tweets with selected *hashtags* at the touch of a button. After the download process, the corresponding collection would be created automatically.
- Another possibility would be to store the generated graphs and include a comparison tool. You could compare two graphs throughout your domain and check if, for example, when one of them has a peak in the other there is a drop in tweets. This could happen, for example, at the Eurovision festival. If a country has made a bad performance, you may not write many tweets while another country "rival" criticize it producing a peak in the latter.

- You could also change the Timeline so that the most representative tweets of the entire event or of the instants chosen by the user are shown, regardless of whether there is a peak or not at those moments.

Finally, we want to indicate that all the code developed in this work is available in the following links:

<https://github.com/estherab/TFG>

<https://github.com/gradog/TEFG>

# Bibliografía

- (2009). Origen y creación de Facebook. [http://www.cad.com.mx/historia\\_de\\_facebook.htm](http://www.cad.com.mx/historia_de_facebook.htm).
- (2009). Origen y creación de Twitter. [http://www.cad.com.mx/historia\\_de\\_twitter.htm](http://www.cad.com.mx/historia_de_twitter.htm).
- (2010). Evolución de los usuarios de Twitter. <https://es.wikipedia.org/wiki/Twiter>.
- (2010). Evolución de Twitter. <http://www.telam.com.ar/notas/201410/82036-la-evolucion-de-twitter-desde-el-hashtag-hasta-la-plataforma-multimedia.html>.
- (2012). Explicación y etapas de TF-IDF. <http://ccdoc-tecnicasrecuperacioninformacion.blogspot.com.es/2012/11/frecuencias-y-pesos-de-los-terminos-de.html>.
- (2012). Manual de la librería csv de python. <https://docs.python.org/2/library/csv.html>.
- (2014). Características de MongoDB. <https://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no>.
- (2015). Características de Python. <https://es.wikipedia.org/wiki/Python>.
- (2017). Estadísticas sobre el uso de las redes sociales. <https://www.fhios.es/20-sorprendentes-estadisticas-sobre-las-redes-sociales/>.
- (2017). Etapa IDF del TF-IDF. <https://es.wikipedia.org/wiki/Tf-idf>.

- (2017). Origen de las redes sociales en web. <http://socialmedialideres.com.ve/origen-y-evolucion-de-las-redes-sociales/>.
- (2017). Origen de las *stop words*. [https://es.wikipedia.org/wiki/Palabra\\_vac%C3%ADa](https://es.wikipedia.org/wiki/Palabra_vac%C3%ADa).
- (2017). Tiempo Unix o POSIX. [https://es.wikipedia.org/wiki/Tiempo\\_Unix](https://es.wikipedia.org/wiki/Tiempo_Unix).
- (2018). ISO 8601. [https://es.wikipedia.org/wiki/ISO\\_8601](https://es.wikipedia.org/wiki/ISO_8601).
- Barnes, J. (2007). Class and Committees in a Norwegian Island Parish. *Human Relations*.
- Boyd, D. M. and Ellison, N. B. (2007). Social network sites: definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1).
- Hull, K. and Lewis, N. P. (2014). Why Twitter Displaces Broadcast Sports Media: A Model. *International Journal of Sport Communication*.
- Kumaran, G. and Allan, J. (2004). Text Classification and Named entities for New Event Detection. *SIGIR*.
- Last, M., Abraham, K., and Bunke, H. (2004). Data Mining in Time Series Databases. *World Scientific Press*.
- Mathioudakis, M. and Koudas, N. (2010). TwitterMonitor: trend detection over the Twitter Stream. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
- Phelan, O., McCarthy, K., and Smyth, B. (2009). Using Twitter to Recommend Real-Time Topical News. *Proceedings of the third ACM conference on Recommender systems*.
- Phuvipadawat, S. and Murata, T. (2010). Breaking News Detection and Tracking in Twitter. *IEEE Computer Society*.
- Ritter, A., Mausam, Etzioni, O., and Sam, C. (2012). Open Domain Event Extraction from Twitter. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.