Detección offline de subtemas en Twitter durante eventos

Beatriz Jiménez del Olmo

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

UNIVERSIDAD
COMPLUTENSE
MADRID

Trabajo Fin Máster
Curso 2017-2018

Convocatoria: Junio 2018
Calificación: 10

Director: **Rafael Caballero Roldán**

Autorización de Difusión

La abajo firmante, matriculada en el **Máster en Ingeniería Informática** de la **Facultad de Informática**, autoriza a la **Universidad Complutense de Madrid (UCM)** a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: **"Detección offline de subtemas en Twitter durante eventos",** realizado durante el curso académico 2017-2018 bajo la dirección de **Rafael Caballero Roldán** en el **Departamento de Sistemas Informáticos y Computación**, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Firma:

Beatriz Jiménez del Olmo
18/06/2018

Resumen

Hoy en día las redes sociales juegan un papel muy importante en nuestras vidas, al convertirse en el principal medio para compartir todo tipo de contenidos, desde la opinión de los usuarios sobre programas de televisión favoritos o las experiencias de su día a día, hasta comentarios sobre las noticias de última hora. Su estudio puede revelar aspectos muy interesantes para analistas, si somos capaces de extraer la valiosa información incrustada en sus inmensos flujos de información.

Por ello, en este trabajo se presenta un método de exploración para Twitter que, a partir del conjunto de tweets emitidos durante un evento, permite detectar los temas que, a juicio de los usuarios, han tenido mayor relevancia durante su desarrollo. Permite, por tanto, analizar el evento obteniendo una imagen general de lo más comentado.

El método propuesto presenta un enfoque que parte de la relación temporal entre los tweets, considerando que tweets que se emiten en una misma ventana de tiempo tienen más posibilidades de formar parte de una noticia, para a continuación agrupar los tweets de la misma ventana según su similaridad textual.

Una vez detectados los temas del evento analizado, se proponen dos métodos de agrupación y representación de los resultados, permitiendo al usuario organizar los temas según dos criterios diferentes, ya sea por la proximidad temporal entre ellos o bien por la relación textual entre los temas que debaten. Para probar la capacidad de detección del método, se ha aplicado sobre conjuntos de datos de eventos de diferentes características, que tienen en común su alto seguimiento en Twitter.

Palabras clave

Detección de temas, Recuperación de información, microblogs, Twitter, Redes sociales, Visualización de datos, TF-IDF, Similaridad textual, Clustering

Abstract

Nowadays Social Media plays an important role in our life. It has become the main way to share all kinds of content, from users' opinion about their favorites TV shows, their daily experiences, or commenting the breaking news. The study of these messages can reveal very important facts to analysts, but only after extracting the important information that is embedded in the huge flows of data.

With this purpose, this work presents a method for Twitter exploration that, starting from a set of tweets published during an event, detects the topics that are more relevant based on the users opinion. Thus, the system analyzes the event, obtaining the highlights of the most commented topics.

The proposed method considers first the temporal relation of the tweets, assuming that messages published on the same time window are more likely to be a member of the same topic. Then, those tweets that belong to the same window are clustered by their textual similarity.

Once we detect the topics related to the analyzed event, we propose two methods to aggregate and represent the results, allowing the user to organize the topics either by temporal proximity among them or by the textual relation among their opinions. In order to test the detection ability of the method, the system has been tested in several datasets with specific characteristics, which have in common their high popularity in Twitter.

Keywords

Topic detection, Information retrieval, microblog, Twitter, Social Media, Data visualization, TF-IDF, Textual similarity, Clustering

Índice de contenidos

Índice de figuras

Agradecimientos

A Rafa, que me ha enseñado dos cosas:
que hacer un TFM puede ser muy divertido y
que no se le debe hacer ni caso al tutor de vez en cuando.
Como ahora mismo.

1. Introducción

Según el informe *Digital in 2018* [1], con la llegada del año 2018 se alcanzaron los 4 mil millones de usuarios de Internet, superando así la barrera del 50% de la población mundial (alrededor de 7'6 mil millones de habitantes). Se estima además que los usuarios activos en redes sociales constituyen ya un 42% de la población.
Si atendemos al tiempo que dedicamos al uso de estas redes, oscila bastante por región, con un máximo de 3 horas y 57 minutos de media diaria en Filipinas y un mínimo de 48 minutos Japón, situando la media española en 1 hora y 38 minutos de uso.
Por otra parte, es destacable el gran auge del uso de los dispositivos móviles como medio de acceso a la información, siendo a día de hoy el principal dispositivo con el que lo hacemos, con un 53% de los accesos desde este tipo de dispositivos y aumentando un 14% el número de usuarios activos desde enero del 2017.

"When it happens it happens on Twitter."
"See what people are talking about." [2]

Twitter [3], red social nacida en 2006 en Estados Unidos, es desde hace años líder del sector de las plataformas basadas en microblogging. Inicialmente sus mensajes (conocidos como "tweets") permitían a los usuarios compartir solo contenido textual con la limitación de un máximo de 140 caracteres por tweet, una de las principales características que la diferencia de otras redes sociales. Esta cifra restrictiva de la longitud de los mensajes fue ampliada a 240 a finales de 2017, aunque la propia plataforma [4] indica que solo el 9% de los tweets alcanzaban el límite inicial. Las cifras de Twitter [5], alcanzando los 330 millones de usuarios activos al mes a finales del 2017, denotan que se trata de una red social en crecimiento, como denota el incremento de un 4% respecto al año anterior.

Otra de las diferencias con otro tipo de redes sociales es que no se exige reciprocidad a la hora de "seguir" a otros usuarios para estar al tanto de sus contenidos publicados. Esta característica ha propiciado que la plataforma sea usada tanto por marcas comerciales como Zara (@Zara) para publicitar sus productos, por personajes públicos como Donald Trump (@realDonaldTrump) como vía para hacer declaraciones, así como por cuentas que proporcionan asistencia, tal y como ofrece la empresa Correos (@CorreosAtiende). Esta diversidad de usos de la plataforma justifica que se generen contenidos de todo tipo y a gran velocidad, tal como indican sus cifras [6] de 500 millones de tweets enviados al día en el 2014.

Twitter se emplea para transmitir todo tipo de contenidos, desde reflexiones, intenciones y actividades de la vida cotidiana de los usuarios hasta actualizaciones en el seguimiento eventos, información meteorológica o de tráfico. Estos últimos usos se ven reflejados en sus cifras, que indican que se produjo un aumento del 15% en el porcentaje de usuarios que usan Twitter para recibir noticias del año 2016 al 2017 [7]. Tal y como indica la propia

plataforma [8], el uso principal de Twitter es, según usuarios, *"descubrir algo nuevo e interesante"*.

El límite de longitud de los mensajes exige que sus usuarios hagan un esfuerzo en condensar sus ideas y pensamientos en unas pocas palabras, lo cual hace fácil la difusión y lectura de información en tiempo real. Todo ello lleva a que se trate de una plataforma muy adecuada para el seguimiento de eventos o noticias de última hora de diferentes contextos, así como para el estudio de las características y relaciones sociales existentes entre sus usuarios.

A día de hoy podemos encontrar distintos enfoques para analizar Twitter, desde la detección de tendencias y eventos [9], a la clasificación de eventos por categorías [10], o para recomendar noticias a los usuarios en función de su interacción [11]. Todos ellos tienen que lidiar con los retos derivados de las características particulares de la red social:

- *Cantidad de información a procesar*: A día 30 de mayo de 2018, se registraron [12] unas tasas de aproximadamente 8000 tweets por segundo, por lo que cualquier análisis sobre los mensajes debe ser capaz de tratar con grandes volúmenes de datos.
- *Tipo de información compartida por los usuarios:* El contenido que recogen los tweets enviados por los usuarios incluye desde los muy populares trending topics como #FelizLunes hasta tweets que sencillamente incluyen información personal que el usuario desea compartir con su red de amigos. Esto ocasiona que los tweets sobre noticias de última hora queden "sepultados" por esta gran cantidad de tweets con información irrelevante para el público general. Se plantea por tanto el reto de conseguir separar la información útil entre tanta información ruidosa y heterogénea. Este problema se incrementa por la falta de contexto de los mensajes, los errores gramaticales, el lenguaje informal e irregular, el uso de abreviaturas o la mezcla de distintos lenguajes que pueden ser encontrados.

Este trabajo se ha centrado en particular en el análisis de eventos de la vida real que son comentados a través de Twitter. Durante estos eventos los usuarios emiten tweets como respuesta a los momentos importantes o *hitos* del evento. Descubrir las reacciones de los usuarios a estos hitos puede ser ser de gran utilidad. Por ejemplo, durante un debate político, los responsables de redes sociales de los partidos pueden estar interesados en saber qué opiniones suscitan las intervenciones de cada candidato. Esta información, de gran valor, no se encuentra fácilmente en medios convencionales.

La motivación de este trabajo es extraer los temas de conversación principales que se dan en Twitter durante un evento real. Además, queremos dar diferentes opciones al usuario para que pueda organizar y mostrar la información obtenida de manera que le resulte de utilidad.

1.1. Objetivos

Nuestro principal objetivo es diseñar e implementar un algoritmo de detección offline de temas a partir de conjuntos de tweets obtenidos durante un evento. Este conjunto ya ha sido seleccionado mediante el filtrado por hashtags o palabras clave, para delimitar la relación de los tweets con el evento.

Este objetivo se puede desglosar en los siguientes 3 subobjetivos, que marcan las líneas generales de nuestro trabajo :

O1.- El algoritmo propuesto debe ser capaz de detectar como diferentes temas de conversación que respondan a un mismo hito repetido en varias ocasiones durante el evento real. Por ejemplo, dos goles en un mismo partido no deben ser confundidos en un solo tema.
O2.- Nuestro método debe evitar asociar cada hito con un tema único. Esto es así porque lo que nos interesa no es el hito en sí, sino su reflejo en Twitter, es decir las reacciones que provoca, que pueden ser numerosas y diversas.
O3.- Como el concepto de "tema" depende de cada usuario, es importante no filtrar ni excluir temas *a priori*. El resultado de esto será un gran número de temas, por lo que nuestro algoritmo debe permitir agrupar los temas bajos distintos criterios para facilitar su presentación.

1.2. Plan de trabajo

Persiguiendo estos objetivos, el trabajo se desarrolla a través de las siguientes etapas:

- *Estudio de trabajos relacionados:* se analizarán los métodos existentes para la detección y seguimiento de temas de conversación en Twitter, analizando y valorando sus diferentes enfoques, presentando al final las características diferenciadoras del método propuesto en este trabajo con respecto a otras propuestas.
- *Descripción, diseño e implementación del método propuesto de acuerdo con los objetivos anteriores*.
- *Evaluación de resultados:* se llevará a cabo una relación de experimentos que muestren si el funcionamiento del método propuesto sobre diferentes tipos de eventos, en efecto detecta las noticias más relevantes.

1.3. Estructura de la memoria

El siguiente capítulo, **Trabajos relacionados**, presenta otras propuestas para el análisis de eventos en Twitter, destacando sus principales características y carencias, permitiéndonos establecer los aspectos diferenciadores con respecto al método presentado en este trabajo.

Seguidamente, el capítulo denominado **Propuesta general** plantea nuestra propuesta de manera general, permitiéndonos tener una imagen global del proceso, desde la recopilación de los datos hasta la presentación de los resultados.

Los tres capítulos siguientes, denominados **Recopilación y preprocesado de datos**, **Detección de temas** y **Presentación de resultados**, respectivamente, definirán cada una de las fases en las que se ha dividido el método propuesto, ofreciendo algunos detalles técnicos del sistema implementado.

En el capítulo **Similaridad entre tweets** se comienza repasando algunas de las técnicas de comparación textual que se pueden encontrar en la literatura, para finalmente escoger la que utilizaremos en el cálculo de similaridad entre tweets, pieza clave de nuestro algoritmo de detección de temas.

A continuación, el capítulo **Experimentos** recoge y comenta los resultados obtenidos tras la realización de pruebas de nuestra propuesta sobre una serie de eventos de diferentes características.

Finalmente, en el capítulo **Conclusiones**, se analizan los resultados obtenidos, el nivel de cumplimiento de los objetivos propuestos, así como posibles líneas de trabajo futuro.

Además, se incluye un **Anexo A**, el cual contiene información sobre el archivo de configuración necesario en la fase de detección de temas, detallando el papel de cada uno sus campos.

El código fuente de este trabajo se puede consultar en el siguiente repositorio:

https://github.com/bejiol/TFM

2. Introduction

According to the report Digital in 2018 [1], at the beginning of 2018 the number of active users on Internet reached 4 billions, passing the barrier of 50% of the world's population (about 7'6 billions inhabitants). Also, the number of social media active users is nowadays around 42%.

If we look at the time that we spend at social media, the maximum is reached by Philippine population, with 3 hours and 57 minutes on average and the minimum by Japan with 48 minutes. The average on Spain around 1 hour and 38 minutes of use.
It's also important to remark the rise of the mobile social media access, being nowadays the main device used to this end, with 53% of the total number of accesses, increasing by 14% that of 2017.

"When it happens it happens on Twitter."
"See what people are talking about." [2]

Twitter [3], is a social media platform born in 2006 on United States, is the leading company in the industry of microblogging. Initially, the user messages in Twitter (known as *tweets*), allowed Twitter users to share only text fragments of 140 characters, one of the main peculiarities between this and other social media. This length constraint was extended to 240 characters by the end of 2017. However, according to data provided by the company [4] only 9% of the tweets reached the initial limit. The figures of active users per month in Twitter [5], which by the end of 2017 reached the 330 millions and suppose an increment of 4% with respect the previous year, indicate that it is still growing.

Another difference between Twitter and other social media is that no reciprocity is required in order to 'follow' and track other users publications. This characteristic has boosted the use of Twitter by commercial companies such as Zara (@Zara) to advertise their products, by public figures such as Donald Trump (@realDonaldTrump) as channel of communication, and by accounts that offer different services, for instance by Correos (@CorreosAtiende). This diversity of uses explain the variety of contents produced in the platform and the huge ration they are published, reaching the 500 millions of tweets every day by 2014 [6].

Twitter is employed to conveyed many different types of contents, from reflections, intentions and activities of everyday life, to news related to events, weather reports, or road traffic news. These last uses are increasing steadily according to the number in [7], which indicate that the percentage of users that choose Twitter to get informed of the latest news increased by 15% from 2016 to 2017. As the own platform indicates [8], the main use of Twitter according to its users is 'finding something new and interesting'.

The limit in the messages length requires from the users to make an effort to express their ideas and thoughts in a few words, which facilitates spreading the information in real time. This makes Twitter a suitable platform for tracking events and breaking news, and hence to study the characteristics and social relations existing among its users.

Currently, we can find different approaches about Twitter analysis, from the detection of tendencies and events [9], to the classification of events by categories [10], or to the recommendation of news to the user according to their interaction [11]. All of them have to deal with the difficulties and specific features of the social media:

- The large amount of information to be processed: as of 30th of May of 2018, a ratio of 8000 tweets per second were registered [12]. Hence, any message analysis must be able to deal with great volumes of data.
- Variety of information shared by the users. Tweets include a great diversity of contents. From the famous #HappyMonday to tweets that simply convey personal information that the user wishes to share with their friends. Often the tweets about breaking news are 'buried' under a great amount of information irrelevant to most of the users. Thus, it is a challenge to try to single up useful information out of this noisy and heterogenous information. The problem is increased due to the lack of context in the messages, the grammar error, the informal language, the use of abbreviations and the many different languages that can be found.

This works focuses on the analysis of the comments in Twitter about real life events. During these events, the users emit tweets as a response to the most relevant moments, or *key moments*. Detecting the user reactions to these key moments can be very useful. For instance, during a political debate, the online community manager can be interested on the opinions obtained as reaction to the speech of each candidate. This information, of great value, cannot be found easily elsewhere.

The main motivation of this work is to extract the main topics of conversation in Twitter during a real event. Moreover, we would like to provide the user with different options in order to organize and display the information in a way that it becomes useful.

2.1. Objectives

The main objective of this work is to design and develop an offline algorithm that can detect topics among a set of Twitter messages which belong to a concrete event. This set has been selected by filtering the messages with certain hashtags or keywords, to identify the tweets related to the event.

This objective can be expanded in the following items, that mark the general lines of our work:

O1.- The proposed algorithm must be able to detect as differents topics about the same key moment that can repeat in several times during the event. For example, if a team scores two goals on a football match, they must be detected as different topics.
O2.- Our method must not assume that a key moment provokes only one topic of discussion. We are not interested on detect this key moments, but their impact on Twitter, which can cause mixed reactions from the users.

O3.- Since the “topic” definition depends on the user, it’s important not to filter or exclude topics beforehand. This creates a big number of topics, so our method must allow to organize the topics by different criteria, to facilitate its presentation.

2.2. Working plan

To achieve this goals, this works follows the following stages

- *Study of the state of the art:* the existing topic detection and tracking systems must be analyzed, comparing their approaches, and summarizing the distinctive characteristics of the proposed method.
- *Description, design and development of the system, according to the previous objectives*.
- *Result evaluation:* in order to test the detection ability of the method, the system has been tested in several datasets with specific characteristics, checking if it’s able to detect the main topics discussed.

2.3. Document structure

The next chapter, **Related work**, presents several proposals to perform Twitter analysis, outstanding their main characteristics and weaknesses, allowing us to highlight the distinctive characteristics of the proposed method.

Then in the chapter **General proposal,** our system is generally defined, showing an overall picture of the process from the tweet gathering to the results presentation.

The **Data gathering and preprocess**, **Topic detection** and **Results presentation** chapters, respectively, will define each stage of the proposed method, offering technical details about the implemented system.

Then, the **Similarity between tweets** chapter includes a review of textual comparison techniques, describing the one that has been selected to measure the similarity betweet the tweets, which is a key element in our proposal.

The next chapter, **Experiments**, include the results obtained from testing our system with different events, each one with specific characteristics.

Finally, the chapter **Conclusions** analyze the final work, measuring the level of compliance with the project objectives, also including several future work lines.

In addition, it’s included an appendix **Anexo A**, which contains information about the configuration file required on the detection stage, providing details on the role of each field.

3. Trabajos relacionados

Twitter resulta muy interesante porque entre su continuo flujo de mensajes se encuentra información que resulta atractiva para públicos muy diversos: desde una empresa que desee estudiar el grado de aceptación de los usuarios frente a su nuevo producto lanzado al mercado, hasta usuarios individuales que quieran informarse de lo que ha pasado en el último programa televisivo que no pudieron ver.

En la literatura podemos encontrar aplicaciones de Twitter a problemas tan dispares como determinar el género de los usuarios en función de su perfil [13], estudiar la influencia de los usuarios relacionándola con el número de seguidores [14] o analizar el papel de la red social durante una campaña política [15].

El problema particular que se trata de resolver en este trabajo se engloba dentro del llamado **Topic Detection and Tracking** (TDT) [16], que trata de detectar la aparición de nuevos temas de conversación dentro de un flujo de información, y estudiar la reaparición, relaciones y evolución de los mismos. Atendiendo al modo en el que la información se difunde en Twitter, estudios [17] han demostrado que se hace manera similar a los medios de comunicación tradicionales, y que aproximadamente el 85% de los temas debatidos son titulares o noticias persistentes.

Dentro de TDT, los distintos problemas se pueden clasificar en función de varios aspectos [18].

Atendiendo a cómo se procesa la información a analizar, podemos dividir TDT en dos subtipos de tareas:

- *New Event Detection (NED):* si se trata del caso en el que los tweets enviados se procesan según llegan, de manera que se trata de un análisis online o en tiempo real.
- *Retrospective Event Detection (RED):* cuando dado el conjunto completo de tweets a analizar se organizan a posteriori de manera offline detectandose temas de conversación.

Ambas tareas cuentan con sus respectivos retos, en caso de NED se debe ser capaz de tratar en tiempo real información que llega con altísimas frecuencias. Además, deberá categorizar los mensajes que lleguen como temas nuevos o perteneciente a alguno de los temas ya detectados. En caso de RED, se cuenta con un alto volumen de información que debe analizar y organizar de manera eficiente.

Por otro lado, dependiendo de la información que conocemos de antemano sobre el evento, se trata de:

- *Especificado*: en caso de que se trate de eventos conocidos o eventos sociales planeados, de los que conocemos datos como el tiempo, el tipo de evento o una descripción, estas características pueden ser explotadas adaptando técnicas

tradicionales de recuperación y extracción de información, como filtrado, generación y expansión de consultas, clustering o agregación de información.

- *No especificado*: si no contamos con información del evento, la tarea consiste en monitorizar el flujo de información para tratar de descubrir cuándo se produce un tema nuevo, generalmente detectando anomalías en la frecuencia de publicación para detectar tendencias. Estas tendencias se agrupan posteriormente en función de sus características similares formando eventos y clasificando finalmente estos eventos en categorías.

Finalmente, como es común en las técnicas de aprendizaje automático, dependiendo del conocimiento *a priori* de los datos lo podemos clasificar en:

- *Supervisado*: los mensajes están etiquetados por temas, por lo que el problema se engloba dentro del aprendizaje supervisado y consiste en crear un modelo capaz de generalizar la clasificación de los tweets a partir del conjunto de datos de entrenamiento. Este es el enfoque menos frecuente, principalmente por el carácter heterogéneo de los mensajes de Twitter y porque suele ser inviable etiquetar a mano conjuntos de datos de tamaño razonable para el análisis.
- *No supervisado*: los datos se encuentran sin etiquetado ni clasificación alguno, por lo que los tweets deberán ser organizados en temas, sin conocimiento *a priori* del número de temas que hay presente.

Dentro de esta clasificación, nuestro sistema estaría englobado como problema **RED**, ya que sobre un conjunto de tweets que se recopilan del evento y analizan a posteriori de manera offline. Se trata también de un método de aprendizaje **no supervisado**, ya que no conocemos el número ni las características de los temas a encontrar y, dado que sabemos datos como el tiempo en el que sucede el evento y conocemos parcialmente de qué clase de evento se trata, hablamos de un evento **especificado**.

A continuación se procederá a analizar y valorar algunos de los sistemas que se han encontrado en la literatura sobre la detección eventos y temas de debate mediante análisis de Twitter o redes similares, indicando dónde se sitúan dentro de esta clasificación. Para cada uno de los métodos estudiados se destacarán los aspectos más relevantes que tengan relación con nuestra propuesta y se señalará aquellos que se trata de mejorar.

3.1. Hotstream

El primero de los métodos analizados, que da lugar a una aplicación llamada *Hotstream* [19], persigue el objetivo de detectar, agrupar y valorar noticias de última hora en tiempo real. En la clasificación anteriormente mencionada se trataría de un problema NED, no especificado y no supervisado. Para cada noticia, el sistema genera un informe de la misma, que incluye el primer mensaje encontrado, los mensajes que la componen, y un gráfico que permite visualizar la evolución de su actividad temporal.

Para ello, siguen un procesamiento de los datos que podemos resumir en los siguientes pasos:

- Recopilación de tweets, filtrando el flujo de mensajes teniendo en cuenta aquellos que contengan determinadas palabras clave como "#breakingnews".
- Indexado de los mensajes empleando Apache Lucene.
- Agrupación de los mensajes similares usando el modelo TF-IDF y potenciando el papel de palabras como nombres propios, hashtags y nombres de usuario detectados por NER, según la noción de similaridad dada por la fórmula:

$$sim(m_1, m_2) = \sum_{t \in m_1} [tf(t, m_2) * idf(t) * boost(t)]$$

donde m_1 y m_2 son los tweets a comparar, t cada una de las palabras de m_1, la función $tf(t, m)$ cuantifica la importancia del término t dentro del mensaje m atendiendo a la frecuencia del término en dicho mensaje y el valor $idf(t)$ pondera la importancia global de t sobre el total de los mensajes.

Para la agrupación de los mensajes se usa un enfoque similar al de nuestra propuesta, donde se evalúa la similaridad de cada mensaje a un grupo en función del valor de la función de similaridad y cierto umbral mínimo exigido, formando un grupo nuevo en caso de no superarlo (Figura 1). Para evaluar la pertenencia de un mensaje se emplea el primer mensaje de la noticia y los 10 términos clave del grupo.

Algorithm 1 Assign message m into a group in G

```
for g in G do
    Score[g] ← Sim(m, g.firstDoc, g.topTerms)
end for
if Max(Score) > MergeThreashold then
    Assign(m, Max(Score).groupId)
else
    groupId ← Group.create()
    Assign(m, groupId)
end if
return G
```

Figura 1: Algoritmo de asignación de mensajes a grupos en el sistema *Hotstream*.

A la hora de valorar la inclusión de un mensaje de un grupo, los autores indican que emplean factores de popularidad y fiabilidad como el número de seguidores de los usuarios y el número de retweets de manera positiva.

3.2. TwitInfo

El objetivo del sistema *TwitInfo* [20] es ligeramente diferente a los otros sistemas estudiados y presenta mayor similaridad al nuestro, dado que trata de resolver el problema de cómo determinar los aspectos (temas debatidos) más importantes de un evento conocido, permitiendo detectar subtemas dentro del propio evento y profundizar en ellos, permitiendo

analizar así eventos de larga duración. Es, como nuestro sistema, un problema no supervisado y sobre un evento especificado, pero el procesamiento de los tweets hace que sea un enfoque NED, analizando los datos en tiempo real.

En concreto, los pasos más relevantes del funcionamiento del método de análisis que proponen son:

- Seguimiento y filtrado de tweets con determinadas palabras clave.
- Representación del evento temporalmente (Figura 2), en función de las frecuencias de publicación de mensajes.
- Detección de los picos relevantes de frecuencia usando técnicas de procesado de señales, que se señalan como momentos potencialmente interesantes. Estos momentos presentarán ratios de publicación inusualmente superiores al histórico y permitirán identificar el máximo local y una ventana temporal alrededor, que delimita el momento en el que se debate dicho subtema.
- Dentro de cada uno de estos picos se aplica el modelo TF-IDF sobre los unigramas y se seleccionan los 5 términos más valorados que servirán además para etiquetar el pico con palabras clave que definan el momento.
- Dados estos 5 términos clave del evento se evalúan los tweets de la ventana de cada pico por la cantidad de ellos que contengan, mostrando al usuario los 5 más valorados que representarán al evento.

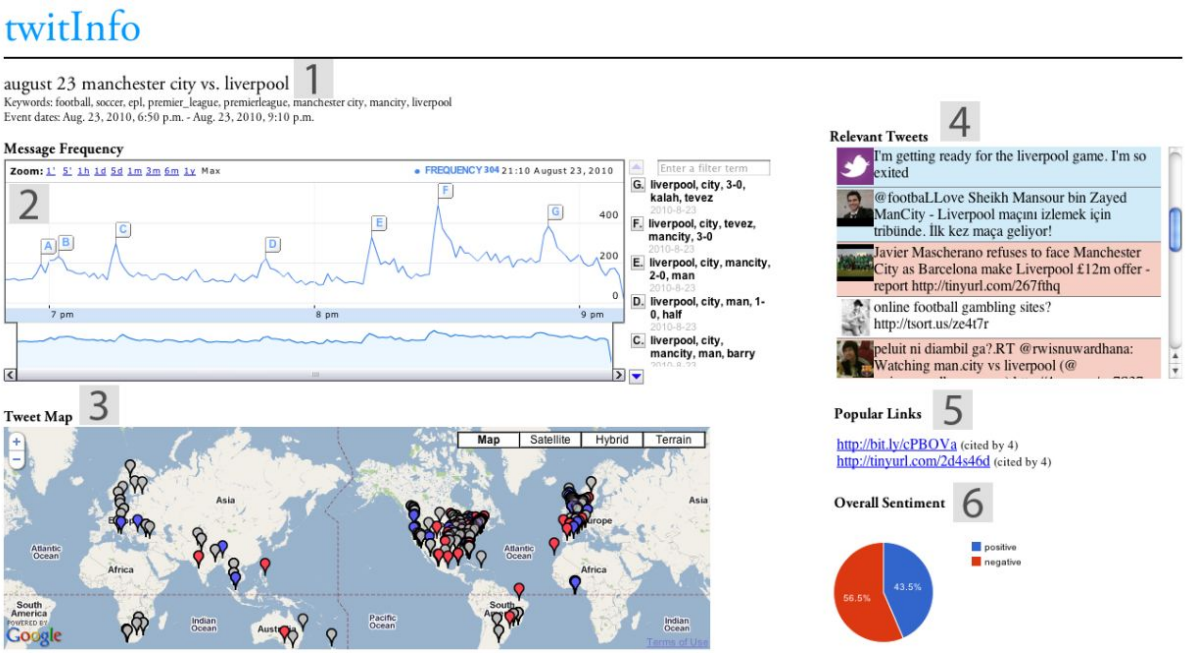


Figura 2: Pantalla principal del sistema TwitInfo analizando un partido de fútbol.

Un aspecto muy interesante de este sistema es que permite seleccionar cada uno de estos picos como evento principal, repitiendo el proceso mencionado anteriormente, haciendo zoom sobre un pico y permitiendo detectar subeventos. Por otro lado incluyen la valoración del sentimiento general de cada momento en grado positivo-negativo.

Es importante destacar que, tal y como se plantea *TwitInfo*, en cada uno de los picos que representan momentos clave dentro del evento solo ocurre un subtema. En cambio, bajo nuestro punto de vista, aunque un "pico" de frecuencias se corresponde con un hito o momento álgido del evento analizado, este suscita en la red una gran variedad de reacciones y opiniones diversas.

Es decir, mientras que el objetivo de *TwitInfo* es detectar qué hito del evento real ha dado lugar al incremento en la frecuencia de emisión de mensajes, nuestra propuesta no es localizar (solo) el hito sino sobre todo las reacciones que ha suscitado entre los usuarios. En nuestro enfoque, en lugar de “hacer zoom” sobre un pico y repetir el algoritmo de análisis, estos subtemas son detectados desde el principio, viéndose representada la relación entre los diferentes temas al agruparse por proximidad temporal.

Además, la agrupación por relación textual que nosotros proponemos nos permite relacionar posteriormente estos “picos”, como puede pasar en caso de que se marquen dos goles en un partido, ocasionando conversaciones similares.

3.3. ETree

El sistema *ETree* [21], al igual que otros sistemas ya comentados, analiza Twitter para tratar de extraer los temas de conversación más relevantes durante un evento determinado. Además, *ETree* llega un paso más adelante en este proceso, dado que establece una jerarquía entre los temas de conversación, tratando de explicar las relaciones causales entre ellos y entender su evolución en el tiempo (Figura 3). Se trata, como en nuestro sistema, de un análisis RED, conociendo de antemano aspectos del evento y siendo un problema de aprendizaje no supervisado.

El funcionamiento de la parte relacionada con nuestro sistema (encargada de la detección de temas) es el siguiente:

- Detección de secuencias de palabras comunes a lo largo de los mensajes relacionados con el evento. Según mencionan, usan un modelo de n-gramas, destacando n=4 como un número adecuado según sus experimentos. Cada uno de estos n-gramas frecuentes formará un *bloque de información* o *tema t*.
- Asignación de los mensajes restantes (que no incluyan los n-gramas más comunes detectados en la fase anterior) a cada uno de los temas detectados, midiendo la similaridad de cada mensaje *m* con cada tema *t*. Para ello, consideran palabras que pertenezcan tanto a *m* como a *t,* evaluando la similaridad por coseno del TF-IDF entre *m* y dichas palabras. Si se encuentra una alta similaridad, se incluirá el mensaje *m* como relativo al tema.

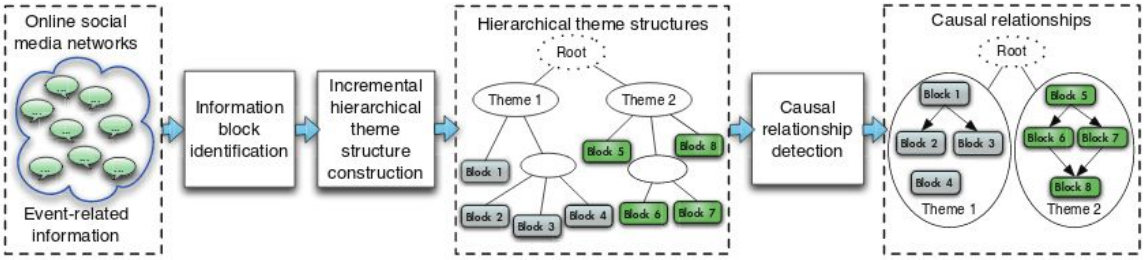


Figura 3: Etapas del funcionamiento del sistema *ETree* al completo.

ETree comparte con nuestra propuesta el cálculo de similaridad de los mensajes usando como función de comparación de textos el coseno de los vectores TF-IDF que representan los documentos. Además, como detallaremos más adelante, realizan una etapa para detectar temas iniciales, los cuales son ampliados en fases posteriores, enfoque que comparte nuestro método. La diferencia es que en nuestro caso tratamos las ventanas de mensajes consecutivos de manera aislada, evitando que se mezclen temas diferentes pero que utilizan palabras similares.

Por otro lado, mencionan que se han basado en un modelo de difusión de mensajes conocido como *"word of mouth"*, que asume que la gente tiende a usar frases clave similares para hablar sobre un evento, transmitidas sin apenas cambios de una persona a otra. En su enfoque, la identificación de estas frases claves de eventos específicos es la que identifica a los núcleos o la esencia de cada tema de conversación. Esta idea sobre el modo en el que se difunden los mensajes en Twitter es compartida por nosotros, tal y como se verá en la etapa de formación de temas iniciales.

3.4. Detección de temas en Twitter usando filtrado agresivo y clustering jerárquico

En la sección anterior de ha descrito un sistema que en lugar de emplear frecuencias busca relacionar temas similares con técnicas de clustering. Este enfoque recibe atención en muchos otros trabajos como [22], [23], [24].

Un ejemplo concreto de este tipo de sistema es [25], cuyas dos principales características son el filtrado agresivo de los tweets y su posterior agrupamiento mediante clustering jerárquico. Según detallan los autores, partiendo de un conjunto de términos clave conocidos relativos al evento, se recopila un conjunto de titulares que resumen los temas comentados delimitado el periodo de interés, incluyendo otros recursos como fotos y los tweets más representativos.

Una de las primeras operaciones a realizar sobre los tweets recopilados es el filtrado de los mismos aplicando medidas bastante restrictivas sobre la estructura y contenido de los

tweets, eliminando los que no cumplan ciertas condiciones, por lo que permiten reducir la cantidad de datos a procesar. Algunas de estas operaciones de filtrado que proponen son:

1. Eliminar stopwords, aquellos tweets que contengan más de dos menciones o hashtags o presenten menos de 4 tokens.
2. Eliminar del vocabulario términos que ocurran en menos de cierto porcentaje de tweets, calculado este valor como un porcentaje a partir del tamaño real del vocabulario.
3. Eliminar los tweets que hayan quedado con menos de 5 palabras del vocabulario tras los pasos anteriores.

Una vez realizado el proceso de filtrado del conjunto de datos, la agrupación de los mensajes en temas se basa en clustering jerárquico que emplea para comparar tweets la similaridad por coseno de la matriz tweet-término. En concreto, se corta el dendograma originado a determinado umbral 0.5, un valor intermedio que según los autores consigue evitar que se formen clusters poco homogéneos y a la vez evita producir fragmentación (mismo tema en dos clusters).

Las fases finales del algoritmo evalúan los clusters resultantes en función de la frecuencia de términos en intervalos temporales según una fórmula denominada *df - idf*$_t$, y dándole mayor importancia a las entidades nombradas reconocidas. Para cada uno de los clusters, se selecciona el primero de los tweets que lo componen ordenados cronológicamente como titular. Posteriormente, indican que aplican clustering de nuevo empleando únicamente estos titulares, permitiendo reducir así la fragmentación que se haya podido producir. El resultado muestra un titular para cada cluster, en concreto el primero según estén ordenados por orden cronológico, igual que en la primera etapa de clustering.

Un aspecto que mencionan, compartido por nuestro método, es que las técnicas de stemming pueden llegar a producir clusters de mala calidad al emplearse en mensajes de poca longitud como son los tweets, por lo que en nuestra propuesta no se aplican.

Por otra parte, de esta propuesta hemos tomado ideas como pedir un nivel de exigencia alto en cuanto al número de palabras de los tweets o eliminar menciones y ser estrictos con los umbrales, permitiendo originar temas compactos, tal y como indicaremos al detallar nuestro algoritmo.

3.5. Un caso de estudio en Text Mining: Interpretando datos de Twitter de la World Cup

Otro de los métodos de análisis de Twitter investigados nos muestra una aplicación práctica a un evento concreto, exponiendo el funcionamiento del sistema al analizar el comienzo de la Copa Mundial de fútbol del 2014 [26], permitiendo encontrar los subtemas más relevantes.

En concreto, el método que proponen aplica las siguientes técnicas:

- Recopilación de tweets que contengan las palabras clave "*world cup*".
- Para solucionar el problema de los tweets outliers, eliminan las stopwords y aplican técnicas de stemming.
- Posteriormente, aplican 3 algoritmos de clustering sobre los tweets para detectar los outliers no eliminados en el paso anterior. Si un tweet es marcado en 2 de los 3 de los algoritmos como ruido o frontera, es eliminado del conjunto de tweets. Los algoritmos empleados fueron K-means y DBSCAN, aplicando clustering repetidas veces y haciendo uso de matrices de consenso.
- Como medida de similaridad emplean el coseno del TF-IDF, destacando que es más adecuado el coseno de los vectores que el cálculo de la similaridad por distancia euclídea porque es más rápido, trabaja mejor con matrices dispersas y calcula las distancias entre los tweets independientemente de sus longitudes.
- Una vez eliminan los tweets outliers, aplican dos técnicas de clustering sobre los tweets resultantes: K-means (empleando una matriz laplaciana para determinar la k) y Non-negative Matrix Factorization (NMF), mencionando que obtienen resultados similares pero con mejor eficiencia de cómputo en NMF.
- Finalmente, incluyen una visualización de los clusters de manera gráfica (Figura 4), indicando sus palabras clave como etiqueta de cada uno de ellos.

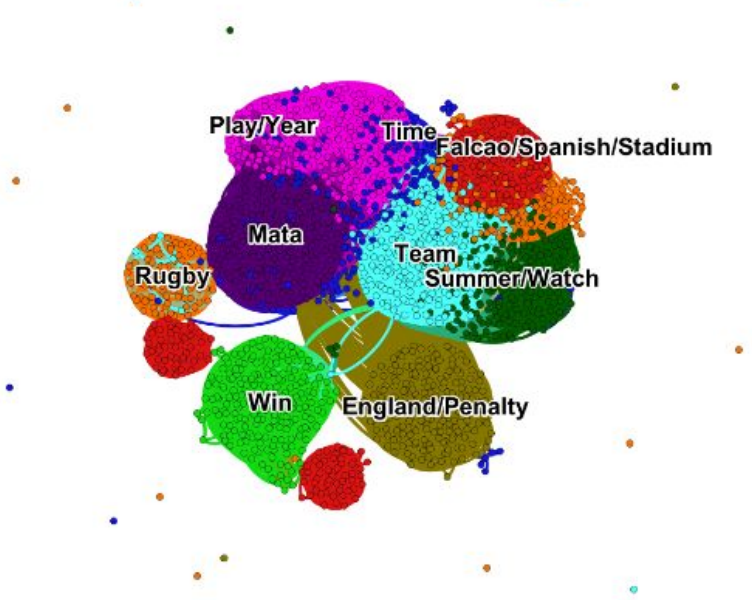


Figura 4: Representación de los detectados por el sistema analizado en la World Cup aplicando el algoritmo K-means.

El análisis de este sistema se vio reflejado en dos medidas aplicadas en nuestra propuesta. La primera de ellas fue la decisión de eliminar los retweets del algoritmo de detección de temas porque, tal y como señalan, pueden llegar a dar una idea de la popularidad del tema pero semánticamente únicamente repiten la información.

La representación visual de los clusters que emplean también fue tomada como idea para visualizar los temas agrupados por relación textual, tal y como se detalla en el capítulo de Presentación de resultados, aunque en nuestro caso lo que se representa son los temas relacionados y no los tweets.

3.6. TweetMotif

El último de los sistemas analizado en profundidad es *TweetMotif* [27]. El método que propone organiza los tweets que contengan ciertas palabras clave en temas. Además, para cada tema permite buscar temas relacionados.

Para ello, aplican inicialmente un proceso de tokenización en n-gramas (indican que en sus experimentos han probado N=1..3) y a partir de ellos identifican las frases más distintivas del conjunto de tweets. Para ello emplean criterios de frecuencia, asumiendo que las frases representativas de un tema estarán muy presentes en un subconjunto de los mensajes, pero no en el resto. En concreto, siguen la siguiente fórmula:

$$Score = \frac{P(phrase \mid tweet\ result\ set)}{P(phrase \mid general\ tweet\ corpus)}$$

Una vez se detectan las frases candidatas a constituir temas, aplican un proceso de unión de temas similares. Por ejemplo si uno de los temas detectados tiene como frase representativa *"swine flu"* y otro *"flu"*, descartan el tema *"flu"*, porque encuentran que *"swine flu"* es un tema más informativo. También aplican un proceso de unión de temas, imponiendo como criterio que sus frases representativas tengan más del 90% de similaridad de Jaccard. Posteriormente, forman conjuntos de mensajes que discuten cada tema evaluando la similaridad de Jaccard de cada trigrama del mensaje con las frases representativas del tema. De esta manera, si se supera un 65% de similaridad se considera que un mensaje pertenece a dicho tema.

Este método comparte con nuestra propuesta el hecho de aplicar a los temas detectados un proceso de fusión, tal y como detallaremos en la propuesta del algoritmo. Así, temas que en principio pueden haberse detectado como diferentes, pueden dar lugar a un tema conjunto, si cumplen determinadas condiciones de similaridad. Es en la función de similaridad empleada donde radica la principal diferencia con nuestra propuesta, dado que ellos emplean una representación en n-gramas sobre la que aplican el índice de similaridad de Jaccard para fusionar temas, mientras que en nuestro caso hacemos fusión de ventanas contiguas empleando la similaridad de los tweets considerados como centro.

Resumen de propuestas

Al comienzo de este capítulo expusimos diversas caracterizaciones en cuanto a la metodología seguida para detectar noticias en Twitter. Tras estudiar los diferentes trabajos nos surge una nueva clasificación en dos grandes grupos generales:

P1. Similaridad entre tweets: basados generalmente en la comparación de tweets, normalmente en su aspecto textual, para agrupar mensajes similares, a menudo mediante técnicas de clustering. El problema asociado a estos enfoques, es que en general no tienen en cuenta la proximidad temporal de los mensajes, por lo que agrupa tweets muy alejados temporalmente. Por ejemplo, imaginemos que en un partido un jugador marca dos goles. Desde este punto de vista, los mensajes de las reacciones de los usuarios a ambos goles serían muy similares, aunque ocurran en momentos muy alejados y se atribuyen a dos hitos diferentes. Sin embargo, este tipo de enfoques los considerarían un tema único.

P2. Análisis de frecuencias: el segundo enfoque se basa en el análisis del ritmo de publicación de los mensajes, permitiendo detectar situaciones anómalas (picos), los cuales se analizarán, marcándose como susceptibles de incluir un "tema nuevo". El problema que suelen presentar estos trabajos es que asumen que en un pico se habla únicamente de un tema. En nuestra opinión un pico corresponde en efecto a un hito del evento (como en el ejemplo anterior, un gol), pero la reacción en la red social pueden ser muchos temas entremezclados y opiniones encontradas (en el caso del gol, mientras unos usuarios lo celebran, otros denuncian un posible fuera de juego…). Este tipo de técnicas no son siempre capaces de distinguir estas diferentes reacciones a un hito concreto. Además, fuera de estos máximos locales también pueden encontrarse temas de interés.

En este trabajo hemos tratado de solventar estos problemas combinando de forma ordenada las dos propuestas. En primer lugar, utilizamos una división en ventanas de tweets, que nos permite separar temas con una diferencia temporal apreciable. En segundo lugar, no nos basamos en frecuencias, sino que dentro de cada ventana, empleamos la similaridad textual. Por tanto, nuestra propuesta explota las dos características que son necesarias para que dos tweets se consideren como altamente relacionados (cercanía temporal y textual), permitiendo además diferenciar conversaciones que debatan aspectos distintos a un mismo hito.

4. Propuesta general

Este capítulo presenta definiciones de conceptos fundamentales que se utilizarán en el resto del trabajo, tales como el concepto de noticia o las propiedades que debe cumplir la similaridad entre tweets. También se muestra el esquema general de la propuesta, así como los principios básicos que representan los objetivos a cumplir.

4.1. Noticias o temas

A lo largo del trabajo, se denomina **noticia o tema** a un conjunto de mensajes de Twitter que presentan cierto grado de similitud entre sí, y que se reparten de forma continua a lo largo de un determinado periodo de tiempo.

Cuando nos referimos a continuidad durante un cierto periodo de tiempo, queremos indicar que un tema no presenta "huecos temporales" de tamaño significativo, es decir, periodos en los que no se encuentra ningún tweet referente a la noticia. Por tanto, el mismo tema comentado en momentos diferentes corresponderá, bajo nuestro punto de vista, a temas diferentes.

Puede argüirse que la misma noticia puede repetirse en diferentes momentos y que debería ser detectada como tal. Sin embargo, pensamos que en ocasiones también tiene sentido considerar que la noticia "repetida" es en realidad diferente. Pensemos por ejemplo en emplear la detección de noticias para monitorizar catástrofes naturales, tales como terremotos. Sería deseable que se detectaran tweets correspondientes a diferentes réplicas del mismo seísmo como noticias diferentes, aunque en una fase posterior de análisis todas estas noticias se pudieran agrupar por similaridad temática.

Por ello hemos pensado que la solución más adecuada es:

1. Considerar noticias separadas por un cierto intervalo temporal como distintas.
2. Presentar al usuario posibles relaciones entre temas de contenido similar.

Como podemos observar, la definición de tema o noticia propuesta depende, al menos, de los siguientes parámetros:

- Tamaño del “hueco”, también denominado en el resto del trabajo como “ventana”.
- Una definición adecuada de la noción de similaridad entre tweets.
- Grado de similaridad entre tweets requerido para formar parte de la misma noticia.

En cuanto al tamaño del hueco o ventana, debemos tener en cuenta que los tweets no se emiten al mismo ritmo durante todo el evento estudiado. Por ejemplo, dos tweets separados por 5 minutos pueden ser considerados próximos temporalmente en mitad de la noche, pero no así en los momentos de mayor frecuencia de publicación.
Por tanto hemos considerado que el intervalo temporal representado por una ventana debe depender de la frecuencia de tweets en el momento considerado. En particular definimos

ventana en un momento T como el intervalo temporal [T,T+h] que contiene una cantidad V de tweets. En nuestro enfoque entendemos que V es una constante (por ejemplo 10.000 tweets), mientras que h es una variable que depende de T.

El algoritmo que presentamos recibe como entrada los tweets publicados durante el intervalo de tiempo a estudiar. Estos tweets se ordenan cronológicamente, y se reparten en ventanas de V tweets. Cada ventana se procesa de manera individual, detectando los temas que han ocurrido en su tiempo de vida. Finalmente examinamos si dos temas de ventanas adyacentes pueden considerarse el mismo tema, procediendo a su fusión. De esta forma reducimos el nivel de fragmentación de los temas.

Una ventaja adicional del uso de ventanas es la eficiencia. El algoritmo que vamos a presentar tiene un orden de complejidad cuadrático, y de esta forma hacemos que dependa del tamaño de la ventana y no del total de tweets. Además, tratar todos los tweets simultáneamente requeriría una cantidad de tiempo excesiva. En nuestro caso, hemos considerado un tamaño V de 10.000 tweets como un compromiso adecuado entre eficiencia y nuestra capacidad de detección de temas.

Hay que señalar que, aunque temas detectados en ventanas diferentes pueden terminar fusionándose dando lugar a un tema único, lo que no puede ocurrir es que se fusionen temas pertenecientes a ventanas no consecutivas.

4.2. Similaridad y distancia

En cuanto a la definición de similaridad, hemos intentado que nuestra propuesta sea independiente de la noción elegida. Para ello, asumimos la existencia de una función *Sim* que, dados dos tweets t_1 y t_2 pertenecientes al conjunto de mensajes M, devuelva un valor normalizado que cuantifique la similaridad entre estos dos tweets en el intervalo [0,1].

$$Sim : M \times M \rightarrow [0, 1]$$

La función debe verificar las siguientes propiedades básicas:

1. $Sim(t_1, t_2) = 1 \Leftrightarrow t_1 = t_2$ *(Identidad de indiscernibles)*
2. $Sim(t_1, t_2) = Sim(t_2, t_1)$ *(Simétrica)*
3. $Sim(t_1, t_2) \geq 0$ *(No negatividad)*

La primera propiedad indica que un tweet es totalmente similar a sí mismo y que, viceversa, si dos tweets tienen la máxima similaridad son realmente el mismo. Como veremos, este concepto depende a su vez del de *igualdad* entre tweets. La propiedad simétrica o conmutativa nos indica que la similaridad no depende del orden. Finalmente, la no negatividad fija un límite inferior a la similaridad (el 0).

En cierto sentido, la función *Sim* recuerda a una función distancia. Las distancias se definen mediante las propiedades:

1. $d(x, y) \geq 0$, y $d(x, y) = 0 \Leftrightarrow x = y$ (*No negatividad + Identidad de indiscernibles*)
2. $d(x,y) = d(y, x)$ (*Simétrica*)
3. $d(x, z) \leq d(x, y) + d(y,z)$ (*Desigualdad triangular*)

Las diferencias entre ambas definiciones son que en lugar de dar el valor máximo del intervalo (1) cuando los elementos son iguales, la distancia asigna el valor 0, así como el añadido de una nueva propiedad, la desigualdad triangular. Cabe preguntarse entonces si una función d, definida como $d(x,y) = 1 - Sim(x, y)$ sería, en efecto, una distancia. La respuesta es que, en general, esta función no será una distancia porque no podemos asegurar se verifique la desigualdad triangular. Veamos un contraejemplo. Partimos de un conjunto M = {a,b,c}, y definimos una función *s* como:

s	a	b	c
a	1	0.7	0.2
b	0.7	1	0.7
c	0.2	0.7	1

Podemos comprobar que *s* verifica las tres propiedades, y que es por tanto una similaridad. Sin embargo, $d(x,y) = 1 - s(x, y)$ no es una distancia, porque en particular no se cumple $d(a, c) \leq d(a, b) + d(b, c)$ ya que sustituyendo queda $1 - 0.2 \leq 1 - 0.7 + 1 - 0.7$, es decir $0.8 \leq 0.6$. Es decir, en general no podemos asumir que una similaridad tenga una distancia asociada a partir de la definición $d(x,y) = 1 - s(x, y)$, aunque evidentemente en ocasiones sí se pueden construir distancias a partir de similaridades por éste o por otros métodos.

Sin embargo, resulta muy sencillo comprobar que un caso particular del resultado recíproco sí se cumple: dada una distancia d con valores entre 0 y 1, la función $s(x,y) = 1 - d(x, y)$ es siempre una similaridad.

Los detalles de la función de similaridad que se ha empleado en la implementación se detallan en el capítulo Similaridad entre tweets, indicando la motivación de su uso en comparación con otras posibles funciones.

4.3. Desde los tweets hasta los temas

Antes de explicar en detalle el método propuesto, es interesante describir de forma general los pasos que llevan desde la obtención del conjunto de los datos inicial formado por los tweets del evento, hasta la identificación de los temas y su presentación al usuario.

El proceso general se puede resumir en los 4 pasos siguientes, representados en el diagrama de flujo de la Figura 5.

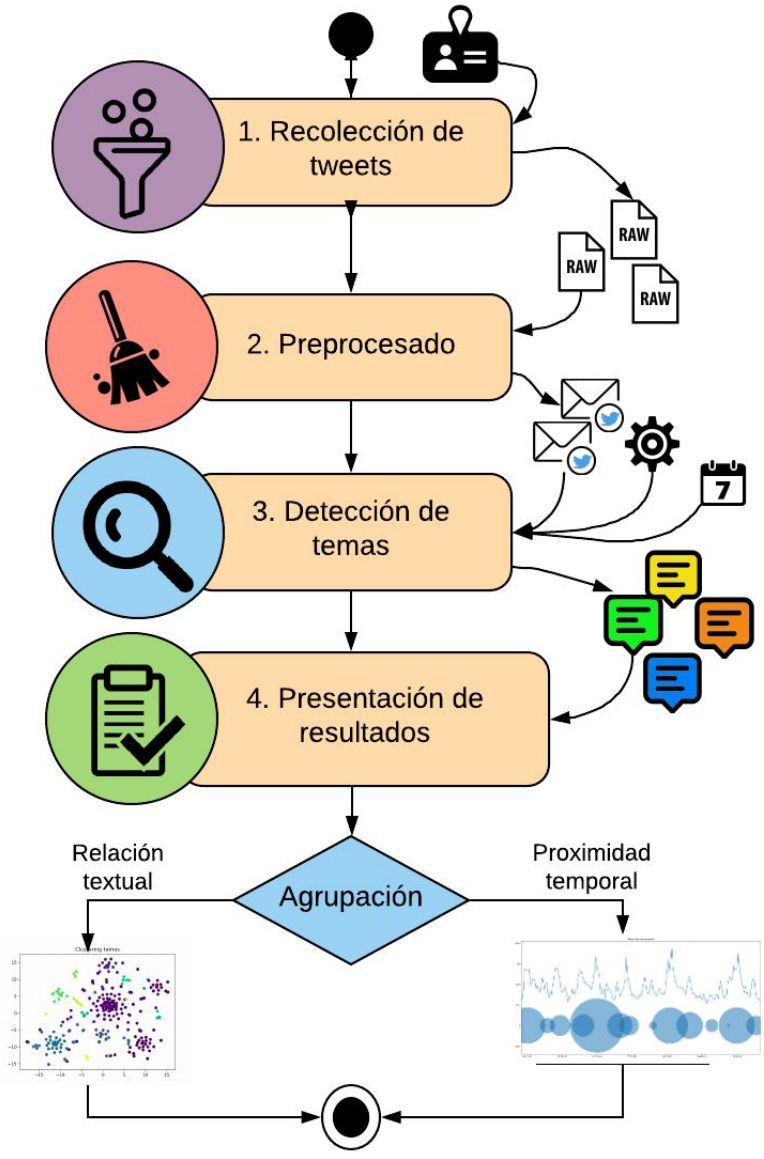


Figura 5: Diagrama de flujo general del método propuesto.

Inicialmente, en una primera fase de **recolección de tweets**, un programa lanzado al inicio del intervalo temporal a estudiar se encarga de monitorizar la actividad de Twitter, almacenando los tweets que contengan determinadas palabras clave o hashtags, que

permiten recopilar únicamente tweets relativos al evento. Esta fase únicamente necesita como datos de entrada las credenciales que permiten la conexión con la API de Twitter y la lista de palabras clave.
Después, se pasa a una fase de **preprocesado** de la información, dado que la API empleada para la recopilación de tweets incluye mucha información irrelevante para la detección de temas, la cual se elimina en esta fase. El resultado generado se almacena en una base de datos de tweets.
La fase más importante del método desarrollado es la **detección de temas**, que recibe como entradas los tweets ya filtrados y limpiados y los parámetros de configuración, así como el intervalo temporal del evento que se desee analizar. Como salida, se genera un archivo que recoge los temas de conversación detectados que se han producido durante el tiempo analizado.
Para terminar, la fase de **presentación de resultados** muestra visualmente los temas, de una manera comprensible y amigable, agrupándolos según dos criterios complementarios:

- *Por proximidad temporal*: se agrupan los temas de conversación de manera que se resaltan los momentos temporales en los que hay una alta concentración, indicando que se trata de momentos clave en el transcurso del evento.
- *Por relación textual: se* agrupan temas de conversación que presentan relación temática de acuerdo con el criterio de similaridad elegido.

4.4. Principios básicos

El método de análisis de eventos en Twitter propuesto en este trabajo se guía por dos máximas principales:

1. No emplear criterios basados en la frecuencia de publicación de mensajes como método de detección de temas.

2. No asumir criterios *a priori* para ponderar la importancia de los diferentes temas.

La motivación de la primera máxima es que una frecuencia baja de mensajes no implica que no haya temas interesantes que se estén discutiendo, y asumir lo contrario iría en contra de la segunda máxima. Por otro lado, aunque la existencia de un máximo de frecuencia de twitteo sí suele implicar un momento álgido o hito en el evento, esto no implica la existencia de una única noticia, sino que es común que, como respuesta a este momento, se generen simultáneamente temas de conversación que versen sobres distintos aspectos del momento.

Respecto a la segunda máxima, se considera que debe ser el propio usuario el que decida qué noticias le parecen más relevantes. Experimentos con muy diversos factores como el número de tweets, la frecuencia temporal de publicación o el número de usuarios participantes en el tema, entre otros, mostraron que cualquier criterio deja fuera temas que para algunas personas pueden ser relevantes. La contrapartida a esta máxima es que, al no haber ninguna fase de filtrado y eliminación de temas poco relevantes, se genera una gran cantidad de temas diversos. Para solucionar este inconveniente, en lugar de filtrar las

noticias encontradas, se ha optado por investigar las posibles formas de agrupación anteriormente mencionadas y que se detallarán en la sección Agrupación de temas

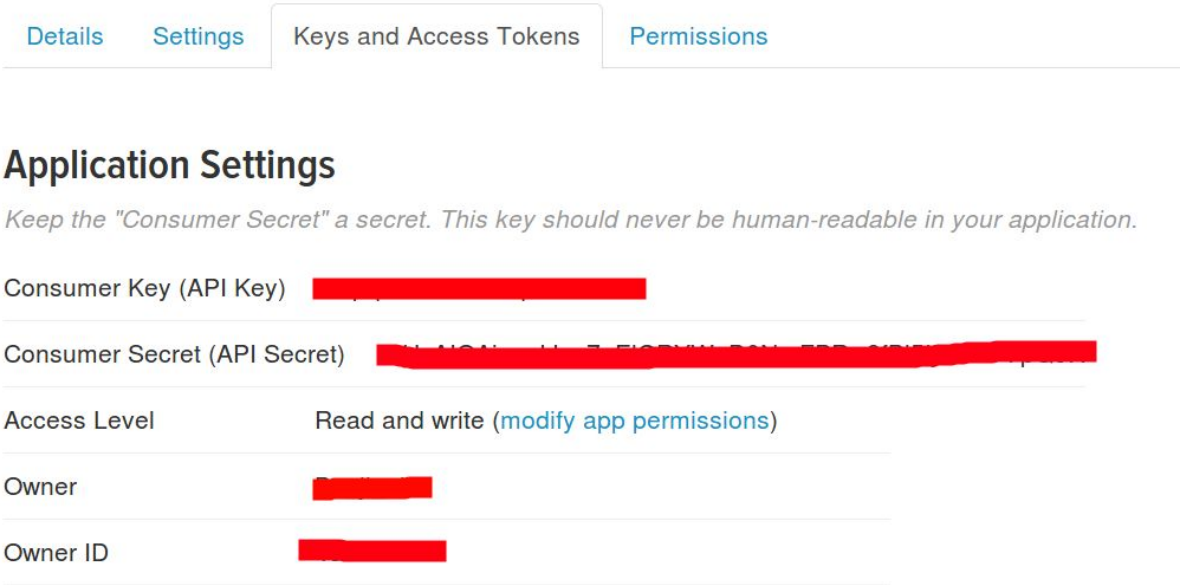
A lo largo de los siguientes capítulos se irá detallando cada una de las fases mencionadas, siguiendo el orden de la Figura 5.

5. Recopilación y preprocesado de datos

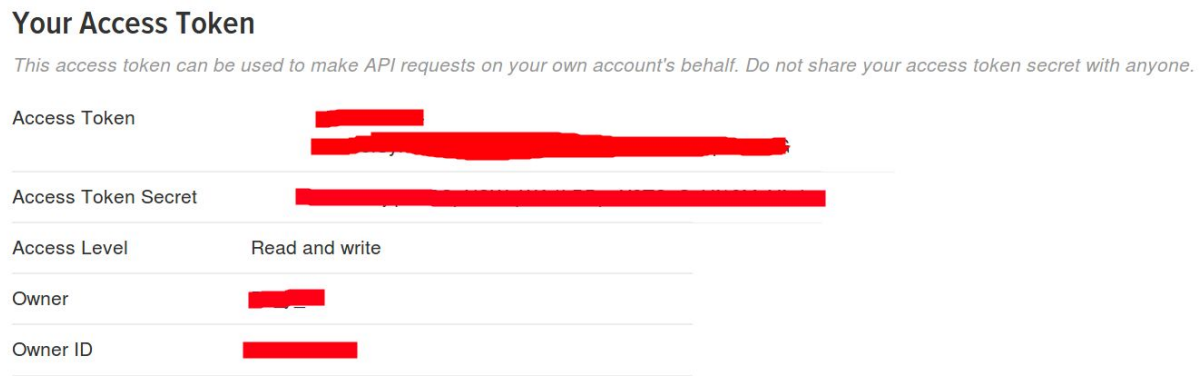
Como cualquier aplicación de análisis de datos, nuestro método de detección de temas comienza con una fase de recopilación de los mensajes de Twitter que serán utilizados. Estos mensajes relativos al evento a estudiar se obtienen mediante la monitorización de Twitter durante el intervalo de tiempo considerado. Para determinar qué mensajes están relacionados con el evento considerado, aprovechamos la posibilidad que proporciona el API de Twitter de descargar solo los mensajes que contienen algunas palabras clave. La selección de estas palabras clave suele hacerse atendiendo a los hashtag o lista de hashtags que, para grandes eventos, proporciona la propia organización o patrocinador, indicando a los usuarios cómo etiquetar los mensajes para que sea más sencillo acceder a información relativa al evento.

En concreto, para llevar a cabo esta recopilación de mensajes, partimos de un script Python desarrollado por el profesor Enrique Martín Martín que emplea Tweepy [28], una biblioteca de fácil uso mediante la cual se monitoriza Twitter usando su API.

Para poder acceder a los datos de Twitter, es necesario que el usuario tenga configurada con la cuenta en la red social una aplicación, la cual proporciona una serie de credenciales que permitirán el acceso a los tweets que se publiquen. Entre las credenciales necesarias se encuentran una clave de la API y un token de acceso para realizar las peticiones, tal y como vemos en la Figura 6.



a) API key y API Secret.



b) Access Token y Access Token Secret.

Figura 6: Credenciales de acceso a Twitter vía API necesitadas por Tweepy (a y b).

El uso de este script es bastante sencillo, siendo necesario únicamente la ruta del directorio en el que almacenar los tweets recopilados, el nombre de fichero deseado en el que se almacenarán y la lista de términos clave por la que filtrar separados por espacios.

El usuario deberá ejecutar el script de la forma:

```
$ python twitter_listener.py <ruta_carpeta> <nombre_ficheros> <t1> <t2>
```

Un ejemplo de esta llamada sería:

```
$ python twitter_listener.py /home/bea/TweetsEventos/ Twitterclasico
clásico clasico elclásico elclasico
```

En caso de errores, normalmente debidos a que los datos se reciben a un ritmo superior al que el script es capaz de procesar, se intentará restablecer la conexión automáticamente.

```
{ ⊟
   "contributors":null,
   "coordinates":null,
   "truncated":false,
   "geo":null,
   "quote_count":0,
   "in_reply_to_user_id":null,
   "is_quote_status":false,
   "timestamp_ms":"1521971997619",
   "filter_level":"low",
   "in_reply_to_screen_name":null,
   "retweeted":false,
   "user":{ ⊞ },
   "lang":"pt",
   "id":977847547959152640,
   "created_at":"Sun Mar 25 09:59:57 +0000 2018",
   "retweet_count":0,
   "reply_count":0,
   "in_reply_to_user_id_str":null,
   "favorited":false,
   "entities":{ ⊞ },
   "text":"EU SOU MUITO OT\u00c1RIO",
   "source":"<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android</a>",
   "favorite_count":0,
   "place":{ ⊞ },
   "in_reply_to_status_id":null,
   "id_str":"977847547959152640",
   "in_reply_to_status_id_str":null
}
```


Figura 7: Ejemplo de mensaje de Twitter recopilado por el script en Python en formato JSON.

Tal y como se observa en la Figura 7, los tweets recopilados se almacenan en formato JSON conteniendo gran cantidad de campos. Estos documentos JSON incluyen desde los datos completos del usuario en el campo `user`, hasta la fecha de publicación del tweet en el campo `created_at`, o los datos de la plataforma en la que se publicó, indicado por el campo `source`, entre otros.

Con esta gran cantidad de información almacenada, si nos encontramos frente a un evento que dure bastante tiempo y sobre el cual se publican tweets con una alta frecuencia, nos podemos encontrar con archivos de gran tamaño, formados únicamente por texto.

Por ello, los datos recogidos accediendo a la API de Twitter se importan a MongoDB [29], un sistema de base de datos NoSQL que proporciona una alta disponibilidad, rendimiento y escalabilidad. MongoDB nos ofrece una capacidad de acceso a los datos que lo hace ideal para el tratamiento de millones de documentos, un volumen de datos que se encuentra en algunos de nuestros conjuntos de tweets.

Además, MongoDB es una base de datos orientada a documentos que almacena su información en un formato similar a JSON, no siendo necesario realizar ninguna conversión de los ficheros recopilados mediante Tweepy para importar los tweets a la base de datos.

En concreto, este paso de importación de los datos se puede realizar fácilmente mediante la herramienta `mongoimport`, en concreto ejecutando:

```
$ mongoimport -d <nombre-db> -c tweets <ruta_fichero>
```

Se sustituirá `<nombre-db>` por el nombre de la base de datos que queramos y `<ruta_fichero>` por la ruta al fichero JSON que contenga los tweets recopilados. Se debe usar el nombre de la colección *`tweets`*.
Un ejemplo de llamada sería:

```
$ mongoimport -d partido -c tweets tweets_psg.json
```

Una vez los datos se han importado a MongoDB, se realiza la denominada fase de preprocesado, limpiando y modificando los tweets de modo que se puedan tratar de manera más rápida y eficiente en la fase de detección.

En concreto, los pasos más importantes del pre-procesado incluyen:

- Marcar los tweets que sean retweet para distinguir entre tweets retwitteados y tweets originales.
- Establecer un campo que indique el número de RT que tiene un determinado tweet original.
- Extraer las menciones, hashtags y urls de cada tweet.
- Establecer un campo en el que se almacene el texto del mensaje eliminando las posibles URLs.
- Establecer un campo en el que se almacene el texto del mensaje eliminando además de las URLs, las menciones de usuarios y los posibles iconos y emoticonos que incluya.
- Establecer un campo que permita detectar los tweets con contenido textual vacío. Esto marcará dichos mensajes como inútiles desde el punto de vista semántico para la comparación de tweets por su similitud textual, evitando que se incluyan en el análisis.
- Eliminar el resto de campos, ya que resultan innecesarios desde el punto de vista de nuestro análisis de temas.

Para llevar a cabo este proceso se deberá arrancar MongoDB sobre la base de datos en la que se importaron los tweets recopilados y ejecutar las operaciones que se detallan en el script *`preprocesado.js`*.

Los mensajes de Twitter resultantes que se almacenen en la base de datos contendrán únicamente los campos pertinentes, reduciendo el tamaño de almacenamiento y por tanto el

tiempo que se tardará en acceder a los mismos. Por esta razón además, se creará un índice por fecha de publicación de los tweets, dado que los tweets serán organizados y procesados en orden cronológico, permitiendo así agilizar las consultas a la base de datos.


```
{ ⊖
   "_id":"5af8768314461e5a7b284a34",
   "text":"I just hope we don't end last again. They don't deserve it 🇪🇸 #Eurovision",
   "created_at":"2018-05-12T19:16:45.000Z",
   "id":"995382286664306688",
   "user":{ ⊞ },
   "RT":false,
   "num_RT":0,
   "hashtags":[ ⊞ ],
   "text2":"I just hope we don't end last again. They don't deserve it 🇪🇸 #Eurovision",
   "text3":"I just hope we don't end last again. They don't deserve it  #Eurovision",
   "string_created_at":"Sat May 12 19:16:45 +0000 2018"
}
```


Figura 8: Ejemplo de documento de un tweet almacenado en MongoDB tras el preprocesado.

Los mensajes de Twitter que se almacenarán en MongoDB después del preprocesado tendrán un formato como el del mensaje de ejemplo de la Figura 8 y, en concreto, en los documentos encontraremos los campos siguientes, de los que detallamos su uso:

- ***_id:*** el identificador único que genera automáticamente MongoDB para cada documento importado.
- ***created_at***: campo de tipo fecha que almacena la fecha de publicación del tweet que nos permite ordenar cronológicamente los mensajes.
- ***id***: identificador del tweet concreto, que permite distinguir un tweet individual de otros que, como los retweets, pueden tener su mismo texto.
- ***user:*** documento que incluirá los campos `screen_name` con el alias del usuario en Twitter e `id`, que identifica de manera única al usuario, al poder modificar su alias. Este documento inicialmente contenía numerosos campos con información del perfil del usuario, como el color de fondo o de la letra escogidos, que resultan innecesarios para nuestro análisis.
- ***RT:*** que contendrá el valor `true` en caso de tratarse de un mensaje retweet o `false` en caso de ser un mensaje original. De esta manera podremos recuperar únicamente los mensajes originales, dado que los retweets no suponen más que información repetida que no aporta ningún significado adicional al original.
- ***num_RT***: número que indicará el número de retweets que tiene el mensaje, en caso de ser un mensaje original. Este campo nos permitirá evaluar los temas formados, incluyendo el impacto general que ha tenido, atendiendo al número total de retweets de los mensajes que lo forman.
- ***hashtags***: array que contendrá los diferentes hashtags extraídos del campo `text`. Este campo se estableció con la intención de comprobar en un futuro posibles relaciones entre hashtags de distintos temas, aunque en esta propuesta del algoritmo de detección no se usa.

- ***mentions***: array que contendrá las menciones a otros usuarios extraídas del campo *text*. Establecido, al igual que los hashtags, para estudiar en un futuro relaciones entre usuarios de temas detectados.
- ***urls***: array que contendrá los urls de recursos como imágenes extraídas del campo *entities*, el cual es eliminado. En determinado momento del desarrollo se sopesó la idea de realizar la comparación de imágenes, motivo por el cual se incluye este campo, pero finalmente quedó pendiente como propuesta de trabajo futuro.
- ***text***: cadena de texto que almacena el mensaje original del usuario. Aunque para el cálculo de la similaridad entre tweets se use el campo *text3* descrito más adelante, este campo nos permite acceder al mensaje original.
- ***text2***: cadena de texto resultante de eliminar del campo *text* las posibles urls. Aunque para la operación de similaridad hemos usado *text3* (descrito a continuación), hemos decidido almacenar esta versión del tweet sin urls para posibles trabajos futuros.
- ***text3:*** cadena de texto resultante de eliminar del campo *text* las posibles urls, menciones e iconos o emoticonos. Este será el campo que se emplee para calcular la similaridad entre los tweets, dado que las urls, menciones e iconos no han sido tenidos en cuenta en esta tarea.
- ***voidtext:*** tal y como se ha mencionado, es importante poder distinguir cuándo un tweet contiene información semántica útil de cuándo no, en función de si el campo *text3* queda vacío. Este campo estará presente en los mensajes en los que esto ocurra, estableciendo su valor a true, permitiendo excluir esos mensajes del análisis.

Para evitar que un usuario de este sistema tenga que realizar este proceso de importación y preprocesado de tweets ejecutando comando por comando, se ha creado un script que ejecuta todas las operaciones automáticamente.

Este script se invocará de la manera siguiente:

```
$ python tweet-import-preprocess.py <ruta_bd> <nombre_bd> <ruta_tweets>
```

donde *<ruta_bd>* es la ruta completa de la carpeta donde se quiera almacenar la información de la base de datos (no debe existir o se borrará), *<nombre_bd>* es el nombre que le deseemos dar a la base de datos de MongoDB y *<ruta_tweets>* es la ruta del archivo generado por el paso de recopilación de tweets.

Un ejemplo concreto de de llamada sería:

```
$ python tweet-import-preprocess.py carpeta/ partido PSG/datospsg2
```

6. Detección de temas

Una vez se ha realizado la importación del fichero de tweets en formato JSON a la base de datos MongoDB y se ha llevado a cabo el preprocesado, da comienzo la fase principal del método propuesto, encargada de detectar los temas o noticias más relevantes que se han producido en Twitter durante el evento concreto a analizar.

Para la implementación se ha empleado Python 3.5.5, en concreto la distribución Anaconda [30] en su versión 4.3.34, que incluye instalados por defecto numerosos paquetes básicos para aprendizaje automático y análisis de datos.

El proceso desde la obtención de tweets mediante la conexión a la base de datos hasta la escritura en un fichero de los temas resultantes, está formado por un conjunto de etapas, las cuales podemos visualizar de forma general en el diagrama de flujo de la Figura 9.

Cada una de estas etapas comprende un conjunto de acciones sobre los mensajes. Se irá explicando de manera progresiva cada una de ellas en las siguientes secciones, con objeto de entender los detalles del procesamiento.

6.1. Obtención de tweets

El primer paso de nuestro algoritmo de detección es la carga de los tweets que se van a analizar. En esta etapa son necesarios principalmente 3 elementos de entrada del algoritmo, los cuales contienen detalles necesarios para acceder a los datos y realizar el análisis:

1. Nombre de la base de datos en la que se han almacenado los tweets preprocesados en la colección *tweets* y puerto en el que corre MongoDB (asumiendo que lo hará en localhost).
2. Ruta del directorio de trabajo, en el que se deberá encontrar un fichero de configuración llamado `config.json` que define los valores de los parámetros que recibe el algoritmo de detección. En esta ruta será además donde se almacenen los archivos de salida del algoritmo. El formato de este fichero se detalla en el Anexo A.
3. Detalles temporales del intervalo de tiempo a estudiar del evento.

Los parámetros (1) y (2) serán indicados en la propia llamada al script de detección, que deberá ser de la forma:

```
$ python tweet-analysis.py <ruta_trabajo> <nombre_bd> <puerto>
```

Un ejemplo de ejecución sería:

```
$ python tweet-analysis.py PartidoPSG/ PSG 27017
```

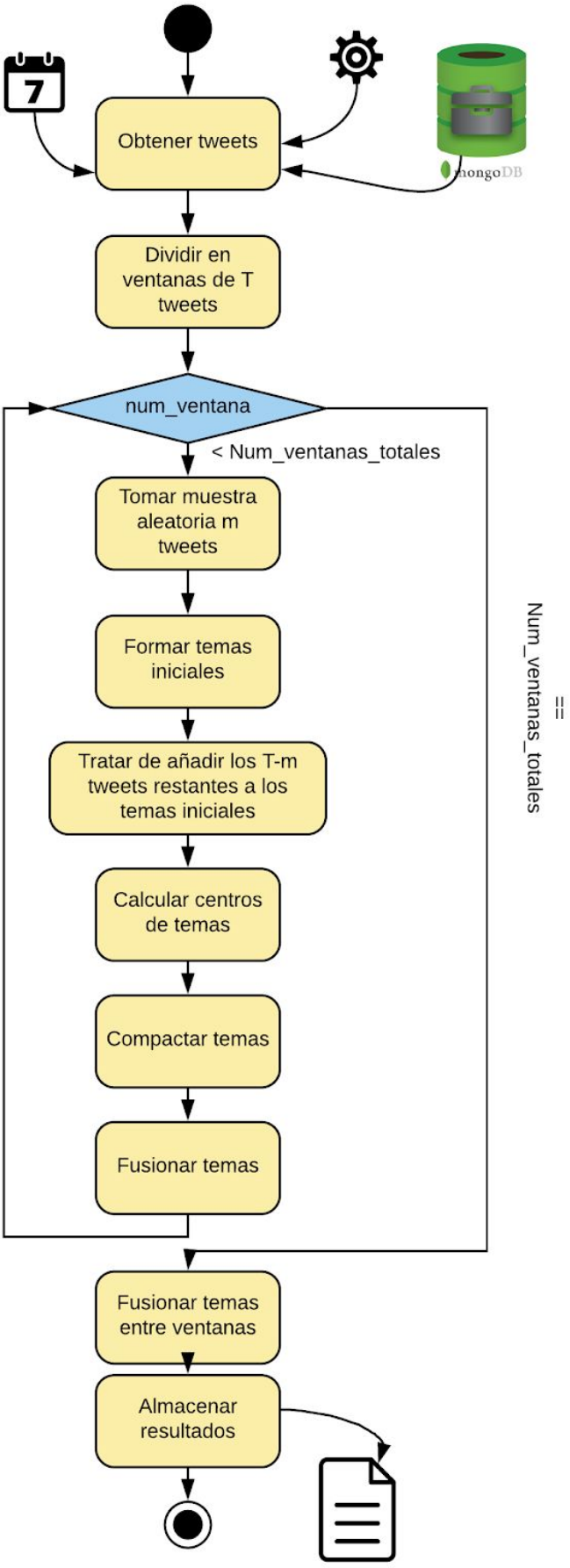


Figura 9: Diagrama de flujo de la fase de detección de temas.

Aunque en un principio podríamos analizar todos los tweets obtenidos durante la fase de recopilación de tweets, a veces puede ser interesante estudiar cierto intervalo de tiempo de un evento, por lo que se deberán introducir las fechas y horas de inicio y fin del análisis, por lo que es necesario el parámetro (3).

Para obtener esta información, se mostrará nada más ejecutar el script una interfaz donde podremos seleccionar de manera sencilla los límites del intervalo temporal, tal y como vemos en la Figura 10.

Figura 10: Selección de los detalles temporales del evento a analizar.

Una vez proporcionadas estas entradas necesarias para el algoritmo, dará comienzo el proceso de obtención de tweets, resumido en las siguientes acciones:

- Establecimiento de la conexión con la base de datos *<nombre_bd>* en el puerto *<puerto>* a través de la biblioteca PyMongo [31].
- Carga del archivo *config.json* de configuración, presente en la ruta del directorio de trabajo *<ruta_trabajo>*, que contiene la definición de todos los parámetros necesarios para el análisis del evento.
- Calcular la información de frecuencias de tweets/minuto del intervalo de tiempo que se quiere analizar, los cuales se almacenarán en un fichero de salida llamado *info_frecuencias.csv*, el cual será necesario para el programa de agrupamiento de los temas por proximidad temporal.
- Obtener, mediante la consulta a la base de datos <*nombre_bd*>, los tweets originales (no marcados como retweet) que tengan información útil (al eliminar menciones y urls, el tweet sigue teniendo texto, indicado porque no presenten el campo *voidtext*). Se recuperarán todos los tweets del intervalo de tiempo que cumplan estas condiciones, ordenados cronológicamente por fecha de publicación *created_at*.

6.2. División en ventanas

Una vez obtenidos los tweets a analizar, estos se repartirán en **ventanas** (Figura 11), cada una conteniendo un número idéntico *V* de tweets, con excepción de la última. Cada una de estas ventanas se procesará por separado, detectando temas que incluyan, en principio, únicamente tweets que pertenezcan a dicha ventana.

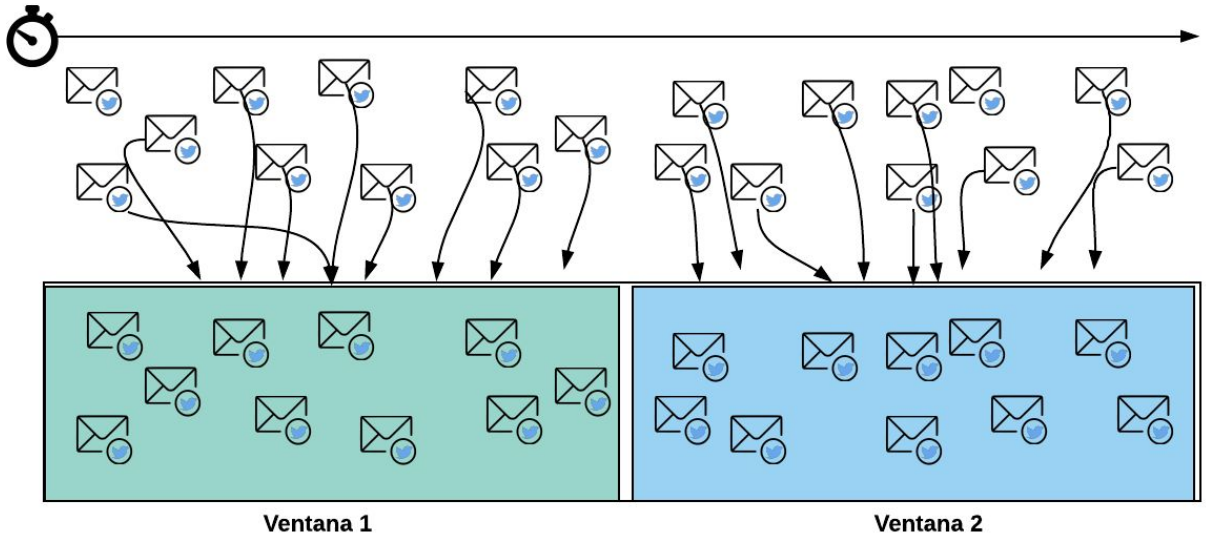


Figura 11: División de los tweets en ventanas.

El tamaño de ventana *V* (parámetro `tweets_ventana`, Anexo A), es decir, el número de tweets que se analizarán conjuntamente, ha sido determinado en 10000, cifra que ha dado buenos resultados en cuanto al equilibrio entre eficiencia y capacidad de detección de temas, y que nos ha permitido controlar los tiempos de ejecución aproximados del algoritmo.

Inicialmente se sopesó la posibilidad de realizar la división en ventanas empleando criterios temporales en lugar del número de tweets, pero esto iba en contra de la idea principal de no usar criterios de frecuencia. Esta diferencia de tamaño complicaría además el cálculo del tiempo aproximado de procesamiento.

6.3. Elección de un subconjunto aleatorio de tweets y formación de temas iniciales

Una vez repartidos los tweets entre las diferentes ventanas, da comienzo el procesamiento individual de cada una de ellas. La idea es agrupar los tweets de cada ventana según temática relacionada. Podemos pensar en este proceso como un algoritmo de clustering o segmentación.

Los pasos resumidos para llevar a cabo este proceso se ven reflejados en la Figura 12, y serán detallados a continuación.

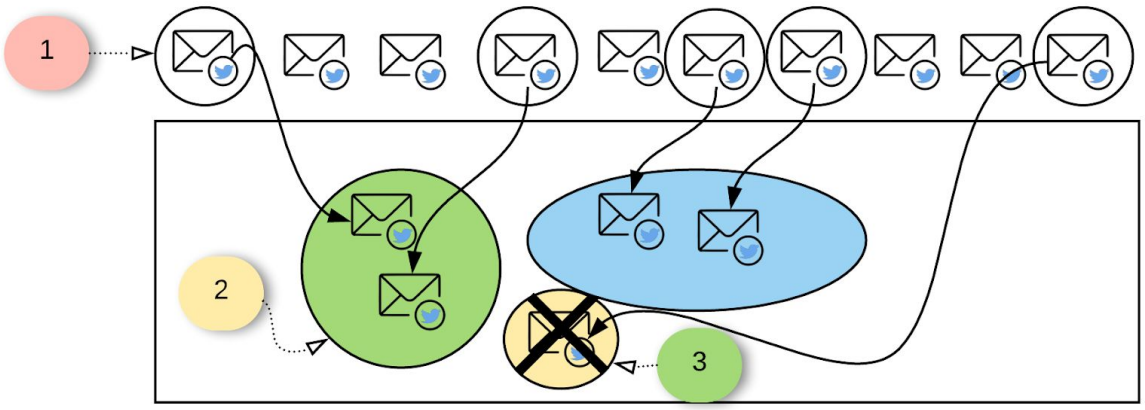


Figura 12: Etapa de formación de temas iniciales.

En el paso inicial, marcado por la etiqueta 1 en la Figura 12, se seleccionan de forma aleatoria un número arbitrario de tweets iniciales (parámetro `muestra_ventana`, Anexo A), que se agruparán en temas. Nuestros experimentos nos indican que un valor inicial de 30% resulta adecuado, lo que supone asumir que tomar 3000 tweets de la ventana permitirá obtener los temas más relevantes con una eficiencia razonable en términos de tiempo de cómputo.

En el paso 2 se agrupan en temas los tweets de esta muestra aleatoria inicial. Este proceso implica un recorrido por todos los tweets de la muestra, donde:

Para cada tweet *t*:

a. Se comprueba si *t* puede formar parte de alguno de los temas ya existentes. Si es el caso se incorpora al primer tema que cumpla las condiciones mínimas detalladas más adelante (F3). Conviene recalcar que en esta fase un tweet puede pertenecer como mucho a un tema.
b. Si *t* no se ha podido incluir dentro de ningún tema por no cumplir dichas condiciones, formará un tema nuevo, donde *t* será el único tweet.

Este proceso puede ser bastante lento y conlleva la formación de muchos temas compuestos por un único tweet. Por ello, cada cierto número de tweets (parámetro `filtrar_unitarios`, ver Anexo A) analizados se realiza un filtrado, eliminando los grupos que continúen siendo unitarios.

Para evaluar si un tweet *t* puede formar parte de un tema *g*, se realizan las siguientes fases:

F1. Se toma una muestra aleatoria *S* de *m* tweets (parámetro `num_muestra_tema`, Anexo A) pertenecientes al tema *g*. Si g no incluye *m* tweets, *S* contendrá todos los tweets del tema.

F2. Para cada uno de los tweets en *S*, se computa:

- $\overline{s}$: media aritmética de la similaridad de *t* con cada elemento en *S*.

- $S' = \{sim(t, t') > umbral\_inicial \mid t' \in S\}$, es decir el subconjunto de tweets en *S* cuya similaridad con *t* excede un cierto valor `umbral_inicial` (Anexo A).

F3. El tweet *t* se considerará similar al tema *g* si se cumplen las dos siguientes condiciones:

- $\bar{s} > umbral\_inicial$, es decir, la similaridad media de *t* con los tweets de la muestra aleatoria de *g* se encuentra por encima de `umbral_inicial`.
- $|S'| > min\_tweets\_similares$, es decir, el **número de tweets en S** cuya **similaridad** está **por encima del** `umbral_inicial`**,** supera un determinado número **mínimo** `min_tweets_similares`**.**
- En caso de que el número de tweets en *S* sea inferior al tamaño de la muestra *m*, lo que ocurre cuando los temas aún están empezando a formarse, simplemente se comprobará que la similaridad media $\bar{s}$ se mantenga por encima del **umbral** `umbral_inicial`.

F4. En cualquier otro caso, el tweet no se considerará similar al tema y formará un tema nuevo, tal y como se explicó anteriormente.

Es decir, se utilizan dos criterios para determinar si un tweet pasa a formar parte de un tema existente. En primer lugar tiene que tener una cierta similaridad media con los tweets que ya están en el tema. Sin embargo, con temas grandes, la media puede resultar insuficiente, por lo que exigimos además, que haya unos cuantos tweets a los que sea muy similar.

El último paso de la etapa de formación de temas iniciales, reflejado con la etiqueta 3 en la Figura 12, realiza un filtrado de los temas formados, eliminando los que se consideran que no cumplen las condiciones mínimas para llegar a formar un tema. Esto se hace porque, en algunos eventos, se pueden formar una cantidad excesiva de temas. Este filtrado en una etapa de formación inicial tiene alta repercusión en el tiempo de cómputo del algoritmo, sin afectar a la capacidad de detección de los temas más relevantes, que se asume que cumplirán las condiciones mínimas.

Los temas considerados válidos, y que pasarán a la siguiente etapa, serán aquellos que:

- Estén compuestos por al menos 5 tweets.
- Hayan sido publicados por al menos 5 usuarios distintos.
- Al menos el 10% de los tweets sean diferentes textualmente, entendiendo que en otro caso es muy posible que nos encontremos frente a tweets publicados por bots.
- La longitud media de sus mensajes, en palabras únicas y sin contar con stopwords, sea de un mínimo de 2 palabras. Esto elimina temas sin apenas contenido.

6.4. Unión de los tweets restantes a los temas iniciales

En la etapa anterior hemos seleccionado una cantidad prefijada (3000 tweets) de los V que componían la ventana (V=10000 en nuestro caso). Estos 3000 tweets se han agrupado, como hemos visto, dando lugar a una primera división en temas. Tras la formación y el filtrado de los temas iniciales, la siguiente etapa comprobará si los tweets restantes (los

7000 no seleccionados al tomar la muestra aleatoria de la ventana), pueden ser incluidos dentro de alguno de los temas iniciales, tal y como muestra la Figura 13.

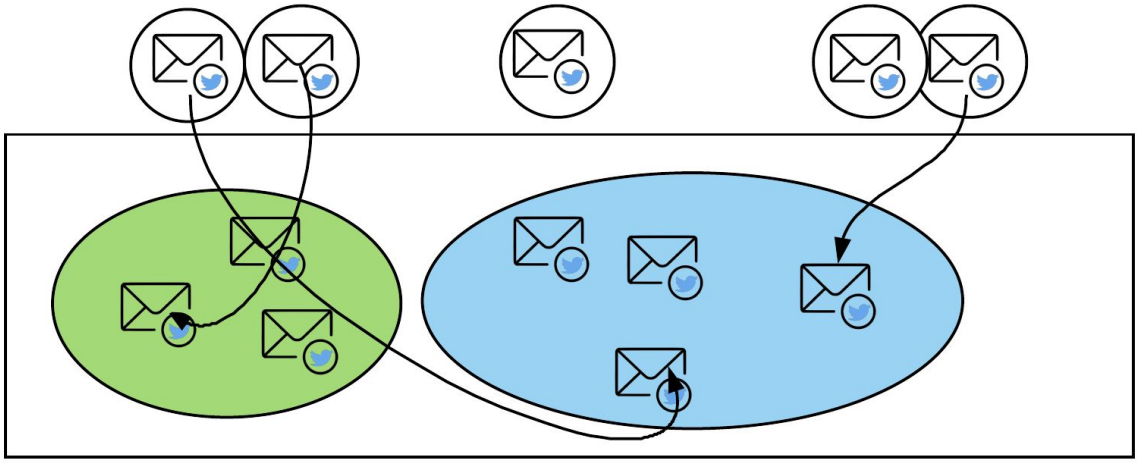


Figura 13: Unión de tweets restantes a los temas iniciales.

Hay que señalar que en esta etapa no se crean temas nuevos, solo se adscriben tweets a temas ya existentes, y que además algunos tweets pueden quedar excluidos, tal y como muestra la figura. Este proceso de tratar de unir los tweets a los temas ya formados es similar al realizado en la etapa anterior, con las salvedades de permitir en este caso que un mismo tweet pueda formar parte de varios de los temas ya formados, y de dejar a elección de la configuración del usuario si desea exigir el mismo umbral de similaridad que en la fase anterior o modificar este valor (parámetro `umbral_restantes`, Anexo A).

La razón por la que se permite que los tweets puedan ser incluídos en más de un tema es que estos tweets harán de “puente” entre temas surgidos de la etapa anterior que guardan una relación muy estrecha y que pueden llegar a fusionarse posteriormente en un mismo tema. En la etapa anterior no se permitía que un tweet se asignara a varios temas, porque esto puede ocasionar que al final se obtengan muy pocos temas y todos ellos con una gran cantidad de tweets repetidos, lo que los hace casi indistinguibles.

6.5. Cálculo de los centros de los temas

Los temas resultantes tras las dos etapas anteriores están formados por un conjunto de tweets publicados en determinado intervalo de tiempo que presentan una similaridad textual alta. En otras palabras, cada tema encontrado incluye mensajes que contienen opiniones o conversaciones muy parecidas y que se han publicado en instantes próximos en el tiempo.

Una vez tenemos estos temas, la siguiente etapa de nuestro algoritmo se encarga de excluir tweets que realmente no corresponden al tema. Esto puede suceder porque el

criterio para incluir el tweet solo compara, por razones de eficiencia, al candidato con un número limitado de tweets. Este factor aleatorio, repetido numerosas veces, puede llegar a admitir tweets que realmente no encajan en el tema, provocando temas demasiado dispersos.

Para ser capaz de reconocer a estos tweets outliers, es preciso introducir un nuevo concepto, que es el de centro de un tema. El **centro del tema** será un tweet encargado de servir como su representante, tal y como quiere resaltar la Figura 14.

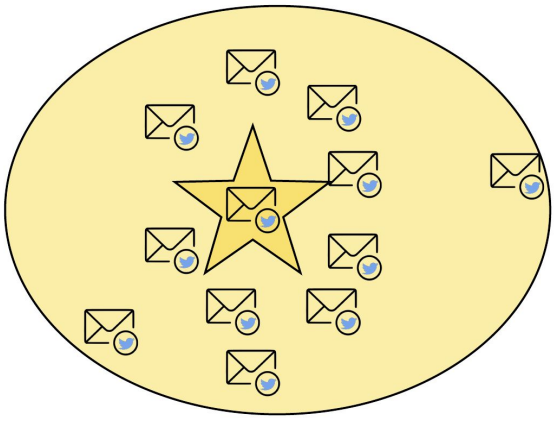


Figura 14: Centro de un tema.

En nuestro caso, definimos como **centro de un tema** a un tweet que maximiza la similaridad media con el resto de tweets del tema. En caso de haber varios centros, consideramos uno de ellos al azar.
El centro se emplea tanto para compactar los temas, como para comprobar si dos temas son susceptibles de fusionarse. En las siguientes secciones detallamos estas aplicaciones.

6.6. Compactación de temas

Vamos a emplear el centro de cada tema para evaluar su compacidad y detectar tweets que pueden ser excluidos. Para lograr esto, a partir de un tema y su centro, comenzamos obteniendo la media y la desviación típica de la similaridad de cada tweet con respecto al centro.

A la media la denominamos *índice de compacidad* y sirve para poder evaluar lo “lejano” que se encuentra un tweet dado respecto al centro. A partir de esta noción, se eliminarán los tweets que presenten una similaridad respecto al centro menor al índice de compacidad menos dos veces su desviación típica, como trata de reflejar la Figura 15.

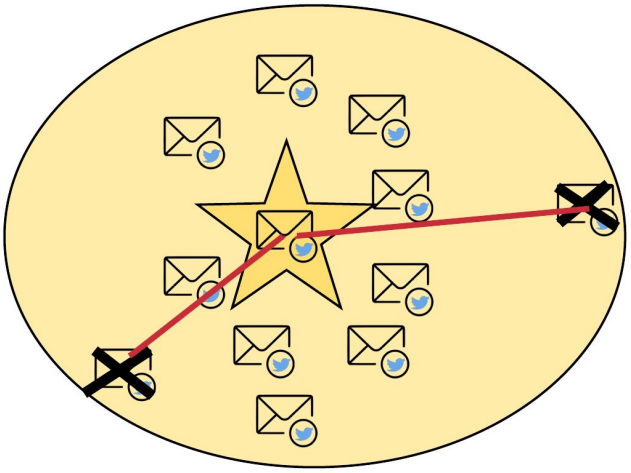


Figura 15: Compactación de un tema por similaridad respecto al centro.

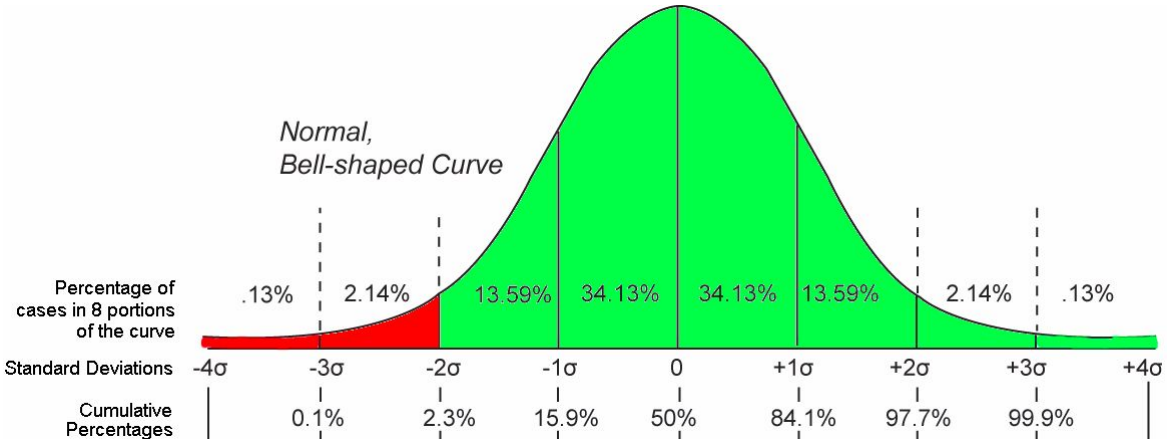


Figura 16: Distribución normal de probabilidades. Porcentajes según desviación típica.
Fuente: *https://commons.wikimedia.org/wiki/File:PR_and_NCE.gif*

Si asumimos que la variable aleatoria subyacente sigue una distribución normal, esto equivale a eliminar aproximadamente el 2.3% de tweets más alejados, que pueden ser considerados outliers (Figura 16).

6.7. Fusión de temas pertenecientes a una misma ventana

El componente aleatorio de nuestro algoritmo tiene dos efectos. El primero, ya tratado en la sección anterior, es la posibilidad de incluir en un tema tweets que en realidad no deberían estar. El segundo efecto, tratado en esta sección, es que se pueden generar como diferentes, temas que en realidad deberían ser considerados una única noticia, lo que se

conoce como fragmentación. Por ello, esta etapa comprobará si dentro de la ventana que se está procesando se pueden producir fusiones entre temas, de manera que pasen a formar un tema único que incluya tweets de los dos temas originales (Figura 17). En una sección posterior se tratará el problema de la fragmentación entre ventanas adyacentes.

Para poder evaluar si dos temas de la misma ventana son susceptibles de ser fusionados, es necesario definir el concepto de núcleo de un tema. Llamamos **núcleo de un tema** a un conjunto de *N* tweets (parámetro `tweets_núcleo`, Anexo A) que incluye el centro y sus *N-1* tweets más similares, de manera que constituyen un conjunto de mensajes que representan el tema de manera más amplia de la que haría solo el tweet central. En nuestro caso hemos encontrado el valor *N*=5 como el más adecuado. Valores demasiado altos para *N* pueden ocasionar que se fusionen temas con poca relación, mientras que valores demasiado pequeños pueden impedir fusiones que serían deseables.

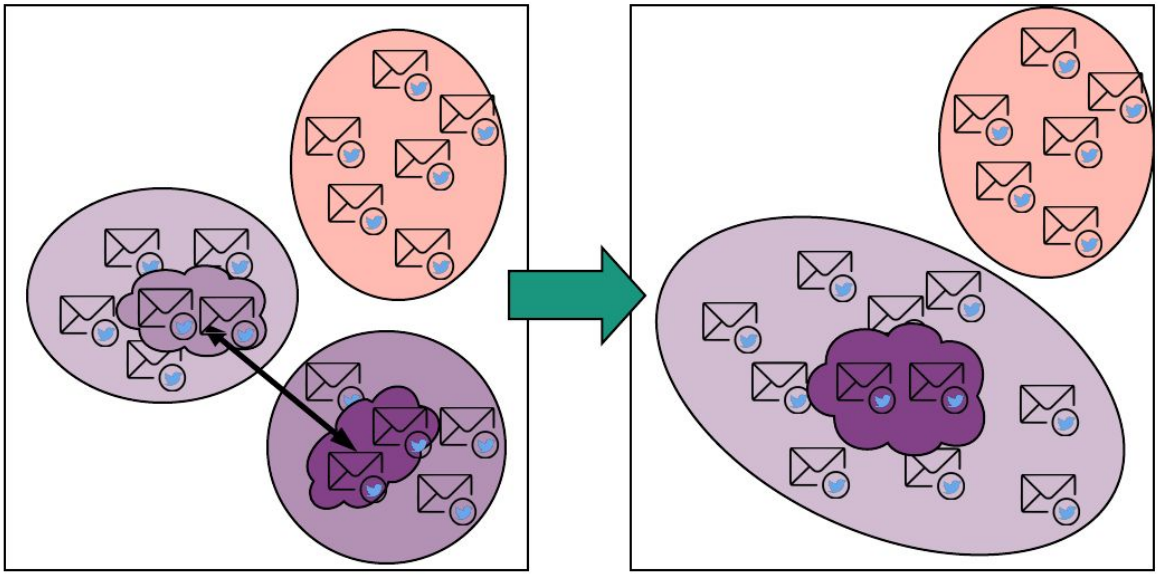


Figura 17: Fusión de dos temas detectados en la misma ventana.

Dos temas se fusionan en un tema único, si se cumple **alguna** de las dos siguientes condiciones:

- El número de tweets de la intersección de ambos temas, es decir el número de mensajes que tienen en común (resultado de la posibilidad de añadir un tweet a dos temas tal y como vimos en Unión de los tweets restantes a los temas iniciales) constituye más del 50% de los tweets que resultaría de la unión de ambos temas (parámetro `porcentaje_fusion_interna`, Anexo A).
- La similaridad media de los pares de tweets que componen los núcleos es mayor que cierto umbral de fusión (parámetro `umbral_fusion_interna`, Anexo A).

Nótese que, después de la fusión de dos temas, será necesario repetir el proceso de compactación, eliminando los tweets que queden lejos del centro del tema resultante, tal y como se explicó en la sección Compactación de temas. En nuestro caso solo se aplica la

fusión seguida de la compactación una sola vez, aunque tiene sentido plantearse como trabajo futuro una repetición de esta secuencia hasta alcanzar un punto fijo.

6.8. Fusión de temas entre ventanas adyacentes

Una vez se han realizado todas las etapas anteriores, tenemos un conjunto de temas detectados dentro de cada una de las ventanas en las que dividimos inicialmente los tweets. En esta etapa final buscamos fusionar temas que, debido a la división en ventanas tomada al principio, se hayan visto separados entre dos ventanas distintas (Figura 18).

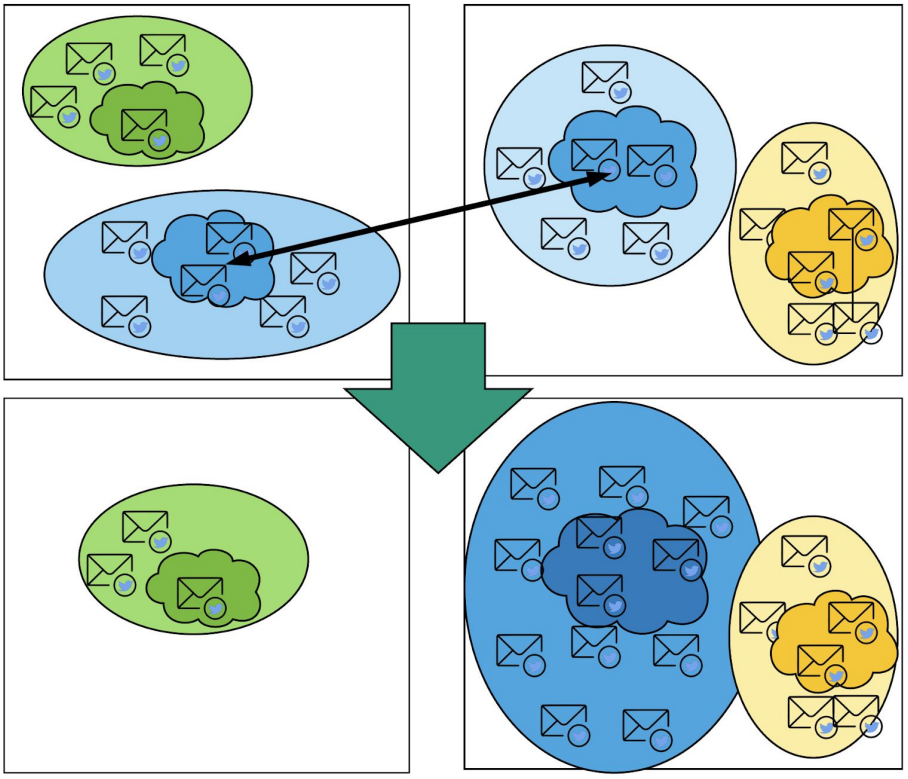


Figura 18: Fusión de temas de dos ventanas diferentes.

Una alternativa podría ser desplazar las ventanas y quedarse con los grupos de tamaño mayor tras todos los desplazamientos, pero la fusión nos ha parecido una solución sencilla, eficiente y que parece resolver bien este problema.
Como resultado de esta etapa, aquellos temas de ventanas adyacentes que estén relacionados se fusionarán, dando lugar a un tema que recoge todos los tweets de los temas origen.

Para que dos temas de diferentes ventanas se fusionen, se deberán cumplir las siguientes condiciones:

- La proximidad media de sus núcleos sea mayor que cierto **umbral de fusión de temas entre ventanas** (parámetro `umbral_fusion_externa`, Anexo A).

- Si formamos un tema unión con los tweets de ambos grupos y tratamos de compactar, no se pierden más del **10%** de los tweets, es decir no se genera una gran cantidad de outliers, que indicaría una elevada dispersión del tema resultante.

Las condiciones para fusionar los temas de una misma ventana (sección anterior) o entre temas de dos ventanas diferentes (sección actual) son distintas, porque en caso de estar frente a fusión dentro de una ventana, podemos encontrar que ciertos tweets pueden formar parte de varios temas, haciendo de “puente” entre ellos, como se indicó en Unión de los tweets restantes.

Por otra parte, en la fusión de grupos entre dos ventanas ningún tweet puede formar parte de los dos temas simultáneamente, eliminando una de las condiciones del caso de la misma ventana, por lo que se exige a cambio una pérdida mínima de tweets al compactar la posible fusión. Esta condición adicional se ha añadido porque este proceso, repetido sobre toda la secuencia de ventanas consecutivas, puede llevar a la formación de temas que ocupen varias ventanas de longitud y que realmente contengan tweets de temas diversos.

7. Similaridad entre tweets

Tal y como adelantamos, una parte muy importante del método propuesto y que emplean numerosas de las etapas del algoritmo de detección de temas, consiste en la **capacidad de comparar pares de tweets** para evaluar su **grado de similaridad** dentro de nuestro conjunto de datos.

En nuestra implementación, la medida de similaridad empleada tiene únicamente en cuenta el contenido textual del tweet, pero es importante destacar que cualquier otra función que cumpla las condiciones anteriormente detalladas en la sección Similaridad y distancia podría sustituir a la función actual, permitiendo comparar los tweets por otros aspectos, como bien podrían ser la localización geográfica desde donde se publican o las imágenes que puedan contener.

A la hora de evaluar la similaridad textual entre dos mensajes, la literatura [32] recoge numerosos enfoques, como se puede observar en la Figura 19, dependientes en gran medida del modo en el que se representan los mensajes.

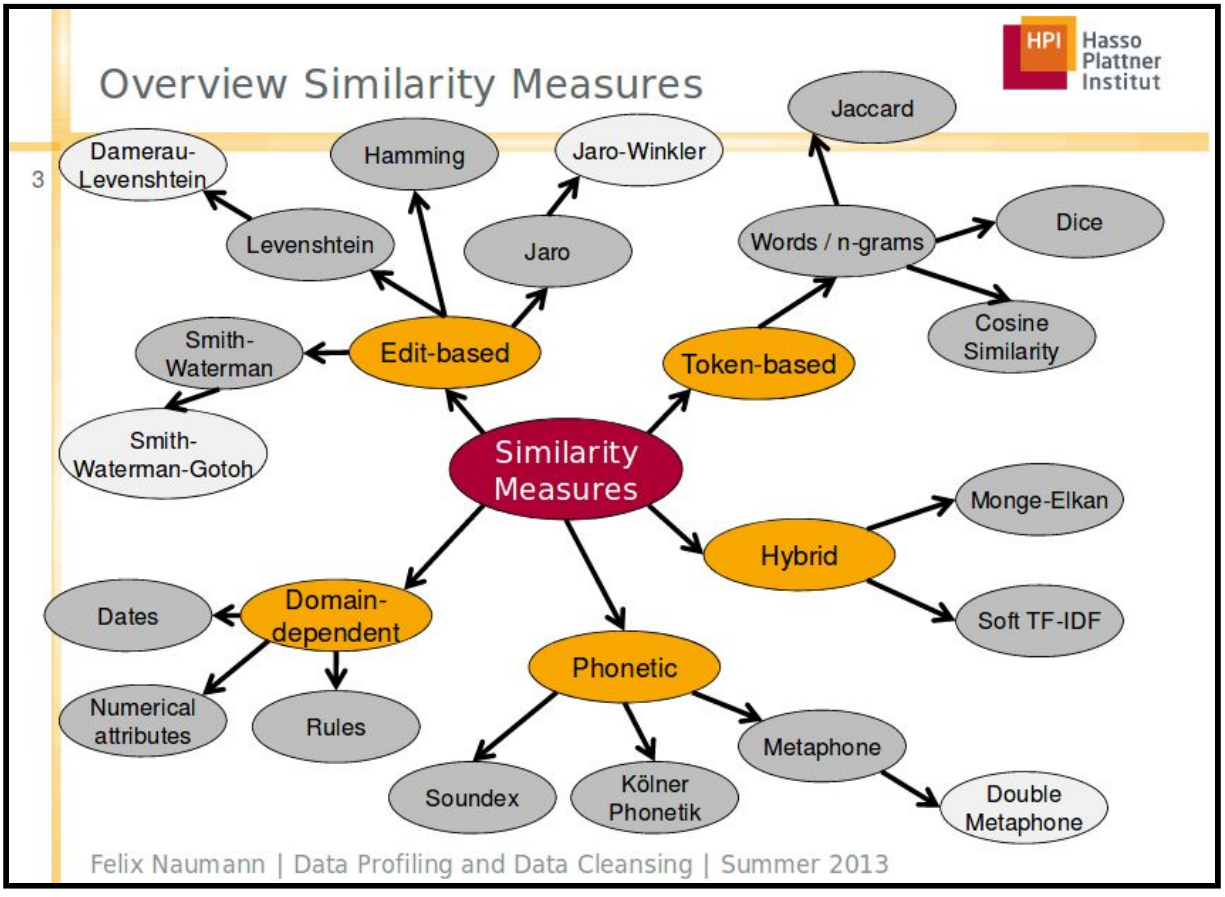


Figura 19: Mapa de las posibles medidas de similaridad entre textos [32].

Una medida de las más usadas para comparar textos basada en edición es la denominada *distancia Levenshtein*, que cuantifica la similaridad en función del número de operaciones de edición (inserción, eliminación o reemplazamiento de caracteres) necesarios para transformar un mensaje en el otro. Por otra parte, la *similaridad de Jaccard* cuantifica en función del ratio del tamaño de la intersección entre el de la unión, una vez se transforman las frases en conjuntos, que bien pueden estar compuestos por cada una de las palabras de los mensajes o tratar grupos de n caracteres o palabras consecutivos, denominados n-gramas. Por ejemplo, un mensaje *m= "Paralelo",* daría lugar al conjunto {"Pa", "ar", "ra", "al", "le", "el", "lo"} si tenemos en cuenta bigramas según caracteres.
Otro enfoque totalmente diferente a los dos ya mencionados es aplicar similaridad fonética, tal y como hace *Soundex*, que transforma cada palabra de un mensaje en su correspondiente código fonético, de manera que palabras que fonéticamente sean parecidas, obtendrán una alta similaridad, sin tener en cuenta su escritura [33].

7.1. Modelo de espacio vectorial

De todos los enfoques revisados, se ha tomado el que divide cada documento en tokens, concretamente en las palabras que conforman el mensaje, quedando representado como lo que se conoce con el nombre de *bag of words,* debido a que el orden de las palabras no es importante, tratándose como un conjunto de términos.
En concreto, cada documento *d* se representará siguiendo el conocido **modelo vectorial** [34]. Dado un conjunto de documentos $D = \{d_1, \ldots, d_n\}$ y $T = \{t_1, \ldots, t_m\}$ el conjunto de los términos presentes en *D*, cada documento quedará representado como un vector de m dimensiones:

$$\vec{di} = (w_{1i}, w_{2i}, \ldots, w_{mi})$$

donde cada w_{ji} indica el peso del término clave t_j en el documento d_i, es decir, la importancia de ese término en el documento (Figura 20).

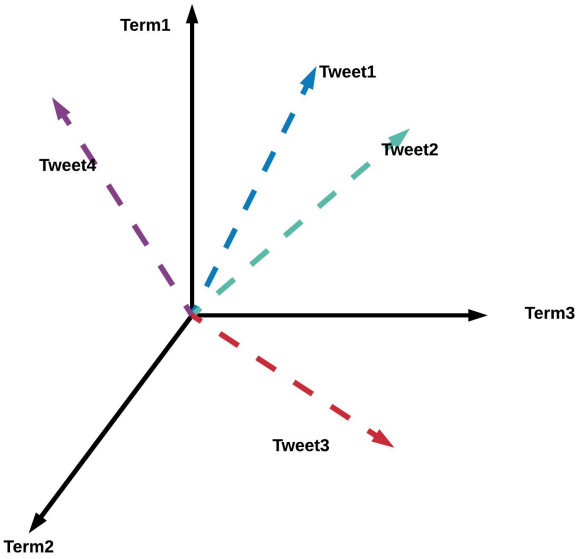


Figura 20: Representación de tweets en espacio de vectores.

A la hora de determinar los valores de estos pesos w_{ji} que cuantifican la importancia de los términos, es importante tener en cuenta dos cuestiones principales:

- Si un término clave t_j aparece muchas veces en un documento, este término suele ser importante dentro del contenido del mismo.
- Si un término clave aparece en gran cantidad de documentos, este término no será relevante a la hora de discriminar unos documentos de otros.

Uno de los métodos más comúnmente usado para definir estos pesos teniendo en cuenta estos dos aspectos, es el denominado **Term Frequency-Inverse Document Frequency (TF-IDF)** [35] [36].
Como su propio nombre indica, combina el cálculo de dos factores para obtener los pesos de los componentes del vector:

- ***Term Frequency*:** la frecuencia del término *t* dentro del documento *d*.
  Se define como:
  $$tf(t,d) = f(t,d)$$
  donde *f(t,d)* es el número de veces que aparece el término *t* en el documento *d*.

- ***Inverted Document Frequency***: la frecuencia inversa del término *t* en toda la colección de documentos a analizar, representando lo importante que es el término globalmente.
  Se define como:
  $$idf(t) = log\left(\frac{N}{df(t)}\right)$$
  donde *N* es el número de documentos de la colección y *df(t)* es el total de documentos en los que aparece el término t.

Estas son las fórmulas clásicas y más simples para calcular estos valores, pero existen múltiples variantes de estas que mejoran el cálculo en uno o varios aspectos.

Por ejemplo, una de las variantes más comunes es aplicar el logaritmo al valor TF original y sumar 1 al resultado, de manera que se amortigua el valor (un término que aparece el doble de veces no tiene que ser necesariamente el doble de importante) y además así la magnitud es comparable con el valor de IDF de manera directa.
$$tf(t,\ d) = 1 + log(f(t,\ d))$$

Por otro lado, a la hora de calcular el valor IDF, una de las aproximaciones más recomendables es sumar 1 al numerador y al denominador (como si de un documento extra se tratase), de manera que se eviten las divisiones entre 0.
De igual manera, se suele emplear un parámetro de suavizado, que consiste en sumar 1 al resultado, de manera que los términos que ocurran en todos los documentos no sean totalmente ignorados.
$$idf(t) = log\left(\frac{1+N}{1+df(t,d)}\right) + 1$$

Una vez calculados estos valores, el peso w_{ji} de cada término clave t_j en el documento d_i se define como:

$$w_{ji} = tf(j,i) \cdot idf(j)$$

Teniendo presente esta representación de documentos como vectores de m-dimensiones donde cada componente representa el peso de un término *t* en el documento *d*, es necesario definir además la **función** a aplicar a los vectores para **cuantificar la similaridad** entre ellos.
Estudios como [37] indican que existen diversas funciones para calcular la similaridad entre estos vectores, como son el cálculo de la distancia Euclídea, el coeficiente de correlación de Pearson o la divergencia de Kullback-Leibler (también conocida como entropía relativa) basada en probabilidad.

La función seleccionada para calcular la similaridad entre vectores compara estos dos vectores mediante el ángulo que formen entre ellos (Figura 21). Así, al comparar y obtener el valor de similitud, se obtendrán valores continuos, lo que nos permite ordenar por relevancia o exigir un umbral mínimo de similaridad, uso que se aplica en nuestro método.

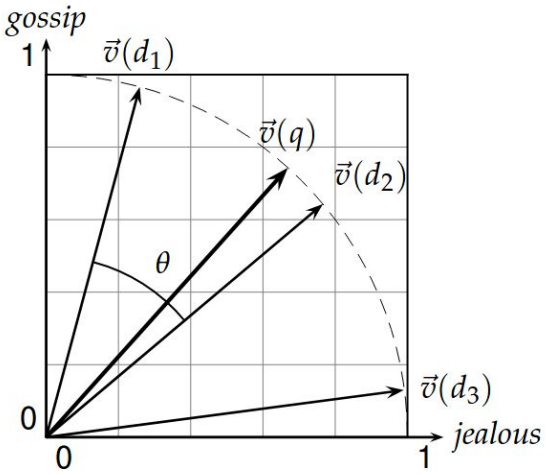


Figura 21: Similaridad por coseno de documentos representados como vectores
Fuente: https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf *(Página 121)*

En concreto, en lugar de comparar el ángulo θ que forman los vectores, se suele calcular el coseno de este ángulo, que habitualmente toma valores en el intervalo [-1,1] pero en este caso concreto será siempre en el intervalo [0, 1], dado que los valores de los pesos de las componentes del vector son siempre positivos.

Para ello, se usa la fórmula del producto escalar, donde podemos despejar el coseno del ángulo que forman ambos vectores, quedando reducido al producto escalar de los vectores dividido entre la multiplicación de sus módulos:

$$d1 \cdot d2 = |d1| \cdot |d2| \cdot cos(\theta) \rightarrow cos(\theta) = \frac{d1 \cdot d2}{|d1| \cdot |d2|}$$

Así, por tanto, la máxima similaridad entre dos documentos será cuando formen un ángulo de 0º y su coseno sea 1, mientras que la mínima similaridad será cuando sean

perpendiculares, cuyo coseno será 0. Es fácil comprobar que esta medida es, en efecto una similaridad tal y como se indica en la sección Similaridad y distancia.

Esta medida, comúnmente conocida como similaridad por coseno, posee una serie de características que la hacen muy adecuada para nuestro caso de uso. En concreto, como indica uno de los trabajos analizados [26], es una medida muy sencilla de calcular (especialmente si los vectores están normalizados, reduciendo la operación a un producto escalar), funciona muy bien con matrices dispersas y es independiente de la longitud de los documentos de manera que un tweet largo con varias palabras puede ser considerado aún muy similar a otro más corto con menos palabras.

Ejemplo del cálculo de la similaridad por coseno

Para ilustrar el funcionamiento concreto de esta medida de similaridad de documentos, se incluye un pequeño ejemplo, permitiendo destacar los aspectos más importantes de su implementación.

Imaginemos la siguiente colección de documentos:

d1 = *"The sky is blue"*
d2 = *"The sun is bright"*
d3 = *"The sun in the sky is bright"*
d4 = *"We can see the shining sun, the bright sun"*

En la siguiente tabla podemos observar los valores de frecuencia de todos los términos en los siguientes documentos, sin el valor amortiguador de aplicar logaritmos.

tf	**The**	**sky**	**is**	**blue**	**sun**	**bright**	**in**	**we**	**can**	**see**	**shining**
d1	1	1	1	1	0	0	0	0	0	0	0
d2	1	0	1	0	1	1	0	0	0	0	0
d3	2	1	1	0	1	1	1	0	0	0	0
d4	2	0	0	0	2	1	0	1	1	1	1

Por otro lado, si calculamos los valores de *frecuencia inversa de documentos*, encontramos los siguientes resultados.

t	**The**	**sky**	**is**	**blue**	**sun**	**bright**	**in**	**we**	**can**	**see**	**shining**
idf	1	1.7369	1.3219	2.3219	1.3219	1.3219	2.3219	2.3219	2.3219	2.3219	2.3219

Una vez calculados tanto los valores tf como los idf, los documentos quedan representados por vectores multiplicando ambos valores. Por ejemplo, para el documento 1:

d	**The**	**sky**	**is**	**blue**	**sun**	**bright**	**in**	**we**	**can**	**see**	**shining**
tf-idf	1	1.7369	1.3219	2.3219	0	0	0	0	0	0	0

Lo más común es normalizar los vectores, aplicando la norma euclídea $v = \frac{v}{|v|} = \frac{v}{\sqrt{v1^2 + v2^2 + ... + vn^2}}$, resultando los valores mostrados en la tabla.

Wij	**The**	**sky**	**is**	**blue**	**sun**	**bright**	**in**	**we**	**can**	**see**	**shining**
d1	0.3439	0.5197	0.4207	0.6591							
d2	0.4268		0.5221		0.5221	0.5221					
d3	0.5262	0.3975	0.3218		0.3218	0.3218	0.5042				
d4	0.3661				0.2239	0.2239		0.3507	0.3507	0.3507	0.3507

Si por ejemplo quisiéramos calcular la similaridad del documento d1 con los documentos d2 y d3, bastaría con calcular los productos escalares de d1 con d2 y de d1 con d3, para después dividir cada uno por la multiplicación de los respectivos módulos, pero si se normalizan los vectores se puede simplificar y calcular directamente los productos escalares.

$$d1 \bullet d2 = (0.3439 \cdot 0.4268 + 0.5197 \cdot 0.... + 0 * 0)$$
$$d1 \bullet d3 = (0.3439 \cdot 0.5252 + 0.5197 \cdot 0.3975.... + 0 * 0)$$
$$|d1| = \sqrt{0.3439^2 + 0.5197^2 + ... + 0^2} = 1$$
$$|d2| = \sqrt{0.4268^2 + 0.5221^2 + ... + 0^2} = 1$$
$$|d3| = \sqrt{0.5262^2 + 0.3975^2 + ... + 0^2} = 1$$

$$Sim(d1, d2) = cos(d1, d2) = \frac{d1 \cdot d2}{|d1| \cdot |d2|} = d1 \cdot d2 = 0.3661$$

$$Sim(d1, d3) = cos(d1, d3) = \frac{d1 \cdot d3}{|d1| \cdot |d3|} = d1 \cdot d3 = 0.7185$$

Por tanto, según TF-IDF, la frase *"The sky is blue*" se parece más a "*The sun in the sky is bright"* que a *"The sun is bright".*

7.2. Adaptación de la similaridad por coseno a la comparación de tweets

A la hora de comparar tweets midiendo su similaridad, es importante destacar que nuestro problema a resolver no consiste en determinar a cuáles se parece más un tweet *t* dentro del conjunto de los tweets, sino que se persigue detectar tweets referidos a un mismo tema de conversación en Twitter durante un evento.
Por ello, se ha decidido modificar la función de similaridad por coseno de manera que se ajuste a las necesidades del método propuesto, cuya motivación se explica a continuación.

Atendiendo a los valores devueltos por este cálculo del coseno que forman los vectores evaluados según el modelo TF-IDF, si tomamos una frase aleatoria de un libro y calculamos la similaridad mediante este método con el libro completo, se observa que hay un gran número de frases que no presentarán ninguna similaridad, mientras que para unos pocos valores la similaridad se distribuyen a lo largo de los valores hasta 1, como queda representado en la Figura 22.

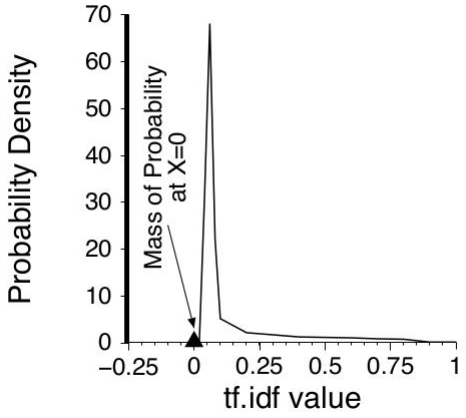


Figura 22: Típica distribución de los valores del producto escalar del TF-IDF [38].

Más allá de la gran concentración de valores con similaridad 0, si representamos el resto de valores, obtenemos una distribución como muestra la Figura 23.

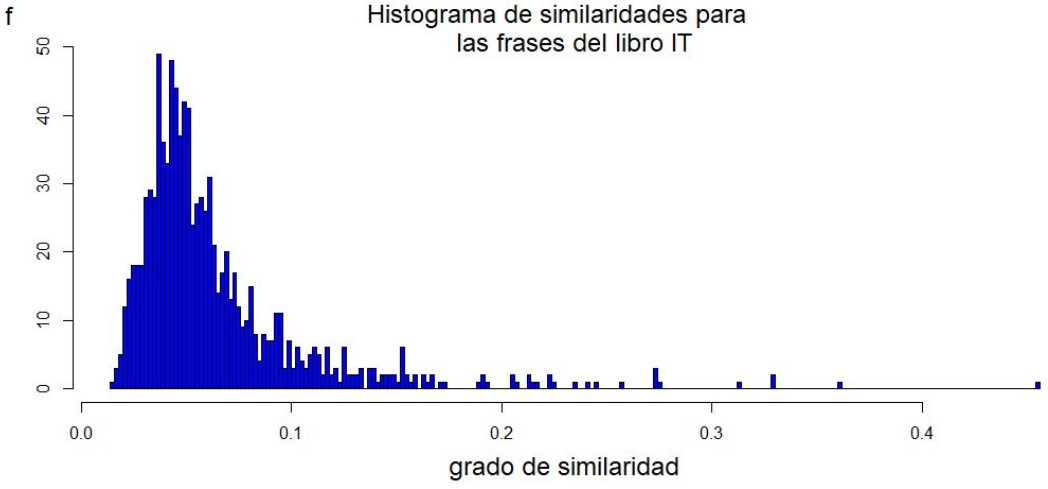


Figura 23: Distribución de los valores positivos de similaridad por coseno del libro IT de Stephen King.

La probabilidad de estos valores no sigue una distribución clara, aunque algunos autores [38] defienden que sigue una distribución *inversa normal*, también llamada *inversa gaussiana* o *distribución wald*. Realizando test de ajuste en R sobre los datos recopilados mediante *descdistr* (Figura 24), las pruebas realizadas no muestran una distribución concluyente, aproximándose principalmente a una distribución beta o gamma.

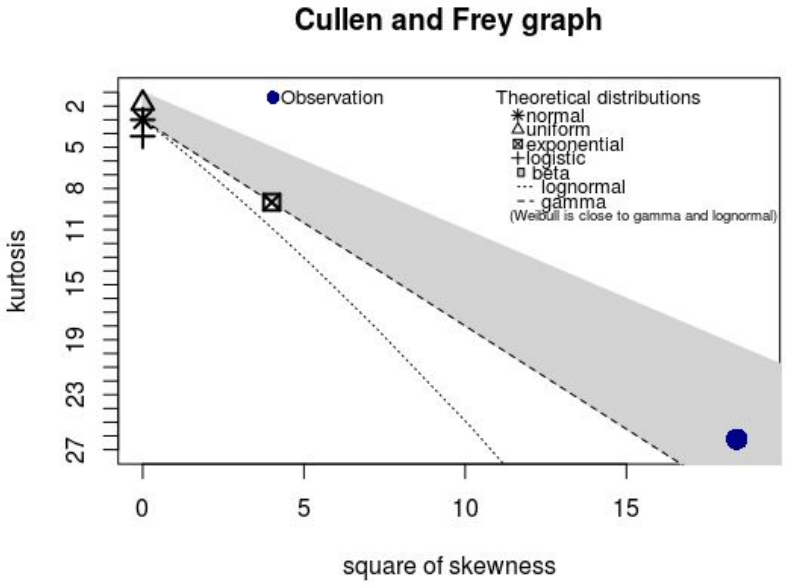


Figura 24: *Descdist* sobre datos de similaridad por coseno de una frase aleatoria de IT.

De estas gráficas se deduce que una similaridad definida a partir del coseno de TF-IDF solo obtendrá valores relevantes para tweets muy similares. Sin embargo, en nuestra aproximación al concepto de tema o noticia se precisa que tweets con un grado de similaridad no tan cercano se consideren pertenecientes a la misma noticia. Por ello, se ha decidido atenuar la gradación de la similaridad por coseno aplicando el logaritmo a este valor.

El resultado se ajusta a una distribución *aproximadamente* normal, aunque más apuntada, como podemos ver en el ejemplo de la Figura 25. De esta manera se pueden aplicar nociones sobre la distribución de los datos, conocidas la media y la desviación típica, que se emplean a la hora de calcular los centros y compactar los temas formados.

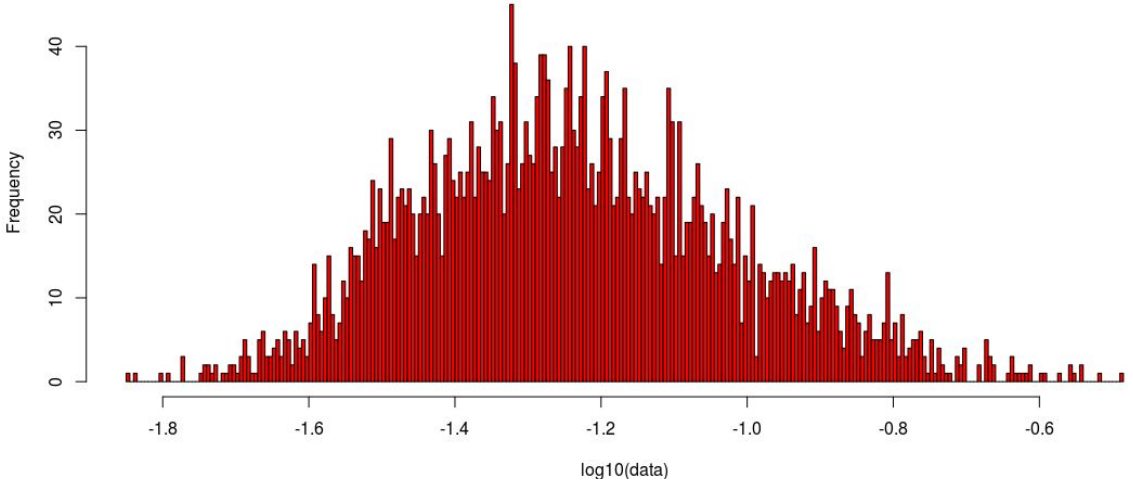


Figura 25: Distribución de valores al aplicar el logaritmo sobre la similaridad por coseno.

Para mantener la medida normalizada tras aplicar el logaritmo a la similaridad por coseno, se ha definido la función *Sim* final de similaridad entre dos tweets *tw1* y *tw2* por partes:

$$Sim(tw1, tw2) \begin{cases} log10(cos\_sim(tw1, tw2)) + 1 & \text{si } cos\_sim(tw1, tw2) >= 0.1 \\ 0 & \text{si } cos\_sim(tw1, tw2) < 0.1 \end{cases}$$

De esta manera, los valores seguirán una distribución más gradual que aplicando la similaridad por coseno y se encontrarán recogidos en el intervalo [0, 1].
Dado que lo que se pretende es agrupar tweets que se parezcan bastante, establecer como 0 valores en los que la similaridad por coseno sea menor de 0.1 no perjudica al funcionamiento del cálculo de la similaridad, dado que cualquier valor por debajo de este umbral no tendrá como resultado la incorporación de un tweet a algún tema ya formado.

8. Presentación de resultados

La representación de los resultados es un aspecto importante que no debe ser infravalorado. De nada sirve conseguir implementar un método que extraiga con gran exactitud las opiniones más relevantes de los usuarios de Twitter sobre un evento si no logramos mostrar de manera adecuada los resultados, de modo que sean fácilmente visualizados e interpretados.

Además, como en nuestro caso concreto se ha tomado la decisión de no eliminar *a priori* noticias (como se detalló en Principios básicos), el resultado es la detección de una gran cantidad de temas, que deberán ser presentados de forma adecuada al usuario.

8.1. Caracterización de temas

A la salida de la fase de detección de temas los resultados se almacenan de manera textual, anotando cada tema encontrado en un fichero de salida en formato de texto plano llamado `salida.txt`.

8.1.1. Factores de caracterización de temas

Este archivo de texto contiene todos los temas encontrados, almacenados de forma consecutiva. Para cada tema, el archivo almacena los tweets que lo componen, seguidos por los factores que lo caracterizan. De cada tweet se registra el alias del usuario que lo ha emitido, la fecha de publicación y el texto.

En cuanto a los factores que caracterizan a un tema, el archivo contiene:

1. ***Número de tweets***: de manera intuitiva podemos establecer que el número de tweets que forman un determinado tema es uno de los factores más relevantes.
2. ***Inicio*****:** marcamos el momento en el que comienza un tema de conversación como el instante en el que se publica el primero de sus tweets.
3. ***Fin*****:** de manera análoga a ***Inicio***, ***Fin*** será el momento en el que se publique el último de los mensajes que componen en tema.
4. ***Tiempo de vida***: número de segundos transcurridos entre ***Inicio*** y ***Fin***.
5. ***Centro temporal*****:** para poder representar el instante temporal en el que tiene lugar un tema, se ha decidido tomar el valor de la mediana de las fechas de publicación de los tweets que lo componen. A este factor de caracterización se le ha denominado **centro temporal**, y será utilizado durante el proceso de agrupación por proximidad temporal de los temas.
6. ***Frecuencia del tema*****:** el cociente entre el ***Número de tweets*** de la noticia y su ***Tiempo de vida***.

7. ***Longitud media de los tweets*****:** uno de los principales problemas a la hora de analizar los mensajes de Twitter es que muchos tweets apenas incluyen información, contando con una o dos palabras únicamente. Por ello, la longitud media de los tweets es un buen indicador del nivel de información asociado al tema; noticias con mayor longitud media suelen corresponder a discusiones y opiniones más detalladas. Este fue uno de los factores empleado en la fase de formación de temas iniciales para descartar temas.
8. ***Similaridad media con el centro*****:** entendida como la media de las similaridades de todos los mensajes de la noticia con el centro de la misma. Este factor se emplea en la etapa de compactación de temas.
9. ***Número de retweets*****:** se ha considerado el número de retweets de los mensajes de una noticia como un indicativo positivo del impacto de la misma.
10. ***Frecuencia absoluta*****:** se define como ***Número de tweets*** dividido entre el número de tweets totales publicados durante ***Tiempo de vida***, de manera que representa el porcentaje de tweets que los tweets de la noticia representan en ese intervalo temporal.
11. ***Similaridad media*****:** calcula la similaridad media aproximada de los tweets del grupo, tomando una muestra aleatoria de 50 tweets como máximo y calculando la media de sus similaridades, para tomar una noción de lo similares que son todos los tweets del grupo entre sí, no sólo con respecto al tweet central.
12. ***Tweets diferentes***: medida del número de tweets que tienen un texto diferente. Este factor permite detectar aquellas noticias que que pueden haber sido originadas por la presencia de bots, dado que es común encontrar bots que, desde cuentas diferentes, publiquen el mismo mensaje a intervalos regulares.

Además, para poder representar de manera sencilla el contenido semántico del tema, se incluye el texto del tweet centro del tema y el de un tweet adicional al que hemos denominado resumen, cuya motivación se explica a continuación.

8.1.2. Tweet resumen

En nuestra propuesta distinguimos entre el tweet centro del tema y el tweet resumen del tema. El centro, que ya hemos definido, se emplea durante el algoritmo de formación de temas. Debe representar bien el tema pero sin información adicional, que haría que se incluyeran tweets de otras noticias. En cambio el resumen tiene como objeto presentar el tema de forma breve al usuario. Aunque en un principio se pensó en utilizar el centro, se comprobó que resultaba demasiado escueto. En cambio el resumen es un tweet generalmente más largo, que incluso puede contener algún detalle no demasiado relevante, pero que en general recoge mejor los matices que definen la noticia.

Mientras que el centro era el tweet que maximizaba la similaridad media, el resumen se calcula de la siguiente manera:

1. Calcular la matriz que contiene los vectores TF-IDF de los tweets del tema.

2. Calcular la suma de los vectores TF-IDF de cada tweet, llamada `suma_TFIDF`.
3. Calcular la longitud de cada tweet, llamada `longitud_tweet`.
4. El **resumen** será el tweet que tenga menor cociente `suma_TFIDF` entre *longitud_tweet*.

La idea es que cuanto más pequeña sea `suma_TFIDF`, más comunes serán las palabras del tweet. La división entre `longitud_tweet` se realiza porque si no llevan ventaja los tweets que contienen pocas palabras.

8.2. Presentación de los temas al usuario

En esta sección se describen las distintas maneras en las que se ha decidido organizar y presentar la información resultante.

8.2.1. Representación cronológica global

Comenzamos por discutir una presentación de los temas de forma temporal que, aunque permite visualizar la distribución cronológica de las noticias, no las agrupa. Esta representación se genera a la vez que el fichero de texto de salida al terminar de ejecutar la fase de detección de noticias.

No se considera en la Figura 5 porque no es un método de agrupación, sino una descripción visual de todas las noticias de forma conjunta que puede ser útil al usuario y que nos sirvió de punto de partida para el método de agrupación temporal descrito posteriormente.

Esta representación relaciona la frecuencia de publicación de tweets por minuto y los temas detectados, como podemos observar en el ejemplo de la Figura 26.

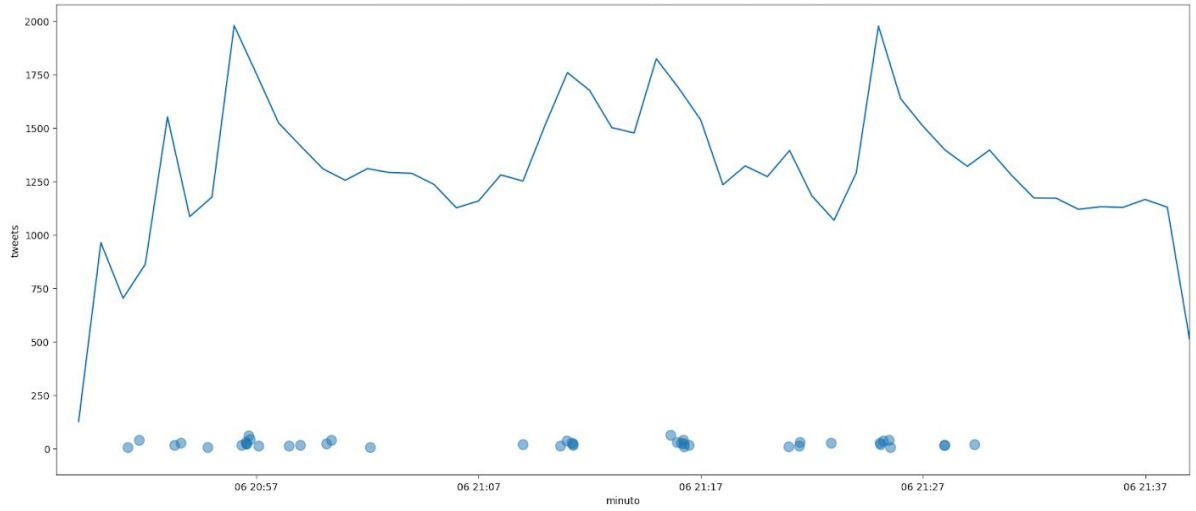


Figura 26: Primera representación visual de los temas detectados.

En la parte superior se incluye el gráfico de frecuencias de publicación de los mensajes analizados, mientras los círculos de la parte inferior corresponden a los temas detectados, ordenados por el factor de caracterización al que denominamos centro temporal. La representación en altura de los círculos está relacionada con el número de tweets del tema.

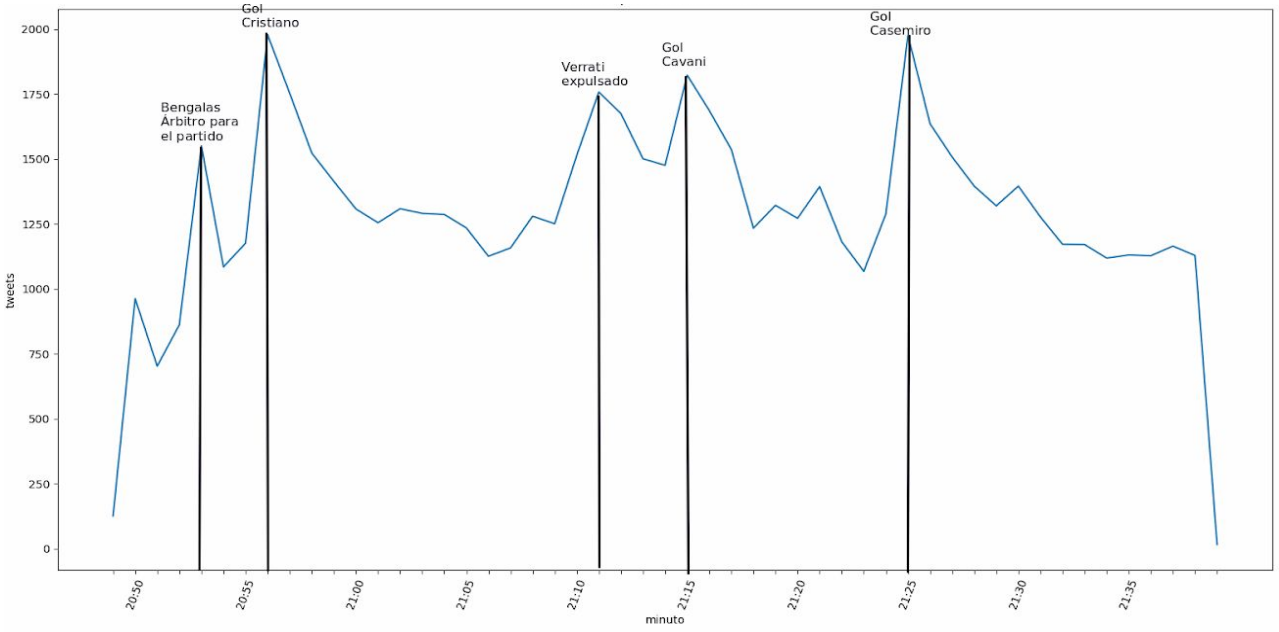


Figura 27: Hitos de la segunda parte del partido de fútbol entre el PSG y el Real Madrid del 06/03/2018.

Aunque nuestra técnica de detección de noticas no utiliza la frecuencia de tweets, la gráfica permite comprobar que las noticias se concentran de forma natural alrededor los momentos de mayor participación en el evento, que a su vez se corresponden con los hitos principales del evento, como podemos ver en la Figura 27.

8.2.2. Agrupación de temas

Aunque tanto los resultados almacenados en formato de texto como esta primera representación visual de los temas detectados permiten al usuario situar los temas detectados temporalmente y, pasando el cursor por encima, visualizar el resumen de los mismos, pensamos que sería conveniente realizar de manera automática la agrupación de temas que facilite su presentación desde diferentes perspectivas.

En particular, tal y como se adelantó al presentar el funcionamiento general del sistema, se ha decidido agrupar y mostrar los resultados de dos maneras distintas, en función del tipo de análisis que se quiera hacer del evento.

La motivación de contar con dos modos de agrupación surge porque, para algunos eventos, puede ser interesante conocer sus momentos clave, entendidos como aquellos instantes temporales en los cuales se produjeron muchos temas de conversación simultáneos o próximos temporalmente. Esta agrupación permite resaltar que en esos momentos pasó algo relevante y digno de analizar, por encima de otros momentos de menor concentración. Sin embargo, para otro tipo de eventos puede ser deseable analizar la relación entre los temas detectados desde un punto de vista textual, permitiendo comprobar por ejemplo si un tema de debate se ha repetido a lo largo de un evento o existe relación temática entre las noticias detectadas.

La manera de agrupar los temas detectados según uno u otro criterio se detalla en las secciones siguientes.

8.2.2.1. Proximidad temporal

Con objeto de mostrar al usuario las noticias agrupadas por "momentos clave", se parte de los **centros temporales** de cada tema que, tal y como se han definido anteriormente, se corresponden con la mediana de las fechas de publicación (incluyendo hora minuto y segundo) de todos los tweets que forman parte de un tema.

Nuestro objetivo entonces es buscar centros temporales próximos, con el fin de mostrar sus noticias asociadas de forma conjunta. La idea principal es buscar los intervalos de tiempo con mayor densidad de temas. La densidad para un intervalo dado *I,* puede definirse como:

$$densidad(I) = \frac{NumTemas(I)}{Tiempo(I)}$$

donde *NumTemas* es el número de temas detectados en el intervalo y *Tiempo* mide la longitud del intervalo en minutos.

Si por ejemplo partimos de un total de *K* temas, la máxima densidad se daría si el centro temporal de los *K* temas ocurriera en un mismo minuto, lo que supondría una densidad de *K*/1 = *K* temas/minuto para el minuto considerado.

Nuestro algoritmo comienza buscando esta máxima densidad, recorriendo todos los minutos del evento considerados. Si no la encuentra -y generalmente es difícil tal concentración se dé-, va bajando el nivel exigencia para la densidad, tanto considerando el número de temas del intervalo como el tamaño del intervalo considerado.

Los dos factores, el que disminuye el número de temas exigidos y el que amplía el tamaño del intervalo entre paso y paso se calculan a partir de un número de niveles de densidad *n*, parámetro que inicialmente debe fijar el usuario. De forma intuitiva, el valor *n* indica la granularidad de la división en temas que desea el usuario.

El factor *g* de reducción del número de temas requeridos en cada intervalo, así como el factor *t* de crecimiento de la longitud de los intervalos, se pueden definir a partir del valor *n* de la siguiente forma:

$$g = \sqrt[n-1]{numtemas} \qquad t = \sqrt[n-1]{tiempoevento}$$

donde *numtemas* es el número total de temas detectados y *tiempoevento* es el intervalo temporal de tiempo analizado del evento.

De esta forma, para un determinado nivel *k*, *0 ≤ k < n*, el número mínimo de temas considerado para formar un momento *nt* y la longitud del intervalo *nm* se calculan de la siguiente manera:

$$nt = \frac{numtemas}{g^k} \qquad nm = t^k \ minutos$$

El nivel inicial *k=0*, corresponde con la máxima densidad (todos los centros temporales en el mismo minuto) mencionada anteriormente. El nivel *k=n-1* exigirá al menos un tema repartido en todo el intervalo de tiempo, lo que corresponde con la menor densidad posible.

El algoritmo agrupa los temas en lo que hemos denominado como **momentos**, cada momento correspondiente a un nivel de 0 a *n-1*. Ocasionalmente puede haber niveles vacíos, sin encontrar ningún momento, indicando “saltos” en la densidad.

Algoritmo: Agrupación de temas en momentos

Entradas:
- ***n*:** número de niveles
- *tiempoevento:* tiempo total del evento en minutos
- *numtemas:* número de temas detectado
- *centrost:* centros temporales de los temas detectados

Salidas:
- ***momentos*:** momentos de los niveles *k*, 0 ≤ *k* < *n*, con los temas que los componen

$g = \sqrt[n-1]{numtemas}$
$t = \sqrt[n-1]{tiempoevento}$
k = 0
while *k* < *n* && **len**(*centrost*) > 0:
    nm = t^k
    nt = $\frac{numtemas}{g^k}$
    // Calcula los grupos de nt temas en intervalos de nm minutos presentes en centrost
    momentos_k = **buscaMomentos**(*centrost*, *nt*, *nm*)
    for *m* **in** *momentos_k*:
        momentos.**append**((*m*, *k*)); // Anota el momento *m* de nivel *k*
        centrost.**eliminaTemas**(*m*) // Elimina los temas que compongan m de centrost
    k = k + 1

En cada nivel k entre *0* y *n-1* el algoritmo exige al menos *nt* temas en *nm* minutos, es decir una densidad mínima de *nm/nt*. Esto nos permite calcular el ratio de crecimiento de la densidad requerida según avanza *k*. Si llamamos *d(k)* a la densidad en el nivel *k*, podemos comprobar que

$$\frac{d(k+1)}{d(k)} = \frac{1}{\sqrt[n-1]{numtemas \cdot tiempoevento}}$$

Es decir, para valores altos del tiempo total y del número de temas, podemos utilizar valores mayores de *n*, para generar densidades similares. Esto tiene sentido porque el disponer de más temas, o de intervalos mayores de tiempo permite realizar particiones de granularidad más fina.

Ejemplo 1 - Distribución de la densidad mínima por niveles:

Supongamos que hemos encontrado 100 temas en un tiempo analizado de 200 minutos y que el usuario ha decidido repartir estos temas entre 10 niveles. La siguiente tabla resume estos datos junto con los factores de modificación del intervalo de tiempo *t* y de número de temas requeridos *g*.

numtemas	tiempoevento	n	t	g
100	200	10	1.80164823	1.66810054

A partir de estos valores y siguiendo el algoritmo, podemos calcular el número de grupos y el tamaño del intervalo asociado a cada nivel *k*, junto con su ratio que se corresponde con la densidad:

k	nt	nm	Densidad
0	100	1	100
1	59.94842503	1.80164823	33.2742119
2	35.93813664	3.24593635	11.0717318
3	21.5443469	5.84803548	3.6840315
4	12.91549665	10.5361028	1.22583245
5	7.742636827	18.9823509	0.40788609
6	4.641588834	34.1995189	0.13572088
7	2.782559402	61.6155028	0.04516005

8	1.668100537	111.009462	0.01502665
9	1	200	0.005
10	0.59948425	360.329646	0.00166371

Podemos mostrar esta tabla gráficamente para visualizar la curva de disminución de la densidad mínima requerida al aumentar *k* (Figura 28).

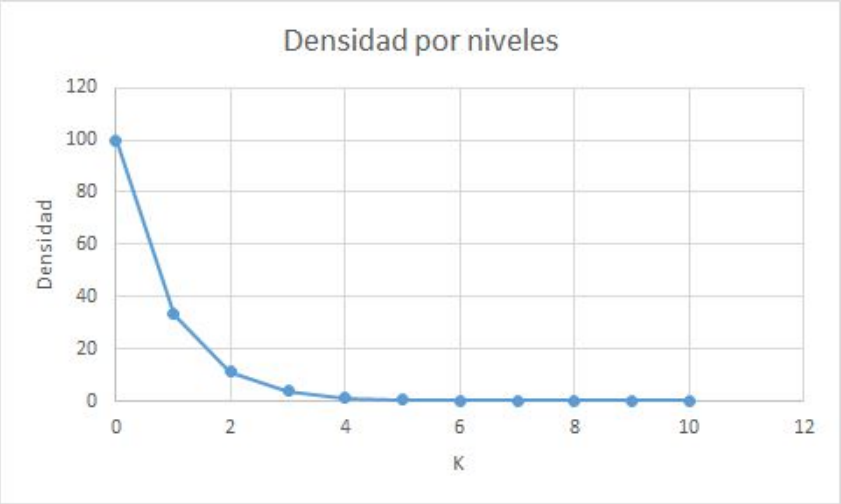


Figura 28: Relación entre la densidad y los niveles.

Ejemplo 2 - Caso práctico de cálculo de momentos:

Para ejemplificar el funcionamiento del algoritmo en un caso real, imaginemos un evento con los siguientes temas detectados, mostrando los tweets centrales y los centros temporales:

T1 - 20:51:42 - *Foot - C1 - PSG/Real - PSG - Real Madrid : les stats de la mi-temps https://t.co/LaP5KMq5ji*
T2 - 20:52:02 - *Vamos Real Madrid! Vamos Cristiano Ronaldo! ○🏆⚽ #halamadrid*
T3 - 20:52:03 - *Vamo PSG vamo que dá*
T4 - 20:52:10 - *Bora bora PSG!! #EuAcredito*
T5 - 20:54:00 - *El psg queria los ultras tengan sus ultras https://t.co/jwZauvyVSf*
T6 - 20:56:21 - *Cristiano. Acabou pro PSG*
T7 - 20:56:33 - *Já era PSG kkk já era previsto*
T8 - 20:56:34 - *Chau chau psg*
T9 - 20:56:34 - *Adios psg ronaldo to good lmao*

T10 - 20:56:40 - *Bye Bye Emre Bye Bye PSG*
T11 - 20:56:43 - *Tchau tchau Psg kkkkkkkk*
T12 - 20:56:44 - *Gol del Real Madrid! ¡Gol de Cristiano Ronaldo! PSG 0-1 Real Madrid*
T13 - 20:56:55 - *PSG 0 Real Madrid 1 Game over*
T14 - 20:57:00 - *It's over for PSG no Neymar no life Bruh*

A simple vista, podemos ver que a las 20:56 se concentraron bastantes de estos temas, indicando que se trata de un momento clave.
Para agrupar estos temas, escogemos *a priori* un número máximo de niveles en los que queremos detectar momentos, tomando en este caso 3 niveles.
Nuestro evento analizado (muy breve para simplificar los cálculos del ejemplo) duró desde las 20:49:00 hasta las 20:58:00, con un total de 9 minutos de duración.

El análisis por niveles se haría de la siguiente manera:

Nivel	Minutos de cada intervalo	Grupos por intervalo
0	1	14
1	3	3.7416
2	9	1.0

Los resultados serían los siguientes:

Momentos nivel 0	--
Momentos nivel 1	M1: T2 - 20:52:02 T3 - 20:52:03 T4 - 20:52:10 T5 - 20:54:00
	M2: T6 - 20:56:21 T7 - 20:56:33 T8 - 20:56:34 T9 - 20:56:34 T10 - 20:56:40 T11 - 20:56:43 T12 - 20:56:44 T13 - 20:56:55 T14 - 20:57:00
Momentos nivel 2	M3: T1 - 20:51:42

Tal y como se ha detectado, el minuto 20:56 fue un momento clave del evento que originó 9 temas de conversación, en concreto coincidiendo con la anotación de un gol en los primeros

minutos del partido. Se observa que el círculo aparece retrasado temporalmente con respecto al evento. Esto es normal, considerando que las reacciones al gol comienzan nada más suceder éste y se alargan durante el periodo de varios minutos.

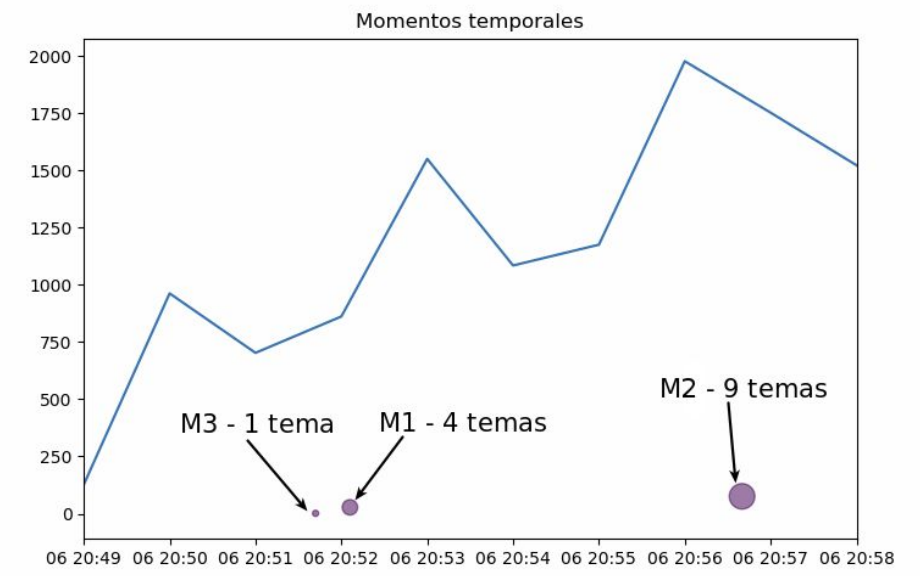


Figura 29: Momentos clave en un evento, indicando el número de temas detectados.

Como se muestra en la Figura 29, los momentos se representarán con puntos de diferente tamaño en función de la suma del número de tweets de cada tema que forma parte del momento. Un punto muy grande indicará que hay una mayor concentración de tweets (que generalmente se corresponde con una alta concentración de temas). La altura en el eje Y de cada uno de los momentos se encuentra determinada por la relación entre este número total de tweets y el intervalo de tiempo en el que se concentran.

Si pasamos el cursor por encima de cada uno de los puntos que representan los momentos, podremos ver un listado de los temas que lo componen, visualizando además los mensajes de los centros de cada tema, tal y como se ilustra en la Figura 30.

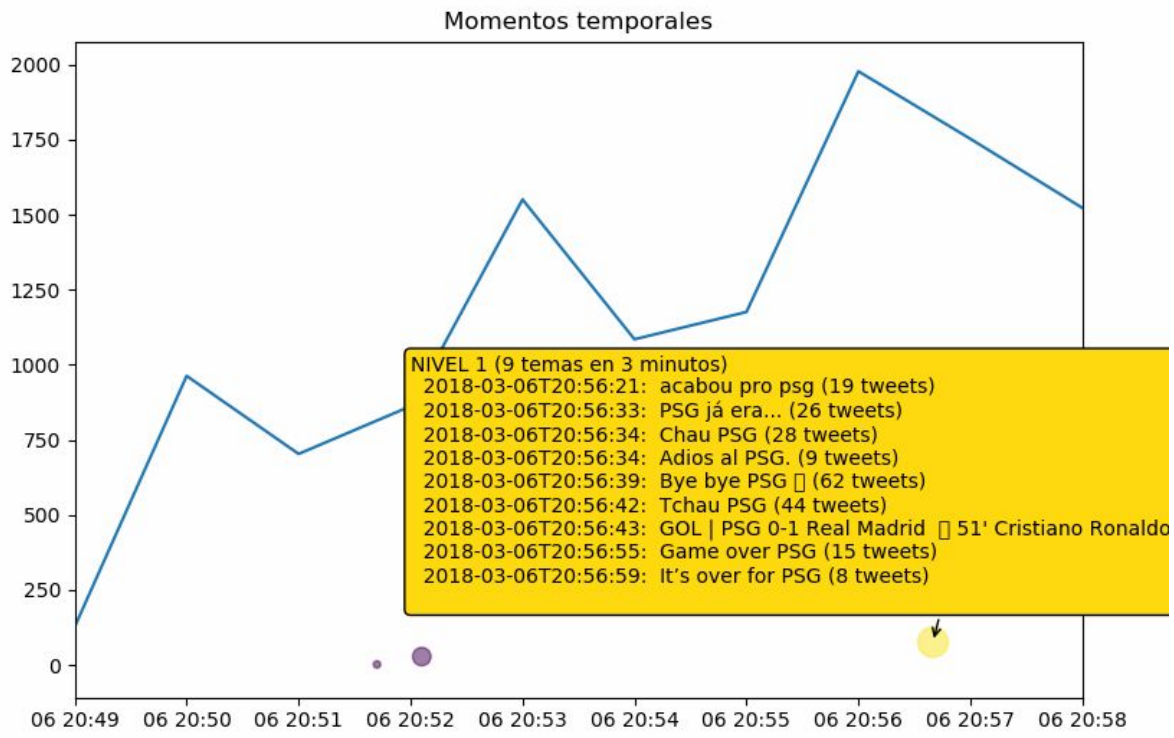


Figura 30: Detalle de los temas que componen un momento del evento.

El script que realiza este proceso de agrupación de temas por proximidad temporal y los representa gráficamente se ejecutará indicando la ruta donde se encuentran el archivo de texto plano con las noticias *salida.txt* y el de la información de frecuencia de twitteo, *info-frecuencias.csv*, que generalmente coincide con la ruta de trabajo de la fase de detección de temas.

El script se ejecutará de la siguiente forma:

```
$ python tweets-time-clustering.py <ruta_trabajo> <num_niveles>
```

Un ejemplo de ejecución sería:

```
$ python tweets-time-clustering.py PartidoPSG/ 3
```

8.2.3. Relación textual

La segunda manera de representar y agrupar las noticias resultantes del análisis de un evento es mediante su relación temática. Para esto, se emplean las nociones de similaridad de tweets mencionadas en la sección Modelo de espacio vectorial, usando los vectores generados por cálculo del esquema de recuperación de información TF-IDF.

En concreto, partiendo de los resultados almacenados en el fichero en formato texto de salida, se calcula la matriz de los vectores TF-IDF de todos los tweets que forman los temas detectados, pero sin incluir la multiplicación por el factor IDF. Recordemos que el papel de IDF es reducir el peso de las palabras que aparecen en muchos documentos. La motivación de excluir este valor surge de que, en este caso, tenemos ya las noticias formadas, en las que las palabras clave se repetirán a menudo dentro de un tema en la mayoría de sus tweets, y queremos que sean justo estas palabras las que nos sirvan para relacionar estos temas. Si incluyéramos la multiplicación por el factor IDF y, por ejemplo, tratásemos de agrupar los temas que hablan sobre los diferentes goles de un partido, la palabra “GOL” perdería peso en la comparación y no serviría como elemento discriminante, que es lo deseado.

Se trata por tanto de un agrupamiento por palabras clave, para lo que es suficiente con la frecuencia de los términos en los tweets. Una vez calculada esta matriz, se aplica la técnica de clustering K-means para agrupar los vectores que representen los **centros de los temas**. Se usan únicamente los vectores de los centros de los temas porque los temas ya constituyen en sí una agrupación de tweets similares, tratando ahora únicamente de establecer relaciones entre los temas ya formados.

Como el número de clusters o “temáticas” no es definible de antemano, dependiendo en gran medida del tipo de evento, se ha tomado la decisión de calcular el número óptimo de clusters realizando el clustering con valores k=3..20, y puntuando los clusters formados para cada k mediante dos índices de evaluación interna de los clusters resultantes. Estos índices son:

- ***Silhouette*** [39]: evalúa la calidad en función de la distancia media entre cada elemento de un cluster respecto a los elementos del cluster más cercano y la distancia media de ese mismo elemento respecto a los elementos de su mismo cluster. Devuelve un valor en el intervalo [-1, 1], donde -1 indica poca calidad de clustering y 1 alta densidad dentro del cluster.
- ***Calinski & Harabaz*** [40] ***:*** evalúa la relación entre la media de dispersión entre clusters y la dispersión dentro de cada cluster y cuanto mayor sea este valor, mejor definidos se encontrarán los clusters.

Los clusters formados se evalúan según estos dos índices, calculando su media, y de esta manera podemos obtener el valor k óptimo, que será el que mayor coeficiente obtenga. En caso de discrepancia entre ambos índices se ha tomado la decisión de escoger siempre el que agrupe los temas en mayor número de clusters (estando este número siempre entre 3 y 20).

Nuestro objetivo es representar los clusters en un gráfico de 2D, representando los centros de temas próximos textualmente del mismo color. Sin embargo, los centros son vectores con tantas dimensiones como palabras han aparecido en los tweets del dataset, por lo que necesitamos un método para reducir la dimensionalidad hasta las dos dimensiones.
Para ello, se les aplica a los vectores TF-IDF de los centros inicialmente una reducción de dimensionalidad mediante Latent Semantic Analysis (LSA) [41] hasta 100 dimensiones, el valor adecuado según [42]. LSA es un modelo estadístico muy conocido en recuperación de información que permite extraer relaciones entre palabras por el contexto en el que aparecen. Una de sus claves principales es que, partiendo de la matriz documento-palabra como hace TF-IDF, emplea una técnica algebraica de descomposición en valores singulares (SVD) sobre ella, realizando una descomposición de la misma en factores, permitiendo reducir las dimensiones de sus vectores [43].

Posteriormente, se aplica una segunda técnica de reducción de dimensionalidad t-SNE [44] para reducir hasta 2 dimensiones. t-SNE construye un modelo donde asigna a cada par de vectores a representar una probabilidad en función de su similaridad. Establece también un modelo en las dimensiones de salida deseadas, asignando otra función de probabilidad y trata de minimizar las diferencias entre ambas distribuciones en un proceso iterativo. Esta técnica resulta muy adecuada para la representación visual de los datos, pero muy ineficiente con grandes dimensiones de datos, lo que justifica la reducción previa mediante LSA.

Para realizar la agrupación por similaridad temática y visualizar los temas gráficamente, se deberá invocar el script de la siguiente forma, de manera análoga al método de agrupación mencionado en la sección anterior:

```
$ python tweets-textual-clustering.py <ruta_trabajo>
```

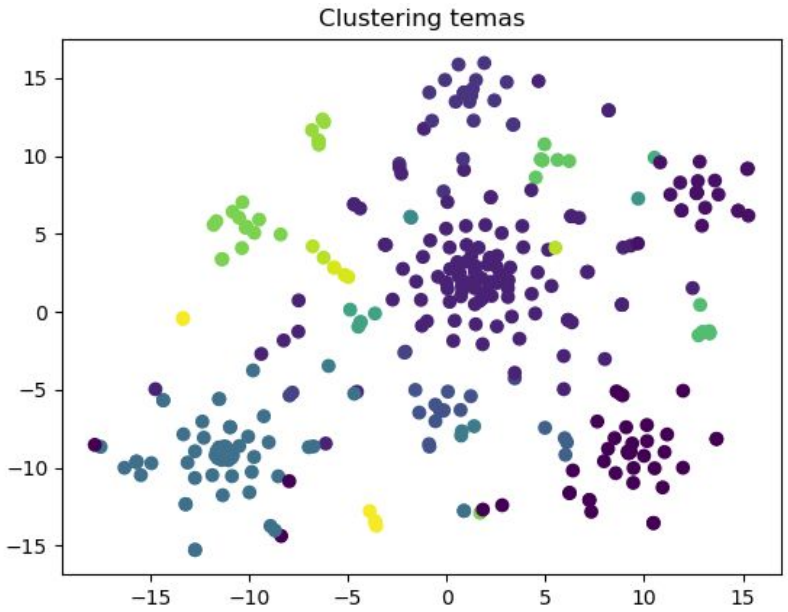


Figura 31: Representación visual de clusters de temas relacionados textualmente.

En los experimentos realizados, se ha observado (Figura 31) que generalmente se genera un cluster muy disperso, cercano al centro de la imagen (coordenadas (0,0)), que contiene temas con centros con poco contenido textual (y de pocas palabras en los tweets que lo forman), o que no ha sido posible agrupar con ningún otro cluster. En cambio, clusters que se alejaban del punto central suelen ya estar mejor formados, siendo más pequeños y densos, con una clara relación textual en su contenido, y por tanto de mayor interés para nuestros propósitos.

Si pasamos el cursor por encima de cada uno de los puntos, podremos ver el mensaje del tweet central del tema y si hacemos clic con el botón derecho encima de ellos podremos ver la ficha (Figura 32) que incluye los aspectos más relevantes del tema, así como los tweets que lo componen.

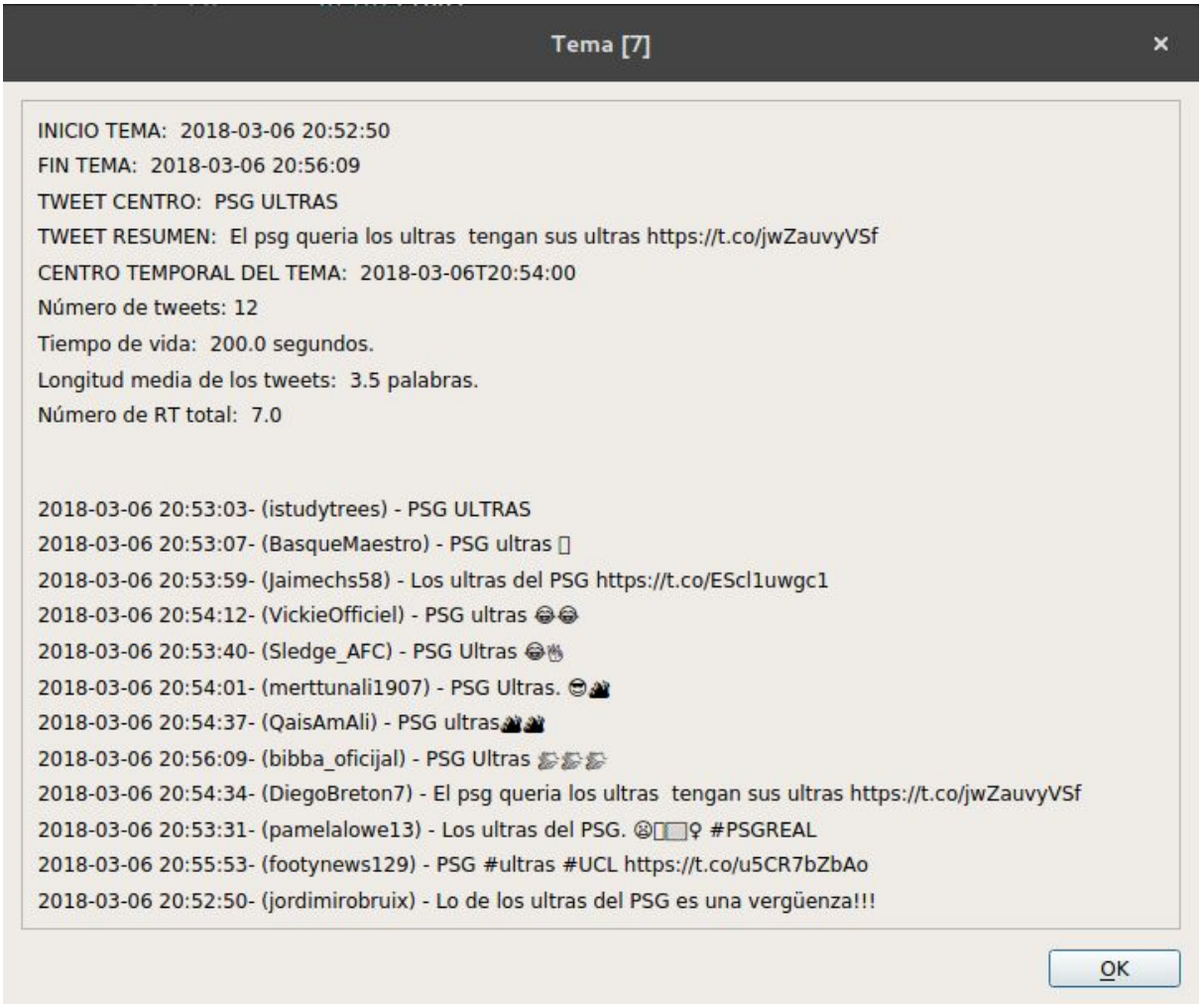


Figura 32: Ficha de un tema detectado, incluyendo sus tweets y factores de caracterización.

9. Experimentos

Para comprobar la eficacia del método propuesto se ha realizado una serie de experimentos que nos han permitido estimar la capacidad de detección de temas sobre eventos de distintas características, tanto en frecuencia de publicación, como temática y duración.

En particular se han analizado diversos partidos de fútbol (PSG - Real Madrid, Juventus - Real Madrid, Final de la Champions, Atlético de Madrid - Real Madrid) y concursos "musicales" (Eurovisión y final de Operación Triunfo). En este capítulo nos centramos en uno de cada tipo, en particular el partido de fútbol PSG-RMA y la gala final de OT.

Una cuestión importante es la estimación de la "calidad" de los temas encontrados. Hay que notar que al tratarse de un método no supervisado no tenemos un conjunto test para la validación, ni para el ajuste de parámetros. En su lugar, proponemos comparar los resultados obtenidos con los hitos del evento reflejados en la prensa, o bien apuntados manualmente durante el evento.

En el primero de estos experimentos se describen de manera detallada los pasos seguidos, para poder tener una visión completa del proceso desde la recopilación de los tweets hasta la presentación de los resultados.

Experimento 1 - PSG vs RMA 06/03/2018 - Segunda parte:

El primer evento analizado fue un partido de fútbol, en concreto el partido del PSG contra el Real Madrid celebrado el día 6 de marzo de 2018 (1-2).

1. Recopilación de tweets:

El primer paso es obtener los tweets relativos al evento, por lo que unos minutos antes de que empezase el partido se lanzó el script *twitter-listener.py* y se almacenaron sus resultados en un fichero de salida, en este caso al que hemos llamado *datospsg.* Como palabras clave por las que filtrar los tweets relativos al evento, hemos usado los nombres de los dos equipos enfrentados.
En concreto, la llamada ejecutada ha sido:

```
$ python twitter-listener.py PSG/ datospsg "PSG" "Real Madrid" ...
```

Solo se incluyen algunas de las palabras clave empleadas.

2. *Importación a MongoDB y preprocesado:*

Una vez contamos con los tweets en bruto del evento, es necesario importarlos a MongoDB y realizar las operaciones de preprocesado, para lo cual ejecutamos:

```
$ python tweet-preprocess.py PSG/base_datos psg PSG/datospsg
```

Esta llamada creará un directorio llamado `base_datos` dentro del directorio *PSG*, donde MongoDB almacenará la información de la base de datos a la que llamará *psg.* El tercer parámetro de la llamada indicará el archivo de los tweets recopilados a procesar, en este caso `datospsg`, que se encuentra también en la ruta `PSG/`. En nuestro ejemplo la base de datos resultante contiene 134197 tweets. El script Python marca cada tweet como original o retweet, además de otras tareas como la eliminación de urls o menciones. El proceso se lleva a cabo en pocos segundos, aunque para conjuntos de datos realmente grandes, del orden de millones de tweets, puede llegar a tardar minutos. En todo caso la aplicación no tiene una limitación de tamaño máximo más allá de las que imponga el propio MongoDB, que ha sido diseñado para trabajar en entornos Big Data.

3. *Detección de temas:*

Una vez se han importado los tweets a MongoDB y se realizado el preprocesado, ya podemos ejecutar la aplicación de detección de temas.
La aplicación emplea un archivo de configuración `config.json` con el formato indicado en Anexo A, que puede ser adaptado a cada caso concreto. En particular, en este ejemplo, se ha usado la siguiente configuración:

```
{       "lang" : "spanish",
        "tweets_ventana" : 10000,
        "muestra_ventana": 3000,
        "num_muestra_tema" : 5,
        "min_tweets_similares": 4,
        "umbral_inicial" : 0.8,
        "filtrar_unitarios": 500,
        "umbral_restantes" : 0.7,
        "porcentaje_fusion_interna": 0.5,
        "umbral_fusion_interna": 0.7,
        "tweets_nucleo": 5,
        "umbral_fusion_externa": 0.7
}
```

PSG/config.json

Una vez se ha creado este archivo, realizaremos la llamada a la aplicación de detección de temas de la siguiente manera:

```
$ python tweet-analysis.py PSG/ psg 27017
```

De esta manera se indica que la ruta de trabajo es *PSG/*, donde se debe encontrar el archivo *config.json* y donde se almacenarán los archivos de salida.

Como el partido tuvo sus momentos de mayor interés en la segunda parte, cuando se marcaron los 3 goles del partido, en esta sección incluímos únicamente las opiniones de esta segunda parte.
Para ello, cuando lanzamos el script con la llamada anterior, podemos seleccionar en la interfaz gráfica el intervalo de análisis desde las 20:48 hasta las 21:39 del día 6 de marzo de 2018.

En concreto, la detección de temas tarda menos de 7 minutos en un ordenador portátil modelo Asus X555-LD, con 8GB de RAM, un procesador i7-4510U a 2.0GHz y empleando un disco duro SSD, analizando un total de 65891 tweets originales, los cuales se ven repartidos en 7 ventanas.
Se detectaron un total de 47 temas, encontrando 4 grandes concentraciones de temas a simple vista, como podemos ver en la Figura 33.

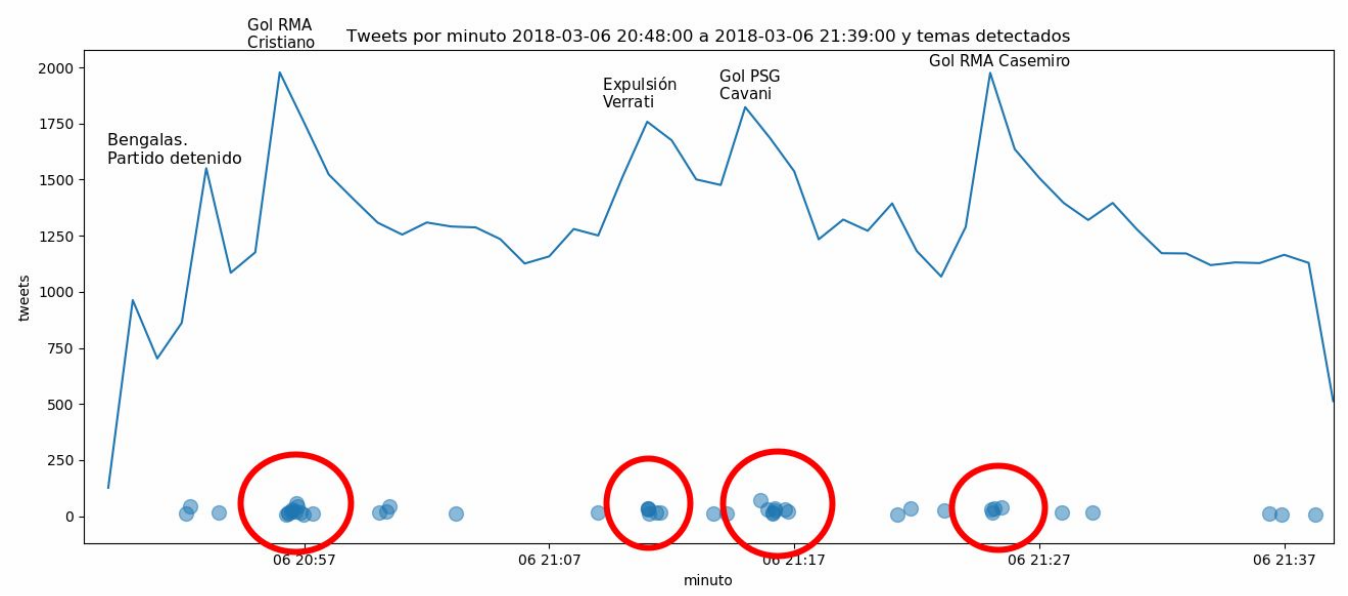


Figura 33: Concentración de temas en la vista general del Experimento 1.

Para validar la capacidad de detección de noticias se han marcado los hitos del periodo considerado. En particular, hemos utilizado la crónica minuto a minuto del periódico deportivo online Marca [45], seleccionado 3 criterios para filtrar los hitos relevantes entre los comentarios minuto a minuto recogidos por este medio:

- Uso de mayúsculas combinadas con negrita en el texto.
- Inclusión de fotografías junto al texto.

Con estos criterios, vamos a mostrar los hitos (en el formato horario seguido en nuestro conjunto de datos), indicando además cómo se reflejan en el diagrama de frecuencias y si han sido detectados como noticias por la aplicación.

Hora	Comentario	Reflejado/Detectado
20:53	Lanzamiento de bengalas de los aficionados del PSG, interrupción del partido.	Detectado como agrupación menor con 3 noticias.
20:56	Gol del RMA por Cristiano Ronaldo.	Un grupo de tamaño considerable de noticias detectado en este momento (primer círculo rojo).
21:03	Remate de Cavani.	No reflejado, ni como incremento de frecuencia ni como detección de noticia por nuestra aplicación.
21:04	Cambio en el PSG.	No reflejado como cambio de frecuencia ni detectado como noticia.
21:11	Expulsión de Verrati.	Máximo local de frecuencias y detectado por varias noticias en la aplicación (segundo círculo rojo).
21:14	Tiro al palo de Asensio.	No reflejado en el diagrama de frecuencia ni como tema.
21:15	Gol del PSG por Cavani.	Incremento en frecuencia y nuestro tercer círculo de noticias.
21:19	Ocasión fallada por Benzema.	En frecuencias se nota un pequeño incremento, y es detectado como noticia.
21:21	Segunda sustitución PSG y en el RM.	Sin incremento de frecuencias y sin noticia.
21:23	Amarilla para Ramos.	Sin reflejo en frecuencias ni en noticias.
21:25	Gol del Real Madrid por Casemiro.	Gran “pico” de frecuencias y el cuarto círculo rojo de noticias.

Atendiendo al fichero *salida.txt* podemos comprobar que los temas detectados coinciden con el momento de los hitos principales. No se reflejan hitos “secundarios” como tarjetas amarillas o sustituciones, pero sí los principales como goles o expulsiones. Además, la aplicación ha detectado otros temas que no corresponden con los hitos. Suelen ser, bien reacciones tardía a un hito, o comentarios animando a uno de los equipos. En general, podemos decir que la aplicación detecta correctamente las reacciones a los momentos más importantes del evento.

Para hacernos una idea de qué ocurrió concretamente en el primero de los momentos de alta concentración (primer círculo rojo), lo más sencillo es fijarnos en el **resumen** y el **centro temporal**, que nos dan una descripción bastante completa de cada tema.

Por ejemplo, encontramos, entre otros:

Centro temporal	Resumen
2018-03-06T20:56:27	*GOOOOOOOOLLLLLLLLL CR7 REAL MADRID REAL MADRID REAL MADRID REAL MADRID*
2018-03-06T20:56:21	*Cristiano. Acabou pro PSG*
2018-03-06T20:56:39	*Bye Bye Emre Bye Bye PSG*
2018-03-06T20:56:49	*51' Goal Real Madrid (Cristiano Ronaldo) Paris St.-Germain 0:1 Real Madrid #PSGRealMadrid https://t.co/QqNBJpcTi5*

Como podemos ver, en el minuto 51 del partido, 20:56 en nuestro análisis, el Real Madrid marcó un gol. Los resúmenes mostrados corroboran que un hito en el evento da lugar a varios temas, que aunque coincidentes en el tiempo, pueden tener contenidos muy diversos. Sería erróneo considerar que el máximo de frecuencias lleva asociada una noticia, porque esto supone confundir el concepto de hito (suceso real acaecido en el evento), con las reacciones que despierta este hito en Twitter. El hito en sí, es a menudo uno de los temas, pero para alguien interesado por ejemplo en conocer la opinión de los aficionados, puede ser poco relevante. En cambio en este ejemplo sí llama la atención que los aficionados inmediatamente tras el gol den por finalizado el trabajo del entrenador del equipo contrario (“*Bye Bye Emre Bye Bye PSG”).*

4. ***Agrupación por proximidad temporal:***

Tal y como indicamos, contamos con dos métodos de agrupación de temas, uno temporal y otro temático. El análisis por proximidad temporal de los temas, debe ser capaz de representar los grupos que se aprecian en la Figura 33.

Para obtener la agrupación temporal de temas en momentos, tal y como los llamábamos en la sección Proximidad temporal, ejecutaremos:

```
$ python tweets-time-clustering.py PSG/ 5
```

El script muestra un gráfico donde las altas concentraciones de temas se representan como puntos de tamaño mayor. En concreto, en la Figura 34 vemos que se forman 6 momentos, correspondiendo los 4 mayores a los que pudimos detectar a simple vista en la primera visualización y que agrupan con gran cantidad de temas.

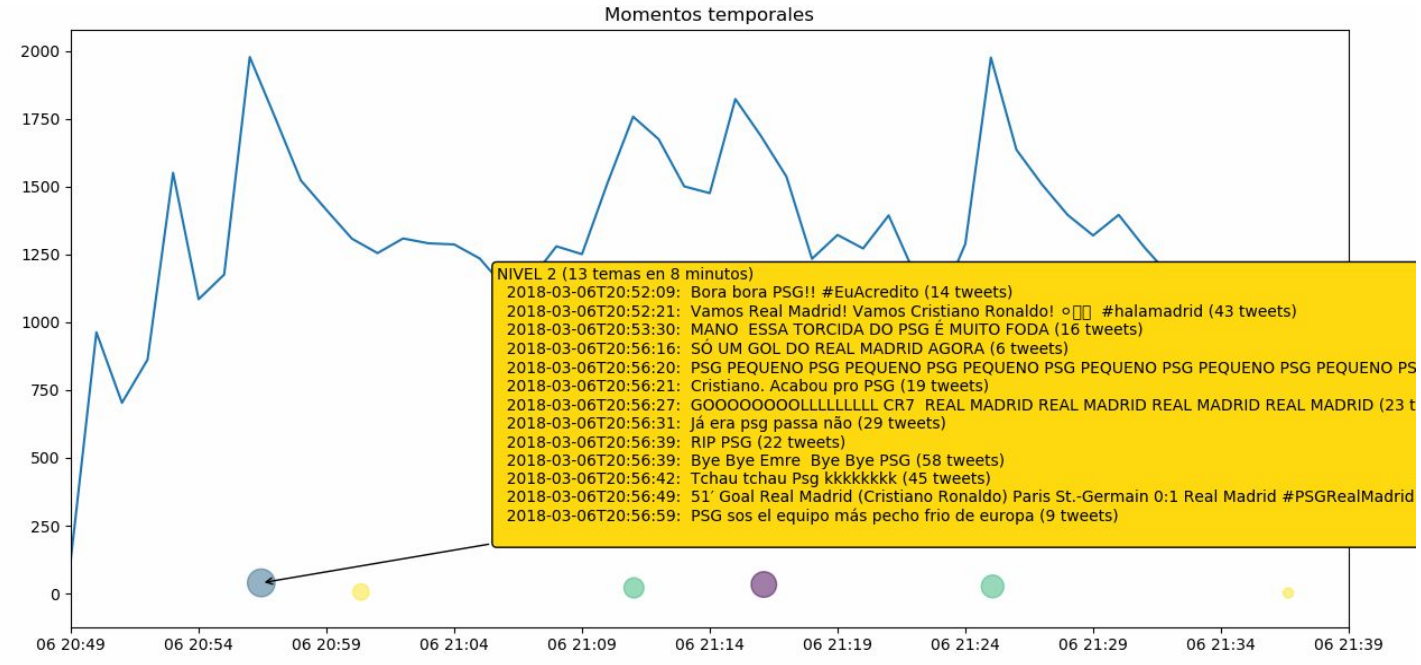


Figura 34: Detalle de un momento por agrupación temporal de temas del PSG vs RMA (n=5).

Además, al pasar el cursor sobre cada momento, podemos ver los temas agrupados, tal y como muestra la Figura 34.

5. ***Agrupación por similaridad textual:***

Si, por otro lado, deseamos ver las relaciones de contenido entre los temas, ejecutaremos:

```
$ python tweets-textual-clustering.py PSG/
```

La siguiente tabla recoge algunos de los clusters detectados, permitiendo comprobar que, en efecto, se agrupan por relación temática. En el encabezamiento de cada clúster se muestra el tweet central del tema que a su vez ocupa el centro geométrico del clúster. Hemos optado por el centro en lugar del resumen porque al ser más escueto nos da una idea de las “palabras clave” del clúster.

Cluster 1: acabou pro psg
0: 2018-03-06 20:56:21→ *Cristiano. Acabou pro PSG* **1:** 2018-03-06 21:11:00→ *Verrati expulso. Acabou pro PSG.*
Cluster 2: Real Madrid
0: 2018-03-06 20:56:16→ *SÓ UM GOL DO REAL MADRID AGORA* **1:** 2018-03-06 20:56:27→ *GOOOOOOOOLLLLLLLLL CR7 REAL MADRID REAL MADRID REAL MADRID REAL MADRID*

2: 2018-03-06 21:00:21→ *¡GOL de Cristiano (Real Madrid)! PSG - Real Madrid 0-1. En directo: https://t.co/i3uSDUkMx8 https://t.co/fNhgsvnoQ0*
Cluster 3: Nada de PSG
0: 2018-03-06 20:56:20→ *PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQ… https://t.co/tl96OaCOHQ* **1:** 2018-03-06 21:11:01→ *El PSG*😂😂 **2:** 2018-03-06 21:11:21→ *Madrid Madrid Madrid adeus PSG time pequeno https://t.co/XLQJ5xZ9D0* **3:** 2018-03-06 21:15:37→ *Mais cagado que esse gol do psg só o benzema jogando* **4:** 2018-03-06 21:36:54→ *PSG muito pequeno*

Por ejemplo, en el primer clúster, encontramos dos temas separados temporalmente, el primero como reacción al primer gol del RMA, y el segundo a la expulsión de un entrenador. A pesar de las diferencia de los hitos correspondientes, ambos provocan una reacción muy similar en los usuarios, que señalan que el equipo PSG ya no tiene nada que hacer en este partido.

En este caso, al ser pocos los temas detectados, la representación visual de la agrupación temática que se presenta en la Figura 35, da una impresión de nube algo dispersa, pero que nos permite visualizar la cercanía entre los temas agrupados conjuntamente.

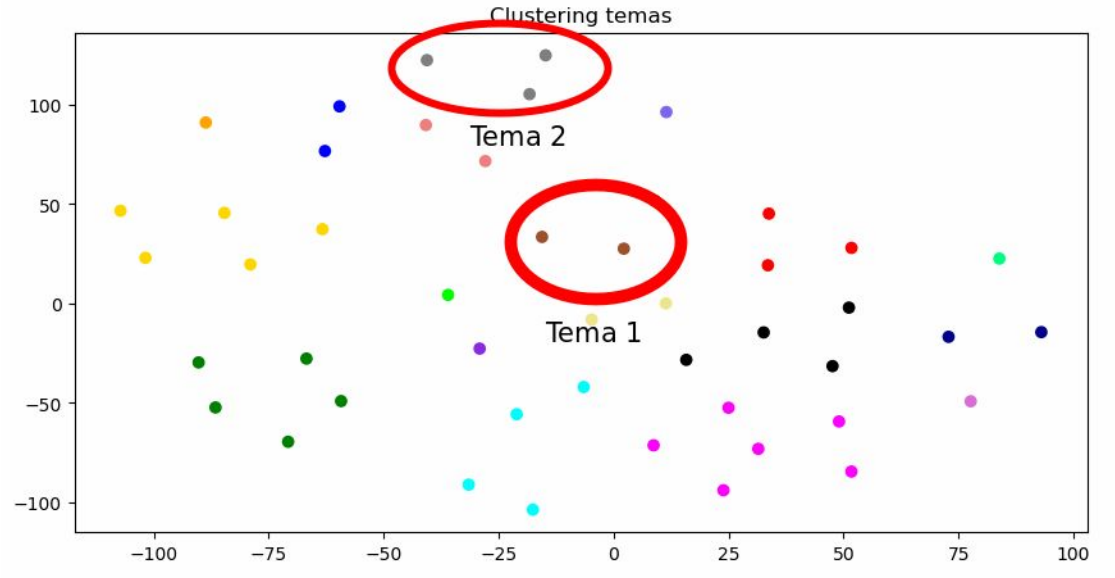


Figura 35: Clustering de temas por similaridad textual PSG vs RMA.

En este caso la representación gráfica no es muy ilustrativa, mientras que como vimos anteriormente en el caso de la agrupación en momentos sí que lo era. Esto nos puede a llevar a pensar que este segundo método de agrupación es innecesario pero, como veremos a continuación, en otro tipo de eventos sucede lo contrario, lo que justifica que dispongamos de ambas opciones de agrupación.

Experimento 2 - Final Operación Triunfo:

Un segundo evento analizado fue la final de Operación Triunfo 2017, celebrada el 5 de febrero de 2018. El programa fue un éxito desde el punto de vista televisivo, con una audiencia de casi 4 millones de espectadores. Esto se reflejó en las redes, donde el programa fue comentado día tras día por sus jóvenes seguidores. Por ello, la gala final (de más de 3 horas de duración) fue seguida y debatida por Twitter con un gran entusiasmo.

Tal y como hemos hecho en el apartado anterior, plasmamos gráficamente los hitos clave del evento en la Figura 36. Podemos encontrar que los hitos en los que se produjeron las actuaciones durante la gala coinciden con los momentos más comentados en la red social.

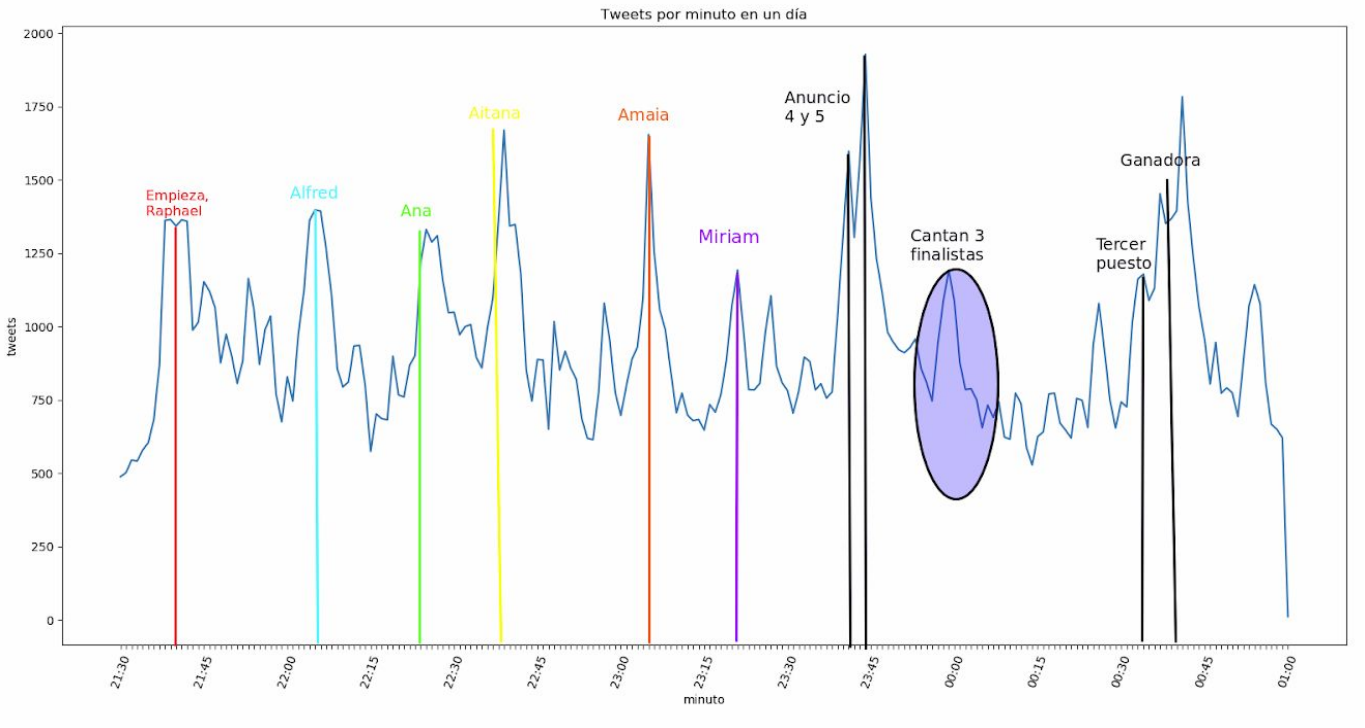


Figura 36: Hitos de la gala final de Operación Triunfo 2017.

En concreto, se analizó un intervalo temporal desde las 21:35 del día 5 de febrero hasta las 01:00 del día 6, cuando terminó el programa tras anunciar el ganador del concurso.
De un total de 946,213 tweets publicados durante el evento, se publicaron 194354 tweets originales que se analizaron en 20 ventanas, tardando un tiempo aproximado de 35 minutos, y dando lugar a un total de 332 temas.

Al contrario de lo que ocurría en el ejemplo anterior, la representación inicial de las noticias detectadas (Figura 37) en este caso no nos permite intuir a simple vista ningún tipo de relación entre ellas, por la gran cantidad de temas detectados.

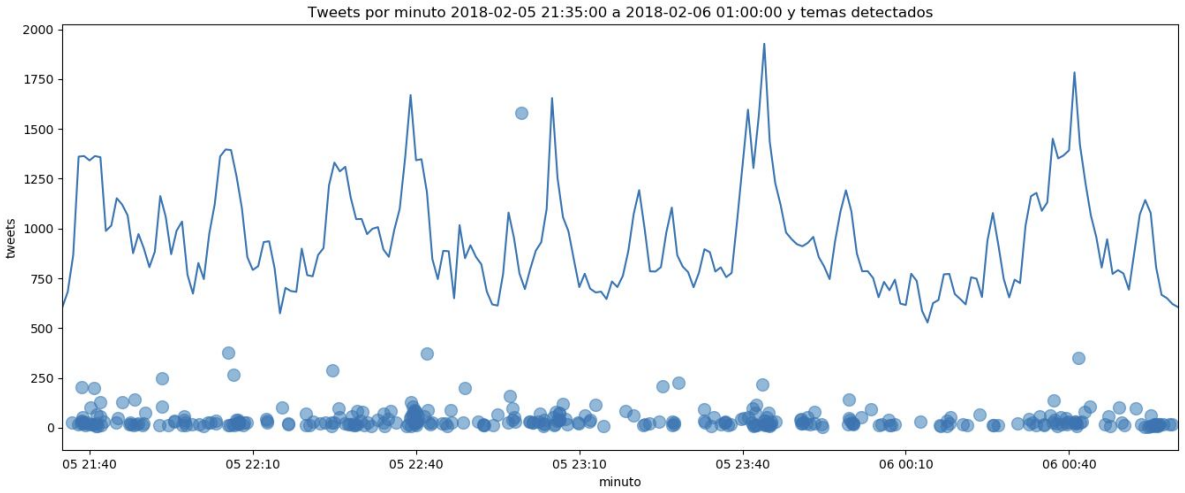


Figura 37: Representación inicial de las noticias detectadas en el Experimento 2.

Por ello, aplicamos nuestro algoritmo de agrupación de temas por proximidad temporal, indicando en este caso un número mayor de niveles, para que el grado de densidad requerido en cada uno de los niveles disminuya de manera más progresiva.

En concreto, la representación visual de los momentos detectados que mostramos a continuación (Figura 38) resalta los momentos de alta densidad, coincidiendo en gran medida con los picos de alta frecuencia de publicación y a su vez con los hitos anteriormente representados.

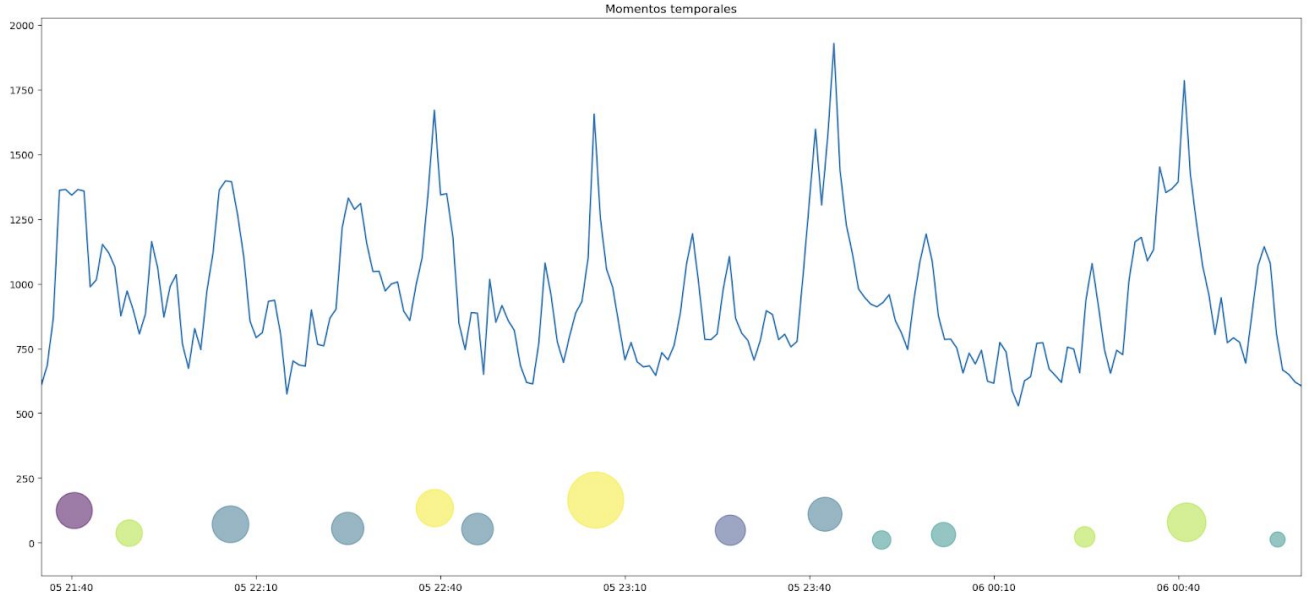


Figura 38: Agrupación temporal de temas en el Experimento 2 (n=10).

Si pasamos el cursor sobre cada uno de estos círculos, podremos ver el conjunto de temas que agrupa. Por ejemplo, en el primero de estos círculos, que coincide con el inicio del programa y con la actuación del cantante Raphael, muestra los siguientes temas.

Centro temporal	Resumen	Tweets
2018-02-05 21:36:48	*No sé si quiero que empiece #OTFinal https://t.co/c44SZolfIq*	28
2018-02-05 21:37:55	*OJALÁ UNA GALA FINAL DE HORA PUNTA #OTFinal*	19
2018-02-05 21:38:23	*@iris_jabonero @Anserlam EMPIEZAAAAA EMPIEZAAAAA #OTFinal https://t.co/loDPPCLSrs*	21
2018-02-05 21:38:29	*Dios estoy nerviosa de verdad soy una madre nerviosa #OTFinal*	30
2018-02-05 21:38:32	*Ahora si empiezaaaa 🙈🙈🙈🙈 AITANA GANADORA*	35
2018-02-05 21:38:36	*Ay ay ay que empieza. #OTFinal*	203
2018-02-05 21:38:43	*Nervios nervios nervios #OTFinal*	54
2018-02-05 21:38:48	*No estoy preparado no estoy preparado no estoy preparado no estoy preparado no estoy preparado no estoy preparado no… https://t.co/dDcMh8E8bV*	25
2018-02-05 21:38:53	*Que no estoy preparado he dicho!!!!! EL PRINCIPIO DEL FIN😏 #otfinal*	23
2018-02-05 21:39:15	*YA EMPIEZA ME MUERO #OTFinal*	14
2018-02-05 21:39:36	*Vamos a llorar todos Aitana VAMOS A LLORAR. #OTFinal*	27
2018-02-05 21:39:51	*Ya estoy llorando y acaba de empezar #OTFinal*	18
2018-02-05 21:40:07	*No estoy preparada no estoy preparada no estoy preparada no estoy preparada no estoy preparada no estoy preparada… https://t.co/xLlOaAXXu9*	103
2018-02-05 21:40:25	*Entre los bailarines de burbujas freixenet y Raphael parece que hemos vuelto a Navidad. #OTFinal*	15
2018-02-05 21:40:44	*AMAIA Y ALFRED POR QUÉ ESTÁIS TAN PRECIOSOS OS AMO #OTFinal*	19
2018-02-05 21:40:47	*Que guapos que guapos QUE GUAPOS. #OTFinal*	198
2018-02-05 21:41:07	*OJALÁ GANE A AITANA SÍ LO DIGO #OTFinal*	10
2018-02-05 21:41:15	*Llega el momento suena "Mi gran noche". Su gran noche ¡Nuestra gran noche! #OTFinal https://t.co/KcGOLos2kD*	64
2018-02-05 21:41:17	*Amaia está preciosa va de ganadora. #OTFinal*	23
2018-02-05 21:41:25	*Ahora son coristas de Raphael. Como debe ser. #OTFinal*	8
2018-02-05 21:41:54	*RAPHAEL RAPHAEL RAPHAEL #OTFinal*	130

2018-02-05 21:41:58	*QUE GUAPÍSIMOS TODOS LOS PROFES joder como los quiero de verdad #OTFinal*	55
2018-02-05 21:42:07	*Por favor mis niños maravillosos 🥰♥♥♥ #OTFinal*	12
2018-02-05 21:42:41	*Empezamos la noche con la gran final!!!🎤🎶 #OTFinal*	32
2018-02-05 21:44:50	*"Primero está la personalidad de uno y luego vienen los consejitos" Pues sí Raphael qué razón #OTFinal*	25
2018-02-05 21:45:13	*Q no quiero q se acabe #OTFinal*	47
2018-02-05 21:45:59	*Mi ganadora es Aitana #OTFinal*	126

Tal y como vemos en la tabla, en un intervalo de 11 minutos se dieron un total de 27 temas u opiniones relativas al hito que marca el comienzo del programa y la actuación del *cantante*, y que podemos ver resumidos detallando su centro temporal, resumen y número de tweets que agrupa.

Si, por otro lado, analizamos los temas detectados atendiendo a su relación temática, se agrupan los 332 temas en 12 clusters, quedando representados visualmente como podemos ver en la Figura 39.

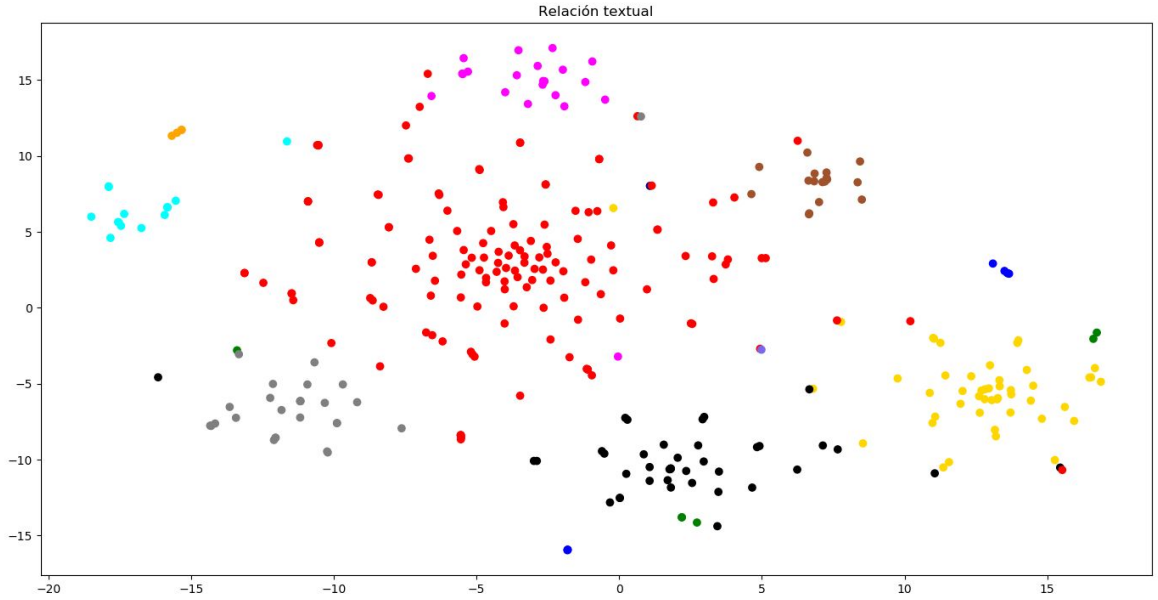


Figura 39: Clusters por similaridad textual formados en el Experimento 2.

Atendiendo a los resultados, podemos observar un grupo cluster central de temas (en rojo), que como se menciona en la subsección Relación textual, está formado por temas difícilmente clasificables. Si nos fijamos en los clusters formados lejos de este punto central, es muy interesante comprobar que 5 de los clusters mejor definidos corresponden a cada uno de los concursantes, agrupando para cada uno de ellos las opiniones más relevantes de los usuarios, tal y como se ve en la siguiente figura.

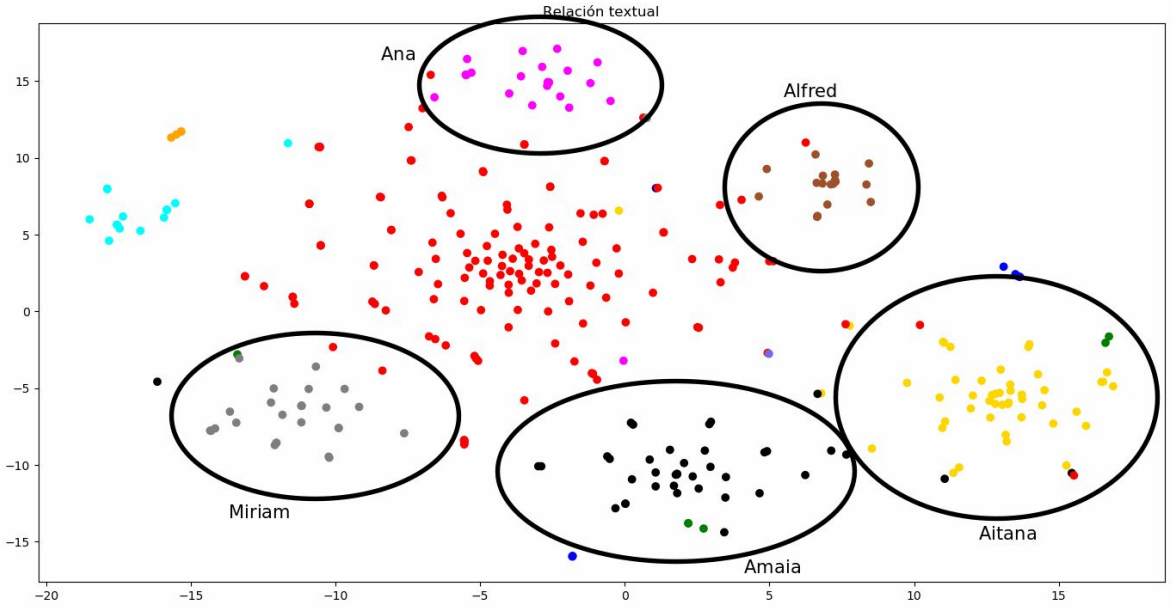


Figura 40: Representación de clusters de concursantes.

Los cluster con mayor número de puntos, corresponden a las finalistas de la gala: Miriam, Amaia y Aitana, que además actuaron dos veces lo que justifica la mayor cantidad de noticias asociadas. El cluster de color azul celeste corresponde a opiniones acerca de las votaciones y deseos de que gane algún candidato.

Por ejemplo, si analizamos el cluster 8 (Figura 40, color marrón), todos los temas que se han agrupado están relacionados con Alfred, como podemos ver en el detalle de algunos de los resúmenes de este cluster.

Centro temporal	Resumen
2018-02-05 21:54:29	*Alfred habla primero porque va a ser el primero en irse muchas gracias!! #OTFinal*
2018-02-05 21:55:39	*“Yo con Alfred me iría al fin del mundo”. “Alfred como persona es un ángel”. MADRE MIA ES QUE ME REPRESENTAN #OTFinal*
2018-02-05 21:57:31	*El niño es un ángel... Viva Alfred!!! #OTFinal*
2018-02-05 21:57:44	*Alfred es tan puro y tan especial... #OTFinal*
2018-02-05 22:02:16	*Alfred bailando "Lo Malo" es lo mejor que vas a ver hoy #otfinal*
2018-02-05 22:06:30	*Alfred es Alfred y es lo bueno de Alfred. Y ya está #OTFinal*
2018-02-05 22:06:52	*Madre mía Vicky. Madre mía Alfred. Madre mía el temazo ❤. #OTFinal*
2018-02-05 22:07:15	*a mí es que la voz de Alfred me tiene enamorada desde el principio #OTFinal*

2018-02-05 22:07:18	*Alfred sublime. Es que no hay más. Alfred please don't stop the music. Eres mágico. #OTFinal*
2018-02-05 22:07:21	*Alfred no es lo más maravilloso que le ha pasado a OT es lo más maravilloso que le ha pasado a la música. https://t.co/63a32Z76EZ*
2018-02-05 22:07:27	*Primera actuación y ya estoy llorando qué artistazo es Alfred #OTFinal*
2018-02-05 22:07:29	*Numerazo el de Alfred genial Alfred y Vicky #OTFinal*
2018-02-05 22:08:52	*Música en MAYÚSCULAS... ALFRED GANADOR GANA ALFRED 27734!!!! #OTFinal @OT_Oficial*
2018-02-05 23:44:22	*Ahora Alfred. Ahora Alfred. Ahora Alfred. Ahora Alfred. Ahora Alfred. Ahora Alfred. Ahora Alfred. Ahora Alfred. https://t.co/j5ScMUxUsr*
2018-02-05 23:44:42	*No es Alfred cuarto es Miriam tercera. #OtFinal*
2018-02-05 23:51:51	*Lo siento pero el cuarto puesto no era para Alfred ni mucho menos #Alfred #OTFinal*
2018-02-06 00:55:40	*raoul yendo a cantar con alfred son hermanos*

Los temas agrupados corresponden en su mayoría a las reacciones cuando actuó este concursante, alrededor de las 22:07, pero también podemos apreciar reacciones cuando se proclamó cuarto finalista, tal y como vemos a continuación en la Figura 41.

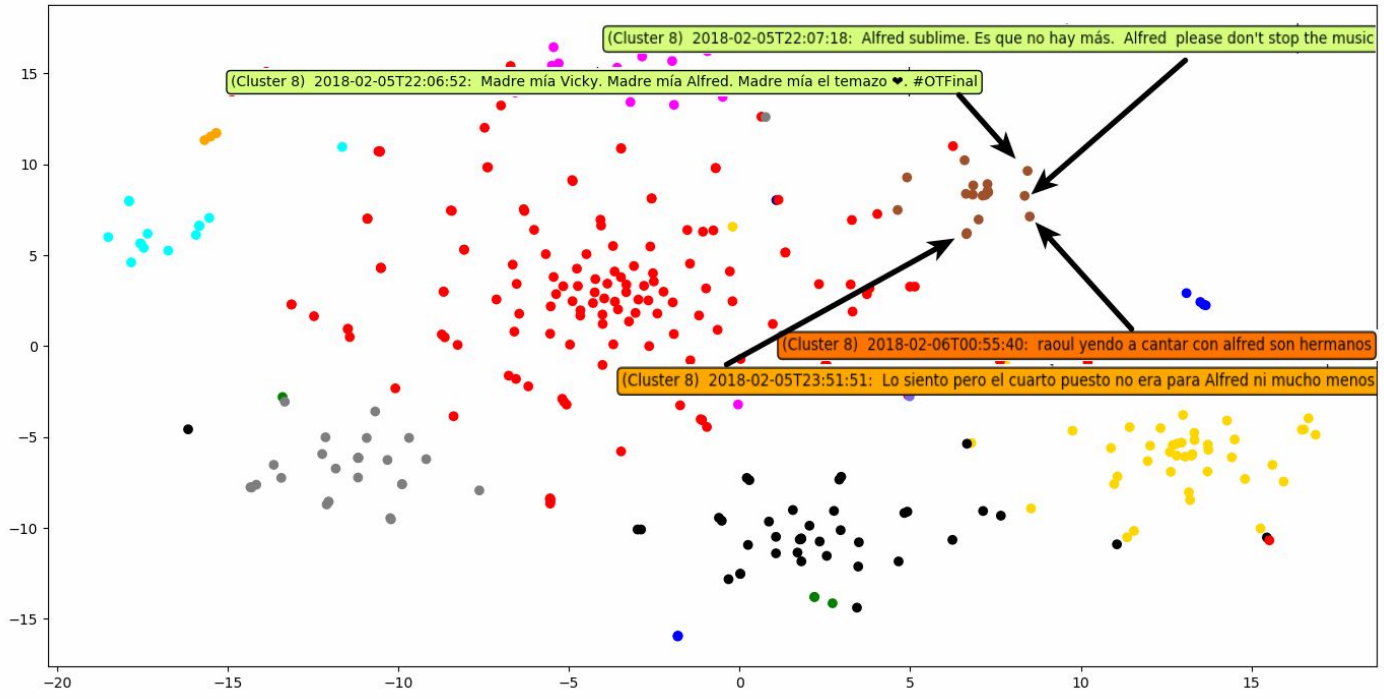


Figura 41: Representación de resúmenes de algunos de los temas del cluster sobre el concursante Alfred.

10. Conclusiones

10.1. Contribuciones

En este trabajo se ha presentado un método para analizar los mensajes publicados en Twitter sobre cualquier evento concreto y extraer información de interés.
Nuestra aplicación toma como entrada los tweets correspondientes a un evento y los agrupa en noticias o temas de contenido similar, permitiendo ofrecer al usuario una visión global de los hitos más importantes del evento.

El esquema propuesto utiliza una similaridad basada en el modelo de representación de documentos como espacio vectorial y el modelo numérico TF-IDF, pero esta función de comparación podría sustituirse por cualquier otra función de similaridad siempre que cumplan ciertas condiciones, como es la capacidad de asignar a cada pareja de tweets un valor de similaridad situado en el intervalo $[0, 1]$.

Nuestro trabajo intenta combinar las dos alternativas propuestas habitualmente en la literatura: proximidad temporal y similaridad entre tweets. Cada una de estas propuestas lleva aparejada una deficiencia. La proximidad temporal, basada en el análisis de frecuencias, tiende a localizar hitos del evento y no temas de discusión suscitados por el hito. En cambio la similaridad temática puede agrupar tweets muy alejados temporalmente que corresponden en la práctica noticias diferentes, pero expresadas de forma similar.

La propuesta integra la proximidad temporal dividiendo el dataset inicial en ventanas de tweets cronológicamente consecutivos. La segunda propuesta se aplica entonces a estas ventanas, buscando similaridad textual, pero ya dentro de un entorno temporal delimitado. De esta manera, se evita identificar temas de contenido similar pero que han ocurrido en momentos diferentes, y a la vez se distinguen diferentes noticias asociadas a cada momento. Con ello, conseguimos cumplir los objetivos propuestos **O1** y **O2**.

Una conclusión importante del trabajo es que no debe confundirse hito (suceso real en el evento) con tema en las redes sociales. Un hito suele suscitar varios temas, uno de los cuales puede ser el reflejo directo del propio hito en la red. Sin embargo, limitarse a la búsqueda de hitos significa reducir, y por tanto perder la riqueza de temas asociados. Los hitos ya vienen reseñados en los medios de comunicación, y los momentos en los que ocurren pueden obtenerse directamente del análisis de frecuencias. En cambio, las opiniones que generan estos hitos sí pueden ser de interés y constituyen la aportación principal de la red social.

Por ejemplo, hemos visto que uno de los temas relacionados con un gol del Real Madrid durante el partido con el PSG, contiene la opinión de que el entrenador del PSG debe abandonar su puesto. Esta reacción de los aficionados puede ser de gran importancia para

los analistas de redes del club, ya que indican a quién señalan los aficionados como responsable del gol.

Nuestro enfoque genera una gran cantidad de temas. En lugar de filtrarlos, descartando algunos y por tanto perdiendo posible información útil, se ha optado por agruparlos a la hora de la presentación al usuario. Esta agrupación se puede realizar o bien de forma temática, relacionando temas con contenido similar, o bien de forma temporal. De esta manera, se ha cumplido el objetivo propuesto **O3**.

Puede parecer que, en realidad, en lugar de decidir *a priori* si usar un criterio temporal o temático, como hacen otras propuestas, este trabajo solo retrasa esta decisión hasta el momento de la presentación de resultados, pero que en realidad esto aporta poco con respecto a trabajos previos. La diferencia notable es que esta agrupación final se hace sobre temas ya formados, no sobre tweets. Es decir los tweets se agrupan en temas sin fijar un criterio *a priori*, mediante el algoritmo descrito en el capítulo Detección de temas, por lo que se evitan los defectos que hemos mencionado asociados a la adopción de un criterio inicial, temporal o temático.

El trabajo presenta dos contribuciones principales, en forma de dos algoritmos de clustering, el primero destinado a la localización de noticias, y el segundo orientado a la detección de "momentos" o agrupaciones de temas por proximidad temporal. En particular:

- *Algoritmo de detección de temas:* presentamos un algoritmo novedoso que en primer lugar genera noticias a partir de una muestra de la ventana de tweets considerada. El resto de los tweets se incorporan posteriormente a los temas ya formados. El algoritmo incorpora además fases de refinamiento que incluyen compactación, eliminación de temas irrelevantes y un proceso de fusión de temas tanto dentro de una ventana como entre ventanas. El resultado es un método eficiente en cuanto al tiempo requerido para la detección de temas.
- *Método de agrupación de temas por proximidad temporal:* se ha descrito un método que permite formar conjuntos de noticias que ocurren en un entorno temporal próximo, a los que se ha denominado momentos. Este método de agrupación se basa en el concepto de densidad de temas en los intervalos temporales, y propone un algoritmo que detecta estos momentos disminuyendo progresivamente la densidad mínima exigida.

Otras contribuciones de este trabajo son:

- En el apartado de clustering de temas por contenidos similares incorporamos, con respecto a otras propuestas analizadas, la detección automática del número de temáticas óptimo.
- La aplicación de logaritmos al resultado del coseno de los vectores TF-IDF, para lograr una gradación más progresiva de la similaridad. Esta idea ha surgido al ver la distribución de la similaridad basada en el coseno de los vectores TF-IDF, que tiende a dar una gran similaridad a tweets casi iguales, bajando muy rápido a una similaridad cercana a 0 para tweets con solo ligeras diferencias. De esta manera

logramos temas más amplios, que contienen una variedad mayor de tweets, aunque sin perder la coherencia temática.

10.2. Trabajo futuro

Durante el desarrollo del trabajo han ido surgiendo posibles líneas de continuación del trabajo. Mencionamos a continuación algunas de ellas.

Ajuste semi-automático de parámetros

La técnica presentada depende de una gran cantidad de parámetros, que se pueden consultar en el Anexo A. En los métodos de aprendizaje automático supervisados el ajuste de estos parámetros para lograr mejores resultados se puede realizar de manera completamente automática para cada dataset, dividiendo el conjunto de datos entre entrenamiento, validación y prueba. Esto no es tan inmediato en métodos no supervisados como el nuestro. Algunos parámetros, como el valor k óptimo en k-means ya son determinados en nuestra propuesta, pero quedaría definir métodos que proporcionen o bien el valor óptimo o al menos un intervalo de valores para el resto de parámetros.

Extensión de la noción de similaridad

Aunque los experimentos realizados han dado buenos resultados en cuanto a la detección de temas, un aspecto que ha quedado pendiente de investigar es la incorporación de otras características de los tweets que permitan extender y mejorar la noción de similaridad más allá de la similaridad textual de los mensajes empleada en este trabajo.

Las imágenes, en concreto, juegan un papel fundamental en Twitter y en general en las redes sociales, y sin embargo habitualmente no son consideradas. Para hacerlo se deben seguir dos pasos. El primero, posiblemente el más complicado, es definir medidas adecuadas de similaridad entre imágenes, que pueden ir desde el reescalado y parcelado de ambas imágenes comparando características como la proporción de píxeles de cada color, hasta técnicas sofisticadas basadas en la detección de contornos [46]. El segundo paso es redefinir la similaridad total entre dos tweets, donde ahora cada tweet x viene representado por dos componentes $x = (x_1,x_2)$, con x_1 el texto y x_2 la imagen. La definición de similaridad s, dependerá por tanto de la similaridad s_1 entre textos y de la similaridad s_2 entre imágenes, teniendo además en cuenta que, dependiendo del tweet, alguna de las dos componentes puede no existir, es decir el tweet puede ser solo texto, solo imagen o ambas cosas. Podemos imaginar que usamos un valor especial para denotar la ausencia del componente, por ejemplo *None*. Así, el vector *("Gol de ...",None)* indicaría un tweet con texto y sin imagen.

Una definición de similaridad capaz de tratar con estas situaciones es el llamado *coeficiente de similaridad general* [47], definido de la forma:

$$s(x,y) = \frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k)\, s_k(x_k, y_k)$$

donde *x,y* representan, en nuestro caso, los vectores con los componentes (texto e imágenes) de los dos tweets, *d=2*, $w(x_k,y_k)$ vale 0 o 1 dependiendo de si la comparación tiene sentido para los valores (en nuestro caso *w(a,b) = 0* si *a* o *b* son *None*, y *1* en otro caso) y las funciones s_k representan las dos similaridades para los dos componentes, extendidas para tratar el caso del valor *None* devolviendo cualquier valor arbitrario, ya que será multiplicado por *0*. Esta similaridad se puede extender además de forma sencilla para dar lugar a una noción adecuada de distancia, tal y como se muestra en [48].

Mejora de la eficiencia

El uso de MongoDB hace que nuestra propuesta sea escalable en cuanto a espacio. Sin embargo el tiempo el requerido para datasets grandes puede llegar a ser excesiva.

Para superar esta limitación, de la división de los mensajes en ventanas surge, casi de manera natural, la idea de realizar el procesamiento en paralelo de cada una de estos conjuntos de tweets, realizando también un proceso de distribución de los datos, lo que sin duda llevaría a altas mejoras en el rendimiento del algoritmo de detección propuesto.

Relaciones entre temas

Otra de las líneas de investigación que han quedado pendientes es el estudio de las posibles relaciones de causalidad entre los diferentes temas detectados. Un estudio en esta línea, tal y como realizaba el sistema ETree [21] nos permitiría comprobar la evolución de cada uno de los temas. Además, yendo un paso más allá, se podría realizar comparaciones entre la evolución de distintos temas, para estudiar si temas aparentemente sin ninguna relación son debatidos de manera similar por los usuarios de Twitter.

Palabras derivadas y sinónimos

La desambiguación de las palabras, a menudo realizada con un proceso de stemming, permite reducir palabras derivadas a su raíz semántica, aspecto que ayuda en ocasiones a las funciones de similaridad, pero que no da demasiado resultado con los mensajes de Twitter por su brevedad y falta de contexto. Nuestra última línea de investigación propone tratar de mejorar este proceso siguiendo el modelo Word Embedding [49] de manera que, mediante la medida de similaridad entre palabras (teniendo además en cuenta el contexto) que este modelo proporciona, podamos implementar una función de similaridad entre tweets que consiga resolver el problema de las palabras derivadas y sinónimas. Se han encontrado

estudios recientes [50] que emplean esta aproximación para el problema de la detección de temas en Twitter, pero consideramos que presentan la carencia de no contar con la cercanía temporal de los mensajes como factor relevante para considerar mensajes pertenecientes al mismo tema.

Nuestros experimentos iniciales utilizando algoritmos de deep learning mediante la biblioteca TensorFlow [51], nos indican que esta técnica permite en efecto extraer palabras que aparecen en un mismo contexto. Un ejemplo de representación visual de la relación contextual de las palabras que hemos obtenido a partir de un dataset con tweets sobre las elecciones presidenciales americanas lo encontramos en la Figura 42.

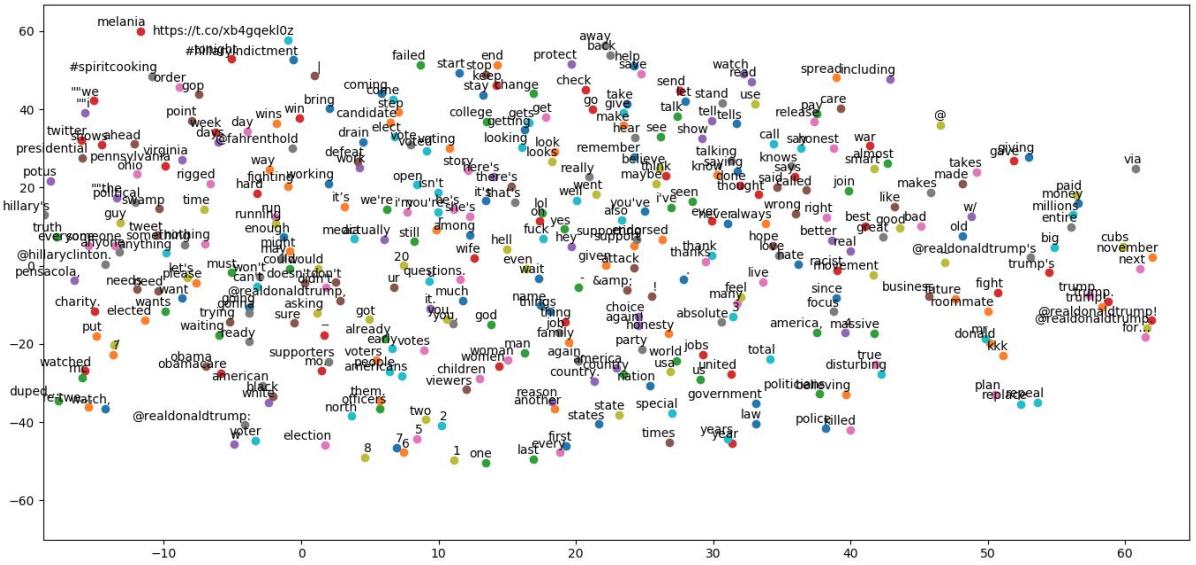


Figura 42: Representación de las relaciones entre palabras según Word Embedding.

Obsérvese que se agrupan términos contrapuestos, como love/hate o good/bad. Esto no es un problema porque tweets como *“Trump attitude is bad*” y *“Trump attitude is good”*, verían incrementada su similaridad, lo que es deseable porque, en efecto, se trata del mismo tema.

11. Conclusions

11.1. Contributions

This work has presented a technique for analyzing the Twitter messages about any particular event, extracting interesting information. The application takes as input the tweets associated to the event, and aggregate them into topics of similar content, allowing the user to have a global vision of the key moments of the events.

The proposed framework, uses a notion of similarity based in the model of representation of documents in a vectorial space and the numerical model TF-IDF. However, this function could be replaced by any other similarity function verifying certain conditions, such that producing a value in the interval $[0,1]$ for each pair of tweets.

Our work tries to combine the two alternatives proposed usually in the literature: temporal proximity and textual similarity. Each of these proposals present some flaw. The temporal proximity, based on frequency analysis, locates the key moments of the event, but it doesn't extract the messages emitted as reactions to these key moments. On the other hand, the textual similarity can aggregate tweets occurring on different times but with similar contents, thus mixing up different news.

Our framework integrates the temporal proximity by dividing the initial dataset into windows of chronologically consecutive tweets. The second proposal is then applied to these windows, looking for textual similarity, but now inside of a temporally limited interval. In this way to avoid to identify as part of the same topic tweets with similar content that occurred in different moments, while at the same time we distinguish among the several topics associated to every key moment. In this way we fulfill the goals **O1** and **O2**.

An important conclusion of this work is that the key moments of a real event must not be confused with their reflection in the social media. A key moment usually generates several topics, one of them can reflect directly of the key event. However, limit the search of topics to those corresponding to the key moments implies to reduce, and thus to miss, the richness of topics associated to the event. Usually the key moments can be found at the newspapers, and the precise time of their occurrence can be obtained directly using a frequency analysis. However, the opinions that these key moments generate can be very interesting and constitute the main asset of the social media.

For instance, we have seen that one of the topics associated with one goal of Real Madrid during a match against PSG, indicates that, under the users point of view, the PSG trainer should quit his job. This reactions of the fans can be of great importance for the online community manager of the club, since this topic points out to the trainer as the responsible of the goal.

Our approach generates a great amount of topics. Instead of filtering these topics, discarding some of them and therefore missing some maybe important information, we have chose to aggregate the topics during the presentation to the user. This aggregation can be done either relating thematically close topics, or using temporal aggregation. In this way we achieve the goal **O3**.

It might seem that actually, instead of deciding at the beginning if we use a temporal or a thematic criterium, we simple delay this decision until the moment of the presentation to the user, and that in fact this is a very small contribution with respect to other, previous, works. The main difference is that the final aggregation considers topics already established, not tweets. That is, in our proposal the tweets are aggregated into topics without setting any *a priori* criterium, following the algorithm described on the chapter Detección de temas, thus avoiding the cons associated to the adoption of either the thematic or temporal approach.

The work presents as two main contributions two new clustering algorithms, the first one devoted to finding topics and the second one oriented to the detections of “moments’’,or aggregations of topics by temporal proximity. In particular:

- *Algorithm for the detection of topics*. The work presents a new algorithm that first of all generates a set of topics from a random sample in a previously established window of tweets. The remainder of the tweets are then incorporated to the already established topics. Also, the algorithm includes phases for compacting, removing irrelevant topics, and a process of fusion of topics inside of a windows as well as between consecutive windows. The result is an efficient technique from the point of view of the computation time.
- *Clustering algorithm to aggregating topics according to temporal proximity*. We describe a technique of temporal clustering that allows aggregating topics that occur in the same temporal interval, which we denominate moments. The technique uses the idea of density of topics in different temporal intervals, and proposes an algorithm that detects these moments, decreasing progressively the degree of required density.

Other contributions of this work are:

- The section devoted to thematic clustering we incorporate with respect to other proposals the automatic detection of the optimum number of clusters.
- The application of logarithms to the result of the cosine of the TF-IDF vectors, to achieve a smoother gradation of similarity. This idea arose after observing the probability distribution of the cosine of the TF-IDF, which scores as very similar to tweets almost identical, and with scores near 0 to tweets only slightly different. In this way to broader topics that contain a greater number of tweets without decreasing the thematic integrity.

11.2. Future work

During the development of the work, different lines of possible future work have arisen. Here we mention some of them.

Semi-automatic parameter adjustment

The proposal depends on a great amount of parameters which can be found at Anexo A. In the supervised machine learning methods the tuning of this parameters can be achieved automatically dividing the initial dataset in training, validation and test set. This is not so straightforward in unsupervised methods. Some parameters, like optimum k in k-means have been determined, but the definitions of methods that yield the optimal value, or at least an interval for the rest of the parameters remains future work.

Extending the notion of similarity

Although the experiments produced good results regarding the detection of topics, an open question is considering other features of the tweets in order to extend and improve our notion of similarity beyond the textual similarity used in this work.

Images, in particular, play a very important role in Twitter and in general in all the social media. However they are usually discarded. To consider images in the similarity two steps must be considered. The first, and possibly the most complicated, step, is to define suitable definitions of similarity for images, which can include scaling and parcelling of both images, comparing features like the proportion of images of every color, to those techniques based on detecting contours [46]. The second step is then to redefine the concept of similarity between two tweets, where now each tweet is represented by two components $x = (x_1,x_2)$, with x_1 the text and x_2 the image. The definition of the similarity s, will depend both on the concept of similarity s_1 between tweets and the concept of similarity s_2 between images, taking into account that depending on the tweet, some of these two components might not be present. That, is, the tweet can consists only of text, only of image, or contain the two components. We can imagine that in these cases we use a special value *None* to indicate the absence of some component. Thus a vector like *("Gol de ...",None)* would indicate that the tweet has text but not image. A definition of similarity that can cope with these situations is the *coefficient of general similarity* [47], defined as:

$$s(x,y) = \frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k)\, s_k(x_k, y_k)$$

where x,y represent, in our setting, the vectors with the two components (text and images) of the two tweets. Thus, d=2. The function $w(x_k,y_k)$ takes the value 0 or 1 depending on weather the comparison of the values makes sense (in our case *w(a,b) = 0* if either *a* or *b* are *None*, and *1* otherwise), and the functions s_k represent the similarity function for each component,

extended to deal with the value None returning any arbitrary value in this case, since it will be multiplied by 0. This similarity can also be extended easily to define a suitable definition of distance, as explained in [48].

Improving efficiency

Using MongoDB makes our proposal scalable in terms of space. However, the time required for large datasets can become excessive.
To overcome this limitation, from our idea of using windows of tweets rise in a natural way the idea of processing each window in parallel, distributing the data among several computers, thus increasing heavily the performance in terms of time of the algorithm of detecting topics.

Relationships among topics

Another future line of research is the study of the possible relationships of casualty among the different topics found. A study in this line, similar to that of the system ETree [21] , will show the evolution of each topic. Also, going one step beyond, we could compare the evolution of different topics in order to detect if topics with no apparent relation are debated in a similar pattern by the Twitter users.

Derived words and synonyms

The disambiguation of words, often carried out by a process of steaming, converts derived words into their semantic root, which is often useful for detecting similarity. However, this technique is not very successful when applied to tweets, due to their short length and lack of context. Our last line of future research proposes to improve this method by using Word Embedding [49]. Using the similarity of words (and taking into account the context) of this model, we can implement a version of the similarity that solves the problem of derived words and synonyms. Recent research [50] employ this idea for the detecting topics in Twitter. However, in our opinion, these works can be improved if the considered also the temporal proximity as a relevant factor when deciding if two tweets belong to the same topic or not.

Our initial experiments using deep learning algorithms using the library TensorFlow [51], show that this technique indeed can be employed to detect words that occur in the same context. Figure 42 depicts an example of the visual representation of the contextual relation of words in a dataset corresponding to the last american presidential elections

Observe that the technique represents opposite terms as very close, such as love/hate or good/bad. This is not a problem, because tweets such as *"Trump attitude is bad*" and *"Trump attitude is good"*, will have a higher similarity in the new setting, which is desirable since, indeed, both tweets correspond to the same topic.

Bibliografía

[1] “Digital in 2018: World’s internet users pass the 4 billion mark” , We are social (30/01/2018). Online: https://wearesocial.com/blog/2018/01/global-digital-report-2018 (Accedido 26/03/2018)

[2] About Twitter https://about.twitter.com/en_gb.html

[3] Twitter https://twitter.com/

[4] “Tweeting Made Easier”, Blog de Twitter (07/11/2018). Online: https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html (Accedido 25/03/2018)

[5] “Q4 2017: Selected Company Metric and Financials”, Twitter (08/02/2018). Online: http://files.shareholder.com/downloads/AMDA-2F526X/5978502153x0x970886/28A12845-4E0D-4F74-B82D-077241BCD7FE/Q4_2017_Selected_Company_Metrics_and_Financials.pdf (Accedido 26/03/2018)

[6] “The 2014 #YearOnTwitter”, Blog de Twitter (10/12/2014). Online: https://blog.twitter.com/official/en_us/a/2014/the-2014-yearontwitter.html (Accedido: 27/03/2018)

[7] “News use across Social Media Platforms 2017”, Pew Research Center (07/09/2017). Online: http://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017/ (Accedido 27/03/2018)

[8] Business Twitter. Online: https://business.twitter.com/en/video-on-twitter.html (Accedido 25/03/2018)

[9] Mathioudakis, M., & Koudas, N. (2010). TwitterMonitor: Trend Detection over the Twitter Stream. *Sigmod*, 5–7. https://doi.org/10.1145/1807167.1807306

[10] Ritter, A., Mausam, Etzioni, O., & Clark, S. (2012). Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* - KDD ’12 (p. 1104). https://doi.org/10.1145/2339530.2339704

[11] Phelan, O., McCarthy, K., & Smyth, B. (2009). Using twitter to recommend real-time topical news. *Proceedings of the Third ACM Conference on Recommender Systems - RecSys* ’09, 385. https://doi.org/10.1145/1639714.1639794

[12] Internet live stats. Online: http://www.internetlivestats.com/one-second/#tweets-band (Accedido 30/05/2018).

[13] Vicente, M., Batista, F., & Carvalho, J. P. (2015). Twitter gender classification using user unstructured information. In *IEEE International Conference on Fuzzy Systems*. https://doi.org/10.1109/FUZZ-IEEE.2015.7338102

[14] Cha, M., Haddai, H., Benevenuto, F., & Gummadi, K. P. (2010). Measuring User Influence in Twitter : The Million Follower Fallacy. *International AAAI Conference on Weblogs and Social Media*, 10–17. https://doi.org/10.1.1.167.192

[15] Larsson, A. O., & Moe, H. (2012). Studying political microblogging: Twitter users in the 2010 Swedish election campaign. *New Media and Society*. https://doi.org/10.1177/1461444811422894

[16] Allan, J., Carbonell, J., Doddington, G., Yamron, J., & Yang, Y. (1998). Topic Detection and Tracking Pilot Study: Final Report. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 194–218. https://doi.org/10.1007/BF02985802

[17] Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a Social Network or a News Media?

[18] Atefeh, F., & Khreich, W. (2015). A survey of techniques for event detection in Twitter. *Computational Intelligence*, 31(1).

[19] Phuvipadawat, S., & Murata, T. (2010). Breaking news detection and tracking in Twitter. *In Proceedings - 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IAT 2010. https://doi.org/10.1109/WI-IAT.2010.205

[20] Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S., & Miller, R. C. (2011). TwitInfo: Aggregating and Visualizing Microblogs for Event Exploration. *CHI '11: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*, 227–236. https://doi.org/10.1145/1978942.1978975

[21] Gu, H., Xie, X., Lv, Q., Ruan, Y., & Shang, L. (2011). ETree: Effective and efficient event modeling for real-time online social media networks. *In Proceedings - 2011 IEEE/WIC/ACM International Conference on Web Intelligence*, WI 2011. https://doi.org/10.1109/WI-IAT.2011.126

[22] Becker, H., Naaman, M., & Gravano, L. (2010). Learning similarity metrics for event identification in social media. *In Proceedings of the third ACM international conference on Web search and data mining - WSDM '1*0 (p. 291). https://doi.org/10.1145/1718487.1718524

[23] Sechelea, A., Huu, T. Do, Zimos, E., & Deligiannis, N. (2016). Twitter data clustering and visualization. *2016 23rd International Conference on Telecommunications (ICT)*, 1–5. https://doi.org/10.1109/ICT.2016.7500379

[24] Alnajran, N., Crockett, K., McLean, D., & Latham, A. (2017). Cluster Analysis of Twitter Data: A Review of Algorithms. *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, 239–249. https://doi.org/10.5220/0006202802390249

[25] Ifrim, G., Shi, B., & Brigadir, I. (2014). Event detection in Twitter using aggressive filtering and hierarchical tweet clustering. *In CEUR Workshop Proceedings.*

[26] Godfrey, D., Johns, C., Meyer, C., Race, S., & Sadek, C. (2014). A Case Study in Text Mining: Interpreting Twitter Data From World Cup Tweets. Recuperado de http://arxiv.org/abs/1408.5427

[27] O'Connor, B., Krieger, M., & Ahn, D. (2010). TweetMotif : Exploratory search and topic summarization for Twitter. *4th International AAAI Conference on Weblogs and Social Media,* (Mayo), 2–3. https://doi.org/10.1145/1390334.1390389

[28] Tweepy http://www.tweepy.org/

[29] MongoDB https://www.mongodb.com/

[30] Anaconda https://www.anaconda.com/

[31] Pymongo https://api.mongodb.com/python/current/

[32] Naumann F. (2013), "Similarity Measures", *Information Systems*. Instituto Hasso-Plattner.

[33] Ganti, V. and Das Sarma, A. ed., (2013). Similarity Functions. In: *Data Cleaning: A Practical Perspective,* 1st ed. Morgan & Claypool.

[34] Salton, G., Wong A., and Yang, C. S. (1975), A Vector Space Model for Automatic Indexing. *Communications of the ACM*, vol. 18, nr. 11, pages 613–620.

[35] Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. Philadelphia, PA. *American Association for Artificial Intelligence retrieval.*

[36] Manning C.D., Raghavan P., & Schütze H.,(2008). Scoring, term weighting and the vector space model. In: *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[37] Huang, A. (2008). Similarity measures for text document clustering. *Proceedings of the Sixth New Zealand*, (Abril), 49–56. Recuperado de http://nzcsrsc08.canterbury.ac.nz/site/proceedings/Individual_Papers/pg049_Similarity_Measures_for_Text_Document_Clustering.pdf

[38] Tata, S., & Patel, J. M. (2007). Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM SIGMOD Record*, 36(2), 7–12. https://doi.org/10.1145/1328854.1328855

[39] Peter J. Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*. 20: 53–65. doi:10.1016/0377-0427(87)90125-7

[40] T. Caliński & J Harabasz (2007) A dendrite method for cluster analysis, *Communications in Statistics*, 3:1, 1-27, DOI: 10.1080/03610927408827101

[41] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3), 259-284.

[42] Truncated SVD, Scikit Learn v0.19.1. Online: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD (Accedido 24/04/2018)

[43] Susan T. Dumais (2005). "Latent Semantic Analysis". *Annual Review of Information Science and Technology*. **38**: 188–230. doi:10.1002/aris.1440380105

[44] Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov), 2579-2605.

[45] PSG vs RMA 06/03/2018. Marca, Online: http://www.marca.com/eventos/marcador/futbol/2017_18/champions/octavos/vuelta/psg_rma/asilovivimos.html (Accedido 13/06/2018)

[46] Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, *13*(1), 146-166.

[47] Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, 857-871.

[48] Wishart, D. (2003). K-means clustering with outlier detection, mixed variables and missing values. In *Exploratory Data Analysis in Empirical Research* (pp. 216-226). Springer, Berlin, Heidelberg.

[49] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 1–12. https://doi.org/10.1162/153244303322533223

[50] Dai, X., Bikdash, M., & Meyer, B. (2017). From social media to public health surveillance: Word embedding based clustering method for twitter classification. *SoutheastCon 2017*, 1–7. https://doi.org/10.1109/SECON.2017.7925400

[51] TensorFlow, https://www.tensorflow.org/

ANEXO A: Parámetros de configuración

El archivo de configuración (Figura 43) necesario para la fase de detección de temas contiene los siguientes parámetros, no pudiendo ejecutarse de manera correcta sin contar con todos ellos definidos. Este fichero debe ser almacenado con el nombre *config.json* y encontrarse en el directorio que requiere como parámetro de entrada la llamada al script *tweet-analysis.py.*

1. "*lang*": idioma mayoritario en el que se encuentran los tweets a analizar, empleado para eliminar las stopwords.
2. "*tweets_ventana*": número de tweets que formarán parte de una misma ventana.
3. "*muestra_ventana*": tamaño de la muestra aleatoria de tweets de la ventana tomados para formación de los temas iniciales.
4. "*num_muestra_tema*": número de tweets aleatorios de cada tema que serán comparados para comprobar si un tweet es similar a un tema.
5. "*min_tweets_similares*": número mínimo de tweets que deben ser similares al tweet que se está comparando para formar parte del tema.
6. "*umbral_inicial*": umbral mínimo para considerar un tweet como similar a un tema. Empleado en la fase inicial de formación de temas, al agrupar los tweets de la muestra aleatoria de la ventana.
7. "*filtrar_unitarios*":número de tweets tras los cuales se realiza un filtrado de tweets que no han conseguido formar un tema en la fase de formación de temas iniciales.
8. "*umbral_restantes*": Umbral mínimo para considerar un tweet como similar a un tema usado en la fase en la que se añaden los tweets restantes a los temas iniciales.
9. "*tweets_nucleo*": número de tweets que componen el núcleo de un tema.
10. "*porcentaje_fusion_interna*": porcentaje de tweets que debe contener la intersección de dos temas de una misma ventana para fusionarse.
11. "*umbral_fusion_interna*": umbral mínimo de similaridad para la fusión de temas dentro de una misma ventana.
12. "*umbral_fusion_externa*": umbral mínimo de similaridad entre núcleo. Empleado para la fusión de temas entre distintas ventanas.

```
{   "lang" : "spanish",
    "tweets_ventana" : 10000,
    "muestra_ventana": 3000,
    "num_muestra_tema" : 5,
    "min_tweets_similares": 4,
    "umbral_inicial" : 0.8,
    "filtrar_unitarios": 500,
    "umbral_restantes" : 0.7,
    "tweets_nucleo": 5,
    "porcentaje_fusion_interna": 0.5,
    "umbral_fusion_interna": 0.7,
    "umbral_fusion_externa": 0.8
}
```

Figura 43: Formato del fichero *config.json* necesario por el script *tweet-analysis.py*.