UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

Proyecto Fin de Carrera

Departamento de Telemática

Ingeniería de Telecomunicación

TÉCNICAS DE CLUSTERING APLICADAS AL ANÁLISIS DE TRENDING TOPICS EN CONJUNTO DE TWEETS

Autor: Adrián García Diéguez

Tutor: Julio Villena Román

Leganés, Julio 2014

AGRADECIMIENTOS

Con este proyecto concluye una etapa muy buena de mi vida, la universitaria.

Parece que fue hace dos días cuando había acabado la selectividad y estaba eligiendo qué carreras me gustaban más, y en donde quería estudiarlas.

Inconscientemente, el primer día de clase supuso el comienzo de un nuevo periodo que sin darme cuenta, ya ha pasado. Al principio, la carrera se veía como un camino muy largo, casi sin fin, en el que nos encontraríamos todo tipo de dificultades, de las que en ese momento, no éramos muy conscientes. Año tras año, iba avanzando hasta el punto que a día de hoy, finalmente he terminado. Parece mentira que el tiempo pase tan rápido.

En todos estos años, he de agradecer a varias personas el apoyo incondicional que me han ofrecido.

Por un lado a toda mi familia y a Anita, que sin presionarme, han sabido confiar en mí, brindándome todo su apoyo tanto en los buenos como en los malos momentos.

Por otro lado, a mis compañeros de universidad. No podía finalizar mis agradecimientos sin mencionar al team “Machachas” qué tantos momentos hemos compartido. Aquellas largas jornadas en la biblioteca de las que tanto nos quejábamos, ahora incluso se echan de menos. Sin la ayuda que nos hemos ofrecido, esto habría sido mucho más difícil y aburrido.

A todos ellos sin duda, muchas gracias.

"No es más feliz el que más tiene,

sino el que menos necesita"

RESUMEN

Dado que los servicios de las redes sociales se expanden geográficamente desde hace varios años, cada vez en mayor medida, y penetran cada vez más en diversos segmentos de la población, el valor de la información que se genera en este tipo de plataformas en línea aumenta drásticamente. La comunicación e interacción en las redes sociales a menudo reflejan los acontecimientos y la dinámica de la vida real, por lo que el continuo aumento de usuario provoca que las redes sociales se conviertan en precisos sensores de los acontecimientos del mundo real.

En este contexto, el tratamiento y procesamiento de toda esa información, capta un gran interés. Este hecho es el que ha inspirado el presente proyecto, centrado en la detección de temas emergentes, llamados Trending Topics en Twitter.

Para ello se proponen y evalúan distintos algoritmos y estrategias de detección que nos permiten descubrir temas más generales o más específicos, según lo que más interese. Esto nos dará la capacidad de conocer los temas sobre los que se habla y saber cuáles de ellos y en qué momento captan un mayor interés por parte de los usuarios, lo que nos permitirá simular el posible comportamiento de Twitter en la tarea de la detección de Trending Topics.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1 - INTRODUCCIÓN

En los tiempos modernos de la era de la información, la omnipresencia de las redes sociales en línea ha visto una expansión sin precedentes en los últimos años, elevando las tasas de actividad hasta niveles que hasta hace años, eran insospechables. Millones de usuarios participan a diario en redes sociales y foros en línea, en donde el consumo y la difusión de información es clara y evidente.

Toda esa flamante información se reduce a un único factor común; el lenguaje. El lenguaje, es el instrumento que los seres humanos utilizamos para comunicar el conocimiento. Desde los inicios de la humanidad, el lenguaje ha sido el medio que ha permitido la comunicación entre los distintos individuos, lo que ha posibilitado el desarrollo de las bases de la sociedad que actualmente conocemos. Sin él, no se podría haber desarrollado ninguno de los avances que hoy dominamos en cualquiera de los campos existentes. Por lo tanto, el estudio científico del lenguaje, la lingüística, adquiere un papel muy importante.

Ahora bien, el origen de Internet en Estados Unidos en el año 1960 [1], fue el comienzo de un punto de inflexión en la aparición de nuevas formas de comunicación. Una nueva época surgiría de aquel acontecimiento. Varias décadas después de su aparición, en el año 1995, Internet había logrado convertirse en una herramienta prácticamente masificada. La aparición de la denominada Web 2.0 [2], sistema con una clara orientación social, permitió que surgieran las primeras redes sociales. Éstas brindaban la posibilidad de que personas de todo el mundo pudieran recuperar o continuar manteniendo contacto con sus antiguos amigos, en medio de un mundo totalmente globalizado. Este suceso fue el desencadenante de la futura aparición de numerosas y diversas redes sociales.

Actualmente el abanico de redes sociales del que disponemos, es inmenso. Twitter es una de ellas, una de las más famosas redes sociales a día de hoy [3]. Se trata del principal servicio de microblogging con más de 500 millones de usuarios registrados en todo el mundo. Esta gran cantidad de usuarios provoca que el flujo de datos y de información en tiempo real sea inmenso. A pesar de que una parte de los contenidos de Twitter no están relacionados con ningún caso real en concreto, la abundancia de mensajes sobre eventos informativos y la prevalencia de los medios sociales, ha impulsado el desarrollo de la investigación orientada a sistemas cuyo

objetivo sea descubrir los aspectos más importantes de la actividad en los medios de comunicación social.

Concretamente, el objeto de estudio se centra en los llamados "*Trending Topics*" (TT), término que hace referencia a los temas tendencia o a los temas que más se hablan en un determinado momento. Tal es la magnitud que ha adoptado este concepto, que incluso ha traspasado las fronteras del propio mundo de Twitter, para pasar a ser una frase de uso común en los medios de comunicación, siendo noticia aquello que se vuelve tema del momento en esta red social. Esto ha propiciado, por ejemplo, que se creen secciones especializadas en periódicos online que se basen en aquello que es tendencia en Twitter.

En este punto entra en juego la ingeniería lingüística. Tal volumen de información ofrece importantes oportunidades y desafíos para la investigación en el Procesamiento del Lenguaje Natural (PLN) y aprendizaje automático.

Uno de los primeros retos se centra en la detección de dichos Trending Topics. Sin embargo, la identificación de los eventos en tiempo real en Twitter no es una tarea trivial, debido a la heterogeneidad y la inmensa escala de los datos Hay que tener en cuenta que los usuarios de Twitter envían mensajes con una temática muy amplia. Además como desafío adicional, y por el propio formato de la mensajería en Twitter, los mensajes se caracterizan por contener poca información textual y encima, de baja calidad. Por este motivo, el manejo y procesado directo de los datos en brutos es algo inviable. La necesidad de un tratamiento previo adquiere un papel fundamental, condicionando directamente la calidad final de los resultados. Ante esta situación, el PLN es el encargado de proporcionar diversos mecanismos que faciliten el procesamiento y tratamiento de los datos, y por consiguiente, ayuden en la tarea de detección.

1.1 MOTIVACIONES Y OBJETIVOS

La detección de TT es una herramienta de un alto valor para numerosas profesiones. Periodistas, analistas, profesionales del marketing online y empresas de seguimiento de opinión, entre otros, se ven involucrados en el interés de conocer cuáles son los temas más comentados en determinados momentos, ya que las tendencias normalmente apuntan a temas que captan la atención del público.

Por ejemplo, de cara al marketing de una empresa, puede resultar muy interesante saber las opiniones de los usuarios reflejadas en los tweets escritos en el canal de la compañía. Esta información sería de gran utilidad para mejorar los servicios ofrecidos por la empresa, lo que acrecentaría el contento y bienestar del usuario. Esto trasladado a grandes multinacionales, supone un campo de aplicación e investigación muy interesante y con mucho futuro.

Para ello el procesamiento y análisis de los tweets es esencial, permitiendo detectar los TT del momento, y por lo tanto, detectar el pulso de opiniones de los usuarios en las redes sociales.

Por este motivo el objetivo principal del proyecto es la detección de TT en Twitter. Para ello, se presentan y evalúan simultáneamente dos mecanismos distintos.

Por un lado, se proponen e implementan un conjunto de algoritmos mediante los que obtenemos una serie de puntuaciones que nos ayudan en la tarea de determinar el carácter TT de un término.

Por otro lado, se plantean diversas estrategias que permiten abarcar distintas posibilidades en cuanto a tipos de detección se refieren.

La aplicación de una u otra estrategia, junto con la ayuda de los algoritmos, nos permitirá contemplar distintos escenarios y por lo tanto, detectar temas más generales o más específicos según nuestros intereses. Esto implica que además de detectar TT, podremos averiguar con más detalle los temas sobre los que se hablan.

Con todo lo mencionado, seremos capaces de evaluar qué mecanismos son los que mejores resultados ofrecen y por lo tanto, deducir cuáles son los que más se asemejan al uso real que hace Twitter en la detección de TT, que a priori, no tienen por qué coincidir con los propuestos en el trabajo.

1.2 ESTRUCTURA DE LA MEMORIA

La memoria se divide en una serie de capítulos que facilitan la comprensión de la estructura que sigue el proyecto.

En el primer capítulo se ha realizado una breve introducción al fundamento y origen de las redes sociales, presentando las características y retos que surgen en Twitter.

Además se han presentado las motivaciones y objetivos del proyecto, junto con la presente descripción de la estructura.

En el segundo capítulo se describen las principales características y prestaciones de Twitter, se presenta el PLN y los distintos tipos de sistemas que engloba y se abordan una serie de algoritmos de detección de TT clasificados según distintos modelos.

En el tercer capítulo se presenta y explica, tanto a un alto nivel como en mayor detalle, el sistema diseñado para la detección de TT.

En el cuarto capítulo se evalúan las distintas pruebas realizadas aplicando los distintos algoritmos y estrategias planteadas.

En el quinto capítulo se exponen las conclusiones según los resultados obtenidos en el capítulo anterior y se explica el posible trabajo futuro a realizar para completar el sistema propuesto, y por lo tanto, los resultados obtenidos.

Por último, se detalla la bibliografía utilizada para desarrollar y apoyar cada una de las partes del proyecto.

CAPITULO 2 - ESTADO DEL ARTE

En este capítulo realizaremos una introducción teórica de todos los conceptos, herramientas, tecnologías y campos de investigación relacionados con la detección de Trending Topics en Twitter.

Por un lado, en primer lugar, presentaremos la red social Twitter. Tras una breve reseña histórica, se abordará el formato de la mensajería, las prestaciones que ofrece y qué tipo de terminología concreta se usa, destacando en todo momento los conceptos más importantes relacionados con nuestra aplicación. Todo ello proporcionará una visión global de la red social y de los elementos fundamentales de la misma.

Por otro lado, presentaremos la disciplina del procesamiento del lenguaje natural. Explicaremos brevemente qué es, en qué consiste, cuáles son sus principales campos de aplicación y qué tipos de sistemas engloba.

Por último abordaremos todo lo relacionado con la detección de TT. Explicaremos el proceso general a seguir en la detección para a continuación describir un conjunto de algoritmos de detección clasificados según el tipo de procesamiento que realizan.

2.1 TWITTER

En este apartado se va a presentar la red Social Twitter, principal servicio de microblogging en Internet. Comenzando por su origen y evolución, introduciremos su característico formato y qué prestaciones nos ofrece. Profundizaremos en los denominados 'Hashtags' y explicaremos cómo estos se relacionan con los Trending Topics, término objeto de nuestro estudio. Se analizarán sus principales características y cómo éstas condicionan su propia detección.

2.1.1 ORIGEN Y EVOLUCIÓN DE TWITTER

Los primeros orígenes de Twitter distan mucho respecto a la percepción actual de la tan famosa red social. La actual red de microblogging no nació con la esencia que todos conocemos y percibimos ahora. Twitter fue un proyecto al que se le fue dando forma con el paso del tiempo. Sus orígenes se remontan al año 2005 [4], en donde solamente era un proyecto menor dentro de la empresa estadounidense Odeo, una

start-up de San Francisco, fundada por Noah Glass y Evan Williams, en la que se trabajaba en el desarrollo de aplicaciones informáticas.

Por aquel entonces la empresa estaba trabajando en un servidor podcast, una idea innovadora que pretendía revolucionar el panorama de la escucha de contenidos de audio en la red, reinventado el concepto de radio online. Sin embargo el auge que sufría Apple con su plataforma iTunes eclipsó por completo el lanzamiento de ese nuevo proyecto. Ante tal fracaso, Odeo decidió invertir todas sus fuerzas en intentar volver a crear algo nuevo. De esta iniciativa y con la idea de crear una red social en la que se pudiesen compartir experiencias con grupos de amigos, en el año 2006 surgió Twitter [5]. Concretamente lanzaron la plataforma con el nombre de "Twttr", un servidor de microblogs que nació con el fin de superar a los famosos SMS e incluso al email, generando así un nuevo modelo de interacción a través de la red [6].

Su lanzamiento coincidió con el clímax de agitación social en Estados Unidos, ambiente en el que la irrupción de una nueva herramienta de comunicación social podría cuajar. A pesar de que el número de seguidores que consiguió no fue el esperado, la cantidad de mensajes que estos enviaban mostraban el potencial del que disponía. En sus inicios consiguió generar 100.000 mensajes mensuales gracias a sus 100 seguidores, lo que mostraba indicios en el comportamiento de uso en los usuarios [4]. Hay que destacar que al principio Twitter no tenía ni hashtags ni retuits (términos de los que hablaremos más adelante). Fueron los propios usuarios quienes inventaron y dieron lugar a estos nuevos, y actualmente conocidos, conceptos, tras probar la red de microblogging.

Fue en el Festival tecnológico SXSW de Estados unidos, en donde se reunían los profesionales de la tecnología, donde se produjo un punto de inflexión en el nacimiento y la evolución de Twitter. Allí, y de manera casual, Twitter se convirtió en la aplicación que mucha gente utilizaba para conocer, por ejemplo, donde se reunía la gente para celebrar las mejores fiestas. De esta forma Twitter empezó a expandirse y comenzó ser utilizada muy frecuentemente por personas fascinadas por la tecnología e informática. Fue entonces, 3 años después de su nacimiento, cuando Twitter comenzó verdaderamente a tener éxito.

A partir de entonces, el ritmo de crecimiento de Twitter ha sido vertiginoso llegando a cotizar en bolsa a finales del año 2013. Desde su creación, Twitter ha crecido hasta el punto en que se ha convertido en la tercera red social más usada en todo el mundo,

por detrás de Facebook y Youtube [3]. Actualmente Twitter cuenta con más de 500 millones de usuarios en el mundo registrados en 2013, de los cuales, más de la mitad son activos [7]. En España ya se alcanza la cifra de 5.7 millones de usuarios.

2.1.2 FORMATO Y PRESTACIONES

La principal característica y el principal atractivo de Twitter se halla en el formato de sus mensajes. Twitter permite mensajes con un máximo de 140 caracteres, lo que obliga a estructurar bien su contenido, al mismo tiempo que permite a los usuarios identificar de un solo vistazo el interés del mismo. A estos mensajes se les llama "Tweets" (micro-mensajes).

Cualquier persona registrada en Twitter puede disfrutar de su principal funcionalidad: el envío y recepción de tweets. Estos mensajes podrán serán vistos por tus "followers" (seguidores), de la misma forma que tú podrás ver los mensajes de tus "following" (personas a las que sigues).

Las diversas prestaciones que Twitter ofrece en el lenguaje de sus mensajes, provoca que, para comprender de manera correcta el comportamiento de Twitter, sea necesario explicar los términos y vocablos más característicos e importantes.

Entre los más importantes, podemos destacar las siguientes expresiones:

- "**@**" (Mención), es un código importante en Twitter. Se usa para referirse a los distintos usuarios de la red. Se combina con el nombre de usuario y se introduce en los tweets.
- "**RT**" (abreviatura de "Retweet" o "Retuit" en español), permite enviar un mensaje publicado por otra persona. En ocasiones se inserta RT en los mensajes para indicar a los demás que es un retweet. El "Retweeting" es una actividad muy común en Twitter que refleja la popularidad de los tweets individuales
- "**#**" (Hashtag), símbolo muy importante y utilizado en Twitter que se usa para identificar, destacar y agrupar una palabra clave o tópico específico. Se llama *hashtag* a una palabra o término que va precedida del símbolo #. De esta forma se consigue crear una etiqueta para aquellas palabras, lo que permite poder agruparlas y separarlas de otros temas que incluyen el mismo término, pero que puede que se estén usando con un sentido distinto al que se desea otorgarle. La multitud de usos y ventajas de los *hashtag* en Twitter, además del mencionado

antes, le otorgan una gran importancia en la mensajería de esta red social. Entre sus usos podemos destacar los siguientes [8]:

1. *Ahorro de espacio*

   Al tener limitados los caracteres de los mensajes, los hashtag permiten acotar los mensajes reduciendo un término popular, que puede tener varias palabras, en tan solo unas pocas letras.

2. *Obtención de resultados de búsqueda*

   Como consecuencia del uso de hashtag, los tweets se añaden a una lista de mensajes con hashtag similares. De esta forma, al hacer clic sobre cualquier hashtag, podemos obtener una lista de todos los tweets escritos por otros usuarios que incluyan dicho hashtag. Esto permite a los usuarios obtener resultados rápidamente y bajo un mismo tema.

3. *Popularidad*

   Éste es uno de los usos más conocidos. Según su popularidad, un hashtag puede convertirse en uno de los temas tendencia de Twitter, más conocidos como *Trending Topics* (TT).

Además de lo expuesto, existen otros términos y palabras características del lenguaje de Twitter [9], pero en nuestro caso no influyen en el desarrollo del sistema a diseñar.

2.1.3 LOS TRENDING TOPICS

Los Trending Topics son uno de los términos más característicos y populares de Twitter. Se refieren a palabras o conjunto de palabras clave (tanto con el símbolo del hashtag como sin él) más usadas en un momento dado, haciendo referencia a las distintas tendencias y temas del momento en Twitter.

Suelen ser convenciones de escritura para grandes eventos, pero también pueden ser palabras que en un momento determinado se detectan como muy nombradas. Esto sirve como pista para saber si un tema se ha vuelto inesperadamente popular como para descubrir si ha sucedido algún acontecimiento nuevo y novedoso.

El auge que ha sufrido Twitter en los últimos años ha propiciado una mayor relevancia en TT, convirtiéndolos en un barómetro de popularidad, en donde se considera un gran logro el alcanzar el estado de “trending”. Por este motivo los TT en ocasiones son forzados o impulsados por un grupo de usuarios, marcas o movimientos. Además, y

como complemento, tienen la posibilidad de elegir los Treding Topics del área geográfica que prefieran (mundial, nacional) o incluso personalizar la opción en base a otros parámetros, entre los que pueden quedar especificados los temas del momento en función de las cuentas que sigue cada usuario en la red.

Sin embargo lograr que un tema llegue a ser TT no es una tarea tan sencilla como pueda llegar a parecer.

2.1.3.1 Características de los Trending Topic

En Twitter los TT son los temas más novedosos del momento, los que marcan tendencia, y no los más repetidos. Es decir, además de la cantidad hacen falta una serie de aspectos más complejos. Por este motivo existen varios factores que influyen en la creación de TT en Twitter [10]. Entre ellos podemos destacar los siguientes:

- *Cantidad*

  La cantidad, es un factor importante e indispensable para que un término llegue a ser TT. Por ello es de vital importancia que aparezca en una gran cantidad de tweets. En esta situación, los retweets son primordiales [11]. Cuantos más retweets se conozcan de un tema, mayor cantidad de mensajes contendrán dicha temática, por lo que serán mayores las opciones de que se convierta en Trending Topic. Además esta cantidad debe ir incrementándose, debe adoptar un crecimiento exponencial.
- *Velocidad*

  La velocidad es un factor adicional muy importante también. Hay que tener en cuenta la velocidad de emisión de tweets que contengan dicho término. Esta debe ir incrementándose con el paso del tiempo, es decir, que el intervalo de tweets que no contengan el término en cuestión, cada vez sea menor.
- *Usuarios*

  Además de la cantidad y la rapidez, es necesario que el término o hashtag esté siendo usado por una gran cantidad de usuarios. Un término que esté siendo utilizado en gran cantidad por un número de usuarios muy reducido nunca podrá convertirse en TT.
- *Evolución de los usuarios*

  El número de usuarios que usan dicho término debe ir aumentando, deben irse sumando más usuarios a la conversación. Es decir, se debe incluir el término en un número mayor de Tweets y cada vez desde más cuentas nuevas y distintas.

Para ello es de gran importancia el número de usuarios diferentes que estén usando el término y la velocidad con que éstos aumentan. Todo ello, sin que baje el ritmo de las que ya estaba participando.

- *Permanencia*

  Todos los factores mencionados hasta el momento, no deben sólo mantenerse, sino que deben seguir incrementándose para que el término en cuestión se mantenga como TT.

- *Nivel de base*

  Hay que tener en cuenta el promedio habitual que cada uno de los factores tiene normalmente. Esto permite calcular el nivel base a partir del cual un término puede sufrir o no un aumento de uso, y en consecuencia, una posibilidad de llegar a ser TT. Si la aparición de un término no mantiene ascendente su porcentaje en todos los valores respecto al periodo anterior, no se podrá mantener como TT.

Todos estos factores influyen en la creación y mantenimiento de un TT. La fuerte interconexión entre todos ellos, junto con la necesidad de su cumplimiento, otorga a los TT un carácter fugaz y relativamente instantáneo. Aunque según la noticia esto puede variar, la duración de un TT suele ser muy corta, llegando a ocupar durante escasas decenas de minutos la lista de los top 10 temas del ranking.

2.2 PROCESAMIENTO DEL LENGUAJE NATURAL

Una vez conocido Twitter y los aspectos más importantes que éste engloba, presentaremos en este apartado brevemente la disciplina del Procesamiento del Lenguaje Natural (PLN). El procesamiento del Lenguaje Natural es una subdisciplina de la inteligencia artificial y de la rama ingeniera de la lingüística computacional. Se encarga de producir sistemas informáticos que posibiliten la comunicación hombre – computadora por medio del lenguaje humano o natural, proporcionando así mecanismos de comunicación computacionalmente eficientes. Sin embargo, no sólo hace referencia al reconocimiento de la información expresada en lenguaje humano, sino que también atañe la comprensión de textos escrito en esa misma lengua.

Actualmente, una buena parte del saber humano se encuentra en formato digital en distintos tipos de colecciones de datos, lo que supone grandes volúmenes de información. Es aquí donde entra en juego el PLN, facilitando el tratamiento de toda

esa información. Para ello es necesaria la transformación de todos esos datos a una representación formal, que posteriormente permitirá manipularla, comprenderla y procesarla según sea necesario. Por lo tanto, en el contexto del PLN, los sistemas se caracterizan por realizar un preprocesado de los datos con el objeto de obtener una representación formal que facilite su comprensión y posterior tratamiento. En el siguiente punto se presentar algunos de los sistemas principales del PLN.

2.2.1 APLICACIONES

La continua y creciente integración de la tecnología en todos los ámbitos de la sociedad durante la última década, ha propiciado el uso e incorporación del PLN como respuesta a la problemática inherente en la interacción con las nuevas tecnologías. La aparición de distintos sistemas o aplicaciones en el ámbito del PLN, ha contribuido a una mejora de esta situación. Sistemas de traducción automática, recuperación de información, categorización automática de texto, corrección o revisión lingüística de textos, generación de resúmenes, búsqueda de respuestas y reconocimiento y síntesis de voz, entre otras, conforman los principales campos de desarrollo del PLN.

Atendiendo al tipo de aplicación, existen otros tipos de clasificaciones según diferentes puntos de vista [12]: basadas en textos y en diálogos, basadas en la granularidad del procesado que se hace de la información textual y basadas en el carácter final o no de la tarea. En este segundo grupo se diferencian las aplicaciones a nivel de palabra y a nivel de documento. Aquellas que trabajan a nivel de documento realizan un procesado interno de los documentos, en donde la identificación de las palabras claves es una tarea de gran importancia, ya que categorizan en cierta medida el documento en el que residen.

El uso y aplicación de todas estas aplicaciones y en concreto, de los sistemas de Recuperación de Información, ha propiciado la emersión de diversas tecnologías [13] entre las que podemos destacar la *Minería de Texto* y *la Extracción de Información*.

2.2.2 SISTEMAS PLN

En el apartado anterior, se han abordado las distintas y más importantes aplicaciones del PLN. Sin embargo, los sistemas PLN pueden ser clasificados en diversas categorías atendiendo a la naturaleza de sus métodos. Según este enfoque, se pueden

distinguir los siguientes modelos: simbólicos, estadísticos o empíricos, conexionistas e híbridos [13].

Los modelos simbólicos están basados en el conocimiento, emplean reglas y algoritmos que representan el conocimiento del lenguaje natural. Por otro lado, los estadísticos o empíricos se caracterizan por la búsqueda de distintos modelos estadísticos para el PLN. Los conexionistas, hacen uso de redes neuronales para la representación del conocimiento lingüístico. A estos dos últimos modelos, se les conoce como modelos matemáticos, debido a su fuerte componente estadística y matemática. Por último los modelos híbridos, constituidos por una mezcla de distintos modelos con el fin de potenciar las ventajas de cada uno de ellos frente a problemas de aplicaciones específicos.

En base a esta clasificación, los diversos sistemas o aplicaciones de PLN, en función de su metodología, se pueden dividir en sistemas estadísticos y en sistemas basados en reglas de PLN o conocimiento lingüístico. Los sistemas híbridos, a pesar de ser una combinación de diversos modelos, se caracterizan por un fuerte uso de técnicas de PLN, por lo que se incluyen dentro de los sistemas basados en reglas de PLN. A continuación se abordan las características principales de ambos sistemas.

2.2.2.1 Sistemas Estadísticos

Los sistemas estadísticos se caracterizan por el gran hincapié que hacen en el uso de la matemática, y en concreto, en la estadística. Hacen uso de colecciones de muestras del lenguaje, llamadas corpus, para la creación de modelos estadísticos. Para ello, tratan de inferir conocimiento de los datos a través del manejo de modelos de probabilidad y estadística, con el fin de buscar regularidades textuales significativas. En esta labor, las técnicas básicas tratan de deducir todas las probabilidades medias y condicionadas de aparición de las palabras de un corpus lingüístico, mediante el cálculo y manejo de sus frecuencias de aparición.

Este tipo de sistemas se suele basar en el modelo de bolsa de palabras [14] (del inglés, “bag of words”). Este enfoque, cuyo elemento básico de representación es la palabra, representa un documento mediante un conjunto de términos índice o palabras clave. De este modo, el contenido de un documento se puede representar siguiendo el Modelo de Espacio Vectorial (del inglés, “Vector Space Model”)[15] propuesto por Salton a finales de los años 60. Según él, un documento puede situarse en un espacio

vectorial con tantas dimensiones como términos tenga el vector de representación del documento. Este tipo de representación es una interpretación extrema del principio de composicionalidad [16], que afirma que la semántica de un documento habita únicamente en los términos que lo forman, lo que induce a alegar que si un determinado término aparece en un documento, este trata dicho tema.

Este planteamiento no tiene en cuenta ni las relaciones entre términos ni la polisemia de los mismos, ignorando por completo las ideas expresadas en el texto, lo que resulta insuficiente para una representación completa y adecuada de la semántica del texto. A pesar de ello, la representación sin contenido lingüístico ha tenido gran éxito durante las últimas décadas, como consecuencia de su gran sencillez conceptual, su fácil implementación y su más que aceptable rendimiento en la práctica.

2.2.2.2 Sistemas con Conocimiento Lingüístico

Por el lado contrario, los sistemas basados en reglas de PLN, utilizan el conocimiento lingüístico para llevar a cabo su cometido. Consideran el conocimiento lingüístico como algo innato y que puede caracterizarse mediante principios y procedimientos. Fue Chomsky en los años cincuenta quien propició la aparición de las gramáticas formales, introduciendo su modelo generativo del lenguaje, en contrapartida a los modelos estadísticos, a los que criticaba duramente. El planteamiento del paradigma lógico forma, caracterizado por unidades y reglas entre otros aspectos, cambió la perspectiva, los programas y métodos de investigación en el estudio del lenguaje, elevando la tarea a la categoría de ciencia moderna.

Actualmente, los sistemas basados en reglas de PLN basan su funcionamiento en el diseño de un conjunto de reglas o heurísticas a partir las técnicas lingüísticas utilizadas. El éxito de estos sistemas se basa en la capacidad de simulación de la facultad lingüística como consecuencia del nivel alcanzado en la comprensión de las lenguas naturales. Tarea que durante los últimos años, está evolucionando hacia la mutilingualidad de los sistemas más frecuentes y hacia su implantación en algunos dominios restringidos, permitiendo a los sistemas PLN, por un lado, ser capaces de manejar diversas lenguas y por otro trabajar con documentos de cualquier tipo de dominio.

2.3 DETECCIÓN DE TRENDING TOPICS

La detección y seguimiento de temas tendencia o Trending Topics (TDT, del inglés "Topic Detection and Tracking"), tiene como misión la extracción de los temas emergentes de una secuencia de fuentes de información textual (documentos) y a cuantificar cómo evoluciona su tendencia en el tiempo. En este caso, la detección se centra en mensajes de texto producidos en las posibles plataformas de redes sociales, en concreto, en piezas de texto extraídas de Twitter.

2.3.1 PROCESO GENERAL

La detección de TT lleva implícito un proceso general que se ha de seguir. Según el algoritmo a utilizar, el proceso podrá ser diferente y tendrá ciertas variaciones, pero la estructura general será la misma o muy parecida. Hay que tener en cuenta que la cantidad de información es prácticamente ilimitada y que el contenido de los mensajes generados por el usuario es impredecible y altamente ruidoso. Por este motivo es necesario aplicar una serie de pasos previos a la utilización del algoritmo de detección de TT. El proceso general a seguir se observa en la siguiente figura ([17], [18]).

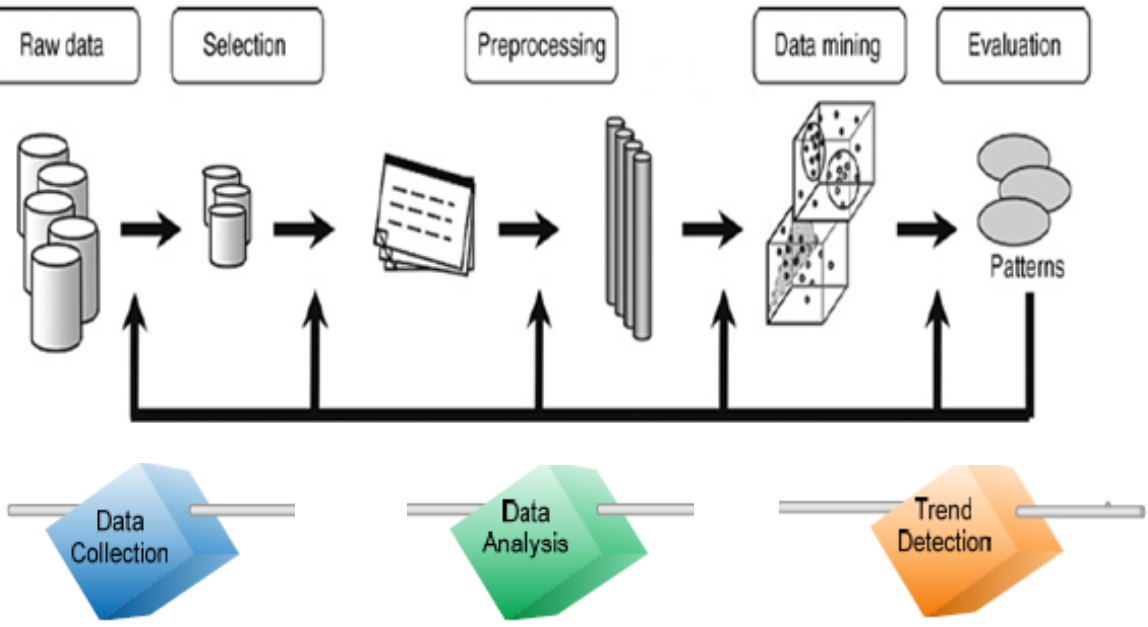

Figura 1. Proceso general en la detección de TT

Como observamos el proceso consta de tres segmentos claramente diferenciados. En primer lugar debemos disponer de los mensajes de los usuarios. Como ya sabemos

la cantidad de información disponible y generada es muy grande, por lo que se deben seleccionar los mensajes que se quieren analizar. Para ello hay disponibles varias herramientas, entre las que podemos destacar el uso de la API de streaming de Twitter, que devuelve aproximadamente el 15% de la actividad total de Twitter.

Una vez obtenido los mensajes y previo a la detección, se realiza una fase de pre procesamiento en la que se elimina el "ruido" de los mensajes recogidos, obteniendo así bolsas de términos más limpios que los mensajes originales.

Como último paso se encuentra la detección, es decir, la utilización del algoritmo de detección de TT. Una vez aplicado, se pueden analizar los resultados y sacar conclusiones sobre la eficiencia de un algoritmo.

2.3.2 PREPROCESADO

También llamado *Normalización* [19], tiene como objetivo la eliminación de cualquier fragmento de texto carente de significado, reduciendo el texto a algún tipo de forma regular que permita su parametrización. Para ello se basa en una serie de operaciones, llamadas *operaciones de texto,* entre las que destacamos el análisis léxico, la eliminación de palabras vacías, la eliminación de signos de puntuación y mayúsculas y la lematización. En los siguientes apartados abordamos cada una de ellas.

2.3.2.1 Tokenización

El proceso de tokenización o análisis léxico, tiene como objetivo la conversión de una secuencia de caracteres en una secuencia de *tokens* o unidades con significado. Básicamente trata de identificar las palabras de un texto. Palabras, que en un principio optan a ser términos clave del documento tratado. En la siguiente figura, se observa un ejemplo.

Texto:	En Twitter, ¡el Procesamiento del Lenguaje Natural es básico!								
Tokens	En	Twitter	el	Procesamiento	del	Lenguaje	Natural	es	básico

Figura 2. Ejemplo de tokenización de una cadena de texto

Sin embargo, el delimitador entre palabras no siempre es el espacio en blanco. Por

este motivo, normalmente se consideran tres tipos de caracteres [20]: caracteres de palabra, interpalabra y especiales. Los caracteres de palabra se refieren a las propias letras y a los números, mientras que los caracteres interpalabra hacen referencia a los espacios en blanco, las comas, los puntos y los signos de puntuación más frecuentes. Por último, los caracteres especiales tratan aquellos símbolos que requieren un procesamiento específico. Distinguimos los guiones (utilizados para continuar una palabra al final de una línea, la conexión de palabras independientes, etc.), los apóstrofos (muy utilizados en el inglés) y las abreviaturas y siglas (que utilizan los puntos con un cometido distinto al de finalizar una oración), entre otros. En esta situación, el uso de una serie de reglas es de vital importancia, ya que ayudan a detectar el caso particular al que nos enfrentamos, permitiendo así un óptimo procesamiento de los datos.

2.3.2.2 Stopwords

Las palabras vacías o palabras de parada (del inglés, "stopwords"), hacen referencia a aquellos términos que por un lado carecen de significado (su contenido semántico es escaso) y por otro suelen ser muy frecuentes, por lo que su aporte al contenido del texto es casi nulo. Como palabras vacías podemos distinguir, entre otras, preposiciones, conjunciones, artículos e incluso en algunos casos, verbos, adjetivos y adverbios ([21], [22]). Por este motivo, el proceso de eliminación de las stopwords es esencial. Ignorar este tipo de términos supone un considerable ahorro de recursos, ya que reducen notablemente la cantidad de términos a procesar, y por consiguiente, reduce el espacio de almacenamiento de las estructuras generadas [23]. Continuando con el ejemplo anterior, en la siguiente figura obtendríamos el siguiente resultado.

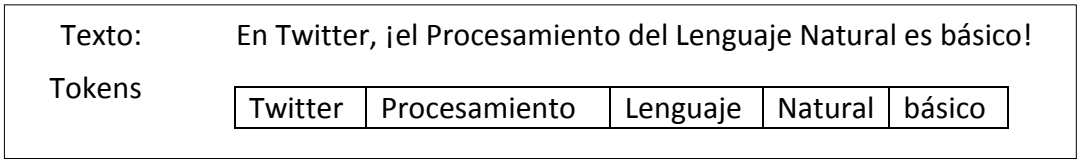

Figura 3. Ejemplo de eliminación de stopwords

Como consecuencia de la eliminación de las palabras vacías, podemos apreciar que el texto se ha visto reducido casi a la mitad de palabras que en un comienzo lo conformaban.

2.3.2.3 Eliminación de Signos Ortográficos

En este paso se trata de homogeneizar el texto. Para ello se eliminan principalmente los signos ortográficos como tildes, diéresis, etc. y se realiza el paso de mayúsculas a minúsculas. De esta forma obtenemos un texto normalizado y uniforme, formado por únicamente términos con significado. Como consecuencia de esta etapa, el ejemplo quedaría tal y como se muestra en la siguiente figura.

Texto:	En Twitter, ¡el Procesamiento del Lenguaje Natural es básico!				
Tokens	twitter	procesamiento	lenguaje	natural	basico

Figura 4. Ejemplo eliminación signos ortográficos

De forma adicional, se eliminan cabeceras, etiquetas, y se controlan parámetros como abreviaturas, acrónimos, etc.

2.3.2.4 Lematización

El proceso de lematización [24] trata de determinar el *lema* o *forma canónica* de cada palabra. Los idiomas provenientes del latín (como el español, italiano, francés, etc.) son lenguas flexivas, es decir, tiene flexión nominal y verbal. Esto quiere decir que las palabras parten de una raíz (lexema) que se junta con terminaciones (morfemas) para indicar rasgos (por ejemplo: "escrib + ía", indicando el pretérito imperfecto del verbo escribir). Según el tipo de terminación usada, se distinguen dos tipos de lematizaciones [25]: la lematización flexiva (se suprimen los plurales, género y terminaciones verbales) y la lematización derivativa (elimina además sufijos derivativos).

Todo este proceso que el ser humano realizamos de forma automática, requiere del uso de un tipo de programas llamados lematizadores, herramientas basadas en una serie de algoritmos, reglas y recursos que tratan de obtener el lemas de las palabras. Para ello utilizan técnicas como el etiquetado POS [26] (del inglés, "Part of Speech") que les permiten conocer la categoría gramatical de las palabras. El proceso de lematización aplicado a nuestro ejemplo, se observa en la siguiente figura.

Texto:	En Twitter, ¡el Procesamiento del Lenguaje Natural es básico!				
Tokens	twitter	procesamiento	lenguaje	natural	básico

Figura 5. Ejemplo de lematización

Algunas de las herramientas de lematización más conocidas son el *Freeling* [27] y el *TreeTagger* [28].

Ahora bien, en idiomas que posean una menor flexión (por ejemplo el inglés), se puede aplicar un proceso similar a la lematización llamado *stemming* [29], cuyo objetivo es la reducción de una palabra a su raíz (del inglés, "stem") mediante la eliminación de sus posibles terminaciones (que a priori contienen la semántica básica del concepto). Para ello se basan en la heurística desechando cualquier tipo de análisis morfológico. Esto implica una serie de inconvenientes y errores [30], por ejemplo que en ocasiones no se distinga adecuadamente el comportamiento de palabras homónimas [31] (por ejemplo "bote" como verbo y "bote" como sustantivo). Por el contrario, una de sus grandes ventajas es la velocidad de procesamiento. Un clásico en los algoritmos de stemming es el algoritmo de Porter [32]. En la siguiente figura se muestra el algoritmo de stemming aplicado a nuestro ejemplo.

Texto:	En Twitter, ¡el Procesamiento del Lenguaje Natural es básico!				
Tokens	twitter	proces	lengua	natural	basic

Figura 6. Ejemplo de stemming

2.3.2.5 N-gramas

Siendo uno de los procedimientos estadísticos más extendidos, el modelo de representación de *n-gramas*, también llamados *modelos del lenguaje* [33], tiene en cuenta las dependencias directas que surgen por la proximidad entre palabras, evitando así la pérdida de información de las relaciones sintácticas entre las mismas, originada por la aceptación de la palabra como unidad básica de significado.

De manera genérica, un n-grama es una secuencia de *n* elementos, palabras o caracteres, extraídos de un texto de forma no necesariamente consecutiva (aunque por regla general suelen ser contiguos). Según el tipo de elementos que formen el n-grama, podremos distinguir diversos modelos [35] (n-gramas de caracteres, de palabra, mixtos, etc.). Para ello, se utiliza una ventana deslizante de longitud fija [34] de *n* elementos que va recorriendo iterativamente el texto, obteniendo así la secuencia de elementos requerida. De esta forma se pueden formar bigramas (2-gramas), trigramas (3-gramas), etc. Considerando n-gramas de palabras, obtendríamos la siguiente figura.

Texto:	Extracción de información y Twitter			
Bigrama:	Extracción de	de información	información y	y Twitter
Trigrama:	Extracción de información	de información y	información y Twitter	

Figura 7. Ejemplo de cálculo de bigramas y trigramas

Podemos observar cómo con el uso de tigramas obtenemos una coocurrencia de palabras de gran significado ("extracción de información"). Por otro lado podemos observar que las *stopwords* no han sido eliminadas para la formación de los n-gramas. Según diversos estudios [36], la *tokenización* de un texto en n-gramas no necesita obligatoriamente de ningún tipo de preprocesamiento tal como el *stemming, lematización, etc.* El simple hecho de tokenizar en n-gramas de caracteres, por ejemplo, permite hacer frente a la problemática de las variaciones morfológicas y errores ortográficos, tal y como se aborda en [37]. Además, existen diversas variaciones del modelo como los n-gramas de longitud variable [38], que indican el éxito que los n-gramas están teniendo en el PLN y en concreto, en la Recuperación de Información. Todo ello otorga a los n-gramas grandes ventajas como la eficiencia, simplicidad, robustez e independencia del dominio.

2.3.3 ALGORITMOS DE DETECCIÓN

La misión de los algoritmos de detección es la extracción de temas tendencia en un cuerpo textual. Para ellos los algoritmos suelen estar basados en modelos estadísticos embarcados en un marco matemático que permite examinar una serie de documentos y descubrir a partir de las estadísticas de las palabras en cada uno, cuales son los temas de los que se habla. Estos documentos, no serán otra cosa que los mensajes generados por los usuarios en Twitter a través de los cuales se podrán obtener los TT.

2.3.3.1 Basados en modelos estadísticos

Este tipo de algoritmos se caracterizan por la simplicidad del concepto que manejan y por ser un mecanismo rápido y bastante eficaz en la detección de temas emergentes. Su objetivo es la identificación de los términos relevantes o términos clave del texto, a través del cálculo de los pesos asociados a dichos términos. A este cómputo se le conoce como *ponderación del término* [39].

2.3.3.1.1 Modelo Binario

Método más elemental, en el que el peso puede adquirir dos valores distintos: 0 en caso que el término no aparezca en el documento, o 1 si el término si aparece.

2.3.3.1.2 TF

La *Frecuencia de Término* (TF, del inglés, "Term frequency") es un criterio que evalúa cómo de relevante es un término para un documento en una colección, midiendo su frecuencia de ocurrencia en la colección de documentos. La TF puede ser definida de dos formas distintas [40], de acuerdo con las siguientes expresiones.

$$tf_{i,j} = n_{i,j} \quad ; \quad tf_{i,j} = \frac{n_{i,j}}{N} \tag{1}$$

$n_{i,j}$ es el número de veces que el término *i* aparece en el documento *j*, y N es el número total de palabras del documento *j*. La primera definición es la propiamente *Frecuencia de Término* mientras que la segunda se corresponde con la *Frecuencia de Término Normalizado* (del inglés, "Normalized Term Frequency"). Existen variantes del algoritmo TF basadas en diversos modelos, tal y como se especifica en [41].

2.3.3.1.3 TF - IDF

El método TF-IDF [42] (del inglés, "Term Frequency – Inverse Document Frequency") es un criterio que evalúa cómo de relevante es una palabra para un documento en una colección, es decir, mide la frecuencia de ocurrencia del término en la colección de documentos [43].

Existen dos tipos de pesos asociados a un mismo término [44]: el *peso local* (la frecuencia de ocurrencia de un término en un documento) y el *peso global* (número de documentos en los que aparece dicho término). De esta forma, el peso local se corresponde con el concepto de TF, mientras que el peso global hace referencia a la *Frecuencia Inversa de Documento* (IDF) (del inglés, "Inverse Document Frequency"). La IDF viene definida según la ecuación (2).

$$idf_i = log(\frac{D}{d_i}) \tag{2}$$

D es el número total de documentos y d_i es el número de documentos que contienen el término *i*. La IDF permite penalizar aquellos términos que están en muchos

documentos, valorando en mayor medida aquellos que se encuentran en menos. Por lo tanto, el cálculo de TF-IDF viene dado por la unión de ambos pesos, tal y como se muestra en la ecuación (3).

$$TF - IDF = tf_{i,j} \cdot idf_i \quad (3)$$

El valor TF-IDF aumenta proporcionalmente al número de veces que un término aparece en el documento, pero es compensada por la frecuencia del término en la colección de documentos. Esto permite manejar el hecho de que algunas palabras son generalmente más comunes que otras. Además, existen variaciones del algoritmo TF-IDF basadas en el uso de diversos métodos de normalización ([40], [41]), en los que se emplea un factor de normalización que permite equilibrar casos en los que se emplean documentos de distintos tamaños.

2.3.3.1.4 NTF

Este método hace uso de una puntuación simple similar al enfoque que proporciona el método TF-IDF. Mientras que en el algoritmo TF-IDF la relevancia de un término en un documento se obtiene en función de la frecuencia de aparición de ese término dentro del documento, normalizado por el número total de documentos en los que aparece el término, el método *NTF* evalúa la frecuencia de los diversos términos en un periodo de tiempo concreto (ventana temporal).

La puntuación definida en este método como $ntf_{t,i}$ se define como el porcentaje del total de tweets que contienen el término *i* dentro de una ventana temporal [45]. La puntuación $ntf_{t,i}$ viene definida según la ecuación (4).

$$ntf_{t,i} = \frac{tf_{t,i}}{cfi} \quad (4)$$

$tf_{t,i}$ *("window term frecuency")* indica el número de tweets que contienen el término *i* dentro de un periodo temporal alrededor del instante *t* y cf_i *("corpus termfrequency")* hace referencia al número total de tweets que contienen el término *i* en la colección de tweets.

El hecho de utilizar la ventana temporal permite identificar aquellos instantes en los que los términos, por un lado son muy frecuentes, y relativamente poco frecuentes por otro. De esta manera se detectan los TT como aquellas zonas donde se alcanza el pico (valor máximo de $ntf_{t,i}$ para el término *i*). El valor máximo que puede alcanzar dicho pico será 1, caso en el que todas las apariciones del término *i* ocurren dentro de

una ventana temporal.

2.3.3.1.5 Entropía

Para calcular la entropía de un término, primero se seleccionan todos tweets que contengan el término a evaluar [40]. Entonces la entropía *(*del inglés, "*Entropy")* del término *i* se define según la ecuación (5).

$$H_i = - \sum_j \frac{n_{j,i}}{N} \log(\frac{n_{j,i}}{N}) \quad (5)$$

$n_{i,j}$ es el número de veces que la palabra *j* aparece en el conjunto de tweets que contienen el término *i* y *N* es el número total de palabras en la colección de tweets que contienen el término *i.* La entropía es un parámetro que puede servir de ayuda para detectar qué términos pueden ser clasificados como spam.

2.3.3.2 Basados en modelos probabilísticos

Por otro lado, y entrando en algoritmos de un mayor nivel de dificultad, tenemos los basados en métodos probabilísticos. Según el tipo de salida que produzcan, podemos distinguir dos grupos de algoritmos [46]: algoritmos basados en documentos y en funciones.

En el primer grupo se pueden destacar los algoritmos LDA y Doc-p [46]. En este tipo de algoritmos un tema está representado por un grupo de documentos. Este método, en algunos contextos, sufre el problema que la inclusión de un nuevo documento a un tema existente depende de umbrales arbitrarios, lo que supone una desventaja.

En el segundo grupo destacamos el algoritmo GFeat-p. En este tipo de algoritmos se hace referencia como función de pivote a la aparición de un grupo de palabras clave en un cierto lugar. Este método se basa comúnmente en el análisis de las asociaciones entre los distintos términos, por lo que a menudo se pueden tener en cuenta correlaciones erróneas.

2.3.3.2.1 LDA

LDA *(*del inglés, "*Latent Dirichlett allocation"*) es uno de los modelos Bayesianos más conocidos y usados en la detección de TT. Se trata de un modelo de aprendizaje supervisado para caracterizar el contenido de los mensajes [40]. En él, cada

documento se representa mediante una bolsa de términos, que son la única variable observada, a través de la cual se intenta extraer los temas tendencia. Es decir, LDA tiene como entrada un conjunto de términos correspondientes a la representación de cada documento de la colección, dando como salida los temas latentes de cada conjunto de palabras, que es lo mismo que los temas latentes de cada documento.

Formalmente, un documento se asocia con una distribución multinomial de temas que a su vez son distribuciones multinomiales de palabras. De esta forma podemos definir la siguiente estructura [47]:

- *Palabra*, es la unidad básica y discreta de datos. Se define como un conjunto de elemento de un vocabulario de tamaño *V*
- *Documento* ($w = \{w_1, w_2, \ldots w_N\}$), es una secuencia de N palabras
- *Cuerpo* ($D = \{ w_1, w_2, \ldots w_M \}$), es una colección de M documentos

En la siguiente tabla se muestra un resumen de las variables utilizadas en LDA [48].

Variable	Descripción
K	Número de temas
N	Número de palabras
M	Número de documentos
Θ	Distribución de temas en un documento
Ø	Distribución de palabras en un tema
α	Parámetro de la distribución Dirichlet de Θ
β	Parámetro de la distribución Dirichlet de ø
w	Palabra observada (a evaluación)
z	Tema evaluado

Tabla 1. Variables del algoritmo LDA

Detallando un poco más podemos especificar los siguientes elementos:

- Cada documento *j* tiene N_j palabras
- w_{ji} es la palabra *i* observada del documento *j*
- Todas las palabras de un documento *j* podrán ser agrupadas en *K* temas
- $Ø_k$, $Θ_j$ y z_{ji} son las variables ocultas que se deben deducir o inferir. Para estas dos primeras se hace uso de la distribución Dirichlet. z_{ji}, como ya explicamos más adelante, puede ser obtenida mediante el procedimiento de muestreo de Gibbs integrando $Ø_k$ y $Θ_j$

Estas variables están interconectadas en el modelo probabilístico en LDA tal y como

muestra la siguiente figura ([47], [49]).

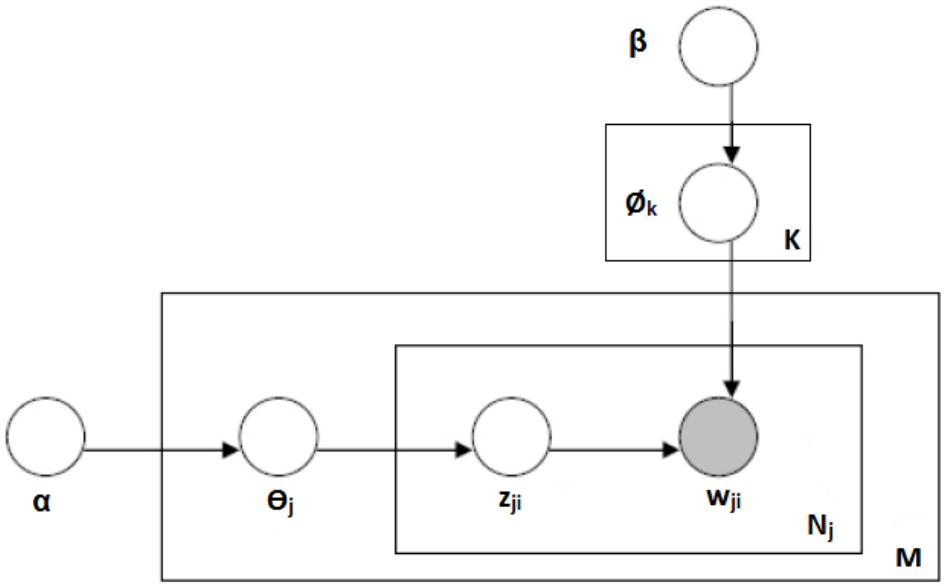

Figura 8. Interconexión de las variables del algoritmo LDA

Como se observa en la figura, existen tres niveles en la representación de LDA semejantes a la estructura antes definida:

- Los parámetros α y β se encuentran en el nivel de cuerpo, asumiendo que se muestrean una vez en el proceso de generación de un cuerpo.
- La variable Θ es un parámetro a nivel de documento, muestreada una vez por documento.
- Por último, los parámetros z y w son variables a nivel de palabra de las que se toman muestras una vez por cada palabra en cada documento.

Al contrario que muchos modelos de agrupación, LDA permite que un documento pueda estar asociado con múltiples temas. Esto se debe a los tres niveles implicados en el algoritmo y en particular, al nodo tema que se muestrea repetidamente en un documento.

El planteamiento de este algoritmo requiere el cálculo y la estimación de distintas probabilidades condicionadas de aparición, además del cálculo de la maximización de alguna de ellas. Esto provoca que en ocasiones nos encontremos con integrales intratables que sólo se puedan hallar mediante distintos métodos de aproximación, lo que entorpece el proceso de detección y aumenta en gran medida su complejidad. Por este motivo vamos a explicar un mecanismo aproximado basado en el muestro deGibbs, por el que se obtienen resultados casi idénticos. En este caso, el proceso

generativo del algoritmo LDA sigue la siguiente estructura [49]:

1) Para cada tema *k*, el parámetro multinomial $\emptyset_k$ se obtiene a partir del muestreo de la distribución Dirichlet $\emptyset_k \sim Dir(\beta)$
2) Para cada documento *j* y sobre los *k* temas posibles, el parámetro multinomial Θ_j se obtiene a partir del muestreo de la distribución Dirichlet $\Theta_j \sim Dirc(\alpha)$
3) Para cada palabra *i* del documento *j*, z_{ji} se obtiene a partir del muestreo de la distribución $z_{ji} \sim Discrete(\Theta_j)$
4) El valor w_{ji} de la palabra *i* en el documento *j*, se obtiene del muestreo de la distribución discreta de los temas z_{ji}, es decir, de $w_{ji} \sim Discrete(\Theta z_{ji})$

Tras ello, la probabilidad de que se dé un cierto tema bajo ciertas condiciones viene dada por la unión de dos ratios ([48], [49]): la probabilidad que tiene la palabra wji bajo el tema *k* y la probabilidad que tiene el tema *k* en el documento *j*. La unión de ambos, se produce según la ecuación (6).

$$p(z_{ji} = k|\mathbf{z}_{-ji}, \mathbf{w}, \alpha, \beta) \propto \frac{n^{(k)}_{-ji,w_{ji}} + \beta_{w_{ji}}}{\sum_{w=1}^{W}\left(n^{(k)}_{-ji,w} + \beta_w\right)} \cdot \frac{n^{(j)}_{-ji,k} + \alpha_k}{\sum_{k'=1}^{K}\left(n^{(j)}_{-ji,k'} + \alpha_{k'}\right)} \qquad (6)$$

$n^{(k)}_{-ji,w}$ es el número de palabras del cuerpo con valor *w*, asignadas al tema *k*, excluyendo la palabra *i* del documento *j* y $n^{(j)}_{-ji,k}$ es el número de palabras en el documento *j* asignadas al tema *k*, excluyendo la palabra *i* del documento *j*. De esta manera obtenemos la probabilidad de que se dé cierto tema en un documento teniendo en cuenta qué palabras contribuyen o no en la generación de ese tema.

3.3.2.2 Doc-p

Doc-p *(*del inglés, "*Document-pivot topic detection*"*)* es un método de detección y de seguimiento de temas que utiliza un enfoque basado en documentos [46]. Este algoritmo se ayuda del método LSH *(*del inglés, "*Locality Sensitive Hashing*"*)* para recuperar rápidamente el vecino más cercano de un documento y acelerar la tarea de agrupamiento. LSH es un método probabilístico cuyo fin es la reducción del número de variables aleatorias que se examinan. Para ello se aplica la función hash (distinta a la función hash en criptografía) a los datos de entrada consiguiendo que documentos similares se asignen a los mismos grupos con una alta probabilidad [50]. Se busca maximizar la probabilidad de colisión entre documentos similares.

Las fases que conforman el algoritmo se muestran a continuación [46]:

- Primero agrupa todos los mensajes y calcula la similitud del coseno de la representación TF – IDF de un mensaje de entrada con todos los mensajes procesados hasta el momento. Si la similitud con el mejor mensaje está por encima de un cierto umbral $\Theta_{tf\text{-}idf}$, asigna dicho mensaje de entrada a la agrupación ya existente como la mejor coincidencia. De lo contrario crea un nuevo clúster con el nuevo mensaje como único elemento. Los mejores grupos de Tweets se recuperan de manera eficiente mediante LSH
- Filtra los clústeres con número de elementos menor que Θ_n
- Para cada clúster *c*, calcula una puntuación según la ecuación (7).

$$score_c = \sum_{i=1}^{|Docs_c|} \sum_{j=1}^{|words_i|} \exp(-p(w_{ij})) \qquad (7)$$

  w_{ij} es el j-ésimo termino que aparece en el i-ésimo documento del clúster y $p(w_{ij})$ es la probabilidad de aparición de un solo término (y se estima a partir de un conjunto de datos de referencia recogidos de Twitter) Por lo tanto, cuanto menos frecuentes sean los términos, mayor será la puntuación del clúster.
- Los clústeres se ordenan según la puntuación obtenida y se devuelven los que mejor puntuación hayan obtenido

La ventaja de utilizar LSH se traduce en una mayor rapidez en la búsqueda de los vecinos más próximos con respecto a la similitud del coseno en una gran colección de documentos, lo que se traduce en una mayor eficiencia.

3.3.2.3 GFeat-p

GFeat-p (del inglés, “Graph-based feature-pivot topic detection”) es el primer método que vamos analizar dentro del grupo de algoritmos función de pivote. Su principal característica es el uso del algoritmo SCAN en la etapa de clúster o agrupamiento. SCAN (del inglés, “Structural Clustering Algorithm for Networks”) es un algoritmo de agrupamiento basado en las definiciones del algoritmo DBSCAN (del inglés, “Density-basedspatial clustering of applications with noise”) [51]. DBSCAN es el primer algoritmo de agrupamiento basado en densidad ya que es capaz de encontrar un número de grupos a partir de una distribución de densidad estimada, construyendo

grupos en los que sus puntos son de dos tipos:

- Puntos centrales, puntos que tienen en su vecindad una cantidad de puntos mayor o igual que un umbral especificado (parámetro MinPts)
- O puntos borde, puntos que no son ni centrales ni ruido (puntos que quedan fueran de los grupos)

Además de detectar comunidades (grupos) de nodos o puntos, SCAN proporciona una lista de todos los centros, cada uno de los cuales puede estar conectado a otro conjunto de comunidades. Relacionado con un enfoque más orientado a la detección de TT, tenemos el siguiente planteamiento:

- Los nodos o puntos del grafo se corresponderían con los términos
- Las comunidades o grupos se corresponderían con los temas

Tras recoger al azar un conjunto de tweets (cuerpo de referencia o tweets de referencia), se selecciona el conjunto de términos que se quiere agrupar (cuerpo a incluir o tweets a incluir). Para cada uno de los términos del cuerpo de referencia, la probabilidad de aparición se estima según la ecuación (8).

$$p\,(w|\text{cuerpo}) = \frac{N_w + \delta}{N + \delta n} \qquad (8)$$

N_w es el número de veces que aparece el término *w* en el cuerpo, N (para un término de longitud m) es el número de términos de longitud *m* que hay en todo el cuerpo y δ es un parámetro de suavizado (con un valor típico de 0.5), que se incluye para regularizar la probabilidad de estimación, de forma que si un nuevo término no aparece en el cuerpo, no se le asigna probabilidad 0.

Para averiguar cuáles son los términos de mayor importancia en el cuerpo, se calcula el cociente de las probabilidades de aparición de cada término en los dos cuerpos [52]. A este cociente se le llama la razón de verosimilitud, definido según la ecuación (9).

$$\text{Razón de verosimilitud} = \frac{p\,(w \mid tweets\ a\ incluir)}{P\,(w \mid tweets\ de\ referencia)} \qquad (9)$$

Los términos con la razón de verosimilitud más alta serán aquellos que tengan una frecuencia de aparición mayor y se espera que se relacionen con los temas más discutidos de forma activa en el cuerpo.

Una vez seleccionados estos términos, se aplica el algoritmo SCAN para extraer

distintos grupos de términos, cada uno de los cuales se consideran que son distintos temas. Para ello el proceso a seguir consta de las siguientes fases [46]:

- *Selección*: se seleccionan los K mejores términos con mejor razón de verosimilitud y para cada uno de ellos se crea un nodo distinto en el gráfico G
- *Enlace*: cada uno de los nodos de G se conectan mediante una estrategia de conexión de términos. En primer lugar se escogen pares de términos y se calcula su similitud. Para ello se utilizan distintos métodos:

  . El número de documentos en los que los términos pueden coaparecer
  . El número de coapariciones divido por la mayor o menor frecuencia de documento de los dos términos
  . El coeficiente de similitud de Jaccard (mide la similitud entre conjunto de muestras finitas)

  Además, también se puede utilizar un enfoque kNN en el que se une cada término con sus k vecinos más próximos o un enfoque en el que se unen pares de nodos que tienen una similitud superior a un valor fijado.
- *Agrupamiento*: en este punto se aplica el algoritmo SCAN, con el que se genera un tema por cada grupo o comunidad de términos detectados.
- *Mejora del agrupamiento*: si la conectividad de cada uno de los centros detectados por SCAN para cada una de las comunidades supera un cierto umbral, los centros se unen a dichas comunidades. Como ya se mencionó antes, SCAN permite que los centros que detecta puedan estar conectados con varias comunidades o grupos de términos al mismo tiempo.

Se puede observar la importancia de la fase de unión entre los distintos términos del gráfico G. El éxito de esta fase y de todo el proceso en general depende directamente del buen cálculo de la similitud entre los distintos términos.

De la importancia del cálculo de la similitud entre pares de términos del algoritmo recientemente analizado, surge el problema de los algoritmos basados en funciones. Este tipo de algoritmos basa su buen funcionamiento en el buen cálculo de la similitud entre pares de términos, es decir, en algún tipo de función que cuente el número de coocurrencias entre pares de términos. En caso de que no sean temas estrechamente relacionados y que no compartan un número relativamente grande de términos, es probable que el cálculo de la similitud produzca temas más genéricos o temas fusionados, lo que provocaría que la detección de temas no fuese tan precisa.

Una opción para afrontar este posible problema, sería tener en cuenta la coocurrencia simultánea entre más de dos términos. Esto nos lleva a considerar el uso de minería de datos.

2.3.3.3 Basados en minería de datos

La minería de datos es un campo de las ciencias de la computación que agrupa una serie de técnicas y tecnologías que tienen como misión la extracción de información a partir de patrones contemplados en grandes colecciones de datos estructurados, almacenados en bases de datos [53]. Trata de detectar patrones repetitivos que expliquen el comportamiento de los datos en un determinado contexto. Por ello se trata de una buena técnica para la detección de temas, ya que es capaz de determinar qué elementos son propensos a coocurrir en un conjunto de datos o transacciones.

En un contexto de redes sociales y concretamente en Twitter, un elemento *w* es cualquier término mencionado en un tweet (sin incluir signos de puntuación, palabras vacías, etc). De esta forma la transacción se corresponde con un tweet y el conjunto de transacciones o datos a analizar son todos los tweets que se producen en un intervalo de tiempo *Tj*. El número de veces que un conjunto de términos dado se produce en un intervalo de tiempo se llama soporte *(support)*, y cualquier conjunto de elementos que cumple un soporte mínimo se llama patrón.

2.3.3.3.1 AIS

AIS *(Agrawal, Imielinski & Swami)* fue el primer algoritmo desarrollado para obtener reglas de asociación, un mecanismo de representación del conocimiento que permite caracterizar posibles regularidades que pueden darse en una gran base de datos [54].

En AIS, una regla de asociación se defina de la forma X $\Rightarrow$ Y *[s,c]* en donde *X* es un itemset (conjunto de ítems, de términos), *Y* es un ítem no incluido en X, *s* representa la relevancia (proporción de transacciones de la base de datos que contienen X ∪ Y) y *c* representa la fiabilidad (*confidence*, proporción de transacciones con X que contienen también a Y). Para evitar la generación de reglas irrelevantes se definen dos umbrales: *MinSupport* y *MinConfidence.* Respecto a la terminología, podemos destacar los siguientes conceptos [55]:

- *k-itemset:* un itemset con k items se denomina k-itemset (k conjuntos de términos distintos, itemset de tamaño k). Los k-itemsets con una relevancia mayor o igual que *MinSupport* forman parte del conjunto *L[k].*
- *L[k]*: es el conjunto de k-itemsets relevantes *(large k-itemsets).*
- *C[k]*: es el conjunto de k-itemsets candidatos *(candidate k-itemsets)* que en principio podrían pertenecer al conjunto L[k], ya que podríanllegar a tener la relevancia mínima.

El proceso que sigue AIS se define acorde a la siguiente estructura:

- Se lee una transacción (tweet) de la base de datos.
- Se determina qué conjunto de términos (itemsets) relevantes de los encontrados en la pasada anterior por la base de datos, están contenidos en la transacción.
- Se generan los itemsets candidatos. Para ello se extienden los itemsets relevantes encontrados en la transacción con otros ítems de la transacción.
- Se calcula la relevancia de cada uno de los itemsets candidatos a medida que se recorre la base de datos. Para ello se hace uso de un contador con el que se va evaluando la frecuencia de cada uno.
- Los candidatos cuya frecuencia sea mayor que la establecida por *MinSupport*, pasan a formar parte del conjunto de términos relevantes *L[k].*

De esta forma, el pseudocódigo del algoritmo quedaría tal y como se muestra en la siguiente figura.

```
L[1] = {large 1-itemsets}
k = 2
Mientras L[k-1] ≠ ∅
      C[k] = ∅;
      Para cada transacción t ∈ D
            L_t = Conjunto de (k-1)-itemsets relevantes contenidos en t
            Para cada itemset l_t ∈ L_t
                  C_t = Extensiones de l_t contenidas en t
                  Para cada candidato c ∈ C_t
                        Si c ∈ C[k]
                              Incrementar el contador asociado a c en C[k]
                        Si no
                              Añadir c a C[k] (contador=1)
      L[k] = { c ∈ C[k] | contador(c)≥MinSupport }
      k++
Solución: ∪ L[k]
```

Figura 9. Pseudocódigo del algoritmo AIS

Así de una transacción *t* (tweet) de *N* ítems (términos) que contiene un determinado k-itemset relevante, se generan (k+1)-itemsets candidatos y seincluyen en *C[k+1]*. En caso de que el itemset ya exista en el conjunto de candidatos de una transacción anterior, se actualiza su contador de frecuencia. Si por el contrario es un itemset nuevo, se la asigna un contador de frecuencia igual a 1.

2.3.3.3.2 SETM

El algoritmo SETM *(*del inglés, "*SET-oriented Mining of association rules*"*)* deriva del algoritmo AIS y fue diseñado para utilizar SQL [56] en la generación de itemsets relevantes, por lo que las transacciones en este caso se representan por conjuntos de tuplas <TID, ítem>. El TID es el identificador de la transacción a partir de la que se obtuvo un determinado término o ítem. SETM se caracteriza porque realiza de manera independiente la generación de los itemsets candidatos de la obtención de la relevancia de cada uno de ellos. A diferencia de AIS, SETM hace uso de dos tablas auxiliares adicionales:

- *CT[k]*: tabla en la que se almacenan los k-itemsets candidatos con su TID correspondiente.
- *LT[k]*: tabla en la que se almacenan los k-itemsets relevantes con su TID correspondiente. Para su obtención, se eliminan aquellos candidatos de CT[k] que no superen la relevancia mínima especificada por *MinSupport.*

El hecho de almacenar también el TID, permite acelerar tanto el proceso de obtención de los itemsets relevantes en una transacción determinada como el proceso de generación de los candidatos [54]. Por un lado permite una mayor rapidez en la obtención de los itemsets relevantes ya que evita tener que comprobar si un determinado itemset está incluido en una transacción dada. Por otro lado, permite mejorar el proceso de generación de los candidatos ya que, tras ordenar la tabla LT[k] según su TID, sólo es necesario recorrerla una vez para la generación de los candidatos (suponiendo que la base de datos está ordenada según su TID).

De esta forma, el pseudocódigo del algoritmo quedaría tal y como aparece en la siguiente figura.

```
L[1]  = { large 1-itemsets }
LT[1] = { large 1-itemsets acompañados de los TIDs correspondientes }
k = 2
Mientras L[k-1] ≠ ∅
      CT[k] = ∅
      Para cada transacción t ∈ D
            L_t = Conjunto de (k-1)-itemsets de LT[k-1] contenidos en t (TID)
            Para cada itemset l_t ∈ L_t
                  C_t = Extensiones de l_t contenidas en t
                  CT[k] + = { <TID,c> | c ∈ C_t }
      Ordenar CT[k] por itemsets
      Eliminar de CT[k] los itemsets con contador < MinSupport ⇒ LT[k]
      L[k] = { < l.itemset, l.contador > | l ∈ LT[k] }
      Ordenar LT[k] según TID
      k++
Solución:  ∪ L[k]
```

Figura 10. Pseudocódigo del algoritmo SETM

Como podemos imaginarnos las ventajas que por un lado nos ofrece este algoritmo, por otro lado se traducen en pequeños inconvenientes:

- En cuanto al tamaño que puede adquirir la tabla de candidatos CT[k], para cada itemset candidato en la tabla aparecerán tantas tuplas como transacciones haya en la base de datos que incluyan al candidato.
- En cuanto a la obtención de la relevancia de los itemsets candidatos, la tabla CT[k] debe reordenarse según las tuplas en las que aparezca un itemset, ya que en el paso anterior se encontraba ordenada según el TID.
- En cuanto a la generación de candidatos, la tabla LT[k] (obtenida tras la eliminación de los candidatos cuya relevancia no llega al umbral mínimo) necesita reordenarse de nuevo según su TID.

Como evolución de los dos últimos algoritmos explicados, surgió el algoritmo Apriori, un método que ofrece una mayor eficiencia en el manejo de los datos.

2.3.3.3.3 Apriori

Las novedades que ofrece este algoritmo se resumen a continuación [54] :

- En la obtención de los conjuntos de itemsets relevantes (L[k]) se realizan múltiples pasadas sobre la base de datos.
- El conjunto de los términos candidatos se obtienen sólo a partir del conjunto de itemsets relevantes encontrados en la pasada anterior. En la primera iteración

se calcula el conjunto de itemset relevantes L[1] que estará formado por los ítems que alcancen la relevancia mínima. En el resto de iteraciones, el conjunto de k-itemsets relevantes L[k] ya obtenido se utiliza para generar el conjunto de itemsets candidatos C[k+1]. Para ello se realiza el producto cartesiano L[k]*L[k] imponiendo la restricción de que los k-1 primeros ítems de los elementos L[k] deben coincidir. Aquellos de los candidatos que no superen la relevancia mínima se eliminan y el resto pasa a formar parte del conjunto L[k+1].

- Tanto los ítems de cada transacción como los ítems dentro de un itemset , se presupone que están ordenados lexicográficamente.

El pseudocódigo del algoritmo Apriori quedaría tal y como se observa en la siguiente figura ([54], [57]).

```
L[1] = { large 1-itemsets }
k = 2
Mientras L[k-1] ≠ ∅
      C[k] = CandidatosAPRIORI ( L[k-1] )
      Para cada transacción t ∈ D
            Ct = Conjunto de candidatos de C[k] contenidos en t
            Para cada candidato c ∈ Ct
                  c.contador ++
      L[k] = { c ∈ C[k] | c.contador ≥ MinSupport }
      k++

Solución:  ∪ L[k]
```

Figura 11. Pseudocódigo del algoritmo Apriori

Es evidente que a medida que aumentamos el conjunto de itemsets candidatos el algoritmo tardará más en encontrar aquellos itemsets relevantes. De hecho, algoritmos como a priori necesitan descubrir del orden de 2^k itemsets relevantes antes de encontrar un k-itemset relevante, por lo que para valores de k elevados, la complejidad exponencial limita la aplicabilidad del algoritmo. Para abordar esta problemática surge el algoritmo DHP, cuya idea básica es uso de una heurística que genere solo los candidatos que tenga una probabilidad alta de ser relevantes. Variaciones del algoritmo Apriori como AprioriTID y AprioriHybrid, tratan de abordar distintos mecanismos para mejorar el comportamiento. En [58] se muestra una comparativa entre las distintas versiones, según distintas propiedades.

2.3.3.3.4 DHP

DHP *(*del inglés, "*Direct Hashing and Prunning*"*)* (basado en el algoritmo Apriori) intenta disminuir el número de candidatos mediante el uso de una tabla hash para (k+1)-itemsets al generar el conjunto de términos relevantes L[k]. El uso de esta tabla cobra especial importancia en las primeras iteraciones, ya que es cuando se producen una mayor cantidad de términos candidatos y, por lo tanto, un mayor coste computacional al obtener los conjuntos L[k] [54]. Destacamos dos comportamientos:

- Dado un itemset *(i)* perteneciente al conjunto de términos relevantes L[k], se incrementan los contadores de las entradas de la tabla hash *h[c]* para todos los (k+1)-itemsets candidatos *(c)* derivados del itemset *i* (como se especifica en el algoritmo Apriori).
- Dado un candidato *(c)*, al generar el conjunto de itemsets candidatos C[k+1]:
  - si h[c] es menor que el umbral *MinSupport*, se descarta el candidato (no se incluirá ya que se sabe que no puede pertenecer al conjunto L[k+1]).
  - si h[c] es mayor que el umbral *MinSupport*, se incluye elcandidato.

A pesar que el algoritmo DHP intenta abordar la problemática que surge con el aumento del conjunto de itemsets candidatos, cuando se manejan conjuntos muy grandes, el problema persiste en mayor o menor medida. Para ello surge el algoritmo Max – Miner.

2.3.3.3.5 Max-Miner

El algoritmo *Max-Miner* define un nuevo concepto; itemset relevante maximal. Un itemset relevante maximal es aquel itemset relevante que no es un subconjunto propio de otro itemset relevante. Este algoritmo es capaz de obtener eficientemente el conjunto de itemsets relevantes maximales mediante el uso de heurística en la búsqueda. Para ello Max-Miner hace uso de un árbol de enumeración *(set-enumeration tree)* para la representación de los ítems. En su construcción se supone la existencia de un orden prefijado entre los ítems. Este árbol está compuesto por los siguientes elementos [59]:

- Raíz del árbol, correspondiente al conjunto vacío
- Nodos. Un nodo situado a una profundidad k, está asociado a un conjunto de k elementos (un k-itemset). A cada nodo del árbol se le asocia un grupo candidato *g*. Este grupo está formado por dos conjuntos de ítems:
  - *h(g) (cabeza)*, se trata del itemset correspondiente al nodo

- *t(g) (cola)*, es el conjunto de ítems que pueden aparecer en los subnodos

  El cálculo de la relevancia de un grupo candidato, se basa en la obtención de la relevancia de los itemsets *h(g), h(g) ∪ t(g) y h(g) ∪ {i} para cualquier i ∈ t(g)*

- Los nodos hijos (subnodos de un nodo) son los k-itemsets derivados del nodo padre

De esta forma, en el caso de 4 items diferentes, la estructura quedaría tal y como se muestra en la siguiente figura [59].

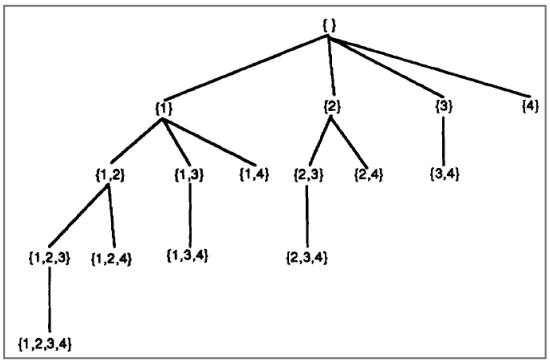

Figura 12. Ejemplo de la estructura del algoritmo Max-Miner para un conjunto de 4 items distintos

En cuanto a su funcionamiento, podemos distinguir los siguientes pasos [54]:

I. Se analiza el dominio de los ítems y se crea el árbol de enumeración (equivalente a la obtención L[1] en el algoritmo Apriori)

II. Se realiza un recorrido en anchura por el árbol para obtener el conjunto de itemsets relevantes maximales

III. Al mismo tiempo, se va podando el árbol. Para ello se usan dos criterios distintos:

   . *Super set-frequency prunning*: Si *h(g) ∪ t(g)* es relevante, cualquier itemset enumerado pornodo será relevante pero no maximal (no hace falta seguirexpandiendo el nodo)

   . *Sub set-infrequency prunning*: Si un itemset *h(g) ∪ {i}* no es relevante, cualquier cabeza de un nodo que incluya *i* tampoco lo será, por lo que se puede suprimir del conjunto *t(g)*

IV. Es decir, se expanden sólo aquellos nodos tales que *h(g) ∪ t(g)* no sea relevante y no esté incluido en ningún itemset de los que son relevantes, tras eliminar de *t(g)* aquellos items que hacen que *h(g) ∪ {i}* no sea relevante

En [60] podemos encontrar el pseudocódigo del algoritmo. Así pues, el algoritmo Max-Miner crece linealmente con el número de itemsets relevantes maximales, independientemente de su longitud. Esto mejora la dependencia de la complejidad del algoritmo Apriori con la longitud máxima de los itemsets relevantes.

Siguiendo con algoritmos basados en minería de datos, nos encontramos con los siguientes algoritmos.

2.3.3.3.6 FPM

El algoritmo FPM *(*del inglés, "*Frequent Pattern Mining*"*)* trabaja con uno de los conceptos más importantes de la minería de datos, un patrón.FPM tiene como misión detectar patrones (conjunto de elementos, subsecuencias, subestructuras, etc) que se producen con frecuencia en un conjunto de datos [46]. En lo que a twitter se refiere, FPM tiene el objeto de detectar los conjuntos de palabras clave más relevantes para cada intervalo de tiempo en un gran torrente de tweets. Este conjunto de palabras clave se pueden considerar los temas que mejor representan las interacciones sociales subyacentes. El algoritmo FP se basa en el lema "divide y vencerás". Para ello es necesario seguir dos pasos: la detección y clasificación.

1. *Detección del patrón (FP detection)*

   Para la detección del patrón, primero es necesario construir una estructura de datos compacta llamada "*FP-tree".* Esta estructura de datos se define como un árbol con la siguiente forma [61]:

   - Un nodo raíz etiquetado como *"null".*
   - Distintos subárboles formados por nodos hijos. Cada uno de ellos está formado por tres campos:
     - Item-name: indica que término está representado por el nodo.
     - Count: indica el número de transacciones que utilizan el nodo en cuestión para llegar a otros nodos.
     - Nodo-Enlace: indica el enlace con el siguiente nodo del árbol.

   Para la creación del árbol se necesita seguir los siguientes pasos [62]:

   - *Lista de palabras clave*: como paso inicial, para cada intervalo de tiempo se crea una lista de palabras claves ordenadas por frecuencia en orden descendente *(F-List)* Para reducir el número de palabras clave a incluir, se

eliminan aquellos términos que no sean frecuentes (aquellos términos que no lleguen a un valor mínimo de frecuencia fijada).

- *Construcción del árbol (FP-tree)*: para cada intervalo de tiempo se construye la estructura de datos en forma de árbol, es decir, se van escaneando el conjunto de términos de una transacción al mismo tiempo que se va creando el árbol. Esto permite clasificar los patrones según sus coocurrencias. Además, al mismo tiempo que se va creando el árbol, se va analizando, lo que permite producir reglas de asociación del conjunto de las transacciones:

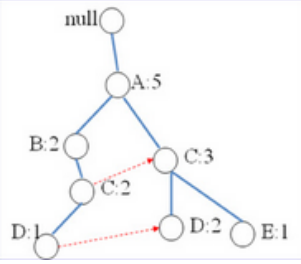

Figura 13. Ejemplo de árbol según FPM

$$\{w_1, w_2\} \rightarrow P_i = \{w_3, w_4\} \ \ con\ support(P_i)$$

Como observamos, la tarea de encontrar conjunto de elementos frecuentes va a depender de la construcción de los árboles.

2. *Clasificación (FP ranking)*

   Una vez creado el árbol *FP-tree*, se extrae el conjunto de patrones frecuentes de la base de datos, se clasifican y los mejores *N* resultados se devuelven como temas candidatos. Para ello se hace uso del algoritmo *FP-Growth* [62]. En [63] se muestra una variante de este algoritmo.

La clave de este algoritmo es la clasificación de los patrones de tal manera que las palabras clave de los temas candidatos estén suficientemente relacionadas entre sí y con la diversidad suficiente para cubrir los diferentes temas subyacentes de las interacciones sociales

Hemos visto cómo el algoritmo FPM es capaz de detectar patrones en un conjunto de términos. Este algoritmo, como ya se comentó en la introducción a los algoritmos basados en la minería de datos, proporciona una solución a los algoritmos basados en funciones en los que sólo se tiene en cuenta las coocurrencias entre pares de términos en el caso de temas densamente interconectados. El algoritmo que se enuncia a continuación, propone un método para detectar patrones que coocurran en un conjunto de términos (de más de dos términos) pero sin exigir que todos los términos de ese conjunto coocurran con frecuencia, es decir, que los temas no necesariamente deben estar altamente interconectados. De aquí surge una versión "suave" del algoritmo FPM.

2.3.3.3.7 SFPM

SFPM *(*del inglés, "*Soft Frequent Pattern Mining*"*)* es una versión de FPM en la que se "suaviza" la búsqueda de patrones en un conjunto de términos [40]. Este método parte de un conjunto de términos *S,* al que se le van añadiendo términos de forma "codiciosa" según la frecuencia con la que estos coocurren con los términos de *S*. Es decir, inicialmente partimos de un conjunto de términos de partida (*S*) y un término candidato (*t*). Con el fin de cuantificar la coocurrencia entre el conjunto S y el término candidato *t*, se crean dos vectores; D_S y D_t respectivamente, ambos de longitud n, donde n en es el número de documentos de la colección. Estos vectores nos indican lo siguiente:

- El i-éismo elemento de D_S indica cuántos de los términos de *S* coocurren en el i-ésimo documento
- El i-éismo elemento de D_t es un indicador binario que representa si el término candidato *t* se encuentra o no en el i-ésimo documento

Por lo tanto, para un término t que coocurra frecuentemente con el conjunto de términos de *S*, el vector D_t tendrá un alto valor de similitud del coseno con respecto al vector D_S. Hay que destacar que algunos de los elementos de D_S pueden tener el valor $|S|$, lo que significa que todos los elementos de *S* coocurren en los documentos correspondientes, mientras que otros pueden tener un valor más pequeño indicando que sólo un subconjunto de los términos de *S* coocurren en los documentos.

De este último tipo de elementos surge la adaptación "suave" entre un término candidato propuesto para la expansión y un conjunto de términos *S.* Para este término a examen, los documentos en los que no todos los términos coocurren también van a influir en el valor del cálculo de la similitud (aunque en menor medida con respecto a los documentos en los que todos los términos coocurren). Por ello se produce una suavización en la expansión de S con el término candidato.

Un aspecto importante del método es saber cuándo un término candidato se añade o no al conjunto de términos, ya que en caso negativo provocaría la detención del proceso de expansión de *S*. Este criterio debe hacer frente a la cohesión de los temas generados, ya que en caso de que no estén densamente conectados, los temas resultantes pueden ser muy genéricos (pocas palabras clave) o pueden ser una mezcla de varios temas (muchas palabras claves relacionadas en ocasiones con temas

irrelevantes). Para ello se hace uso de la similitud del coseno entre *S* y el siguiente mejor término coincidente. Si esta similitud está por encima de un umbral, se añade el término, de lo contrario el proceso de expansión se detiene. Este umbral lo define una función sigmoidea, tal y como se aprecia en la ecuación (10),

$$\theta(S) = 1 - \frac{1}{1 + \exp((|S| - b)/c)} \tag{10}$$

en donde *b* y *c* son parámetros libres cuya función es controlar el tamaño del conjunto de términos y definir el nivel de "suavizado" en relación a las restricciones de las coocurrencias. En la práctica estos valores se establecen de manera que cuando la cardinalidad de *S* sea baja, la adición de términos candidatos sea más fácil (el umbral sea más bajo). En esta situación es posible que los términos que se añadan se asocien a diferentes temas. Por el contrario cuando la cardinalidad de *S* sea alta, la adición de términos deberá ser más difícil (umbral más alto), así *S* no crece sin límite.

Este procedimiento de búsqueda se aplicará tantas veces como términos candidatos tengamos, lo que producirá tantos temas como términos consideremos, de los cuales muchos estarán duplicados, por lo que es necesario un proceso final en el que se eliminen todos los duplicados. El pseudocódigo del algoritmo, se muestra en la siguiente figura.

```
T: conjunto de términos candidatos
Topics = 0
for cada término t en T do
        S = t;
        D_S = D_t;
        SeguirExpandiendo = true;
        repeat
            t^ = ConseguirMejorCuerpoTérminos(D_S, S, T);
            sim = SimilitudCoseno(D_S, D_t^);
            if sim > Θ(S)
                S = S ∪ t^;
                D_S = D_S + D_t^;
                for each D_S^i < |S|/2 poner D_S^i = 0
            else
                SeguirExpandiendo = false;
        until SeguirExpandiendo
        Topics = Topics ∪ S
end for
```

Figura 14. Pseudocódigo del algoritmo SFPM

Hay que destacar que para acotar el procedimiento de búsqueda dentro de unos límites razonables, se seleccionan los n términos con mejor razón de verosimilitud (ratio usado también el algoritmo G-Featp), definido según la ecuación (11).

$$\text{Razón de verosimilitud} = \frac{p\,(w \mid tweets\ a\ incluir)}{P\,(w \mid tweets\ de\ referencia)} \qquad (11)$$

Además de lo antes explicado y cómo se observa en el pseudocódigo, después de cada etapa de expansión, se ponen a cero aquellas entradas de D_S cuyo valor sea menor $|S|/2$. El motivo de esto se debe a que si un término muy frecuente se añade al conjunto de términos, el vector D_S puede incluir muchas entradas no nulas con valores muy pequeños. Esto puede tener el efecto de que un término pueda considerarse relevante para *S* ya que coocurre con frecuencia sólo con un número muy reducido de términos del conjunto más que con la mayoría de ellos.

En estos dos últimos algoritmos, y en general en los algoritmos basados en minería de datos, hemos visto cómo se pueden tener en cuenta las coocurrencias simultáneas que se producen entre más de dos términos. Sin embargo, se puede obtener un resultado similar de una forma algo más simple: mediante el uso de n-gramas en vez de unigramas (n-gramas de un solo término)

2.3.3.3.8 BNgram

En ese método se aborda el uso de n-gramas ya explicado anteriormente pero añadiendo una nueva característica. Se tiene en cuenta el cambio en la frecuencia de los términos a lo largo del tiempo como una fuente de información útil para la detección de los temas emergentes. El objetivo principal de este método es el de encontrar los temas emergentes comparando la frecuencia de los términos en el instante actual con aquellos de otros instantes anteriores [46].

Para ello se introduce el parámetro del tiempo en la puntuación característica del método *TF-IDF*, obteniendo una nueva puntuación llamada $TF\text{-}IDF_t$. Se usan datos del pasado para penalizar aquellos temas que empezaron en el pasado y que aún siguen siendo populares en el presente, y que por lo tanto no se definen como nuevos temas. Los principales pasos a seguir son los siguientes:

a) Se agrupan todas las palabras clave de los mensajes de la colección.
b) Se organizan los índices de palabras clave (quienes tienen en cuenta, además de una sola palabra clave, bigramas y trigramas) en diferentes instantes de tiempo

c) Se calcula la puntuación *tf*-idft. Para ello, la puntuación se calcula para cada n-grama correspondiente al instante actual *i* en base a su frecuencia de documento (para ese mismo instante) y se penaliza por el logaritmo de la media de las frecuencias de los documentos en los *t* instantes de tiempo anteriores. Matemáticamente, la expresión de la puntuación *tf-idf*$_t$ está definida por la ecuación (12).

$$df\text{–}idf_t = \frac{df_i + 1}{\log\left(\frac{\sum_{j=i}^{t} df_{i-j}}{t} + 1\right) + 1} \tag{12}$$

Como resultado de este proceso, se crea un ranking de n-gramas basados en las puntuaciones que han obtenido.

d) Se aplica un algoritmo de clustering para agrupar los n-gramas más representativos en grupos, cada uno representando un tema. El motivo de este paso se debe a que la información que ofrece un grupo de n-gramas es mucho mayor que la que puede ofrecer uno solo. La acción de agrupar se basa en el cálculo de distancias entre los n-gramas o grupos de n-gramas [64]. A partir del conjunto de todas las distancias, aquellas que no superen un determinado umbral se consideran que pertenecen y representan el mismo tema

El umbral se fija según la similitud. La similitud entre dos n-gramas se define como la fracción de los mensajes que contienen ambos. En un comienzo se asigna un clúster para cada n-grama para luego aplicar un algoritmo estándar de clustering para encontrar de forma iterativa y combinar el par más cercano de cluters [65]. Cuando un grupo de n-gramas se une a otro, la similitud de la nueva agrupación se calcula como el promedio de las similitudes de los grupos combinados. Este proceso de agrupamiento se repite hasta que la similitud entre los grupos sin combinar más cercanos cae por debajo de un umbral de similitud fijo Θ

e) Por último, los grupos se clasifican de acuerdo con la puntuación *tf-idf*$_t$ más alta de los n-gramas contenidos en el cluster, tal y como se muestra en la siguiente figura.

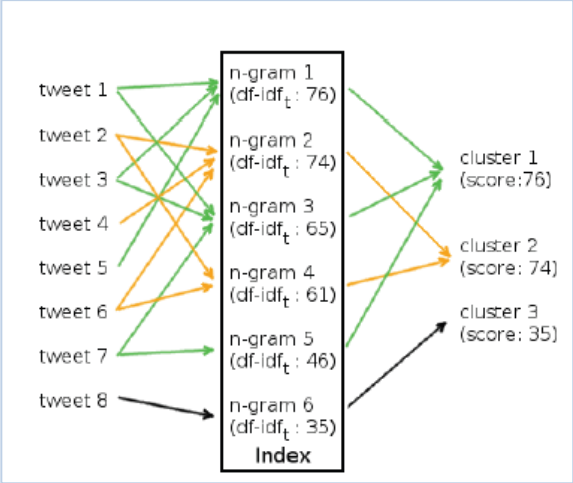

Figura 15. Ejemplo clasificación de n-gramas según el algoritmo BNGRAM

Se observa la organización de cada n-grama según su índice, en la que cada n-grama tiene una referencia a los tweets donde aparecen. Cada cluster o grupo se compone de distintos n-gramas y su puntuación $tf\text{-}idf_t$ se calcula como el valor máximo de $tf\text{-}idf_t$ de los n-gramas que la componen. Este criterio de clasificación se basa en el hecho de que la puntuación de cada grupo debe estar asociado con la puntuación de los n-gramas más representativos de la agrupación.

La principal contribución de este enfoque es el uso de la dimensión temporal de los datos para detectar temas emergentes.

CAPÍTULO 3 - DETECCIÓN DE TRENDING TOPICS EN TWITTER

En el capítulo anterior se ha mostrado una panorámica general de los campos que engloba nuestro trabajo. Por un lado se ha explicado Twitter, sus características y prestaciones. Por otro lado se ha realizado una breve introducción al PLN, sus principales aplicaciones según distintas clasificaciones y los distintos tipos de sistemas que engloba. Por último se han explicado y desarrollado diversos algoritmos de búsqueda de TT basados en distintos modelos.

En el vigente capítulo vamos a presentar el sistema que ha sido diseñado para la detección de Trending Topic en Twitter, utilizando PHP (*Hipertext Preprocessor)* como lenguaje de programación.

En el primer apartado plantearemos la arquitectura del sistema, explicando a un alto nivel su diseño. Se describirá la estructura general, los módulos y fases que la conforman, su funcionalidad, la metodología utilizada y los algoritmos implementados. Todo ello sin entrar en detalles de la implementación.

En el segundo apartado abordaremos la implementación concreta del sistema. Básicamente justificaremos cómo se ha implementado la funcionalidad de cada módulo descrita en el apartado anterior. Para ello se indicará la estructura del código junto con las prestaciones y opciones que este nos ofrece.

En un último apartado ilustraremos un manual que ofrezca la posibilidad que cualquier persona pueda ejecutar, modificar o ampliar el sistema que se ha diseñado. Contendrá detalles técnicos de la implementación, como el lenguaje y compilador utilizado, su instalación, ejecución del programa, ficheros generados, etc.

3.1 DISEÑO DEL SISTEMA

El principal objetivo de nuestro sistema es la detección de temas emergentes en una colección de tweets, tarea de gran interés debido a la heterogeneidad y la inmensa escala de los datos que fluyen por Twitter. Para ello se sigue una metodología

muy clara y marcada, que define una serie de pasos y procedimientos básicos a tener en cuenta.

El sistema está estructurado en distintos módulos, cada uno de ellos con una funcionalidad específica. Esto nos permite por un lado, tener bien identificadas cada una de las fases por las que pasan los datos, y por otro, tener una mayor abstracción en la estructura general del programa, facilitando su comprensión e implementación.

A continuación explicamos el proceso general que rige nuestro sistema, detallando las funciones básicas de cada módulo.

3.1.1 ESTRUCTURA GENERAL

Partiendo de una gran colección de tweets, el primer propósito se centra en la obtención del conjunto de términos representativos. A este procedimiento se le conoce como el *Modelo de Espacio Vectorial*. Se trata de un modelo algebraico que representa documentos en lenguaje natural de una manera formal mediante el uso de vectores, es decir, partiendo de un texto, obtiene como resultado un vector de términos representativos, términos que aportan significado. Este primer modelado consta de dos fases: una fase de preprocesamiento y otra de tratamiento. Este paso es de vital importancia ya que nos facilitará el posterior tratamiento y análisis de los datos.

Una vez obtenido el vector de términos representativos, será necesario llevar a cabo una evaluación de los mismos. Esto se traduce en la aplicación de distintos algoritmos con los que obtendremos los resultados que posteriormente representaremos.

Para finalizar, se llevará a cabo un análisis de los resultados obtenidos, lo que nos permitirá sacar conclusiones sobre el comportamiento de cada método aplicado.

De esta forma, el diagrama de flujo genérico de nuestro sistema sigue la estructura de la siguiente figura.

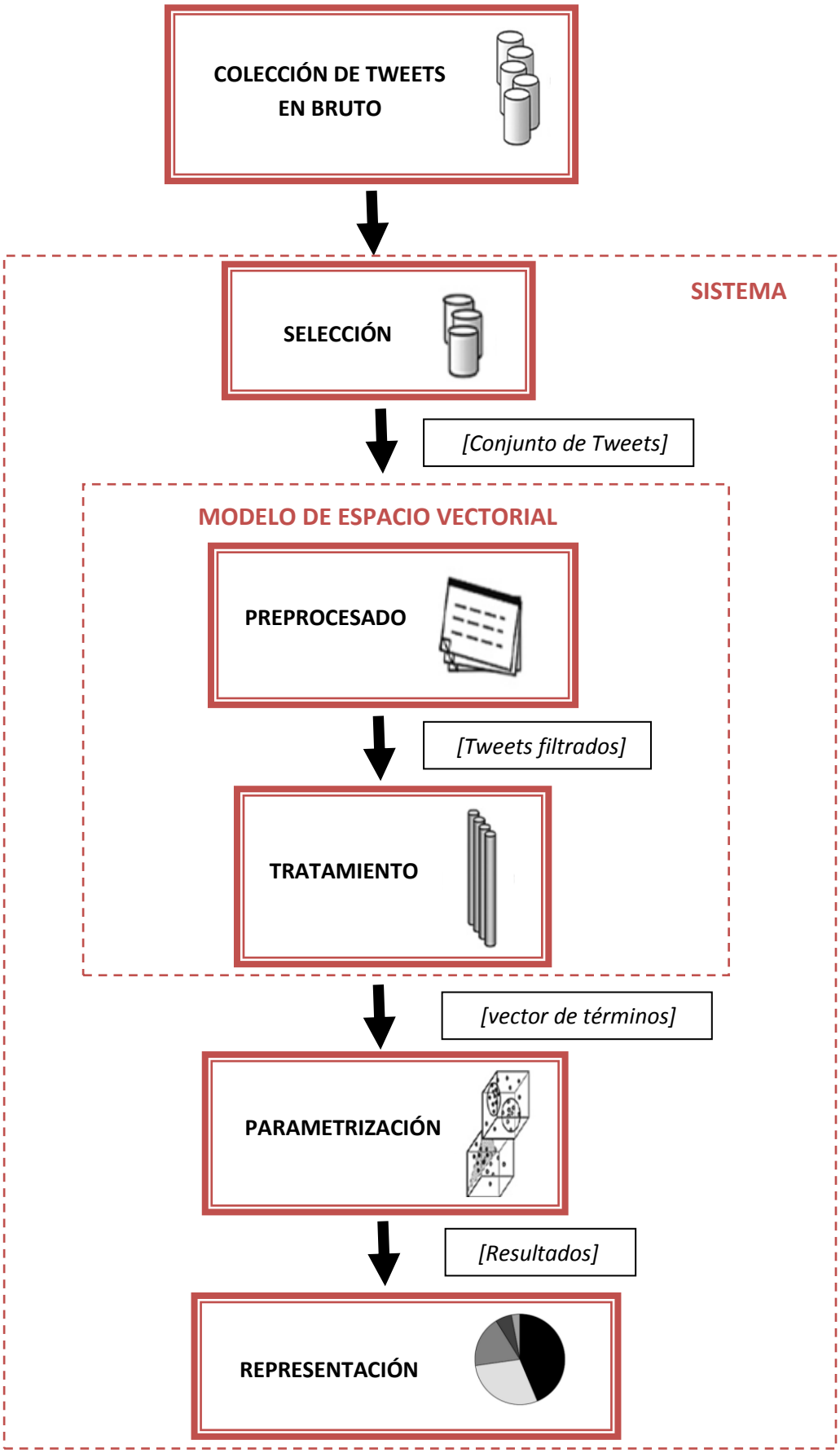

Figura 16. Estructura general del sistema de detección de Trending Topics

Concretamente, nuestro sistema tiene como punto de partida el análisis de un conjunto de tweets ya dado, por lo que el proceso de búsqueda y selección de Tweets a partir de una colección de tweets en bruto, es algo externo a la estructura propia del sistema implementado. No obstante, la fase de búsqueda y selección de tweets es algo totalmente necesario para su posterior tratamiento.

Dependiendo de los requerimientos del algoritmo empleado, puede que la estructura del programa incorpore alguna etapa adicional. Sin embargo, el diagrama recién mostrado representa la base primordial de la detección de Trending Topic en Twitter.

En los posteriores apartados describiremos la funcionalidad de cada una de las fases del sistema, especificando el tipo de datos que intercambian cada uno de los módulos, tal y como hemos adelantado en el diagrama anterior.

3.1.2 RECOLECCIÓN Y SELECCIÓN DE TWEETS

En esta fase se capta el conjunto de tweets a evaluar. Nuestro sistema no implementa dicha funcionalidad. Ha sido diseñado para comenzar el análisis a partir de un conjunto de tweets ya dado. No obstante, llamando directamente a la API de streaming de twitter *(https://dev.twitter.com/docs/api/1.1/post/statuses/filter)* se pueden obtener conjuntos de tweets de ciertas características.

1.2.1 Formato de los Tweets

Antes de tratar el conjunto de mensajes capturados, deberemos tener claro cuál es su formato. Cada tweet lo hemos definido en una línea, siguiendo la estructura definida en la siguiente figura.

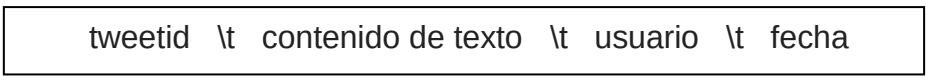

Figura 17. Formato de los tweets

Podemos identificar 4 campos tabulados dentro de cada mensaje:

- *Tweetid*: Identificador del tweet. Etiqueta numérica que identifica a cada tweet.
- *Tweet*: Mensaje emitido. Hace referencia al propio contenido del tweet
- *Usuario*: Usuario que ha publicado el tweet
- *Fecha*: Fecha de emisión

En la siguiente figura se muestra un ejemplo en donde se aprecian cada uno de los campos según el formato mencionado.

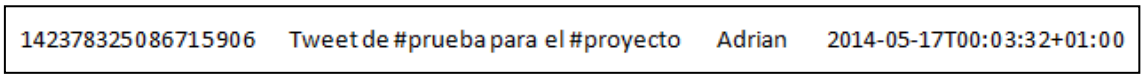

Figura 18. Tweet de ejemplo según el formato requerido

De esta forma, el fichero de tweets de partida, incluirá tantos tweets como mensajes haya con el formato especificado.

3.1.3 PREPROCESADO

El preprocesado de los datos adquiere una gran importancia al tratar con mensajes de Twitter. Partiendo de la amplia temática que se trata en esta red social y del propio formato de la mensajería, hay que enfrentarse a mensajes con poca información textual por un lado, y con baja calidad por otro, en la que los errores ortográficos y usos de signos de exclamación, entre otros, están a la orden del día. Por ello, la misión de este módulo se centra en normalizar el texto de entrada, eliminando todos los términos carentes de significado, dando lugar un texto totalmente plano y limpio, listo para ser procesado.

De esta forma, el modulo de preprocesado en general, recibe como parámetro de entrada el conjunto de tweets que se desea analizar.

En nuestro sistema, este módulo se ha diseñado con un nivel de abstracción mayor; en vez de recibir todo el conjunto de tweets a la vez, recibe solamente tweets individuales, y en general, cualquier fragmento de texto. Internamente el módulo se ha configurado para que procese en una iteración un único texto de entrada. De esta forma, a partir de un tweet de entrada, se obtiene como salida su versión totalmente filtrada.

Para ello, además de las propias operaciones típicas del preprocesamiento de textos, se ha teniendo en cuenta el formato que ofrece Twitter en sus mensajes, incluyendo el tratamiento de alguna de sus prestaciones.

Las distintas operaciones implementadas para la normalización del tweet de entrada, son las siguientes:

- Eliminación de menciones a usuarios (@). Algo muy común en Twitter es mencionar a algún usuario en el propio contenido del tweet.
- Eliminación de URL. En ocasiones los usuarios hacen referencia a diversas url en el contenido de sus mensajes, como páginas externas, vídeos de youtube, etc.
- Eliminación de expresiones horarias, numéricas y fechas.
- Paso a minúsculas.
- Eliminación de los signos ortográficos más comunes, como el acento agudo y grave. Además se trata el carácter "ñ", símbolo inexistente en el código ASCII.
- Eliminación de cualquier símbolo o carácter distinto a letras o números, a excepción del símbolo "#", utilizado como símbolo especial para los hashtag.
- Eliminación de las *stopwords*. Se ha escogido una lista de palabras de parada tanto en español ([66] y [67]) como en inglés [68]. De esta forma nuestro sistema aumenta su eficiencia en el tratamiento de tweets en inglés.
- Lematización. Se ha incorporado un cliente en PHP [69] al que se le han añadido algunos cambios para adaptarlos a nuestro sistema.

Las siete primeras operaciones conforman el concepto propiamente de filtrado. La lematización es un proceso extra que permite la reducción de los términos a su forma canónica, con la idea de mejorar la detección de términos semejantes con distintas terminaciones.
Nuestro sistema ofrece al usuario la opción de elegir si quiere por un lado filtrar el tweet de entrada, y por otro, hacer uso del lematizador, pero en cualquier caso, siempre se realiza un filtrado muy básico que elimina ciertos caracteres ASCII. De esta manera, se pueden observar los efectos de aplicar o no una fase de preprocesado.

Así pues, el tweet filtrado de salida estará formado únicamente por los términos representativos del contenido del tweet origen de entrada.

3.1.4 TRATAMIENTO

El módulo de tratamiento, recibe el tweet totalmente filtrado de la fase anterior, para obtener como salida un vector de términos representativos. Para ello es

necesario reconocer cada uno de los términos del tweet normalizado de entrada.

En Twitter, normalmente se comenta que un determinado hashtag se ha convertido en Trending Topic. Sin embargo, la lista de tendencias también incorpora palabras como Trending Topic, tal y como se observa en la siguiente figura.

Figura 19. Ejemplo de una lista de tendencias en Twitter

Esto quiere decir, que a la hora de detectar TT en Twitter hay que contemplar ambas situaciones, teniendo en cuenta tanto palabras como hashtags.

Este hecho, es el fundamento del módulo de tratamiento. Nuestro sistema se ha diseñado para detectar hashtags, palabras o ambas a la vez. El usuario puede elegir qué tipo de detección quiere emplear. Según la elección, el vector de salida variará. Si se elige la detección de hashtag por ejemplo, el vector de salida solo contendrá los hashtag existentes en el tweet filtrado de entrada.

Esto permite realizar el análisis desde distintas perspectivas, posibilitando un estudio más detallado y completo

3.1.4.1 Formato de los Hashtags

Como ya sabemos, la identificación de un hashtag en un tweet se consigue a través del reconocimiento del símbolo “#”. Cualquier palabra precedida por dicho término será un hashtag.

Ahora bien, existen varios formatos en los que los hashtag pueden aparecer. Tener bien identificadas todas las posibles configuraciones nos ayudará en la detección de TT.

A continuación, se describen los distintos formatos que pueden adquirir los hashtag y que necesitaremos detectar:

- *Hashtag individual*

  Es el formato más simple, una palabra o expresión precedida por el símbolo "#". Por ejemplo:

  "El #PLN en Twitter y sus #aplicaciones"

  Apreciamos la aparición de dos hashtag (#PLN y #aplicaciones) con formatos individuales.

- *Secuencia de N hashtag contiguos*

  Podemos encontrarnos una secuencia de *n* hashtag anidados y contiguos sin ningún carácter de espaciado entre ellos. Cualquier palabra que se añada sin espaciado al final de un hashtag, se considera partícipe de dicho hashtag. Sin embargo, la anidación de varios hashtag adyacentes se considera como una secuencia de hashtag individuales. Esto se debe a que el contenido de un hashtag no puede incluir el símbolo "#". Por ejemplo:

  "El PLN y Twitter #proyecto#estudio#trabajo"

  En este caso podemos distinguir una secuencia de 3 hashtag anidados sin espaciado entre ellos, dando lugar a 3 hashtag distintos en el tweet, equiparados como 3 hashtag individuales.

  Sin embargo, en la siguiente situación solo identificaríamos un hashtag:

  "El PLN y Twitter #proyectoestudiotrabajo"

  El único hashtag de este tweet sería *#proyectoestudiotrabajo*. Una de las ventajas de la anidación de hashtag es el ahorro de caracteres de espaciado, lo que se traduce en una mayor disponibilidad de caracteres totales.

- *Hashtag individual precedido por una expresión*

  Por regla general, la opción del hashtag individual es la más común. Sin embargo es necesario distinguir el caso en el que un hashtag está precedido, sin ningún carácter de espaciado, por una palabra o expresión. Por ejemplo, el siguiente tweet:

  "Detección de TT en Twitter#proyecto"

En este caso, identificamos un único hashtag (#proyecto) precedido por una palabra (Twitter). Distinguir esta situación será de vital importancia cuando se detecten tanto palabras como hashtag.

- *Secuencia de N hashtag precedidos por una expresión*
  Al igual que con un hashtag individual, podemos encontrarnos una secuencia de *n* hashtag anidados y contiguos, precedidos, sin ningún carácter de espaciado, por una palabra o expresión. Por ejemplo:

  "Detección de TT en Twitter#proyecto#estudio#trabajo"

  En esta situación, identificamos una secuencia de 3 hashtag contiguos, pero precedidos, sin espaciado, por una palabra. Al igual que en el caso anterior, esto será esencial cuando se detecten tanto palabras como hashtags.

La siguiente tabla muestra un resumen de los distintos formatos que pueden adquirir los hashtag en un tweet.

FORMATO	EJEMPLO
Hashtag individual	#Twitter
Secuencia de n hashtag contiguos	#Twitter#PLN#Proyecto#. . .
Hashtag individual precedido por una expresión	Twitter#PLN
Secuencia de n hashtag contiguos precedidos por una expresión	Twitter#PLN#Proyecto#. . .

Tabla 2. Formatos de aparición de los hashtag

La unión de la funcionalidad del módulo de preprocesado junto con el de tratamiento, emula el comportamiento del Modelo de Espacio Vectorial mostrado previamente en el diagrama del sistema. En este punto, los datos están totalmente listos para ser procesados.

3.1.5 PARAMETRIZACIÓN

El módulo de parametrización implementa cualquier mecanismo cuya finalidad sea la evaluación de los términos representativos recibidos. Para ello hemos incorporado diversos métodos y algoritmos que nos permiten tratar los datos desde

diferentes puntos de vista.

El proceso de parametrización consta de dos fases de distinta temática: una primera en la se hace uso del método de n-gramas y una segunda contigua en la que se aplican algoritmos puramente estadísticos.

A continuación se abordan cada uno de ellos.

3.1.5.1 N-gramas

La posibilidad de aplicar el método de n-gramas al vector de términos representativos recibidos del módulo anterior (que recordemos que pueden ser palabras, hashtag o ambas), permite realizar un mayor análisis del contenido de cada tweet, permitiéndonos identificar posibles coapariciones de términos.

Nuestro sistema ofrece al usuario la posibilidad de confeccionar n-gramas desde 1 hasta **5** términos.

Por regla general, este método actúa como una ventana deslizante de tamaño *n* mediante la que se consigue combinaciones de *n* términos vecinos. Nuestro sistema va más allá, y aplica la combinatoria para buscar todas las posibles combinaciones sin repetición de *n* términos, entre el total de *m* términos del vector recibido. Básicamente realizamos combinaciones de *m* elementos tomados de *n* en *n*, siendo $m \geq n$. Según esto, el número distinto de combinaciones según el n-grama elegido, se rige por la ecuación (13).

$$\binom{m}{n} = \frac{m!}{n!\,(m-n)!} \tag{13}$$

Observamos que a medida que tratemos con vectores de términos representativos de mayor tamaño *m*, el número de n-gramas posibles va aumentando. Esto ya es un indicio del esfuerzo de procesamiento que requerirá la aplicación del método de n-gramas.

Este modelo se caracteriza por realizar combinaciones de términos sin repeticiones, en donde el orden de los mismos no importa.

Dado el siguiente tweet, por ejemplo; “Red social Twitter”, si queremos buscar los bigramas existentes (2-gramas), obtendremos 3 posibles combinaciones distintas (“red social”, “red Twitter” y “social Twitter”).

Según nuestro concepto e implementación, el bigrama “red Twitter” únicamente aparece una vez en el tweet de ejemplo. Esto se debe a que consideramos el n-grama como una unidad con significada propio, obviando todas las posibles combinaciones de los términos internos que forman dicho n-grama.

Por otro lado, el orden de los términos no se tiene en cuenta. Según el siguiente par de tweets: “red social Twitter” y “Twitter es la red más famosa”, el bigrama “red Twitter” aparece un total de 2 veces en el conjunto de los dos mensajes.

En el primer tweet, distinguimos “red Twitter”, mientras que en el segundo distinguimos “Twitter red”. Es en este sentido por el que el orden del bigrama no importa, ya que ambos se consideran iguales. Este fenómeno deberemos tenerlo en cuenta a la hora de evaluar la frecuencia de aparición de los distintos n-gramas en el conjunto de tweets.

Hay que destacar que la aplicación de unigramas (1-gramas) es equivalente a representar cada término individualmente, tal y como los obteníamos a la salida del módulo de tratamiento, por lo que en esta situación no se requerirá ningún tipo de procesamiento.

Así pues, el método de n-grama, recibe como entrada el vector de términos representativos del módulo anterior, dando lugar como salida un vector en el que cada posición se corresponde con cada una de las combinaciones según el n-grama elegido.

Sin embargo se ofrece también la posibilidad de calcular los n-gramas mediante el método común de la ventana deslizante, aumentando así las prestaciones del sistema.

Una vez explicado el método n-grama utilizado y antes de abordar el conjunto de algoritmos de carácter estadístico, es necesario presentar una nueva funcionalidad de nuestro sistema, mediante la cual se da la opción al usuario de agrupar todos los términos y por lo tanto, los n-gramas definidos, según distintos formatos de fecha.

3.1.5.2 Elección del formato fecha

En Twitter, el número de tweets emitidos en un intervalo temporal reducido, puede ser altísimo y más aún, cuando diversos temas están a la pugna por convertirse en TT. En esos momentos la cantidad de tweets que fluyen por la red es enorme.

Nuestro sistema, detecta TT en con un conjunto de tweets de origen, tal y como

hemos visto previamente. El número de tweets de ese conjunto puede variar según la cantidad de mensajes que queramos analizar. A pesar de que podamos capturar un gran número de tweets, puede ocurrir en ciertos casos que no dispongamos del número suficiente de tweets por unidad temporal mínima como para emular el flujo real de tweets en Twitter.

Se puede dar la situación de que en ciertos minutos por ejemplo, no tengamos ningún tweet emitido, lo que dificulta indirectamente la detección de TT. Por este motivo, se ha implementado una funcionalidad que permite al usuario agrupar los términos según distintos formatos de fecha. Cuando nos referimos a términos, hacemos referencia a los posibles n-gramas, que en caso de unigramas se corresponden directamente con los términos identificados.

El formato elegido será la unidad mínima temporal para la representación y análisis de tweets. De esta forma, si por ejemplo no tenemos suficientes tweets emitidos por minuto, podemos agruparlos por horas, permitiendo así obtener una mayor cantidad de términos por unidad temporal.

Los distintos formatos de fecha disponibles se muestran en la siguiente tabla.

FECHA	FORMATO
Año	'Y'
Año-Mes	'Y-m'
Año-Mes-Día	'Y-m-d'
Año-Mes-Día/Hora	'Y-m-d/H'
Año-Mes-Día/Hora:Minuto	'Y-m-d/H:d'
Año-Mes-Día/Hora:Minuto:segundo	'Y-m-d/H:i:s'

Tabla 3. Formatos fecha

El formato propio de Twitter es aquel en el que se especifica desde el año hasta el segundo en el que se ha emitido el tweet.

Una gran parte de los algoritmos empleados, analizarán los datos según el formato fecha elegido por el usuario.

En los siguientes apartados introduciremos cada uno de ellos y explicaremos su funcionalidad.

3.1.5.3 TF_i

El algoritmo TF_i (del inglés, "*Term* frequency") básicamente calcula la frecuencia de aparición de cada término en el conjunto de tweets, considerándolos como un único documento.

El índice *i* hace referencia a cada uno de los diferentes términos del vector de n-gramas obtenido en la fase anterior, configurado según el formato fecha elegido por el usuario.

Para el cálculo de la frecuencia de los distintos n-gramas disponibles, (a excepción del unigrama, en el que directamente se calcula la frecuencia de aparición individual de cada término) no se tiene en cuenta el orden interno de representación del propio n-grama. Es decir, tal y como adelantamos en el apartado anterior, un tweet que contenga el bigrama "A B" y otro que contenga el bigrama "B A", se considerarán el mismo bigrama. De esta forma al aplicar el algoritmo TF_i obtendremos que el bigrama "A B" (o "B A") aparece un total de dos veces en el conjunto de tweets.

Si tuviésemos en cuenta el orden de los términos internos del n-grama, según el ejemplo anterior, el bigrama "A B" sería distinto del bigrama "B A" y por lo tanto, aparecerían una única vez cada uno.

Esto implica que en cada n-grama deberemos comprobar las distintas combinaciones de los términos que lo forman. El número diferente de combinaciones de cada n-grama viene dado por *n!* (n factorial, en donde *n* es el tamaño del n-grama), por lo que a medida que se formen n-gramas de mayor número de términos, el nivel de complejidad en cuanto al cálculo de la frecuencia aumenta considerablemente.

Una vez que se ha obtenido la frecuencia global de cada n-grama a nivel de documento (todo el conjunto de tweets), se devuelve un vector en el que cada posición indica el n-grama identificado y su frecuencia. Este vector estará ordenado de mayor a menor frecuencia, lo que facilitará la identificación de los n-gramas más frecuentes.

3.1.5.4 TF_{ij}

El algoritmo TF_{ij} es muy similar al anterior. La única diferencia reside en la concepción de qué es un documento. El algoritmo TF_i consideraba todo el conjunto de tweets como un único documento, mientras que el método actual no. Este método se

define según la ecuación (14), en donde n es la frecuencia del término i en el documento j.

$$tf_{i,j} = n_{i,j} \tag{14}$$

El índice j hace referencia a cada uno de los documentos del conjunto de tweets. Cada documento estará definido por el formato fecha que el usuario haya elegido. Si por ejemplo, el usuario ha elegido la configuración de fecha detallada hasta los minutos, tendremos tantos documentos como minutos de distintas fechas haya en el conjunto de tweets. Es decir, todos los n-gramas de los tweets emitidos en una misma fecha exacta (en este caso en el mismo minuto), se tratarán como un único documento, y por lo tanto, serán agrupados en un mismo vector.

Si por ejemplo, según el formato de tweets explicado en apartados anteriores, tenemos los siguientes tweets:

1	Tweet de #prueba	Adrián	2014-05-19T15:30:13+01:00
2	Nueva #prueba	Adrián	2014-05-19T15:31:00+01:00
3	Otra #prueba	Adrián	2014-05-19T15:31:50+01:00

Si elegimos el formato detallado hasta los minutos, podremos identificar dos documentos: uno que incluye el primer tweet y otro que incluye los dos restantes. Esto se debe porque según el formato fecha elegido, los dos últimos tweets han sido emitidos en la misma fecha (en el mismo minuto). Si hubiésemos elegido el formato fecha detallado hasta las horas, únicamente tendríamos un documento, ya que todos los tweets fueron emitidos en la misma hora.

De esta forma, el algoritmo calcula la frecuencia de aparición de un determinado n-grama en cada uno de los distintos documentos definidos según el formato fecha elegido. Esto permite observar cómo evoluciona la frecuencia de un n-grama a lo largo de cada uno de ellos.

Como lo más lógico es aplicar este algoritmo a los n-gramas más frecuentes, se ofrece al usuario la posibilidad de elegir el número n-gramas más populares que quiere estudiar. Para ello, este algoritmo recibe como entrada el vector de salida del algoritmo TF_i, permitiendo identificar cuáles son los n-gramas más frecuentes.

Así pues, este algoritmo nos facilita la frecuencia de aparición de un cierto número de n-gramas populares en cada uno de los documentos existentes en el total de tweets.

3.1.5.5 TF – IDF

En este caso, en vez de ponderar la puntuación con el término TF_{ij}, hacemos uso del término TF_i, lo que nos ofrece una visión más global sobre el carácter TT de un n-grama en todo el conjunto de documentos (definidos según el formato fecha elegido por el usuario). De esta forma, la puntuación se define según la ecuación (15).

$$TF - IDF = tf_i * idf_i \quad (15)$$

Así pues, el algoritmo TF – IDF no solo tiene en cuenta la frecuencia de aparición global de un n-grama, si no también cómo está distribuida a lo largo de todos los documentos. Esto obliga a que la mayor parte se concentre en unos pocos documentos, adquiriendo así un comportamiento inesperadamente popular. De esta forma, aquellos n-gramas muy frecuentes, que a priori tenían mayor probabilidad de ser TT, no tienen por qué ser tendencia.

Al igual que antes, al usuario se le ofrece la posibilidad de elegir el número de n-gramas más populares que quiere estudiar mediante el presente algoritmo, permitiéndole obtener la puntuación de cada uno de ellos.

3.1.5.6 NTF

El algoritmo NTF (del inglés, "Normalized Term Frequency") propone un procedimiento complementario para evaluar la frecuencia de un determinado n-grama en un periodo de tiempo concreto.

Nuestro sistema, ofrece al usuario la posibilidad de elegir varios parámetros. Por un lado, y en primer lugar, el usuario deberá especificar el n-grama que quiere estudiar.

Por otro lado, se le permite determinar tanto la fecha de comienzo y fin (según el formato que especifique) en donde se quiere realizar el análisis. El periodo comprendido entre ambas fechas será el periodo donde se aplicará el algoritmo. El formato de ambas fechas se deberá corresponder con fechas en la que se haya emitido algún tweet. De no ser así se deberá elegir otra de nuevo.

Por último se le da la opción de elegir el tamaño de la ventana. El tamaño hace referencia al número de tweets que se tienen en cuenta alrededor del instante temporal *t* definido por la fecha de comienzo establecida. A medida que vayamos

evaluando cada uno de los tweet, la ventana se irá desplazando, siempre centrada en el instante temporal marcado por la fecha del tweet que se está evaluando. Por este motivo se requiere que la ventana definida sea un número impar.

Así pues, este método nos ofrece otra posibilidad a la hora de observar el carácter TT de un determinado n-grama. Evaluar qué porcentaje de tweet contienen un cierto n-grama dentro de una determinada ventana con respecto al total de tweets, es una manera de saber el nivel de popularidad que alcanza dicho n-grama durante un concreto periodo temporal.

3.1.6 REPRESENTACIÓN

Tras el proceso de parametrización, como acabamos de ver en el apartado anterior, obtendremos una serie de resultados sobre el comportamiento de determinados n-gramas en el conjunto de tweets.

El módulo de representación nos permite disponer de manera visual todos esos datos, pudiendo así posteriormente realizar un análisis mucho más directo. Además, reflejaremos también el tiempo de ejecución de nuestro sistema.

De manera genérica exportaremos automáticamente todos los resultados a ficheros

Excel. Esto nos permitirá obtener una serie de tablas con todos los resultados que posteriormente podremos representar gráficamente. El contenido de las tablas variará según el método que se use.

3.2 IMPLEMENTACIÓN

Tal y como hemos explicado en el sección anterior, nuestro sistema está basado en módulos. En vez de diseñarlo como un único bloque, se ha optado por estructurarlo en distintos módulos, que unidos satisfacen todos los requisitos funcionales que buscamos. Cada módulo implementa una determinada funcionalidad acorde a cada una de las fases por la que tienen que pasar los datos, permitiendo así una mejor estructuración de la arquitectura del sistema.

En la presente sección, por un lado, abordaremos la implementación concreta de cada uno de los módulos. Explicaremos la estructura del código de cada una de las

funciones que nos permiten desarrollar las funcionalidades de cada uno de los módulos. Sin entrar en particularizaciones de cómo se ha optado implementar el propio código, comentaremos cuando sea necesario detalles del código de las funciones, facilitando la comprensión de nuestro sistema.

Tras ello y por último, explicaremos la configuración general de los distintos sistemas planteados para realizar los diversos análisis del conjunto de tweets. Aquí ilustraremos qué estructura concreta adopta cada sistema, y cómo se ha implementado la conexión de los distintos módulos en cada uno de ellos.

3.2.1 MÓDULOS

En este apartado detallaremos en mayor medida cada uno de los módulos que forman nuestro sistema. Explicaremos las funciones que lo componen y cómo se han estructurado y diseñado cada una de ellas. Esto permitirá tener una mejor visión y comprensión de la configuración de nuestro programa. A continuación se desarrollan los distintos módulos.

3.2.1.1 Módulo de preprocesado

El módulo de preprocesado se ha implementado en la clase *Modulo_preprocesado.php*
Esta clase está formada por 2 funciones principales: una de preprocesado y otra de lematización. Esta última a su vez hace uso de una función auxiliar *(function sendPost)*

A. Función preprocess

La función preprocesado *(function preprocesado)* recibe 3 parámetros de entrada, tal y como sigue en el orden indicado:

1. Variable de tipo string *($tweetIN)* con el contenido del tweet
2. Variable de tipo entero *($filtered)*, indica el deseo de filtrar o no el tweet (*$filtered* = 1 si se filtra, *$filtered* = 0 no se filtra)
3. Variable de tipo entero *($lemmatizer),* indica el deseo de lematizar o no el tweet (*$lemmatizer* = 1 si se lematiza, *$lemmatizer* = 0 no se lematiza)

Como salida la función devuelve un string que cumplirá con las características impuestas por los parámetros de entrada.

A lo largo de la función se realizan diversas operaciones de filtrado sobre el tweet de entrada. En algunas ocasiones será necesario disponer del string completo, mientras que en otras será necesario detectar cada término individualmente. Por este motivo, según las necesidades, iremos tratando el tweet de la forma requerida en cada momento.

Para ello hacemos uso de dos funciones propias de php: explode e implode. La función explode realiza conversión de string a array. Según nuestra configuración ($words = *explode(" ", $tweetIN)), d*evuelve un array con tantas posiciones como términos separados por carácter de espaciado tenga el tweet.

La función implode, con un comportamiento contrario, convierte de array a string. Según nuestra configuración ($tweetIN = *implode(" ",$words)),* devuelve un string en el que cada término (que se corresponde con cada una de las posiciones del array) está separado por un carácter de espaciado.

En la siguiente figura se muestra la estructura general del sistema de preprocesado. Entre paréntesis especificamos las variables sobre las que se realiza las distintas operaciones.

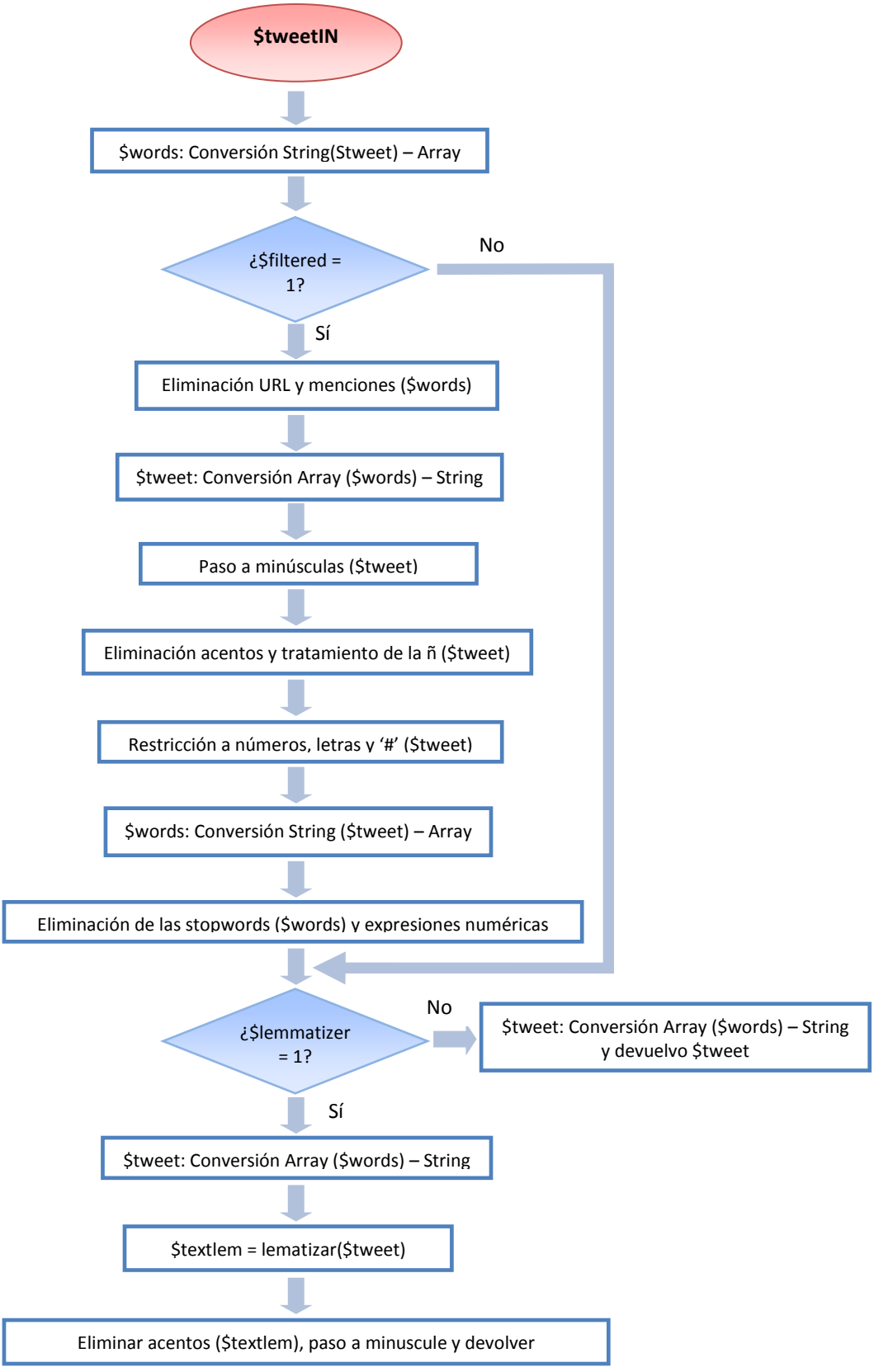

Figura 20. Diagrama de flujo de la fase de preprocesado

El la diagrama recién mostrado podemos apreciar las distintas operaciones que han sido implementadas. Cabe mencionar las siguientes aclaraciones:

- Eliminación de URL y menciones a usuarios. Para ello se recorre cada uno de las posiciones del array *$words*. Para la identificación de las menciones buscamos que el primer carácter del término sea '@'. Para la detección de URL buscamos que los cuatro primeros caracteres sean 'http' (de esta forma podremos identificar también URL con conexión segura, 'https'). Si se detecta cualquiera de ambos casos, el término en cuestión será eliminado.
- Paso a minúsculas mediante el comando *strtolower($tweet).*
- Eliminación de acentos. Se eliminan tanto acentos agudos como graves. Además se sustituye la ñ por la n. Para ello se hace uso de la función *str_replace*, ya comentada anteriormente.
- Restricción a números, letras y el símbolo '#' (reservado para los Hashtags) Se elimina cualquier carácter distinto a los mencionados. Para ello se hace uso de la función *ereg_replace* mediante la definición de una expresión regular, tal y como sigue:

  $tweet = ereg_replace("[^a-z0-9#]", " ", $tweet);

- Eliminación de las stopwords. Para ello será necesario recorrer cada una de las posiciones del array $tweet. Por cada posición se deberá comprobar si se trata o no de alguna de las stopwords definidas en el array $stopwords (array en el que cada posición contiene una stopword).
- Llamada al lematizador. Para ello se hace uso de una función auxiliar ya implementada *(function lematizar)* que explicaremos luego. Esta función devuelve caracteres en mayúscula y con acentos, por lo que es necesario realizar un filtrado posterior.

Un aspecto muy importante es el orden de aplicación de las distintas operaciones de preprocesado. El orden implementado ha sido diseñado concretamente para que cada operación no interfiera en la operación contigua. Esto quiere decir que una operación no puede provocar un mal comportamiento en operaciones futuras.

Por ejemplo, si eliminamos cualquier carácter distinto de letras y número antes de la eliminación de menciones y URL, estaríamos cometiendo un grave error. Por un lado no reconoceríamos las menciones ya que habríamos borrado el carácter '@'. Por otro

lado, no podríamos eliminar correctamente las URL, ya que al eliminar el carácter '/', entre otros, estaríamos borrando sólo una parte de ella.

De la misma forma, si detectamos las stopwords antes de realizar el paso a minúsculas y eliminar los acentos. La lista de las stopwords configura una lista de términos en minúsculas y sin acentos. Si obviamos esto, no identificaríamos muchas de las stopwords de los tweets.

Por este motivo seguir un orden correcto es fundamental.

De esta forma, el pseudocódigo de la función de preprocesado quedaría tal y como se observa en la siguiente figura.

```
$stopwords = array con las palabras de parada;
$words = conversión string ($tweetIN) - array;

if $filtered == 1
        for cada término t en $words do
                Elimino URL y menciones
        end for

        $tweet = conversión array ($words) - string;
        Paso a minúsuculas $tweet;
        Eliminación de acentos y sustitución de ñ por n en $tweet;
        Restricción a números, letras y '#' en $tweet;

        $words = conversión string ($tweet) - array;
        for cada término t en $words do
                for cada término x en $stopwords do
                        Elimino palabras de palabra
                        Elimino expresiones numéricas
                end for
        end for
end if

if $lemmatizer == 1
        $text = conversion array ($words) - string;
        $textlem = lematizar($text); //Llamada al lematizador
        Eliminación de acentos y filtrado de caracteres de $texlem;

        return $textlem
end if

$textfilter = conversion array ($word) - string
return $textfilter
```

Figura 21. Pseudocódigo de la función de preprocesado

Tras la eliminación de signos de puntuación, puede que determinados términos sean solamente numéricos (por ejemplo la expresión horaria 14:00 a la que se le ha eliminado los dos puntos). Estos términos carecen de significado propio. Por ello al mismo tiempo que eliminamos las stopwords desechamos todas las expresiones totalmente numéricas. Para ello se hace uso de la función *is_numeric* de PHP que detecta si un término está formado únicamente por números.

En la siguiente figura se muestra un ejemplo del funcionamiento de esta fase.

Texto entrada: "El #EJEMPLO muestra la funcionalidad de esta fase en el @proyecto "
Texto de salida sin lematización: "#ejemplo muestra funcionalidad fase"

Figura 22. Ejemplo función preprocesado

B. Función lemmatizer

Para poder aplicar la lematización a los tweets, se ha incorporado a nuestro sistema un cliente PHP [69] que implementa dicha funcionalidad. El código adquirido se ha introducido dentro de esta función *(function lemmatizer)*, permitiendo así realizar llamadas al programa siempre que se quiera aplicar este proceso. Para ello, la función recibe un único parámetro de entrada:

1. Variable de tipo string *($textIN),* contiene el mensaje a lematizar

Como parámetro de salida, se devuelve el mismo texto de entrada pero con cada uno de los términos lematizados.

La configuración inicial del programa mostraba por pantalla el resultado de la lematización. Como en nuestro sistema esta funcionalidad no está aislada, sino que forma parte de un proceso, se han tenido que añadir ciertos cambios para acoplarlo según nuestros requerimientos.

Para ello cada uno de los términos lematizados se guardan en un array, que posteriormente convertiremos a string para ser devuelto. Sin embargo, antes de eso, hay que tener en cuenta el trato de los hashtags por parte del lematizador. Este separa el contenido del hashtag del símbolo '#', tratándolos como términos diferentes, lo que provoca que no tengamos bien identificado en una única posición del array el hashtag.

Para solucionarlo, una vez lematizado todo el texto de entrada, se recorre y se busca el símbolo '#'. En caso de que exista, el término siguiente a la posición en la que se ha encontrado la almohadilla será el hashtag. Uniendo ambos términos y borrando la posición en la que se encontraba solo el contenido del hashtag, incorporaremos en una única posición el hashtag completo. Es decir, en la posición donde se encontraba el carácter '#', añadimos el hashtag de la siguiente posición antes de borrar su respectivo contenido.

De esta forma obtenemos un array en el que coexisten tanto términos como hashtag lematizados.

Cabe señalar que el lematizador puede devolver más de un término lematizado según la palabra evaluada. Sin embargo, los hashtags en su mayoría son expresiones únicas que no tienen ningún tipo de forma canónica concreta, por lo que el resultado de la lematización es directamente el mismo hashtag. En muy pocas ocasiones un hashtag tiene varios términos lematizados. Por este motivo solamente tenemos en cuenta el término contiguo al carácter almohadilla.

Continuando con el ejemplo antes mostrado, el resultado de aplicar la lematización se muestra en la siguiente figura.

Texto entrada: "El #EJEMPLO muestra la funcionalidad de esta fase en el @proyecto "

Texto de salida con lematización: "#ejemplo mostrar funcionalidad fase"

Figura 23. Ejemplo función preprocesado con lematización

Por último, simplemente convertimos el array a string. Antes de devolverlo como parámetro de salida, se eliminan los caracteres de espaciado sobrantes, para cubrir aquellas situaciones en las que se haya borrado el contenido de alguna posición del array para obtener las expresiones correctas de los hashtag.

3.2.1.2 Módulo de tratamiento

El módulo de tratamiento se ha implementado en la clase *Modulo_tratamiento.php*

Este módulo está formado por una única función, encargada de la detección y tokenización de todos los términos del tweet que recibe.

A. Función getTerm

La función getTerm *(function getTerm)* recibe como parámetros las siguientes variables:

1. Variable de tipo string *($tweetIN)* con el contenido del tweet
2. Variable de tipo entero *($allow_hash)*, permite detectar o no hashtag (*$allow_hash* = 1 se detectan hashtag, *$allow_hash* = 0 no se detectan)
3. Variable de tipo entero *($allow_words)*, permite detectar o no palabras (*$allow_word* = 1 se detectan palabras, *$allow_words* = 0 no se detectan)

Como parámetro de salida, la función devuelve un array en el que cada posición se corresponde con cada uno de los términos del tweet, según la configuración especificada.

El diagrama de flujo genérico de la función se puede resumir en la siguiente figura.

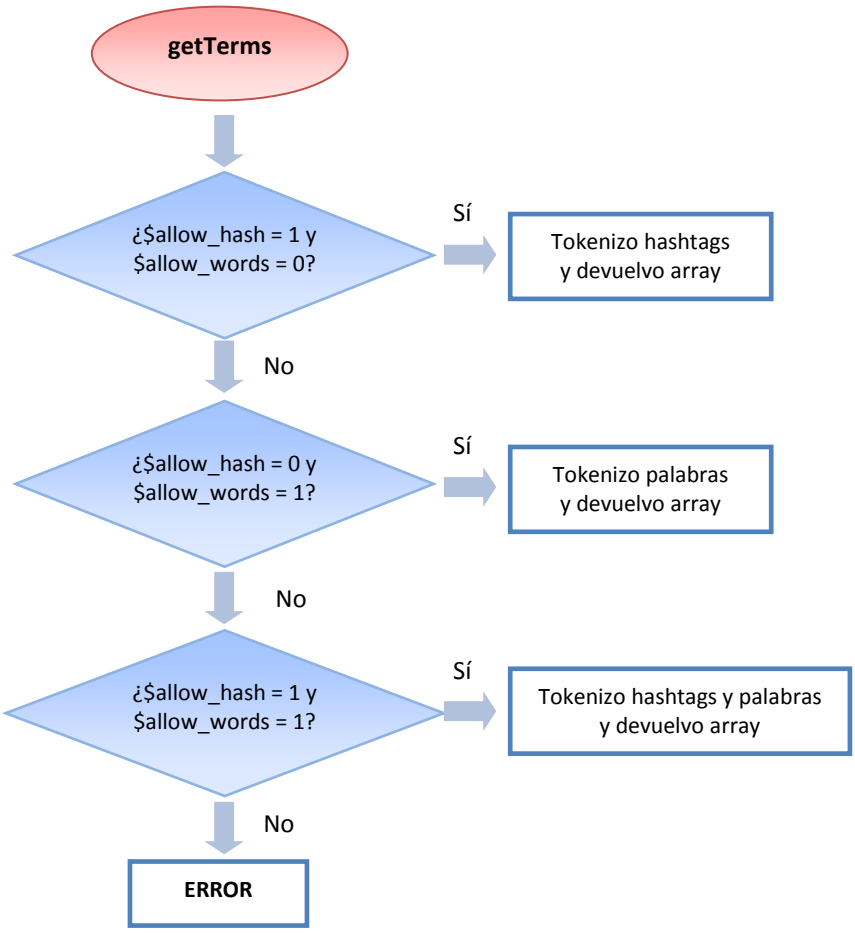

Figura 24. Diagrama de flujo de la función getTerms

Tal y como se aprecia en la estructura, se evalúan las diferentes combinaciones de las dos variables que nos indican el tipo de detección. En caso de que el usuario no quiera detectar nada, se devolverá un mensaje de error y se parará la ejecución. No tiene sentido ejecutar el sistema si no se detecta nada.

En cuanto al proceso de tokenización, necesitamos recorrer cada uno de los términos del tweet recibido y averiguar si se tratan de una palabra o un hashtag.

En el caso de los hashtags, tendremos que identificar todos los formatos previamente presentados en el apartado 3.1.3.1. Hay que tener en cuenta que los casos de una secuencia de n hashtag contiguos, de un hashtag individual precedido por una expresión sin carácter de espaciado y de una secuencia de n hashtag contiguos precedidos por una expresión sin carácter de espaciado, se identifican como un único término. Esto se debe a que la conversión de String a Array se realiza a través de la detección de los caracteres de espaciado entre los distintos términos. De esta forma, al aparecer cada formato sin ningún carácter de espaciado, se consideran como un único término, cuando en realidad contienen varios hashtags y alguna palabra también.

Para ello, lo primordial es la identificación del carácter '#'. La ausencia de este nos indica que el término se corresponde con una palabra. Además, en el caso que se identifique el carácter, será necesario también saber su posición dentro del término evaluado y cuántas veces aparece. Esto nos servirá para identificar cada uno de los formatos.

La estructura de detección y reconocimiento de hashtag y palabras, se rige por el diagrama mostrado en la siguiente figura.

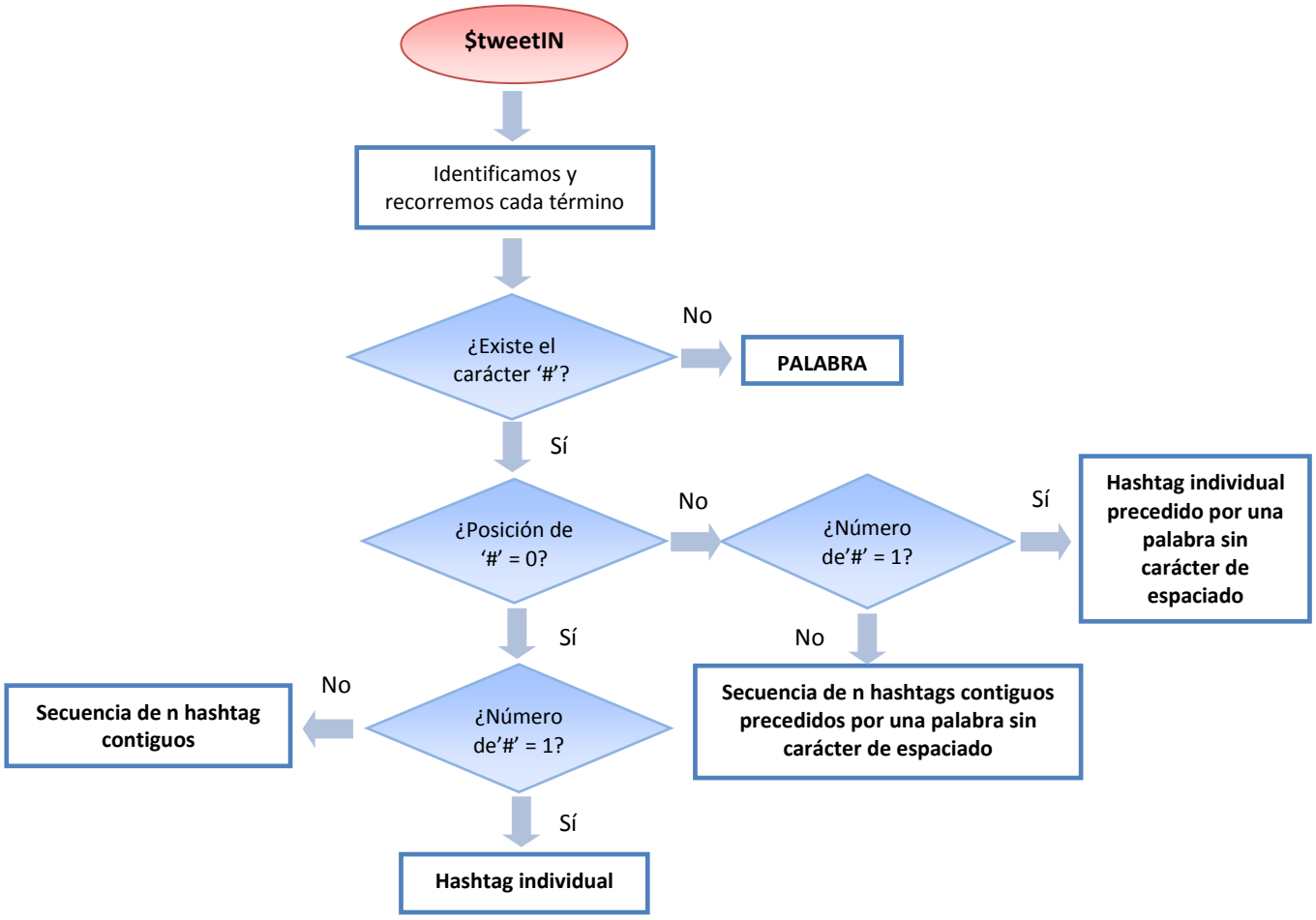

Figura 25. Diagrama de flujo en la detección de los distintos formatos de aparición de los hashtags

El número de apariciones del carácter '#' nos marcará el valor de n en los formatos que contienen secuencias de n hashtag contiguos, indicando cuantos hashtag adyacentes debemos detectar. Una vez detectados, para tokenizar simplemente dividimos la expresión según los distintos caracteres '#', obteniendo así un array con cada uno de los hashtags pero sin el símbolo '#'.

En la siguiente figura se muestra un ejemplo de cada uno de los distintos formatos.

Tweet: "#2#contiguo #individual palabra#individual palabra#2#contiguos"

Figura 26. Ejemplo de tweet con todos los formatos posibles de hashtags

En las ocasiones que dichas secuencias están precedidas por una palabra sin carácter de espaciado, hay que tener en cuenta que el primer término del array se trata de una palabra, por lo que dependiendo de las especificaciones del usuario, se deberá incluir o no. En la siguiente figura se observa la salida del método según los distintos tipos de detección según el ejemplo anterior.

. Hashtags: Array ([0] => 2 [1] => contiguo [2] => individual [3] => individual [4] => 2 [5] => contiguos)
. Palabras: Array ([0] => palabra [1] => palabra [2])
. Ambos: Array ([0] => 2 [1] => contiguo [2] => individual [3] => palabra [4] => individual [5] => palabra [6] => 2 [7] => contiguos)

Figura 27. Resultado ejemplo de la función getTerm según los tipos de detección

B. Función getHash_word

Función auxiliar encargada de buscar en un tweet palabras idénticas a los hashtags contenidos en él. Recibe dos parámetros de entrada:

1. Variable de tipo array ($comb) contiene los hashtags que definirán las palabras a buscar
2. Variable de tipo array ($tweet), contiene el tweet completo

Como parámetro de salida devuelve un array con los hashtags recibidos y las palabras idénticas a esos hashtags (en caso de existir). Para ello recorre cada una de las palabras del tweet de entrada y busca aquellas que sean idénticas a los hashtags contenidos en el tweet. En caso de no existir ningún hashtag, no busca ninguna palabra.

Se ha implemetando además otra función prácticamente similar (*getHash_word2*) que no incluye en el array de salida los hashtags de cada tweet procesado, ya que esta función se incluye en un sistema que previamente ya detecta y almacena los hashtags.

3.2.1.3 Módulo de parametrización

El módulo de parametrización se ha implementado en la clase *Modulo_parametrización.php*

En este módulo se han incluido todas las funciones que implementan los distintos tipos de algoritmos utilizados, así como alguna función auxiliar necesaria para completar su funcionamiento. A continuación abordamos cada una de ellas.

A. Función getComb

La función getComb *(function getComb)* es la encargada de simular el

comportamiento del método de los n-gramas. Para ello recibe dos parámetros de entrada:

1. Variable de tipo array *($termarray)*, contiene los términos a procesar
2. Variable de tipo entero *($max_combination)*, determina el tamaño del n-grama a calcular
3. Variable de tipo entero ($allcomb), indica el tipo de n-grama a calcular. (*$allcomb* = 1 calcula todas las combinaciones sin repetición, *$allcomb* = 0 calcula combinaciones de términos contiguos)

Como parámetro de salida la función devuelve un array con cada uno de los posibles n-gramas según el tamaño especificado.

El propósito del código para esta función es muy simple: identificar el tamaño del n-grama y calcular las posibles combinaciones. Para ello la estructura del código se define tal y como se muestra en el siguiente diagrama.

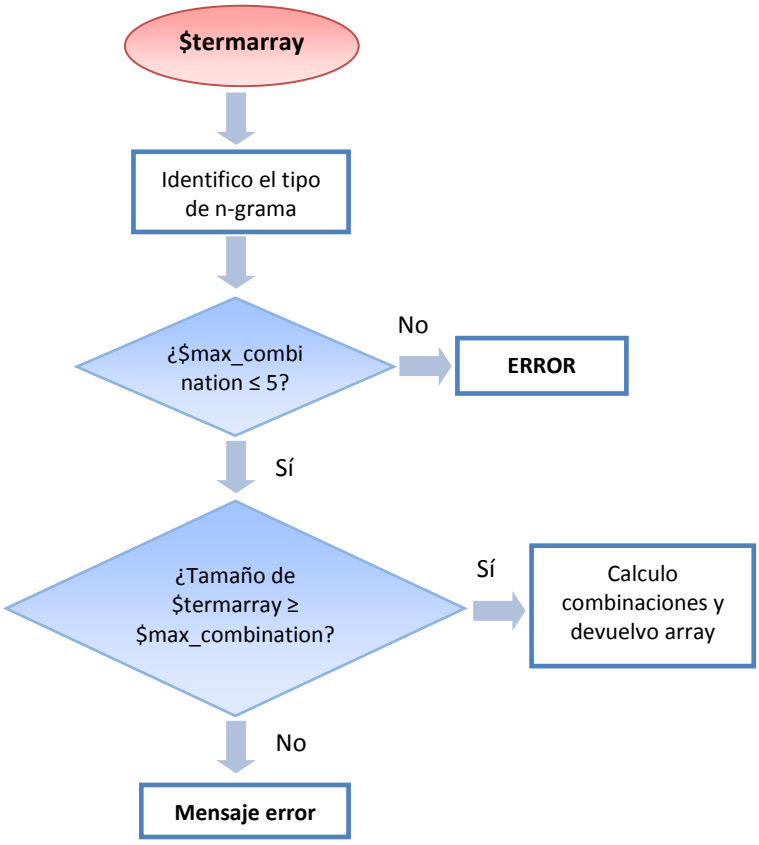

Figura 28. Diagrama de flujo de la función getComb

Tal y como se aprecia, el proceso es siempre el mismo: primero se identifica el tipo n-grama, y en caso de que el n-grama sea válido según nuestro sistema, se comprueba si

se disponen de suficientes términos para realizar las distintas combinaciones. Esto es importante, ya que en ocasiones puede que el array de entrada no contenga los elementos suficientes como para realizar el tipo de n-grama especificado. En este caso, primero se muestra un mensaje de error informando que el número de términos no es suficiente y luego, se devuelve un array vacío. En esta situación no tendría sentido parar la ejecución, ya que nos impediría seguir procesando otros tweets. Sin embargo, como nuestro sistema ofrece la posibilidad de calcular hasta 5-gramas, si el usuario elige un tamaño mayor, sí se parará la ejecución

En el caso de unigramas directamente devolvemos array recibido, ya que el cálculo de 1-gramas se corresponde directamente con cada uno de los términos del array *$termarray* de entrada. No se necesita ningún tipo de cálculo adicional.

En cuanto al cálculo de las distintas combinaciones según el tipo de n-grama, siempre se aplica la misma idea. Hay que destacar, tal y como explicamos en el apartado 3.1.5.1, que se realizan combinaciones de *n* elementos sin repetición en el que el orden no importa. A la hora de la implementación, deberemos tener en cuenta este requisito.

Para ello, el tamaño del n-grama a calcular nos indica la cantidad de bucles *for* anidados que debemos utilizar. Cada uno de ellos representará un índice distinto, permitiendo recorrer y combinar cada uno de los términos del array recibido.

Como se tratan combinaciones sin repetición, deberemos obviar aquellas situaciones en el que algunos los índices o todos ellos, tomen el mismo valor. En esa situación estaríamos formando un n-grama, o parte de él, a partir de un mismo término. Por este motivo, dado el siguiente array y requerido el cálculo de 3-gramas, tendríamos la configuración inicial de índices mostrada en la siguiente figura.

Figura 29. Configuración de los índices para el cálculo de combinaciones sin repetición

Esta configuración inicial, nos permite crear un criterio a partir del cual obtener las distintas combinaciones.

Además, por otro lado, esta situación nos sirve ya para obviar el orden de las distintas combinaciones. No tener en cuenta el orden, se traduce en que aquellas

combinaciones que ya hayamos evaluado, y que aparezcan en otro orden, no se deberán tener en cuenta. Para ello, hay que garantizar en aquellos n-gramas que vayamos a construir e incluir, que cada uno de los índices ocupe una posición más adelantada con respecto a su índice anterior en el array de términos recibido, según el orden ante mostrado. Esto implica que deberemos controlar el valor del primer índice a partir del cual se definen los siguientes. Si no lo hacemos, correremos el riesgo de acceder a elementos del array que no existan.

Según el ejemplo antes mostrado, el índice A podrá adoptar como máximo el valor 1. Este valor se corresponde con la posición que permite construir el último 3-grama del array recibido ("Twitter varias combinaciones").

Generalizando, el primer índice podrá adoptar como máximo el valor de: *tamaño del array - ($max_combination -1)*. Tomando este valor como referencia, cada índice posterior podrá adoptar como máximo una posición más con respecto al valor máximo del índice anterior. Cumpliendo estos requisitos garantizamos la construcción correcta de los distintos n-gramas.

Para la construcción de los distintos n-gramas se separa cada término por un carácter de espaciado.

En la siguiente figura a modo de resumen, se muestra el pseudocódigo del cálculo de las distintas combinaciones sin repetición para el caso de 3-gramas. Los demás casos seguirán la misma estructura, cada uno con el número de bucles anidados que le corresponda y por lo tanto con el número de índices requeridos en cada caso.

```
maximoA = (tamaño de $termarray – ($max_combination- 1));

for ( 0 < A < maximoA )
        valorB = A;
        valor C = A;
        valorB++;
        valorC++;
        for ( valorB < B < (maximoA +1) )
                valorC++;
                for ( valorC < C < tamaño de $termarray)
                        ngrama = $termarray[A].' '.$termarray[B].' '.$termarray[C]
                end for
        end for
end for
```

Figura 30. Pseudocódigo para el cálculo de combinaciones sin repetición para 3-gramas

En el caso que se quieran calcular solamente n-gramas de términos contiguos, simplemente vamos formando las distintas combinaciones desplazando una ventana del tamaño del n-grama a calcular.

En la siguiente figura se muestra el resultado tras calcular las distintas combinaciones mediante los dos métodos propuestos siguiendo el ejemplo antes mencionado.

. Combinaciones sin repetición:
Array ([0] => prueba twitter varias [1] => prueba twitter comb [2] => prueba varias comb
[3] => twitter varias comb)
. Combinaciones con ventana deslizante:
Array ([0] => prueba twitter varias [1] => twitter varias comb)

Figura 31. Ejemplo de la salida de la función getComb para los dos tipos de combinaciones

B. Función getFrequency

La función getFrequency *(function getFrequency)* se encarga de evaluar la frecuencia global de un conjunto de términos. Implementa la funcionalidad del algoritmo TF_i. Para ello, recibe un único parámetro de entrada:

1. Variable de tipo array *($arrayINN)*, contiene el conjunto de términos a evaluar. Este array tiene un formato específico. Se trata de un array multidimensional en el que cada posición contiene a su vez un array de dos posiciones: la primera especifica la fecha que se está evaluando (acorde al formato fecha elegido por el usuario), y la segunda incluye un array que comprende cada uno de los términos, y en general n-gramas, de los tweets publicados en la fecha antes mencionada. De esta forma, el contenido de cada una de las posiciones del array *$arrayINN* adoptaría la siguiente estructura:

Array ([fecha] => 2013-01-01/01 [hash] => Array ([0] => prueba [1] => pln . . .)

Como parámetro de salida, la función devuelve un array mutidimensional *($arrayreturn)* en el que cada posición especifica el n-grama evaluado y su frecuencia total. Este array estará ordenado de mayor a menor frecuencia.

De esta forma, dado el ejemplo de la figura, el array de salida adopta la siguiente estructura.

. Array de entrada:

Array ([fecha] => 2013-01-01/01 [hash] => Array ([0] => ejemplo [1] => array
[1] => entrada [0] => ejemplo)

. Array de salida:

Array ([0] => Array ([hash] => ejemplo [frec] => 2) [1] => Array ([hash] => entrada
[frec] => 1) [2] => Array ([hash] => array [frec] => 1))

Figura 32. Ejemplo de la salida de la función getFrequency

La idea de esta función es muy sencilla. Se van recorriendo cada uno de los n-gramas de entrada de cada uno de los documentos. Si el n-grama a evaluar ya se ha analizado anteriormente, se aumenta su frecuencia. Si por el contrario es nuevo, lo incorporamos con frecuencia unitaria al array de salida donde guardamos los distintos n-gramas ya procesados. De esta forma obtenemos un array con cada uno de los diferentes n-gramas y sus correspondientes frecuencias.

La estructura general de la función se muestra en la siguiente figura.

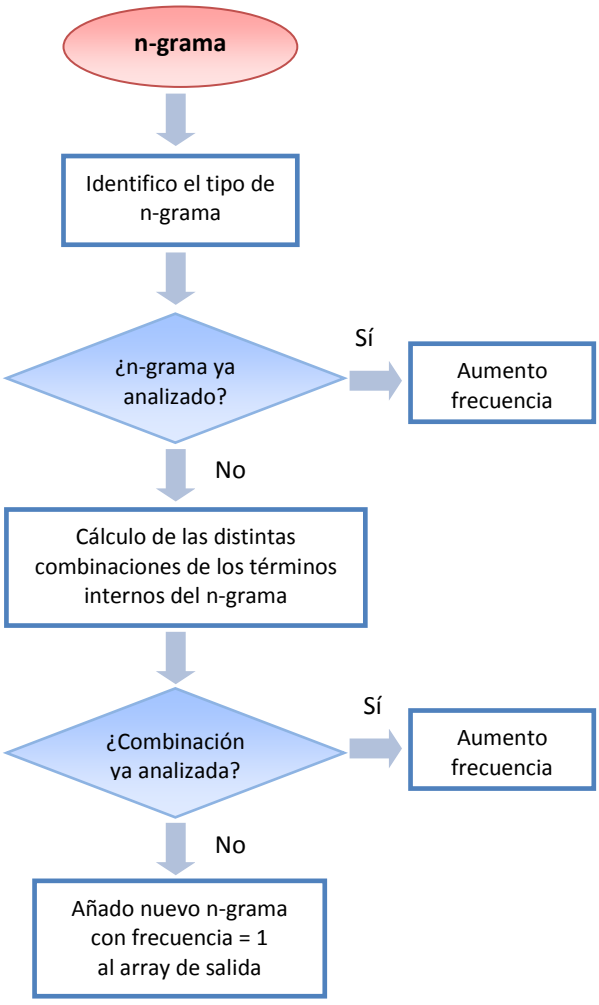

Figura 33. Diagrama de flujo de la función getFrequency

Tal y como explicamos anteriormente, el n-grama “A B” se considera igual que el “B A”, por lo que deberemos comprobar todas las posibles combinaciones de los términos de cada n-grama para asegurarnos de que el n-grama a evaluar no ha sido analizado anteriormente.

En el caso de unigramas, no tiene sentido calcular las distintas combinaciones ya que se tratan de un único término, por lo que si se confirma que no se ha analizado ya anteriormente, se añade directamente al array de salida donde almacenamos los distintos n-gramas que ya han sido procesados.

A partir de unigramas, ya será necesario analizar cada una de las distintas combinaciones. A priori, la manera más directa de abordar todas las combinaciones es el uso de tantos bucles for como indique el tamaño del n-grama, sin embargo esta configuración ralentiza mucho el sistema. Por este motivo se ha optado por otro tipo de implementación.

En el caso de bigramas y trigramas, tenemos un total de 2 y 6 combinaciones respectivamente (2! y 3!). En estas ocasiones, al tratarse de pocas combinaciones, se han implementado manualmente. Para situaciones más complejas se ha abordado de otra forma. Dado un n-grama recibido, se comprueba si las distintas combinaciones de los términos internos de cada uno de los n-gramas almacenados en el array de salida, se corresponde con el n-grama recibido en cuestión. Para ello, por cada término interno del n-grama recibido, se comprueba si se corresponde con alguno de los términos internos de un determinado n-grama almacenado en el array de salida. Cada término correspondido, se elimina del proceso de reconocimiento. Este proceso se repite con todos los n-gramas existentes en el array de salida. De esta forma sabemos si un n-grama ya evaluado es igual al n-grama recibido.

Dado un 4-grama por ejemplo, la estructura del pseudocódigo quedaría tal y como se observa en la siguiente figura.

```
$aux1,$aux2,$aux3,$aux4 = 0;

for cada n-grama i del array de salida de n-gramas ya procesados

        for cada término t del n-grama i

                if((termino t == termino 0 del n-grama recibido) y ($aux1 == 0))
                        $aux1 = 1;
                else if((termino t == termino 1 del n-grama recibido) y ($aux2 == 0))
                        $aux2 = 1;
                else if((termino t == termino2 del n-grama recibido) y ($aux3 == 0))
                        $aux3 = 1;
                else if((termino t == termino 3 del n-grama recibido) y ($aux4 == 0))
                        $aux4 = 1;
        end for
end for

if(($aux1 y $aux2 y $aux3 y $aux4) == 1)
        Aumentamos frecuencia
else
        añadimos n-grama al array de salida con frecuencia 1
```

Figura 34. Pseudocódigo de la función getFrequency para 4-grama

Una vez que un término interno de un n-grama específico del array de salida se corresponde con un término del n-grama recibido, se elimina del proceso *($auxX == 0)*, evitando tenerlo en cuenta en las siguientes iteraciones, garantizando así que el n-grama ya procesado es igual (pero en distinto orden) que el n-grama recibido

Una vez que se han analizado todos los n-gramas del array recibido, se ordenan de mayor a menor frecuencia, y se devuelven como parámetro de salida.

C. Función TFij

Esta función *(function TFij)* implementa el algoritmo TFij. Evalúa la frecuencia de un determinado número de n-gramas más frecuentes en cada uno de los documentos del conjunto de tweets, acordes con el formato fecha establecido. Para ello recibe tres parámetros de entrada:

1. Variable de tipo array *($array_comb_ord)*, contiene el conjunto total de n-gramas evaluados con sus respectivas frecuencias, ordenador de mayor a menor frecuencia (parámetro de salida *$arrayreturn* de la función *getFrequency*).

2. Variable de tipo array *($array_comb)*, contiene el conjunto de n-gramas totales. Se trata del mismo array que recibe la función *getFrequency ($arrayINN).*
3. Variable de tipo entero *($num_paint_term)*, indica el número de n-gramas más frecuentes (según *$termarray*) que se evaluarán.

Como parámetro de salida devuelve un array *($arrayOUT)* de dos posiciones:

- La primera posición contiene una matriz con la frecuencia de cada uno de los n-gramas más populares en cada uno de los distintos documentos. El número de n-gramas populares a evaluar dependerá del parámetro de entrada *$num_paint_term.*
- La segunda posición contiene otra matriz con la puntuación IDF de cada uno de los n-gramas antes evaluados.

Calcular la puntuación IDF no es requerimiento del algoritmo TF_{ij}. Sin embargo, como esta función analiza cada uno de los n-gramas recibidos en cada uno de los documentos, aprovechamos para obtener la puntuación IDF, que más adelante nos hará falta para computar la puntuación de otro algoritmo.

El primer paso es saber el número de n-gramas a los que se va a aplicar el algoritmo. El parámetro *$num_paint_term* nos indica cuántos elementos se cogen del array *$array_comb_ord*. De esta forma obtenemos los *$num_paint_term* n-grama más populares para ser procesados.

Una vez realizado este paso, la estructura es similar a la de la función anterior. Se evalúan todos los n-gramas de cada uno de los documentos del array *($array_comb)* de entrada, con cada uno de los n-gramas más populares.

Si un determinado n-grama popular no se corresponde con ninguno de los distintos n-gramas de un determinado documento, se concluye que dicho n-grama popular no aparece en ese documento y por lo tanto su frecuencia es nula. Repitiendo este proceso en cada uno de los distintos documentos y con cada uno de los n-gramas populares, obtenemos la frecuencia de cada n-grama popular en cada uno de los distintos documentos.

Por ejemplo, dado un n-grama popular concreto a evaluar, el pseudocódigo de la función quedaría tal y como muestra la siguiente figura.

```
$TT = n-grama popular a evaluar;
$encontrado = 0;
$numdoc_TT=0;

for cada documento (fecha según formato) i del array $array_comb
        $indicarIDF = 0;

        for cada n-grama t del documento i

                Identifico el tipo de n-grama;

                if(n-grama t == $TT)
                        Aumento frecuencia de $TT en el documento i;
                        $encontrado = 1;

                        //Número de documentos en el que aparece $TT
                        if($indicarIDF = 0)
                                $numdoc_TT++;
                                $indicarIDF=1

                else
                        Calculo las combinaciones de los términos internos del n-grama t;

                        if(Alguna combinación == $TT)
                                Aumento frecuencia de $TT en el documento i;
                                $encontrado = 1;

                        //Número de documentos en el que aparece $TT
                        if($indicarIDF = 0)
                                $numdoc_TT++;
                                $indicarIDF=1

        end for

if($encontrado = 0)
        frecuencia = 0 de $TT en el documento i;

end for
```

Figura 35. Pseudocódigo de la función TFij para un determinado n-grama

Al igual que la función anterior, estamos evaluando la similitud entre distintos n-gramas, en este caso según el ejemplo, entre un n-grama popular y el conjunto de n-gramas de los distintos documentos del array recibido. Esto implica, al igual que antes, que deberemos calcular las posibles combinaciones de los términos internos de cada uno de los n-gramas evaluados en cada uno de los documentos. De esta manera nos

aseguramos y comprobamos si alguno de ellos se corresponde con el n-grama popular en cuestión.

Además, tal y como se aprecia, estamos detectando paralelamente el número de documentos en los que aparece el n-grama $TT, y en general, en los que aparecen el conjunto de n-gramas a evaluar, para posteriormente realizar el cálculo del peso IDF. De esta forma optimizamos el tiempo de ejecución.

D. Función TF_IDF

La función TF-IDF *(function TF_IDF)* se encarga de calcular la puntuación TF-IDF Recibe tres parámetros de entrada:

1. Variable de tipo array *($arrayidf)*, contiene la matriz con cada una de la puntuaciones IDF calculadas anteriormente.
2. Variable de tipo array *($array_comb_ord)*, contiene el conjunto total de n-gramas evaluados con sus respectivas frecuencias, ordenador de mayor a menor frecuencia (parámetro de salida *$arrayreturn* de la función *getFrequency*).
3. Variable de tipo entero *($num_paint_term)*, indica el número de puntuaciones TF-IDF que se calcularán.

Como parámetro de salida devuelve una matriz asociativa *($arraytotal)* con cada una de las puntuaciones de los distintos n-gramas.

Esta función simplemente multiplica cada una de las puntuaciones IDF del array $arrayidf con cada una de las frecuencias del array $array_comb_ord, de tantos n-gramas como indique el parámetro $num_paint_term.

E. Función NTF

La función NTF *(function NTF)* se encarga de realizar el cálculo de la puntuación NTF. Recibe 4 parámetros de entrada:

1. Variable de tipo String *($TT)*, contiene el n-grama que se quiere evaluar.
2. Variable de tipo entero *($ventana)*, indica el tamaño de la ventana a utilizar. Hace referencia al número de tweets que se quieren tener en cuenta para la ventana.

3. Variable de tipo array *($array_comb)*, contiene un array multidimensional en el que cada posición engloba la fecha y los n-gramas de cada uno de los tweets evaluados.
4. Variable de tipo entero *($numTT)*, indica el número de veces que el n-grama a evaluar aparece en el total de tweets de la colección.

Como parámetro de salida, la función devuelve una matriz asociativa que indica la puntuación NTF de cada uno de los tweets evaluados

Como ya se explico anteriormente, el sistema para calcular la puntuación NTF ofrece al usuario la posibilidad de elegir ciertos parámetros. La función NTF ha sido implementada para trabajar directamente con datos y estructuras ya organizadas según los requisitos del usuario.

De esta forma, previo a la función, el usuario ha elegido ya las fechas entre las que quiere aplicar el algoritmo y la ventana que desea utilizar. Por ello la matriz de entrada engloba directamente los n-gramas del conjunto de tweets entre ambas fechas pero teniendo en cuenta además, el efecto de la ventana. Como para el cálculo de la puntuación, la ventana se centra en cada instante temporal t(definido por la fecha de cada tweet), en los tweets cercanos a las fechas límites se estará accediendo a tweets fuera de ese periodo temporal.

Por ejemplo, dada una ventana de valor 3, en la siguiente figura se muestra el comportamiento en las fechas límite:

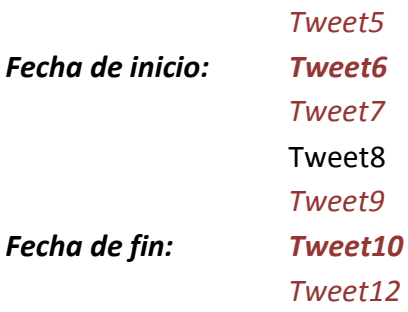

Figura 36. Ejemplo del comportamiento de la ventana en el algoritmo NTF en las fechas límite

Según el ejemplo, la función NTF recibiría un array que contiene ya los n-gramas desde los tweets 5 a 12. Posteriormente presentaremos el sistema NTF y explicaremos como se realiza el cálculo y comprobaciones para la creación de los distintos componentes necesarios para llamar a esta función

Una vez aclarado esto, podemos explicar el funcionamiento de la función. Para el cálculo de cada una de las puntuaciones NTF, se recorre cada uno de los tweets comprendidos entre las fechas límite. En cada fecha, que se corresponde con cada tweet, se debe computar el número de tweets en los que aparece el n-grama especificado a lo largo de toda la ventana definida. De esta forma obtendremos lo necesario para calcular la puntuación NTF en cada una de las fechas que comprenden el periodo temporal analizado.

La estructura del código queda tal y como se muestra en la siguiente figura.

```
$TT: n-grama a analizar
$frec: número de tweets (dentro de la ventana) en los que aparece STT

for cada tweet i entre las fechas límite

        $frec, puntuación NTF = 0;

        for cada tweet j incluido en la ventana (centrada en el tweet i)

                for cada n-grama x del tweet j

                        Identifico n-grama;

                        if(n-grama x = $TT)
                                $frec++;
                                break;
                        else
                                Calculo combinaciones términos internos n-grama x

                                if(combinaciones = $TT)
                                        $frec++;
                                        break;
                end for
        end for

Desplazamos ventana un instante temportal
Calculamos puntuación NTF = $frec/$numTT
end for
```

Figura 37. Pseudocódigo de la función NTF

Al igual que en otras ocasiones, para el caso de unigramas, en caso de no coincidir con el n-grama a evaluar, no hay que calcular las combinaciones restantes ya que se trata de un único término.

Hay que tener en cuenta que a medida que evaluamos cada instante temporal (cada tweet), debemos ir desplazando la ventana para tenerla siempre centrada en el instante que se está evaluando en ese momento.

En este caso, al analizar un conjunto de tweets específicos (definidos por el periodo temporal comprendido entre las fecha límite), se ha optado para calcular las distintas combinaciones por el uso de bucles *for.* El tamaño del n-grama define el número de bucles anidados. Cada bucle representará un índice que permitirá realizar las distintas combinaciones sin repetición de los elementos internos.

Por ejemplo, para un bigrama, tendríamos el pseudocódigo mostrando en la siguiente figura.

```
for (0 < z < tamaño n-grama)
        for (0 < y< tamaño n-grama)
                if(z ¡= y)
                        comb =  term_int[z].' '.term_int[y];
        end for
end for
```

Figura 38. Pseudocódigo del cálculo de las combinaciones en la función NTF

Para implementar combinaciones sin repetición hay que evitar situaciones en el que el valor de los índices es idéntico.

Por último, a medida que se van obteniendo las diferentes puntuaciones NTF, se va creando una matriz asociativa en la que se especifica las puntuaciones correspondientes a cada una de las fechas evaluadas (tweets evaluados) dentro de las fechas límite definidas por el usuario.

F. Función findTerms

Función auxiliar *(function findTerms)* necesaria para el cálculo de la puntuación NTF. Busca un determinado n-grama en un tweet de entrada. Recibe dos parámetros de entrada:

1. Variable de tipo array *($termarray)*, contiene un array con los términos del tweet de entrada.
2. Variable de tipo string *($findTerm)*, contiene el n-grama a buscar.

Si el n-grama especificado se encuentra en el tweet, se devuelve un true. En caso contrario se devuelve un false.

Para ello se recorren cada una de los términos del array recibido y se comprueba si alguno es igual al n-grama especificado. En casos distintos a unigramas, se identifican cada uno de los términos internos del n-grama y se buscan en el array. Si todos los términos del n-grama especificado se encuentran en el tweet, se devuelve un 1, si no un 0.

G. Función get_micro_time

Independientemente de la parametrización de los términos de los tweets, se ha implementado una función para obtener el tiempo en un determinado instante. Simplemente nos devuelve el tiempo en un instante expresado en segundos, con una precisión hasta milisegundos. Esta función nos servirá para medir el tiempo en distintos puntos de nuestro sistema, permitiéndonos evaluar el tiempo de ejecución según distintas pruebas.

Para ello se llama a la función microtime() de PHP (que nos devuelve un string de la forma "mseg seg") e identificamos cada uno de los valores, para posteriormente devolverlos.

El código de la función se muestra en la siguiente figura.

```
// Obtenemos los segundos y milisegundos
list($mseg,$seg) = explode(" ",microtime());
// Devolvemos
 return ((float)$mseg+(float)$seg);
```

Figura 39. Código de la función get_micro_time

3.2.1.4 Módulo de representación

El módulo de representación se ha implementado en la clase *Modulo_representación.php*

En este módulo agrupamos todas las funciones mediante las cuales representamos los diversos datos obtenidos tras la aplicación de los distintos algoritmos. Todas estas funciones, las cuales explicaremos continuación, hacen uso de una clase auxiliar necesaria para la exportación de los datos a Excel. Se trata de la clase e*xcel.php*

Esta clase, ya implementada y de código abierto, ha sido descargada e incorporada a nuestro sistema. De esta forma podemos exportar fácilmente un flujo de datos

directamente a una hoja de cálculo de Microsoft Excel. Estas hojas de cálculo tendrán formato *xls*. Además de escribir, también permite leer datos.

Sin embargo, a la hora de exportar los datos a Excel, se ha de seguir un formato específico. Esta clase permite exportar array de datos, y en general, matrices asociativas de datos. Las matrices asociativas son conjuntos no ordenados de pares formados por una clave y un valor. Utilizan claves en lugar de índices numéricos para organizar los valores almacenados.

Esto implica que a la hora de crear los arrays de datos a exportar, deberemos cumplir con estas especificaciones. Como PHP permite crear arrays con parejas clave-valor, no tendremos ningún problema.

El contenido, tamaño y número de elementos de cada tabla variará según la función usada, sin embargo la creación de la matriz de datos siempre sigue la misma estructura.

Por ejemplo, dada la siguiente tabla en Excel.

PRUEBAS	**VALOR**
Prueba1	20

Tabla 4. Ejemplo del formato de la tabla exportada a excel

Si queremos crear la estructura correspondiente, deberíamos definir una matriz en el que cada posición se corresponde con los datos mostrados. Sin embargo PHP nos deja realizar esto de forma más directa, tal y como se muestra en la siguiente figura.

```
$matrizprueba[0]['PRUEBAS'] = 'Prueba1';
$matrizprueba[0]['VALOR'] = 20;
```

Figura 40. Código ejemplo de creación de la tabla de exportación a excel

Al mismo tiempo que establecemos el valor 'Prueba1', definimos el valor de la columna ('PRUEBAS') con el que podemos acceder a dicha posición. Una vez realizado esto, simplemente añadimos más filas identificando el valor de la columna ya definida. Si quisiésemos añadir una 'Prueba2' simplemente deberíamos escribir lo que se indica en la siguiente figura.

```
$matrizprueba[1]['PRUEBAS'] = 'Prueba2';
$matrizprueba[1]['VALOR'] = 30;
```

Figura 41. Código ejemplo de ampliación de la tabla de exportación a excel

De esta forma podríamos añadir tantas columnas y filas como queramos. El uso de este formato permite una implementación muy rápida y compacta mediante el uso de bucles *for.*

Una vez aclarado el formato de exportación a tablas Excel, abordamos cada unas de las funciones de representación de datos.

A. Función paintExcel

Esta función *(function paintExcel)* exporta directamente los datos recibidos. Tiene dos parámetros de entrada:

1. Variable de tipo string *($name_excel)*, indica el nombre con el que se guardará el archivo Excel
2. Variable de tipo array *($pruebafrec)*, contiene la matriz de datos que se quiere exportar, según el formato antes especificado

La función no tiene ningún parámetro de salida. Simplemente escribe los datos en una hoja de Excel. En la siguiente figura se muestra el código implementado.

```
// Importamos excep.php
require_once("excel.php");
// Creamos la ruta con el nombre del excel
$export_file = "xlsfile://./".$name_excel.".xls";
// Abrimos el ficheros
$fp = fopen($export_file, "w");
// Si no se puede abrir, mensaje de error
if (!is_resource($fp))
        die("Cannot open $export_file");
// Exportamos los datos
fwrite($fp, serialize($pruebafrec));
// Cerramos el fichero
fclose($fp);
```

Figura 42. Código de la función paintExcel

Si la hoja de cálculo especificada con *$name_excel* no existe, se crea y se exportan los datos. Si por el contrario, la hoja ya existe, se sobrescribe. En caso de que la tengamos abierta, dará un error.

Este será el formato de exportación que se usará en todas las funciones de representación.

B. Función paintExcel_Frec

Esta función *(function paintExcel_Frec)* representa los datos obtenidos del algoritmo TF_i. Recibe tres parámetros de entrada:

1. Variable de tipo array *($array_comb_ord)*, contiene el array de salida del algoritmo TF_i *(function getFrequency).*
2. Variable de tipo string *($name_excel)*, indica el nombre con el que se guardará el archivo Excel.
3. Variable de tipo entero *($num_patin_term)*, indica el número de n-gramas de los que representaremos su resultado.

No devuelve ningún parámetro de salida.

A partir del array de entrada, esta función crea una matriz asociativa con dos columnas: una en la que se especifica el n-grama y otra la frecuencia. Esta tabla tendrá tantas filas como indique el parámetro $num_paint_term.

Para ello se recorren los distintos campos del array $array_comb_ord de tantas posiciones como establezca $num_paint_term, asignando cada valor a su posición correspondiente en la tabla.

El pseudocódigo queda tal y como se muestra en la siguiente figura.

```
for (0 < $i < $num_paint_term)
        $tabla[$i][‘TERMINO’] = hash de $array_comb_ord[$i];
        $matrizprueba[$i][‘FRECUENCIA] = frec de $array_comb_ord[$i];
end for
```

Figura 43. Pseudocódigo de la función paintExcel_frec

Por último se exporta a una hoja de cálculo con el nombre especificado en $name_excel.

C. Función paintExcel_Time

Esta función *(function paintExcel_Time)* exporta a Excel distintos tiempos de ejecución del programa. Recibe dos parámetros de entrada:

1. Variable de tipo array *($tiempo)*, contiene los distintos tiempos a representar
2. Variable de tipo string *($name_excel)*, indica el nombre con el que se guardará el archivo Excel.

No devuelve ningún parámetro de salida.

En esta función se crea una matriz asociativa con una estructura de los dos columnas: una que identifica qué tiempo es y qué valor tiene.

Para ello, el pseudocódigo es el mostrado en la siguiente figura.

```
for cada posición i de $tiempo
        $tabla[i]['TIEMPO] = 'Tiempo';
        $matrizprueba[i]['SEG] = $tiempo[i];
end for
```

Figura 44. Pseudocódigo de la función paintExcel_Time

Por último se exporta a Excel con el nombre especificado en $name_excel.

D. Función paintExcel_numTerm

Esta función *(función paintExcel_numTerms)* representa el número de n-gramas evaluados. Recibe 2 parámetros de entrada:

1. Variable de tipo entero *($num_terms)*, indica el número de n-gramas totales que han sido evaluados.
2. Variable de tipo string *($name_excel)*, indica el nombre con el que se guardará el archivo Excel.

No devuelve ningún parámetro de salida. A partir del dato de entrada, crea una matriz de una única columna en la que se exporta el número de n-gramas evaluados.

3.2.2 SISTEMAS

Para probar el funcionamiento de nuestro sistema se han creado unos scripts en los que está definida toda la estructura general de nuestro programa. En ellos se usan los distintos módulos y se crean y configuran las distintas conexiones, para obtener así todos los resultados requeridos tras su ejecución.

3.2.2.1 Sistema 1

En el sistema implementado en este script (llamado *sistema1.php*), se han configurado todos los algoritmos a excepción del algoritmo NTF. Hemos diseñado el sistema para obtener directamente todos los resultados de los algoritmos TF_i, TF_{ij} y TF-IDF.

Además se han añadido algunas prestaciones. Por un lado se realiza el cálculo del número total de términos que se evalúan, permitiéndonos saber el número de n-gramas totales que se procesan en cada prueba.

Por otro lado, se calcula en diversos puntos de interés del sistema el tiempo de ejecución, lo que nos permite conocer qué operaciones tienen un coste computacional mayor.

Para comenzar, el usuario debe definir una serie de parámetros que configuran el tipo de prueba que desea realizar. Se debe elegir el tamaño del n-grama, el tipo de cálculo de n-gramas, si se quiere detectar hashtag y/o palabras, si se quiere aplicar filtrado y/o lematización, el formato fecha, el número de n-gramas a representar en los distintos Excel, el nombre del fichero que contiene el conjunto de tweets y una serie de nombres con los que se guardarán los distintos Excel con cada uno de los resultados.

El formato fecha determinará cada uno de los documentos. Para ello, según el formato fecha elegido, se sigue estructura mostrada en la siguiente figura.

```
fecha anterior = 0;

for cada tweet j del conjunto de tweets
        if( fecha tweet j = fecha anterior)
                Añado n-gramas al último documento
                Actualizo fecha anterior = fecha del tweet j
        else
                Creo nuevo documento y añado n-gramas
                Actualizo fecha anterior = fecha del tweet j
end for
```

Figura 45. Pseudocódigo de la formación de los documentos

Para poder obtener y añadir cada uno de los n-gramas, previamente se ha preprocesado, tratado y parametrizado cada uno de los tweets según las especificaciones iniciales del usuario. De esta forma obtenemos cada documento con sus respectivos n-gramas.

Tras esto, calculamos cada uno de los algoritmos llamando a las distintas funciones y exportamos los datos obtenidos a ficheros Excel.

De cara a evaluar el tiempo de ejecución, los puntos de interés que se han marcado son: momento en el que se ha creado todo el conjunto de documentos con sus

respectivos n-gramas, finalización del algoritmo TF_i y finalización del programa (resto de algoritmos con todas las exportaciones de los resultados a Excel)

3.2.2.2 Sistema 2

En este script (llamado *sistema2.php*) se ha configurado el procedimiento para el cálculo del algoritmo NTF.

De nuevo el usuario debe definir el valor de una serie de parámetros. Deberá especificar el n-grama concreto que quiere analizar, el tamaño del mismo, el tipo de cálculo de n-gramas, el formato fecha, la fecha de inicio y fin entre las que se quiere aplicar el algoritmo, el tamaño de la ventana a usar, el nombre del fichero que contiene el conjunto de tweets y el nombre del Excel donde se exportarán los resultados.

Tras ello, se comprueba si con los datos especificados por el usuario no se produce ningún tipo de error en el cálculo del algoritmo. Para ello la estructura que sigue el programa se muestra en la siguiente figura.

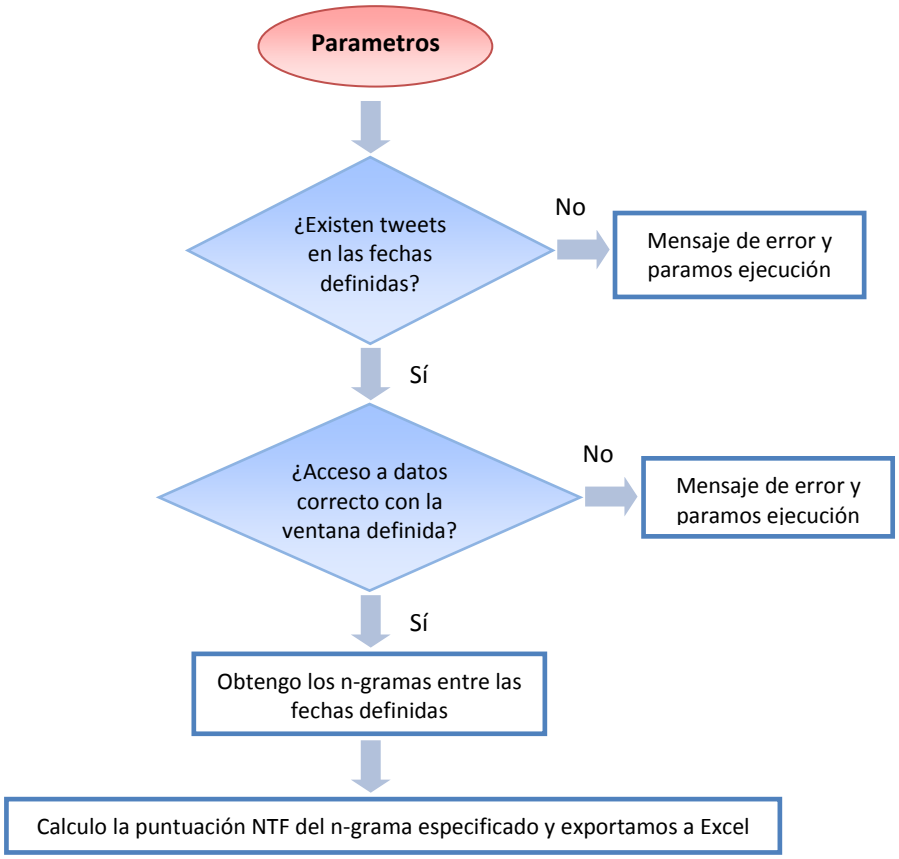

Figura 46. Diagrama de flujo del sistema 2 implementado

En este caso el formato fecha establece la precisión con la que buscamos tweets emitidos en las fechas límites definidas por el usuario. Debe definir fechas exactas, según el formato que elija, en las que se haya emitido algún tweet.

La obtención de los distintos términos de los tweets dentro de las fechas límites, se calculan tweet a tweet, independientemente del formato fecha elegido. Para ello siempre se aplica un filtrado y se detectan tanto palabras como hashtag.

Como para llamar a la función NTF es necesario saber el número de tweets del conjunto total en los que aparece el n-grama a evaluar, se recorre toda la colección de tweets. Sin embargo, solo procesaremos y almacenaremos los n-gramas comprendidos entre las fechas límite definidas.

Luego simplemente llamamos a la función y exportamos los resultados a Excel.

3.2.2.3 Sistema 3

En este script (llamado *sistema3.php*) se ha implementado una prueba en la que se detectan hashtags y palabras idénticas a esos hashtags, sólo en aquellos tweets que contienen al menos un hashtag.
Al igual que en el sistema 1, se calculan todos los algoritmos a excepción del NTF. La única diferencia reside que en este caso se detectan hashtags y palabras que sean iguales que esos hashtags.

Los parámetros a elegir por el usuario son: el deseo de filtrado y lematización, el tamaño del n-grama y el tipo de cálculo, el formato fecha, el número de términos a representar en los distintos Excel, el nombre del fichero que contiene el conjunto de tweets y una serie de nombres con los que se guardarán los distintos Excel con cada uno de los resultados.

La estructura interna es similar a la del sistema 1. En vez de calcular los distintos n-gramas según los requisitos del usuario, para cada tweet primero se detectan los hashtags y, en caso de identificar al menos uno, se llama a la función *getHash_word* para detectar aquellas palabras que sean iguales que los hashtgas detectados. El resto de proceso es totalmente idéntico

3.2.2.4 Sistema 4

En este script (llamado *sistema4.php*) se ha implementado una prueba en la que se detectan hashtags y posteriormente se identifican todas aquellas palabras que se corresponden con alguno de los hashtags detectados, en todo el conjunto de tweets.

Los parámetros a elegir por el usuario son los mismos que los del sistema 3: el deseo de filtrado y lematización, el tamaño del n-grama y el tipo de cálculo, el formato fecha, el número de términos a representar en los distintos Excel, el nombre del fichero que contiene el conjunto de tweets y una serie de nombres con los que se guardarán los distintos Excel con cada uno de los resultados.

Para ello, primero se detectan todos los hashtags del conjunto de tweet. Luego por cada palabra de cada uno de los tweets, se averigua si se corresponden con algunos de los hashtags antes identificados. En caso afirmativo, guardamos esa palabra y la procesamos. En caso negativo, la obviamos.

El esquema del formato fecha es el mismo que el del sistema 2.

3.2.2.5 Ficheros generados

Tras la ejecución de nuestro sistema, se generan una serie de tablas de Excel con los resultados obtenidos. Estas Excel se guardan automáticamente en la carpeta *htdocs*.

Con la ejecución del sistema1 obtenemos los siguientes ficheros *xls* de salida:

- Fichero con los resultados del algoritmo TF_i
- Fichero con los resultados del algoritmo IDF
- Fichero con los resultados del algoritmo TF-IDF
- Fichero con los resultados del algoritmo TF_{ij}
- Fichero con el número de n-gramas evaluados
- Fichero con los tiempos de ejecución del sistema

Con la ejecución del sistema 2 obtenemos los siguientes ficheros *xls* de salida:

- Fichero con los resultados del algoritmo NTF

Con la ejecución del sistema 3 y 4 obtenemos los mismos ficheros que el sistema 1, pero con la detección antes descrita.

3.3 DESPLIEGUE DE LA SOLUCIÓN

El desarrollo del sistema se ha llevado a cabo en Windows. En este entorno la opción más sencilla se basa en usar cualquier servidor web o apache con soporte PHP. Para ello, existen varias opciones:

- XAMMP (http://www.apachefriends.org/index.html) con Apache, PHP y MySql para base de datos .
- WAMP (http://www.wampserver.com/en/)
- Simple PHP (http://tyler.io/simple-php-framework/) que consta de un servidor web con soporte PHP.

En nuestro caso se ha usado XAMMP. Utilizando el servidor Apache integrado en este programa, podemos ejecutar y compilar nuestros scripts en cualquier navegador web. Para ello es necesario acceder un navegador web a la siguiente URL:

localhost/nombrescript.php

De esta forma ejecutamos nuestro sistema, obteniendo la salida de datos por el navegador web.

Para poder ejecutar los scripts, se deben situar todos los ficheros en una carpeta específica de XAMMP. Concretamente en la carpeta *htdocs*, localizada dentro de la carpeta xammp (situada en la ruta donde se halla instalado el programa).

Aquí se deberán colocar todos los ficheros que componen nuestro sistema: *Modulo_preprocesado.php*, *Modulo_tratamiento.php*, *Modulo_parametrizacion.php*, *Modulo_representacion.php*, *excel.php*, *sistema1.php*, *sistema2.php*, *sistema3.php*, *sistema4.php* y el fichero de texto que contiene el conjunto de tweets.

En definitiva, se debe incluir cualquier fichero que forme parte del programa a ejecutar.

Para realizar cambios o añadir mejoras al sistema, se puede utilizar cualquier editor de textos. En nuestro caso se ha utilizado Notepad++.

CAPÍTULO 4 - EVALUACIÓN

En este capítulo expondremos el conjunto de pruebas confeccionadas para abordar la detección de Trending Topics en Twitter. Se realizarán pruebas bajo diversos puntos de vista y con distintas configuraciones, lo que nos permitirá obtener un conjunto de resultados que nos ayudarán a sacar conclusiones sobre el tema tratado.

Por un lado abordaremos la detección de hashtags, la detección de palabras y la detección de ambas a la vez. En cada uno de ellos se evaluarán diferentes n-gramas según el comportamiento obtenido, analizando cada uno de los algoritmos implementados. Además se analizará el escenario particular de la detección de hashtags y palabras idénticas a ellos en sus dos versiones distintas.

Por otro lado se estudiará el comportamiento aplicando el proceso de lematización. Veremos distintas pruebas en las que observaremos la influencia que tiene sobre el tratamiento de los datos y los resultados obtenidos.

Por último, se analizará el número total de términos evaluados y el tiempo de procesado, en cada uno de los distintos escenarios antes tratados.

Para ello evaluaremos el conjunto de tweets dados en el fichero "*tweets.SEPLN.txt*". Este conjunto contiene 78401 tweets (a lo largo de 5 meses) de un corpus usado en el taller TASS (Taller de análisis de sentimientos) de la SEPLN (Sociedad Española para el Procesamiento del Lenguaje Natural) [70]. En varias pruebas seleccionaremos sólo una parte de este conjunto total de tweets. Además, en alguna ocasión analizaremos el conjunto de tweets del fichero "*tweets.2013.txt*" para verificar algún tipo de comportamiento. Este conjunto está formado por 57121 tweets (de un mes) de una serie de personalidades españolas (políticos, periodistas, famosos…).

Destacar que en toda la evaluación recién descrita, se aplica el filtrado previo de los datos. A continuación se muestran las distintas pruebas implementadas.

4.1 DETECCIÓN DE HASHTAGS

En este apartado abordaremos la detección de hashtags. Primero no centraremos en el caso de 1-gramas, para luego evaluar el resto de n-gramas.

4.1.1 UNIGRAMAS

En este punto detectaremos hashtags individuales en todo el conjunto de tweets de SEPLN. Aplicando el algoritmo TF_i obtenemos la siguiente tabla de los hashtags más frecuentes (ordenados de mayor a menor frecuencia).

HASHTAG	TF_i	HASHTAG	TF_i
elcambioandaluz	860	eperiodico	177
ff	674	losdesayunosdetve	176
malaga	540	25m	165
andalucia	405	grinan	142
investidura	403	cmin	141
congreso	385	eres	138
sevillahoy	271	portadaepc	126
17congresopp	250	rajoy	124
yeswespainisdifferent	230	cofradiasmlg	121
38congresopsoe	216	votapp	115

Tabla 5. Hashtags más frecuentes

Observamos que 6 primeros hashtags tienen una frecuencia total bastante superior respecto al resto, y concretamente el primero en la tabla, que destaca sobre el resto.

A priori se puede pensar que el hashtag más frecuente será TT, sin embargo esto no tiene por qué ser así. El carácter frecuente es una cualidad necesaria pero no suficiente. Es obvio que un término con una frecuencia baja, inferior a un cierto nivel, es muy poco probable que sea TT. Sin embargo en el extremo contrario, no se puede afirmar que un término frecuente siempre sea TT.

Acudiendo al algoritmo TF-IDF, podemos averiguar qué hashtags tienen mayor carácter TT. En la siguiente tabla se muestran los resultados del citado algoritmo.

HASHTAG	TF-IDF	HASHTAG	TF-IDF
elcambioandaluz	784,733233	eperiodico	247,873389
ff	730,120365	losdesayunosdetve	225,290901
malaga	514,899372	25m	246,458909
andalucia	455,913996	grinan	218,887766
investidura	**844,58739**	cmin	267,361097
congreso	507,275652	eres	200,758576
sevillahoy	329,506245	portadaepc	173,760499
17congresopp	433,899725	rajoy	214,243922
yeswespainisdifferent	320,43053	cofradiasmlg	226,742093
38congresopsoe	389,634566	votapp	195,248189

Tabla 6. Puntuación TF-IDF de los hashtags más frecuentes

Los hashtags más frecuentes son los que poseen una puntuación TF-IDF más alta. Sin embargo, la mayor puntuación no se corresponde con el hashtags más frecuente, tal y como apreciamos. Es el 5º hashtag el que obtiene la mejor puntuación. Esto confirma lo que antes se ha mencionado: que un término sea muy frecuente no implica que sea TT. Sin embargo ser TT si requiere que el término tenga un nivel de frecuencia alto.

Como podemos observar dentro del conjunto de hashtags más frecuentes, aquellos con menor frecuencia, son los que tienen una puntuación TF-IDF más baja. Esto no hace más que corroborar lo anterior: un TT requiere un cierto nivel mínimo de frecuencia.

Es muy importante tener claro la relación unidireccional entre frecuencia y carácter Trending Topic. Así pues, según los resultados obtenidos, el hashtag *'investidura'* es el "mayor" TT de todo el conjunto de tweets.

Analizando los tweets mediante el algoritmo TF_{ij}, podremos observar en la siguiente figuracómo evoluciona cada uno de los hashtags a lo largo de cada uno de los documentos, comprobando así visualmente su carácter TT. Para ello, el formato fecha definido es el aquel con precisión hasta las horas ('Y-m-d/H').

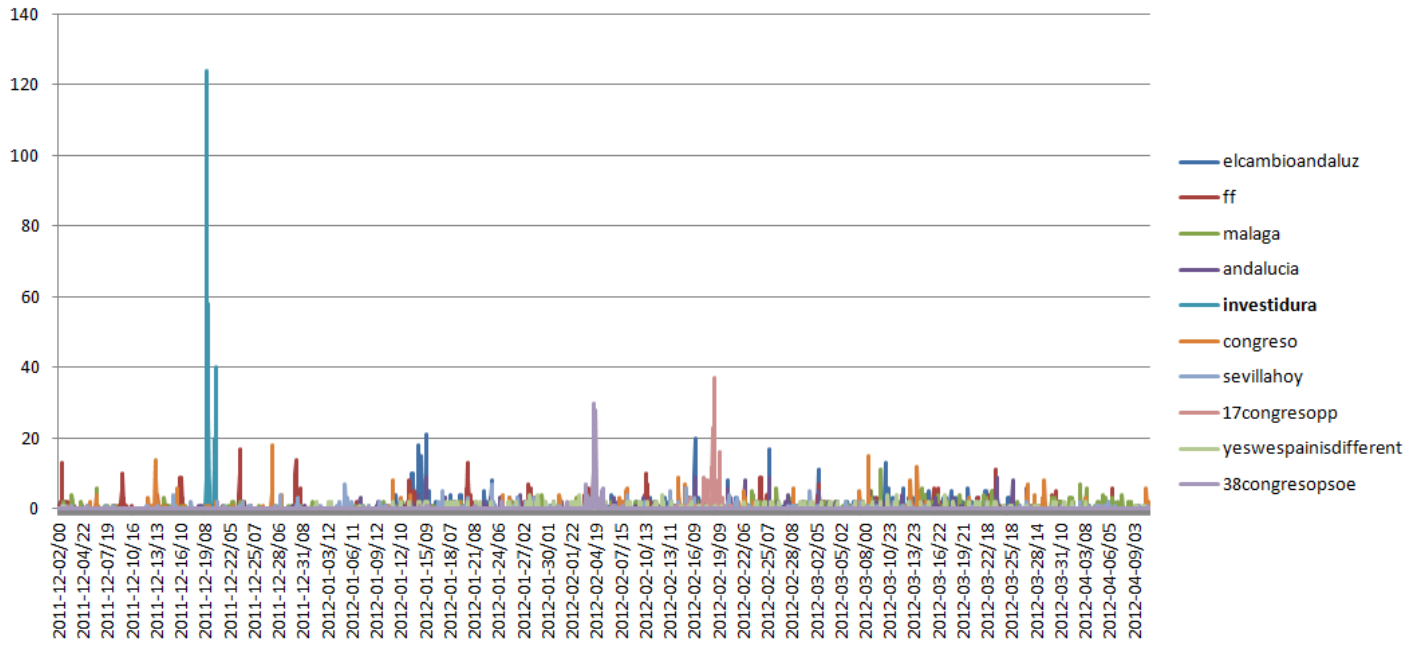

Figura 47. Puntuación TF_{ij} de los hashtags más frecuentes

Contemplando la figura, es evidente el carácter TT del hashtag *'investidura'*. La mayoría de sus apariciones se concentran en unos pocos documentos. Por este motivo alcanza ese pico de frecuencia cuyo valor es muy superior al del resto de los hashtags.

Esta característica, es un requisito indispensable para que un hashtag sea TT. Es necesario que la mayor parte de apariciones se concentren en un periodo temporal bastante acotado. Cuanto más acotado sea, mucho mejor.

Representando solamente la evolución de los 3 hashtags más frecuentes obtenemos la siguiente figura.

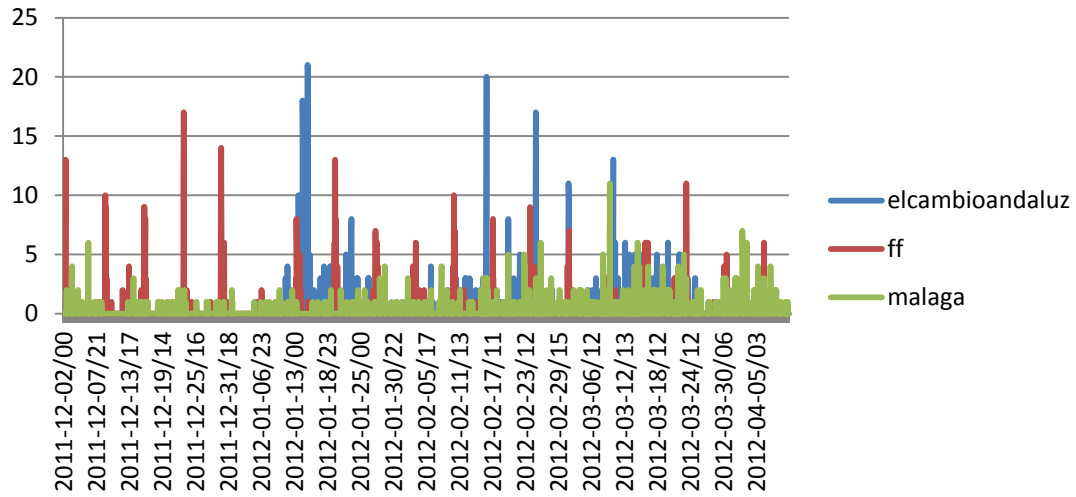

Figura 48. Puntuación TF_{ij} de los 3 hashtags más frecuentes

Si recordamos, estos hashtags tenían las puntuaciones TF-IDF más altas (por debajo del hashtag *'investidura'*).

Podemos observar que el comportamiento de los dos primeros sigue una estructura similar: existen varios periodos de tiempo concretos en donde se concentra un mayor número de apariciones, alcanzando picos de frecuencia en diversos documentos. No obstante, de cara a ser TT esto no es lo óptimo.

El hecho de que existan varios picos de frecuencia en determinados momentos es un factor a favor. Sin embargo esos picos de frecuencia no tienen un valor suficiente como para destacar sobre el resto de hashtags. Partiendo de un número de apariciones constante en un cierto intervalo, la distribución de la frecuencia en diversos instantes provoca que el valor que adquiere en cada uno de ellos sea menor.

El hashtag *'investidura'* tenía un número de picos mucho menor (prácticamente un único pico predominante), tomando así valores de frecuencia mucho mayores, lo que le hacía destacar notablemente sobre el resto.

Respecto al tercer hashtag, apreciamos que a lo largo de todos los documentos posee

un nivel más o menos constante en frecuencia, a excepción de algunos instantes en donde alcanzan picos mayores. El hecho de que posea un nivel constante bajo en frecuencia, junto con la aparición de algunos picos más altos con respecto a ese nivel, provoca que su puntuación TF-IDF sea de las más altas. Sin embargo, como se aprecia su carácter TT es casi nulo con respecto al hashtag *'investidura'*. Hashtags con una frecuencia de aparición casi constante a lo largo de todo el conjunto de tweets, son descartados de cara a convertirse en TT.

Por lo tanto además de la frecuencia, la distribución de cada una de las apariciones a lo largo de todos los documento es de vital importancia. Si observamos individualmente la puntuación IDF, podemos apreciar el peso que tiene este factor.

HASHTAG	IDF	HASHTAG	IDF
elcambioandaluz	0,9124805	eperiodico	1,40041463
ff	1,08326464	losdesayunosdetve	1,28006194
malaga	0,95351736	25m	1,49369036
andalucia	1,12571357	grinan	1,54146314
investidura	**2,09575035**	cmin	1,89617799
congreso	1,3175991	eres	1,45477229
sevillahoy	1,2158902	portadaepc	1,37905158
17congresopp	1,7355989	rajoy	1,72777356
yeswespainisdifferent	1,39317622	cofradiasmlg	1,8739016
38congresopsoe	1,80386373	votapp	1,69781034

Tabla 7. Puntuación IDF de los hashtags más frecuentes

Observamos cómo el hashtag *'investidura'* es el que mayor puntuación IDF tiene. A pesar de no ser el más frecuente, la ponderación que realiza la puntuación IDF le coloca como el TT de todo el conjunto de hashtag.

Por otro lado, apreciamos que la puntuación IDF de los hashtags menos frecuentes es superior a la de los tres primeros. Esto quiere decir que sus apariciones se concentran en un número menor de documentos y el valor frecuencial que se alcanzan en ellos es superior. Si tuviésemos en cuenta sólo esto, todos ellos deberían tener un carácter TT mayor con respecto a los primeros hashtags. Sin embargo, al ponderarse con su baja frecuencia, les deja fuera de la puja por ser TT.

Por todo ello es necesario cumplir ambos los requisitos: por una parte debe ser bastante frecuente y por otro debe concentrar gran parte o el total de sus apariciones en un número mínimo de documentos, o lo que es lo mismo, en un periodo temporal muy concreto y acotado.

Para determinar exactamente qué porcentaje de apariciones del TT se producen en un determinado periodo temporal, podemos hacer uso de la puntuación NTF.

4.1.1.1 Estudio del tamaño de la ventana NTF

Analizaremos la puntuación NTF del TT *'investidura'* en fechas cercanas al momento donde se concentra el máximo número de apariciones. Esto nos permitirá averiguar el porcentaje sobre el total de tweets que contienen dicho hashtag. Concretamente evaluaremos la ventana temporal definida desde la fecha '2011-12-16T10:53' hasta '2011-12-22T05:19'. Este intervalo comprende 4660 tweets.

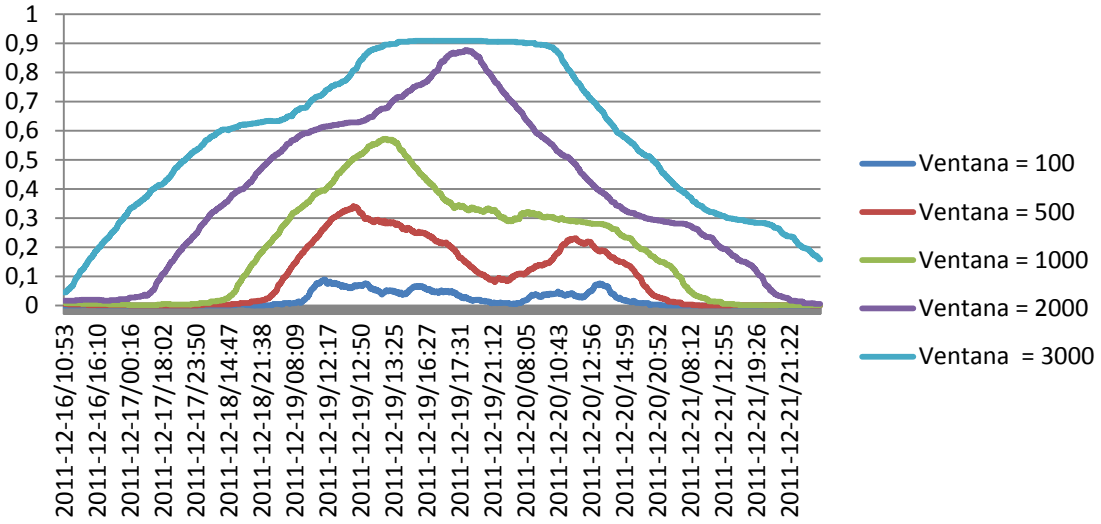

Figura 49. Estudio del tamaño de la ventana NTF

Según el tamaño de la ventana NTF obtenemos distintos resultados. A medida que aumentamos su tamaño, aumenta el porcentaje del total de tweets que contienen el hashtag 'investidura' dentro intervalo temporal analizado. El tamaño de la ventana restringe el número de tweets máximos en los que se pueden concentrar todas las apariciones.

Por este motivo en las situaciones donde la ventana adquiere un menor tamaño (100 y 500 tweets) observamos que el porcentaje de apariciones sobre el total, es muy bajo. Estamos obligando a que las apariciones se concentren en un número muy pequeño de tweets, y por lo tanto, en un periodo temporal muy breve.

Por el contrario, cuando el tamaño es mayor (3000 y 2000 tweets) obtenemos unos picos por encima del 80%. Esto quiere decir que alrededor del instante donde se producen esos picos, se concentran el 80% de las apariciones totales. El mayor tamaño

de la ventana permite abarcar un mayor número de tweets que contienen el término TT, y por lo tanto, un periodo temporal más amplio. Del total de 78401 tweets, en sólo 3000 se concentra el 90 % de las apariciones, lo que corrobora el fuerte carácter TT antes deducido.

Aumentando por encima de 3000 tweets el tamaño de la ventana conseguiríamos obtener un % algo mayor, sin embargo estaríamos perdiendo parte del carácter fugaz e instantáneo. A partir de un determinado tamaño, la puntuación NTF alcanza una cota máxima que indica el límite en donde la puntuación deja de cobrar sentido. En contrapartida, cuánto más pequeño sea el tamaño de la ventana, mayor carácter TT deberá tener el hashtag evaluado, siempre por encima de un tamaño mínimo que permita un análisis coherente.

4.1.2 N-GRAMAS

En este punto procederemos a analizar distintos valores de coapariciones de hashtags contenidos en un mismo tweet, en todo el conjunto de SEPLN.

Evaluando 2-gramas de hashtags obtenemos la siguiente tabla, ordenada de mayor a menor frecuencia.

2-GRAMAS	TF_i	2-GRAMAS	TF_i
malagasanta cofradiasmlg	84	yotrabajomanana yotrabajomanana	28
andalucia elcambioandaluz	83	propone votapp	25
ff ff	83	periodismo medios	22
cofradiasmlg malaga	55	firma sign	20
malaga elcambioandaluz	48	17congresopp comprometidosconespana	18
vespertina eperiodico	41	eres grinan	18
andalucia 25m	40	reformalaboral congreso	18
malaga malagasanta	39	rubalcaba chacon	18
queganechacon queganechacon	36	25m elcambioandaluz	17
ssanta12 sevillahoy	30	grinan elcambioandaluz	17

Tabla 8. Frecuencia de 2-gramas de hashtags

A primera vista, los bigramas de hashtags son mucho menos frecuentes que los ya analizados unigramas. Los términos internos de los bigramas más frecuentes se corresponden con alguno de los hashtags individuales más populares. Esto refleja que se sigue tratando el mismo tema de conversación.

Analizando su evolución a lo largo de todo el conjunto de tweets, obtenemos la siguiente figura.

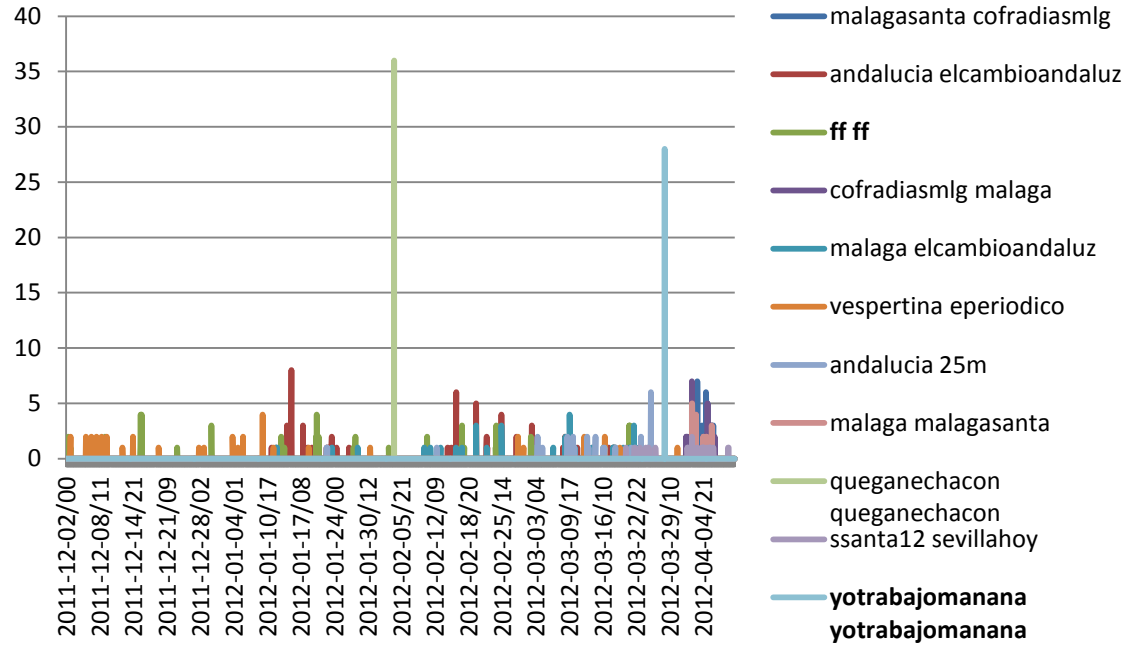

Figura 50. Puntuación TF_{ij} de bigramas de hashtags

A excepción de dos bigramas, el resto no supone ningún TT. Fijándonos bien, observamos que ambos picos se concentran en un único instante. Bigramas con un carácter TT tan fuerte, deberían haber tomado mayor protagonismo en el análisis de hashtags individuales, y dado que los valores que alcanzan son muy altos, el resultado obtenido se encuentra fuera de lo esperado.

De cara a averiguar qué sucede, analizamos 3-gramas y 4-gramas de hashtags. La siguiente tabla muestra los más frecuentes.

3-GRAMAS	TF_i	4-GRAMAS	TF_i
queganechacon queganechacon queganechacon	84	queganechacon queganechacon queganechacon queganechacon	126
yotrabajomanana yotrabajomanana yotrabajomanana	56	yotrabajomanana yotrabajomanana yotrabajomanana yotrabajomanana	70
firma firma sign	40	firma firma sign sign	60
malaga cofradiasmlg malagasanta	37	firma firma sign firma	40
firma sign sign	30	firma sign sign sign	20

Tabla 9. Frecuencia de 3-gramas y 4-gramas de hashtags

Evidentemente el comportamiento es totalmente anómalo. A medida que aumentamos el tamaño del n-grama obtenemos un valor frecuencial mucho mayor de los dos bigramas antes destacados. Los términos internos de cada uno de los trigramas y cuatrigramas son los mismos que lo de los bigramas, por lo que en las pruebas anteriores deberíamos haber obtenido un valor de frecuencia mucho mayor.

Se da la situación en la que dos tweets del conjunto están compuestos única y repetitivamente por el mismo hashtag (#queganechacon y #yotrabajomanana respectivamente). Esto provoca que al calcular las distintas combinaciones sin repetición, el número de n-gramas obtenidos se dispare a medida que aumentamos el tamaño del n-grama, lo que produce ese valor frecuencial tan elevado.

Analizando 3-gramas y 4-gramas de hashtags mediante la ventana deslizante, solucionamos este comportamiento, tal y como se observa en la siguiente tabla.

3-GRAMAS	TF_i	4-GRAMAS	TF_i
malagasanta cofradiasmlg malaga	**33**	firma firma sign sign	6
queganechacon queganechacon queganechacon	7	queganechacon queganechacon queganechacon queganechacon	6
yotrabajomanana yotrabajomanana yotrabajomanana	6	jlove danceagain idol fyitweet	5
andalucia votapp 25m	5	yotrabajomanana yotrabajomanana yotrabajomanana yotrabajomanana	5
danceagain idol fyitweet	5	massocialismo massocialismo massocialismo massocialismo	3
jlove danceagain idol	5	vespertina eperiodico fcblive pilotaor	3

Tabla 10. Frecuencia de 3-gramas y 4-gramas de hashtags con ventana deslizante

El cálculo de los n-gramas mediante la ventana deslizante corrige el comportamiento anterior. Ahora apreciamos la frecuencia real de las estructuras conflictivas.

Hay que destacar el valor tan alto en frecuencia (33) obtenido por el primer 3-grama. A medida que aumentamos el tamaño del n-grama, mayor dificultad tiene éste en conseguir un valor frecuencial alto. Esto se debe a que la repetición de una estructura de 3 términos es mucho menos probable. Por este motivo este trigrama merece una especial atención. Analizando su evolución temporal obtenemos la siguiente figura.

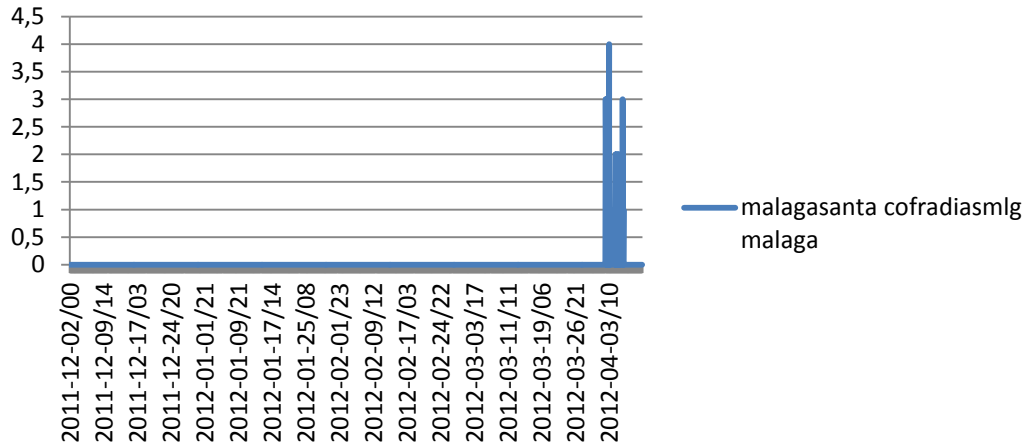

Figura 51. Puntuación TF_{ij} del trigrama 'malagasanta cofradiasmlg malaga'

A pesar de ser un trigrama con una frecuencia muy alta, su carácter TT es muy pequeño. Este comportamiento nos puede ofrecer información del tema sobre el que se trata, pero de cara a detectar TT no serviría.

Generalmente, a medida que aumentamos el tamaño del n-grama, el número de apariciones disminuye drásticamente. A partir de bigramas, la frecuencia de los n-gramas obtenidos se corresponde norlamente con algún tweet que contiene un número elevado de hashtags idénticos. Son muy pocas las ocasiones en las que n-gramas de tamaño elevado poseen una frecuencia alta debido a que distintos tweets contiene sus términos internos.

Esto indica por regla general, que sólo se suele usar un hashtag por tweet. Como consecuencia de ello el análisis de estructuras superiores a bigramas es prácticamente inútil de cara a detectar TT.

4.2 DETECCIÓN DE PALABRAS

En este apartado abordaremos la detección de palabras. De nuevo evaluaremos primero el caso de unigramas. Posteriormente examinaremos el resto de n-gramas en sus dos versiones de cálculo. En esta situación analizaremos sólo una parte del total de tweets de SEPLN. Concretamente, procesaremos el mismo conjunto de tweets que se examinó en el estudio de la ventana NTF, zona en donde se identifica el TT *'investidura'*, tal y como se concluyó anteriormente.

4.2.1 UNIGRAMAS

En el presente punto evaluaremos unigramas de palabras de todo el conjunto de tweets de SEPLN. Aplicando el algoritmo TF_i obtenemos la siguiente tabla de las palabras más frecuentes (ordenados de mayor a menor frecuencia).

PALABRA	TF_i	PALABRA	TF_i
rajoy	3374	congreso	1171
gracias	3228	ano	1160
gobierno	2541	rubalcaba	1140
dia	2441	cont	1092
pp	2326	andalucia	1071
manana	2213	buenas	1049
dias	2197	reforma	1010
psoe	1987	feliz	1002
ahora	1985	noche	916
madrid	1984	presidente	895
espana	1784	millones	864
buenos	1638	publico	850
mejor	1315	noches	829
anos	1302	deficit	824
nuevo	1279	partido	824
portada	1247	vez	823

Tabla 11. Palabras más frecuentes

Apreciamos que la frecuencia de los términos obtenidos es mayor que en el caso de detección de hashtags, algo totalmente lógico.

Esta lista depende directamente de la fase de preprocesado, concretamente de la eliminación de las stopwords. Detectar palabras obviando esta fase, carece totalmente de sentido. En esta situación obtendríamos únicamente una lista de términos sin significado, lo que sería inútil de cara a detectar los temas más tratados.

De igual forma, si obviamos el proceso de normalización en donde se eliminan acentos, mayúsculas, etc. Además de obtener palabras sin significado, no podríamos identificar unívocamente la totalidad de ellas, debido a sus distintas formas de representación.

Analizando la tabla obtenida podemos apreciar que una gran parte de los términos se corresponden con palabras relacionadas con el TT *'investidura'*. Palabras como *'rajoy', 'gobiero', 'pp', 'psoe'*, etc. son del mismo ámbito que el TT antes detectado, el ámbito de la política. Esto quiere decir, que además de ser popular en términos de hashtag, el TT *'investidura'* tiene un ámbito de repercusión mayor.

Por regla general, si un determinado tweet contiene un hashtag, el resto del contenido del tweet está íntimamente relacionado con él. Este comportamiento es el que se refleja en la tabla obtenida.

Evaluando su puntuación TF-IDF obtenemos los siguientes resultados.

PALABRA	TF-IDF	PALABRA	TF-IDF
rajoy	**1458,2646**	congreso	793,940278
gracias	1116,72544	ano	787,285096
gobierno	987,267771	rubalcaba	973,272225
dia	914,296661	cont	1057,17144
pp	1047,27267	andalucia	767,1506
manana	862,976796	buenas	761,810312
dias	1012,86366	reforma	742,15809
psoe	1003,73301	feliz	821,503114
ahora	834,857767	noche	643,192432
madrid	877,995062	presidente	678,670522
espana	800,323775	millones	643,13812
buenos	975,789054	publico	582,227147
mejor	696,951165	noches	717,334882
anos	712,264176	deficit	682,748782
nuevo	734,78968	partido	645,286488
portada	1039,37084	vez	580,9115

Tabla 12. Puntuación TF-IDF de las palabras más frecuentes

Varios de los términos relacionados con el TT *'investidura'* poseen las puntuaciones más altas, lo que nos muestra su acentuado carácter TT. La popularidad del hashtag se traslada al tema tratado en el tweet, y viceversa, lo que origina este comportamiento.

Si nos fijamos, términos como *'buenos'* y *'dias'* tienen una puntuación bastante alta. Estas palabras son el claro ejemplo de palabras muy frecuentes cuyo carácter TT es bajo, a pesar de obtener una puntuación TF-IDF alta. La expresión *'buenos dias'* es una frase utilizada diariamente por muchos usuarios, lo que le perjudica de cara a ser TT. La puntuación tan alta que obtiene se debe a la ponderación que realiza su alta frecuencia.

Para averiguar si los términos relacionados con el TT *'investidura'* adquieren su carácter popular justo en el mismo que éste, analizamos su evolución temporal mediante la puntuación TFij, tal y como se observa en la siguiente figura.

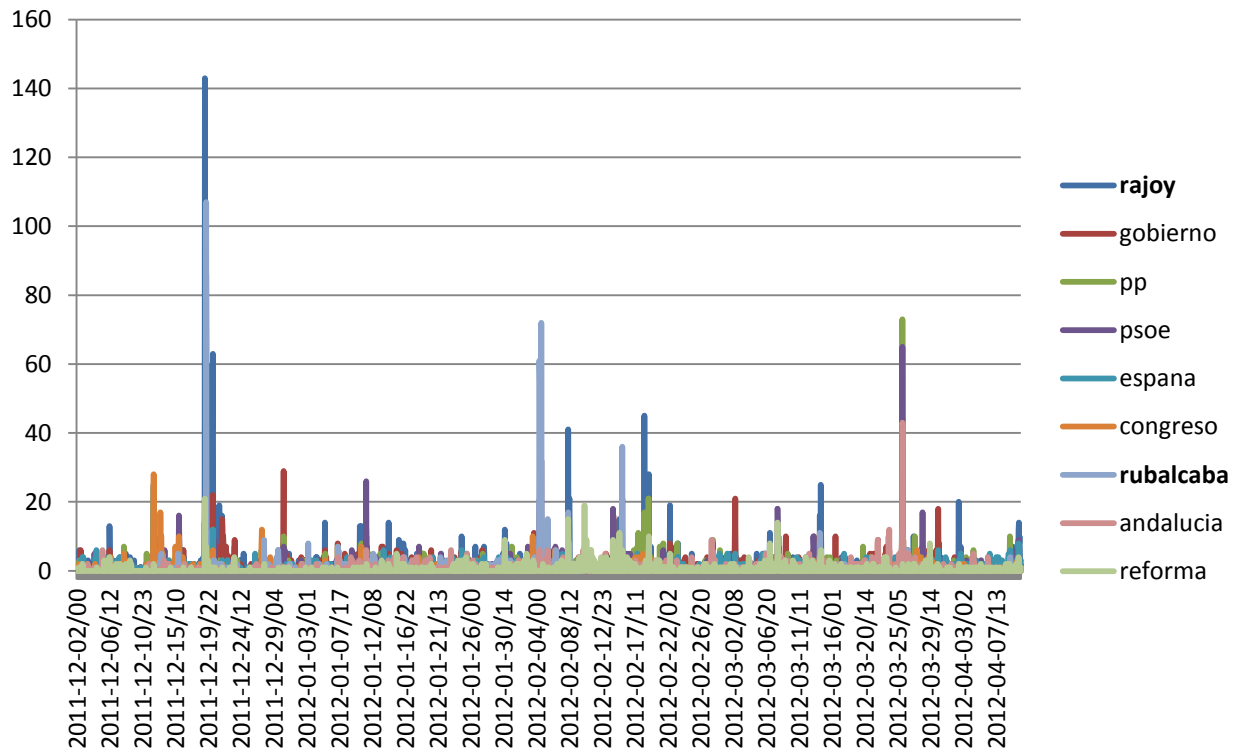

Figura 52. Puntuación TF_{ij} de las palabras más frecuentes

Por un lado, justo en el mismo instante del TT *'investidura'* se producen dos picos de valor frecuencial alto, correspondiente con los términos *'rajoy'* y *'rubalcaba'*. Por otro lado, podemos identificar dos instantes adicionales donde se producen otros dos picos (de menor frecuencia que el anterior). Uno se corresponde con el término *'rubalcaba'* y otro con los términos *'pp'* y *'psoe'*.

El análisis de palabras nos permite detectar nuevos términos TT. Los términos *'rajoy'* y *'rubalcaba'* adquieren un fuerte carácter popular en el mismo instante que el hashtag *'investidura'*. Esto refuerza el carácter TT del tema política tratado en este instante.

Posteriormente, además descubrimos otros dos términos, que a pesar de adquirir un nivel en frecuencia bastante menor, destacan por encima del resto de las palabras.

El término *'reforma'* adquiere la misma característica que el antes comentado *'buenos dias'*. Se trata de un concepto de uso casi diario, lo que le otorga un valor frecuencial cuasi constante a lo largo de todos los tweets. Esto provoca que sea uno de los términos más frecuentes pero que sin embargo, nunca pueda llegar a ser TT.

Para averiguar qué termino entre *'rajoy'* y *'rubalcaba'* posee un mayor carácter TT, evaluamos la puntuación NTF en el mismo periodo temporal definido anteriormente.

En este caso, serán los instantes de tiempo alrededor del pico de los dos términos a evaluar. Con una ventana de 2000 tweets, obtenemos la siguiente figura.

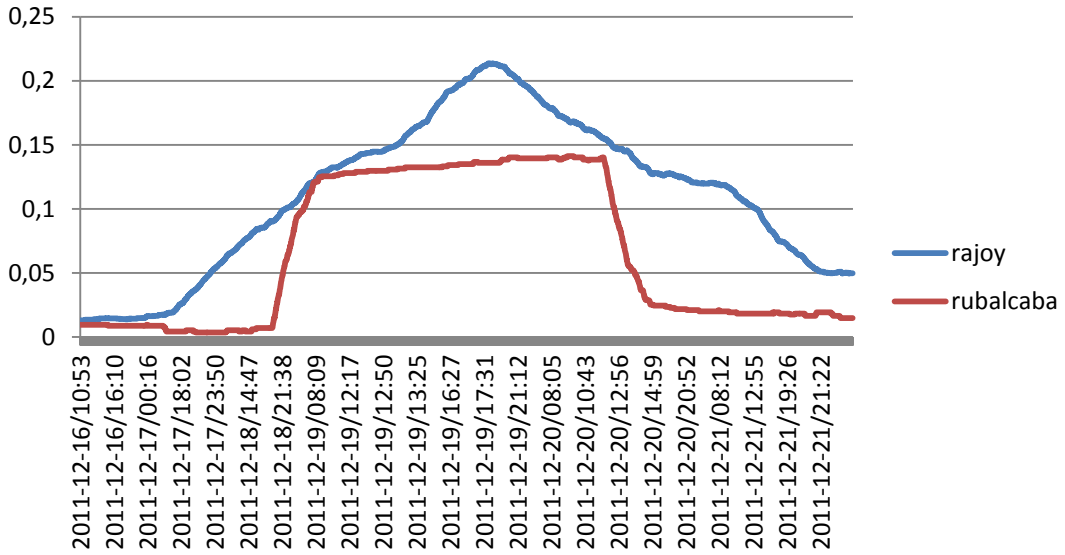

Figura 53. Puntuación NTF de los términos 'rajoy' y 'rubalcaba'

El término *'rajoy'* alcanza un pico de porcentaje mayor, sin embargo, ambos adquieren valores relativamente bajos respecto al caso de detección de hashtags. Al detectar palabras, es mucho más difícil que se encuentren concentradas en un único periodo temporal concreto. La frecuencia de aparición de ambos términos se encuentra más distribuida a lo largo de todos los tweets. A pesar de ello, englobando apenas el 20% del total de apariciones, el término 'rajoy', y 'rubalcaba' en menor medida, adquiere un fuerte carácter TT. La diferencia en frecuencia entre ambos términos, marca la diferencia en la puntuación NTF.

2.2 N-GRAMAS CON COMBINACIONES SIN REPETICIÓN

Para el cálculo de los distintos n-gramas nos centraremos en el periodo donde se produce el mayor pico de los términos *'rajoy'* y *'rubalcaba'*, definido entre las fechas '2011-12-16T10:53' y 2011-12-22T05:19'.

De esta forma, analizando coapariciones de dos palabras (2-gramas), obtenemos la siguiente tabla de los bigramas más frecuentes.

2-GRAMAS	FRECUENCIA	2-GRAMAS	FRECUENCIA
rajoy gobierno	107	espana rajoy	34
rajoy rubalcaba	**88**	rajoy pensiones	32
rajoy amaiur	79	rajoy zapatero	32
rajoy presidente	75	rajoy ley	31
discurso rajoy	60	investidura discurso	30
presidente gobierno	56	antiguedad rajoy	29
buenos dias	54	rajoy ahora	28
investidura rajoy	46	feliz navidad	27
rajoy pp	46	rajoy ministros	27
anuncia rajoy	42	nuevo gobierno	26
eta rajoy	40	antiguedad amaiur	25
debate rajoy	39	pnv rajoy	25
mariano rajoy	38	rajoy reforma	25
debate investidura	36	grupo amaiur	24
buenas noches	35	presidente espana	24

Tabla 13. Frecuencia de bigramas detectando palabras en un conjunto de tweets concreto

Casi el total de los bigramas más frecuentes en el intervalo analizado, están íntimamente relacionados con los TT que surgen en ese momento. Esto muestra el fuerte carácter popular que adquiere este tema en esos instantes. Podemos apreciar que el segundo bigrama más frecuente se corresponde con los dos términos, que en el caso de unigramas, producían un pico TT. Esto indica que en una gran cantidad de tweets ambos términos se repiten al mismo tiempo.

Evaluando la puntuación TF-IDF de los bigramas más frecuentes, obtenemos la siguiente tabla.

2-GRAMA	TF-IDF	2-GRAMA	TF-IDF
rajoy gobierno	65,11922186	investidura rajoy	35,20734027
rajoy rubalcaba	**95,54266596**	rajoy pp	37,99944092
rajoy amaiur	85,77125694	anuncia rajoy	47,33840153
rajoy presidente	49,85301003	eta rajoy	48,96059245
discurso rajoy	52,3099376	debate rajoy	33,0856967
presidente gobierno	42,86110989	mariano rajoy	32,2373455
buenos dias	40,33225206	debate investidura	30,54064311

Tabla 14. Puntuación TF-IDF de los bigramas más frecuentes detectando palabras en un conjunto de tweets concreto

Justo la puntuación TF-IDF más alta se corresponde con el bigrama formado por los dos términos que en el caso de unigramas, formaban dos picos TT en el mismo instante. Representando gráficamente la puntuación TF_{ij}, obtenemos la siguiente figura.

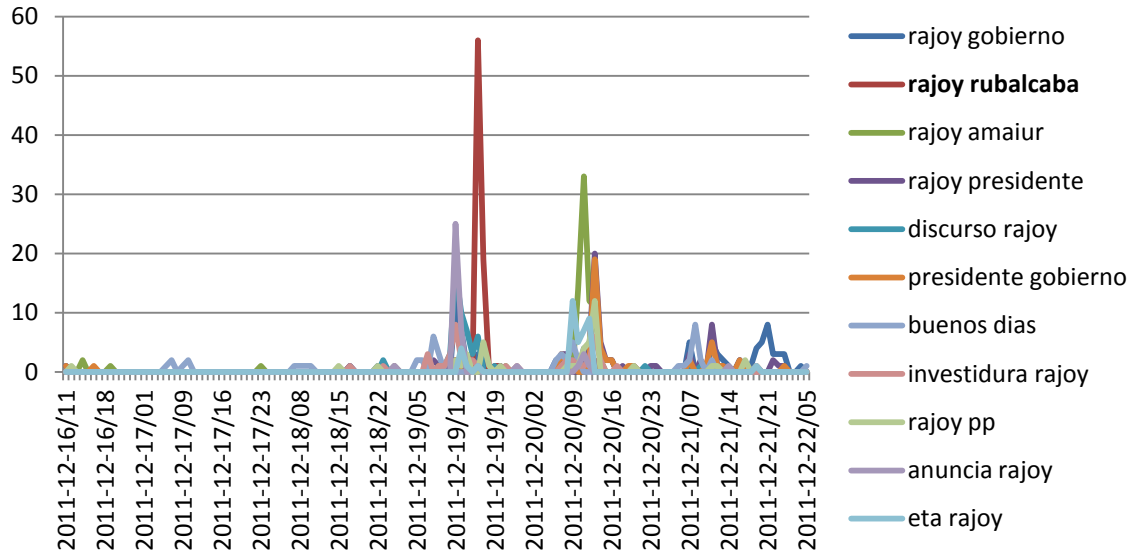

Figura 54. Puntuación TFij de los bigramas más frecuentes en un conjunto de tweets concreto

El hecho de que ambos términos sean TT individualmente justo en el mismo instante, provoca que al realizar en análisis de bigramas, obtengamos que el bigrama formado por *'rajoy'* y *'rubalcaba'* sea el que mayor carácter TT adquiera, a pesar de no ser el más frecuente.

Analizando la puntuación NTF obtenemos la siguiente figura.

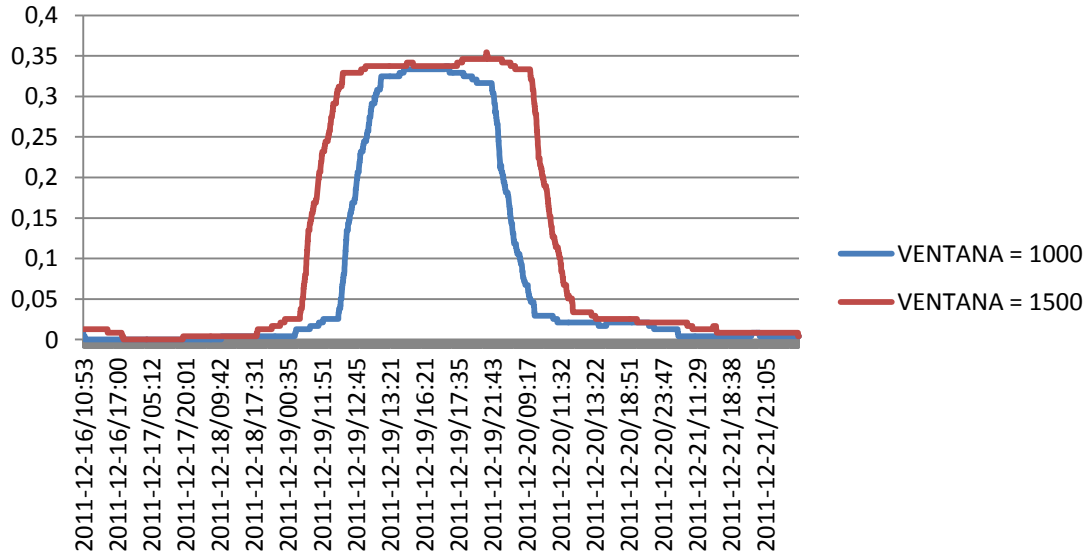

Figura 55. Puntuación NTF del bigrama 'rajoy rubalcaba' donde alcanza su pico TT máximo

Como máximo, el bigrama *'rajoy rubalcaba'* alcanza el 35% sobre el total de tweets que lo contienen. Este pico indica el punto en el que el bigrama alcanza su máximo de

popularidad. Este resultado sigue la misma tendencia que el análisis individual de los términos *'rajoy' y 'rubalcaba'*. En esa situación, cada término por separado alcanzaba como máximo un 20% sobre el total de apariciones.

En este caso, el bigrama formado por ambos términos obtiene una puntuación algo superior. Hay que tener en cuenta que al tratarse de 2-gramas, la probabilidad de coaparición de ambos términos es menor, por lo que aquellos instantes donde los términos tengan simultáneamente una mayor popularidad, serán aquellos momentos en donde el bigrama alcance cotas más altas.

Si analizamos el intervalo en el que identificábamos un segundo pico del término *'rubalcaba'* (definido por las fechas límite 2012-01-26T00:59 y 2012-02-22T02:52), podremos observar que el nivel NTF alcanzado es menor. Con una ventana de 1000 tweets obtenemos la siguiente figura.

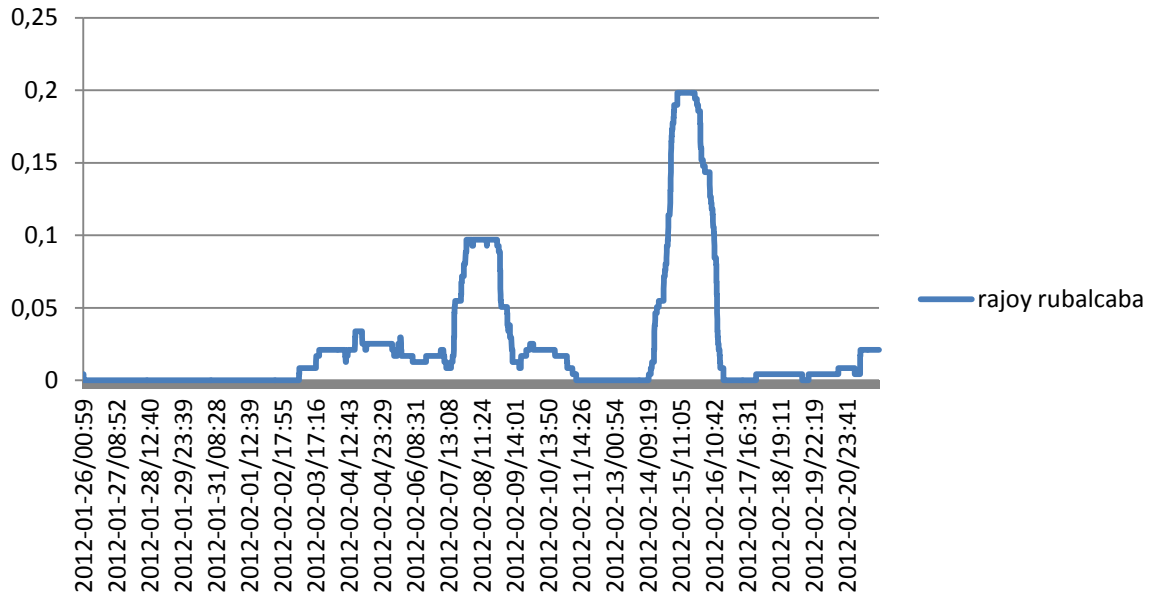

Figura 56. Puntuación NTF del bigrama 'rajoy rubalcaba' en la zona donde alcanza su segundo pico TT

En esos instantes, la popularidad simultánea de ambos términos era mucho menor. Por este motivo el bigrama *'rajoy rubalcaba'* en este caso, alcanza una cota máxima del 20% en un instante concreto. En el resto de momentos, observamos que la puntuación alcanzada es muy baja. Además, el comportamiento a lo largo de todo el periodo analizado no sigue un único patrón de subida de popularidad y posterior descenso de la misma, tal y como sucedía antes, si no que está formada por varios picos de menor valor.

Una vez realizado el análisis de bigramas en el periodo donde se alcanza los mayores TT, se ha visto que los resultados obtenidos, en menor medida, siguen el mismo comportamiento TT que los unigramas de palabras. En este punto podemos plantearnos la siguiente pregunta: ¿el análisis de bigramas en aquellas zonas donde no existen TT de palabras, permite descubrir nuevas estructuras tendencia. Para averiguarlo, a continuación se analiza el conjunto de tweets definido entre las fechas 2012-01-16T22 y 2012-01-30T14, periodo ausente de palabras TT.

Analizando la frecuencia global y la puntuación TF-IDF de cada uno de los bigramas más frecuentes en el citado periodo, obtenemos la siguiente tabla.

2-GRAMAS	FRECUENCIA	TF-IDF
buenos dias	172	103,793354
buenas noches	95	82,8680295
ping ping	72	179,679192
rubalcaba chacon	50	48,8515199
gracias cont	32	46,5328529
madrid barca	31	40,034155
junta andalucia	29	32,8806785
oe oe	29	58,5342694
descombacantes disponible	28	37,7836565
noticias descombacantes	28	103,793354

Tabla 15. Puntuación TF-IDF de los bigramas más frecuentes detectando palabras en un conjunto de tweets concreto

En este caso la frecuencia de los bigramas más usados en general es menor, a excepción de los 3 primeros bigramas. La ausencia de unigramas TT, provoca que bigramas como *'buenos dias' y 'buenas noches'* adquieran un mayor peso en términos de frecuencia.

Respecto a la puntuación TF-IDF se identifica un comportamiento anómalo del bigrama *'ping ping'*, alcanzando un valor por encima de los bigramas populares resultantes del análisis del periodo donde se producía el pico TT de palabras. Al igual que sucedió en el análisis de 3-gramas de hashtags, el hecho de calcular todas las combinaciones sin repetición puede crea una sensación irreal de bigramas tendencia en aquellos casos en los que ambos términos sean el mismo. Existen dos tweets, cuyo contenido repite continuamente la palabra 'ping'. Esto provoca que en un periodo muy concreto el número de n-gramas obtenido sea muy alto, y por lo tanto, la puntuación TF-IDF lo sea también. Si realizamos el mismo análisis mediante ventana deslizante, podremos

observar en la siguiente tabla, su verdadero protagonismo.

2-GRAMAS	TF_i	TF-IDF
buenos dias	172	103,793354
buenas noches	95	82,8680295
descombacantes disponible	28	37,7836565
noticias descombacantes	28	37,7836565
junta andalucia	27	31,1342847
rubalcaba chacon	27	32,8533499
ping ping	16	39,9287743

Tabla 16. Puntuación TF-IDF de los bigramas más frecuentes detectando palabras en un conjunto de tweets concreto con ventana deslizante

El bigrama *'ping ping'* ha desaparecido entre los más frecuentes, y por lo tanto ha adquirido la puntuación TF-IDF correcta. Una vez detectado y corregido el problema, evaluando la puntuación TFij de los bigramas más populares obtenemos la siguiente figura.

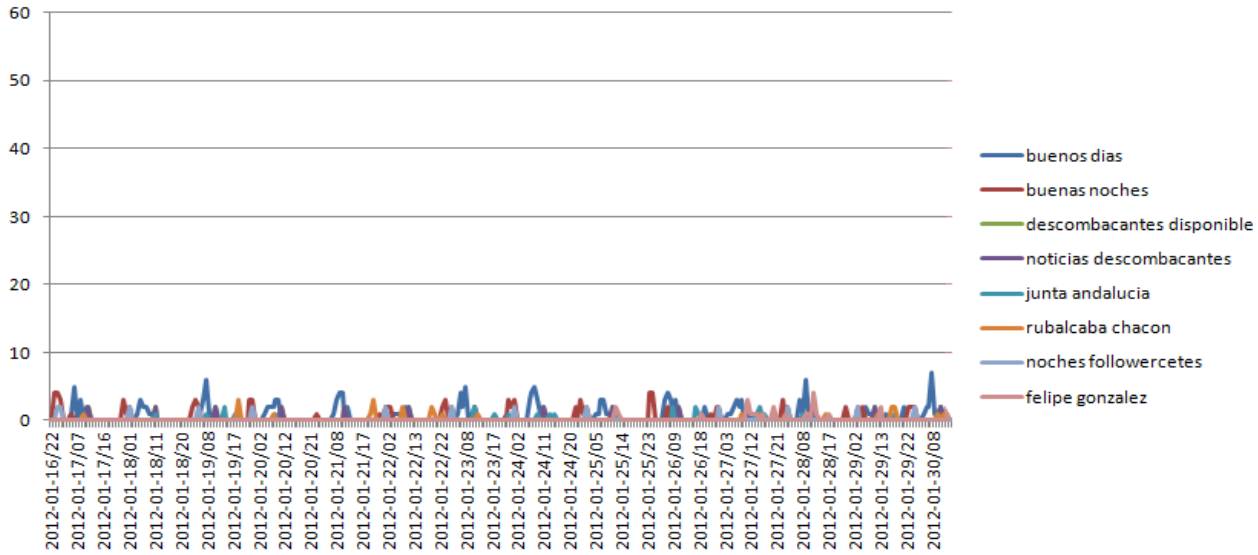

Figura 57. Puntuación TFij de los bigramas más frecuentes en la detección de palabras en un conjunto de tweet concreto, con ventana deslizante

El caráter TT de los bigramas más populares detectados en este nuevo periodo, es prácticamente nulo. Esto implica, que el análisis de bigramas en zonas en donde los unigramas no adquieren ningún carácter tendencia, no ofrece ningún tipo de ventaja. Es prácticamente imposible detectar TT en zonas donde unigramas no adquieren ni un mínimo de popularidad.

Por este motivo centramos el análisis en el instante donde se alcanza un payor pico TT de unigramas, y en donde por lo tanto, se puede obtener una mayor información.

Analizando 3-gramas en el periodo de interés, obtenemos la siguiente tabla.

3-GRAMAS	TF_i	TF-IDF
rajoy presidente gobierno	34	26,0027678
intencion subir impuestos	20	36,1235995
amb leo messi	18	32,5112395
discurso investidura rajoy	18	21,6741597
leo messi great	18	32,5112395
leo messi leo	18	32,5112395
leo messi messi	18	32,5112395
presidente mariano rajoy	17	25,5875496
antiguedad amaiur rajoy	16	24,0823997

Tabla 17. Frecuencia y puntuación TF-IDF de trigramas detectando palabras en un coniunto de tweets concreto

La detección de trigramas permite la emersión de estructuras que ofrecen un mayor nivel de información respecto al tema popular tratado en esos momentos. El trigrama *'intencion subir impuestos'*, quien justo posee el mayor carácter TT, es un claro ejemplo de ello. Además, este tipo de cálculo nos permite unificar varios de los bigramas más frecuentes antes detectados. El trigrama *'rajoy presidente gobierno'*, quien agrupa los bigramas *'rajoy presidente', 'rajoy gobierno' y 'presidente gobierno'*, es un ejemplo.

Se puede observar que los trigramas relacionados con *'leo messi'* adquieren en este tipo de estructura un carácter TT bastante alto respecto al máximo definido por *'intencion subir impuestos'*. A pesar de ello, no suponen estructuras de interés de cara a detectar TT ya que en los análisis previos no adquirían ni un mínimo de importancia.

Ahora bien, ¿merece la pena calcular todas las combinaciones sin repetición para el cálculo de n-gramas, pudiendo formarlos a través de una ventana deslizante? Para averiguar si se obtiene el mismo comportamiento, en el siguiente apartado abordamos la detección a través de n-gramas calculados mediante la ventana deslizante.

4.2.3 N-GRAMAS CON VENTANA DESLIZANTE

Centrándonos, al igual que en el caso anterior, en el periodo donde se concentra el mayor pico TT de palabras, obtenemos la siguiente lista de los bigramas más frecuentes.

2-GRAMAS	TF_i	2-GRAMAS	TF_i
buenos dias	53	nuevo gobierno	20
presidente gobierno	48	rosa diez	20
mariano rajoy	36	kim jong	17
debate investidura	34	antiguedad rajoy	16
rajoy anuncia	34	consejo ministros	16
discurso rajoy	33	tribuna invitados	16
buenas noches	32	vaclav havel	16
rajoy presidente	28	antiguedad amaiur	15
gobierno rajoy	24	gobierno espana	15
investidura discurso	24	jong il	15
investidura rajoy	23	reforma laboral	15
rajoy amaiur	22	subir impuestos	15
feliz navidad	21	victimas terrorismo	15
rubalcaba rajoy	**21**	banco malo	14

Tabla 18. Frecuencia de bigramas (con ventana deslizante) detectando palabras en un conjunto de tweets concreto

A primera vista observamos que el bigrama *'rubalcaba rajoy'* adquiere una frecuencia mucho menor que en el cálculo anterior.

El cálculo de 2-gramas mediante ventana deslizante permite detectar bigramas correspondiente con expresiones típicas ya formalizadas y nombres propios, tales como: *'buenas noches ,'mariano rajoy' ,'feliz navidad', 'victimas terrorismo', 'rosa diez',* etc. Por este motivo el bigrama *'buenos dias'* es el más frecuente en todo el conjunto de tweets analizados.

Esto provoca que el carácter TT del bigrama *'rubalcaba rajoy'* haya descendido bruscamente, tal y como se observa en la siguiente tabla.

2-GRAMA	TF-IDF	2-GRAMA	TF-IDF
buenos dias	39,5853585	rajoy presidente	26,6283791
presidente gobierno	38,6345042	gobierno rajoy	24,3158747
mariano rajoy	31,3859626	investidura discurso	24,3158747
debate investidura	28,8439407	investidura rajoy	20,6230852
rajoy anuncia	**45,8644206**	rajoy amaiur	31,4189655
discurso rajoy	34,5814772	feliz navidad	23,6692008
buenas noches	28,6929881	**rubalcaba rajoy**	**28,3280245**

Tabla 19. Puntuación TF-IDF de bigramas (con ventana deslizante) detectando palabras en un conjunto de tweets concreto

En contrapartida, han surgido nuevos bigramas cuyo carácter TT es mucho mayor que

el bigrama antes mencionado. Evaluando su evolución temporal, obtenemos la siguiente figura.

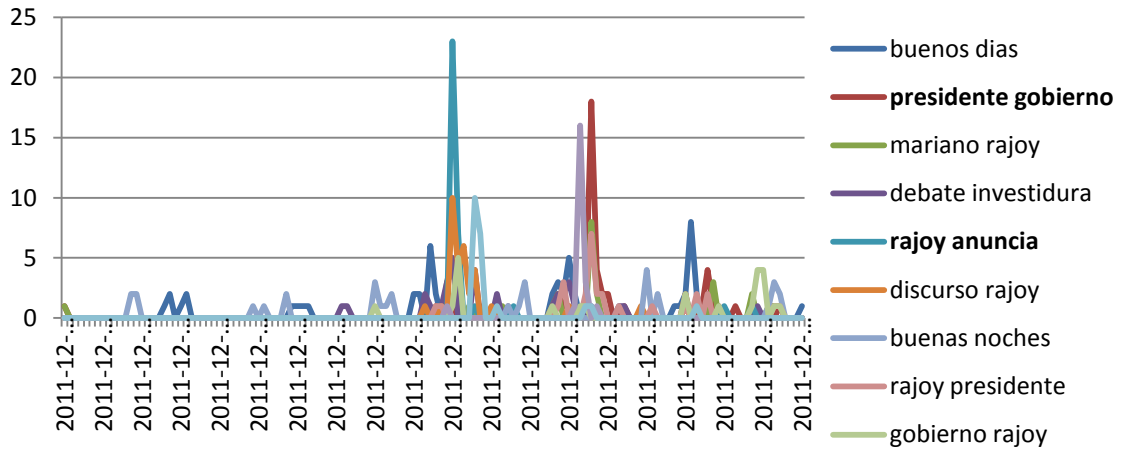

Figura 58. Puntuación TFij de los bigramas (con ventana deslizante) más frecuentes detectando palabras en un conjunto de tweets concreto

Tal y como apreciamos, se producen nuevos picos con este tipo de cálculo. Sin embargo, su carácter TT es mucho menor como consecuencia de su menor frecuencia. El hecho de calcular bigramas con una ventana deslizante, limita la identificación de las distintas combinaciones, ya que sólo se tienen en cuenta términos vecinos. Por este motivo ahora, por un lado, detectamos nuevos bigramas, y por otro, obtenemos una menor frecuencia general.

Analizando 3-gramas y 4-gramas obtenemos la siguiente tabla de n-gramas más frecuentes.

3-GRAMAS	TF_i	4-GRAMAS	TF_i
kim jong il	15	xii edicion premios culturas	12
disponible historias dia	13	edicion premios culturas extremadura	11
presidente gobierno espana	13	descombacantes disponible historias dia	8
edicion premios culturas	12	noticias descombacantes disponible historias	8
intencion subir impuestos	12	crea empleo crece libertad	7
noticias descombacantes disponible	12	alert real vs sevilla	6
xii edicion premios	12	carcel alava trasladado etarras	6
premios culturas extremadura	11	cr7 goal alert real	6
amaiur inaki antiguedad	9	fotos nueva carcel alava	6
amaiur antiguedad rajoy	8	goal alert real vs	6

Tabla 20. Frecuencia de 3-gramas y 4-gramas de palabras (con ventana deslizante) detectando palabras en un conjunto de tweets concreto

Varios de los 3-gramas más frecuentes están relacionados con el tema TT origen. De nuevo este tipo de estructura nos ofrece un nivel mayor de información acerca del tema tratado. En este caso el 3-grama *'presidente gobierno espana'* hace referencia al TT *'rajoy'*. De igual forma que *'intencion subir impuestos'* está relacionado con *rajoy', 'rubalcaba'* y el debate político.

Al mismo tiempo, emergen nuevos conceptos que en los n-gramas previos no se habían identificado. Mediante 3-gramas, podemos apreciar *'edicion premios culturas'* como un nuevo tema tratado, que se apoya en los resultado obtenido en 4-gramas: *'xii edición premios culturas' y 'edicion premios culturas extremadura'*. En este último caso el tema popular político, adquiere menor importancia.

4.3 DETECCIÓN DE HASHTAGS Y PALABRAS IDÉNTICAS A ELLOS

En este apartado detectaremos los hashtags y sólo aquellas palabras que sean idénticas a ellos, en el conjunto de todos los tweets de SEPLN. Evaluaremos dos casuísticas distintas: una en la que sólo se analizan las palabras en aquellos tweets que contienen hashtags y otra que se analizan las palabras de todos los tweets.

4.3.1 TWEETS CON HASHTAG

En este punto se analizan sólo las palabras de aquellos tweets que contienen al menos un hashtag. En aquellos tweets que no contengan ningún hashtag, no se tendrán en cuenta las palabras. Con esto veremos si aquellos tweets ya poseen almenos un hashtag, lo suelen contener también como palabra.

En la siguiente tabla se muestra una comparativa de la puntuación TF_i entre la detección única de hashtag y la detección de hashtag y palabras idénticas a ellos en tweets con hashtag.

HASHTAGS	TF_i	TF_i
elcambioandaluz	860	860
ff	674	678
malaga	540	568
andalucia	405	414
investidura	403	410
congreso	385	393
sevillahoy	271	271
17congresopp	250	250
yeswespainisdifferent	230	230
38congresopsoe	216	216
eperiodico	177	177
losdesayunosdetve	176	176
25m	165	165
grinan	142	144

Tabla 21. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en tweets con hashtags (tweets SEPLN)

Apreciamos que sólo una parte de los términos ha aumentado su frecuencia, y en esos casos, el aumento ha sido casi despreciable respecto a la detección única de hashtags. Gráficamente se aprecia a la perfección.

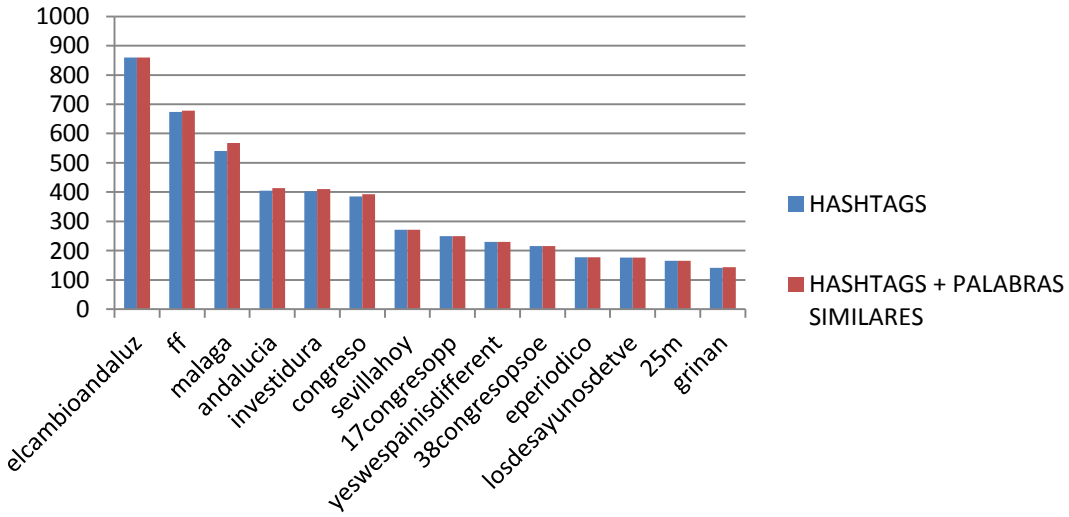

Figura 59. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en tweets con hashtags (tweets SEPLN)

Para comprobar si este comportamiento se repite por regla general, vamos a analizar otro conjunto de tweets. En este caso vamos a evaluar los tweets del fichero *"tweets.2013.txt"*, formado, como ya explicamos al comienzo, por tweets de una serie de personalidades españolas.

En la siguiente tabla se muestra la misma comparativa que en la anterior.

HASHTAGS	TF_i	TF_i
alcorcon	10546	10787
madrid	638	656
ligaadelante	609	610
realmadrid	445	455
vamosalcor	309	309
todosconelcastilla	239	239
yoconfioecf	205	205
psoe	196	197
parla	189	192
40principales	147	150
mostoles	141	147
fuenlabrada	138	138
123alcorcon	132	132
cocoestaperdido	125	125

Tabla 22. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en tweets con hashtag (tweets 2013)

Representando gráficamente, obtenemos la siguiente figura.

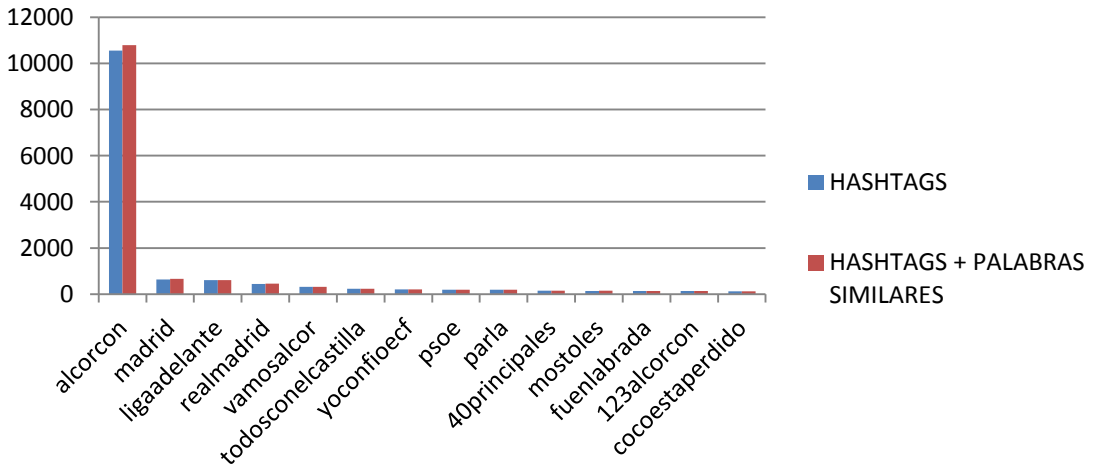

Figura 60. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en tweets con hash (tweets 2013)

Se puede observar que en ambos conjuntos de tweets sólo una parte de los términos ha visto modificada su frecuencia y ese incremento ha sido muy pequeño. En el caso de los tweets 2013, el hashtag *'alcorcon'* ha aumentado su frecuencia en más de 200 unidades. Sin embargo esto, en comparación con su frecuencia total, es una muy pequeña porción.

Por regla general, una parte de los hashtags no se corresponden directamente con palabras. Suelen ser estructuras que unen varios términos, por lo que su contenido no sigue ningún patrón definido. Esto provoca que el contenido del tweet no contenga exactamente el hashtag al que hace referencia. Sólo en aquellos tweets en el que el hashtag se corresponde con el formato de una palabra, se aprecia un mayor uso en su contenido. A pesar de ello en esta última situación, que un tweet contenga al mismo tiempo un hashtag y su equivalente en palabra, no es algo muy común según los resultados obtenidos.

4.3.2 TODOS LOS TWEETS

En este caso se realiza un análisis detectando todas las palabras de todos los tweets. Aquellas palabras que se correspondan con alguno de todos los hashtags detectados, se tendrán en cuenta.

En la siguiente tabla se muestra una comparativa de la frecuencia de los hashtags más populares detectando únicamente hashtags, con su frecuencia según el método actual evaluado.

HASHTAGS	TF_i	TF_i
elcambioandaluz	860	860
ff	674	697
malaga	540	1134
andalucia	405	1476
investidura	403	410
congreso	385	1556
sevillahoy	271	271
17congresopp	250	253
yeswespainisdifferent	230	230
38congresopsoe	216	216
eperiodico	177	177
losdesayunosdetve	176	176
25m	165	204
grinan	142	480

Tabla 23. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en todos los tweets (tweets 2013)

Gráficamente, el resultado es el siguiente.

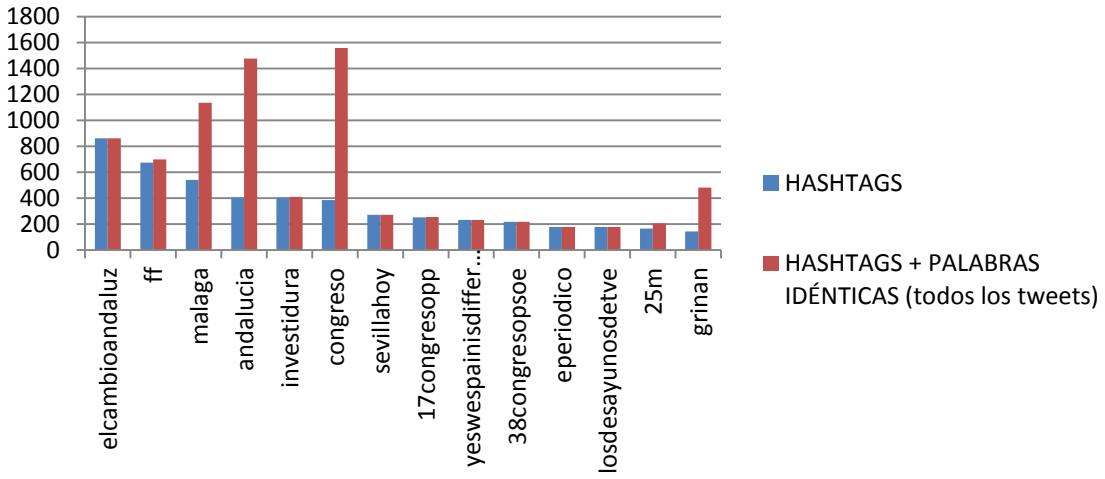

Figura 61. Comparación entre la frecuencia detectando hashtagas y detectando hashtags más palabras idénticas en todos los tweets (tweets SEPLN)

Podemos apreciar que varios de los hashtags se han visto incrementados en gran medida. Por este motivo, la lista de los términos más frecuentes ha cambiado sustancialmente, tal y como se observa a continuación.

HASHTAGS + PALABRAS IDENTICAS (todos los tweets)	TF_i	HASHTAGS + PALABRAS IDENTICAS (todos los tweets)	TF_i
rajoy	3498	presidente	896
gracias	3233	publico	875
gobierno	2553	elcambioandaluz	860
dia	2442	deficit	836
pp	2335	mundo	794
madrid	2056	twitter	752
psoe	2013	politica	738
ahora	2002	foto	702
espana	1801	empleo	698
congreso	1556	ff	697
andalucia	1476	chacon	690
portada	1249	crisis	687
rubalcaba	1167	trabajo	662
malaga	1134	enhorabuena	655
feliz	1006	debate	613

Tabla 24. Términos más frecuentes en la detección de hashtags y palabras idénticas a ellos en todos los tweets

Podemos apreciar que muchos de los términos obtenidos son los mismos que en la detección de palabras. Sin embargo, solo aquellas palabras que tengan su equivalente

en hashtag, se tendrán en cuenta y por lo tanto, podrán optar a ser términos populares.

Por un lado, aquellos hashtags con carácter TT seguirán manteniendo su popularidad, a pesar de no haber aumentado su frecuencia, como es el caso de *'elcambioandaluz'*. Por otro lado, aquellos hashtags cuyo carácter TT no destacaba antes, ahora se pueden convertir en términos muy frecuentes, y a lo mejor en populares.

Analizando la puntuación TFij de los términos con puntuación TF-IDF más alta y de los hashtags cuyo carácter TT era mayor, obtenemos la siguiente figura.

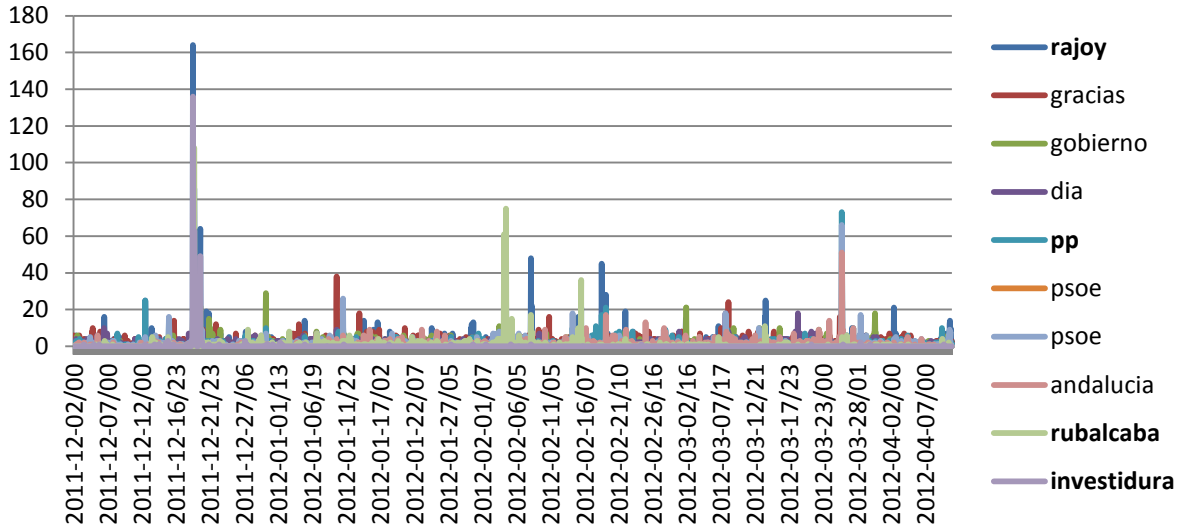

Figura 62. Puntuación TFij de los términos más frecuentes en la detección de hashtags y palabras idénticas a ellos en todos los tweets

Respecto a convertirse en términos populares, 'rajoy' y 'rubalcaba' son claros ejemplos de hashtags que no tenían ningún tipo de popularidad y ahora adquieren un fuerte carácter TT. El pico de popularidad alcanzado ahora es mayor que en el caso de detección de palabras, ya que se ha tenido en cuenta también los hashtags. Términos como *'gracias'* y *'dia'* son un claro ejemplo de hashtags que según esta detección, se encuentran entre los más frecuentes. Sin embargo su carácter TT es muy bajo. Aquellos hashtags que se correspondan con palabras de uso común serán muy frecuentes pero adoptarán un carácter TT bajo.

Por lo tanto, este tipo de detección permite descubrir, además de los hashtags populares, aquellas palabras tendencia que en su equivalente en hashtag puede que no destacasen. Aquellos términos muy frecuentes pero de carácter TT bajo, no perjudican la detección de TT.

La evaluación de todas las palabras de todos los tweets supone una gran diferencia con respecto al anterior análisis. El hecho de que un tweet no contenga ningún hashtag, no indica que su contenido no esté relacionado con el tema popular del momento. De hecho, en los casos en los que los hashtags adopten una estructura sencilla, aquellos tweets que no contienen ningún hashtag tienen una mayor probabilidad de contener palabras idénticas a esos hashtags respecto a los tweets que si contienen alguno.

4.4. DETECCIÓN DE HASHTAGS Y PALABRAS

En este apartado se evalúa la detección identificando hashtags y palabras al mismo tiempo. Esto nos permitirá observar si existe alguna diferencia respecto a la detección única de palabras y la detección de hashtags y palabras idénticas a ellos

4.4.1 UNIGRAMAS

En este punto sólo se tendrán en cuenta los unigramas. En la siguiente tabla se muestra una comparativa entre los términos más frecuentes según la detección actual y la detección única de palabras antes evaluada.

TÉRMINO	TF_i palabras	TF_i pal + hashtags	TÉRMINO	TF_i palabras	TF_i pal + hashtags
rajoy	3374	3498	congreso	1171	1556
gracias	3228	3233	ano	1160	1160
gobierno	2541	2553	rubalcaba	1140	1167
dia	2441	2442	cont	1092	1092
pp	2326	2335	**andalucia**	**1071**	**1476**
manana	2213	2213	buenas	1049	1049
dias	2197	2197	reforma	1010	1010
psoe	1987	2013	feliz	1002	1006
ahora	1985	2002	noche	916	916
madrid	1984	2056	presidente	895	896
espana	1784	1801	millones	864	864
buenos	1638	1638	**elcambioandaluz**		**860**
mejor	1315	1315	publico	850	875
anos	1302	1302	noches	829	829
nuevo	1279	1279	deficit	824	836
portada	1247	1249	partido	824	824

Tabla 25. Comparación de la frecuencia según la detección única de palabras y la de palabras más hashtags

Podemos observar que la gran mayoría se corresponde con los términos obtenidos en la detección única de palabras. En este caso, varios de ellos se han visto incrementados en frecuencia al tener en cuenta también los hashtags. Además, respecto a la detección de hashtags y palabras idénticas a ellos en todos los tweets, observamos que se han incorporado nuevos términos como *'manana','dias','buenos',* etc. Al tener en cuenta todas las palabras, aquellos términos de uso común que no tienen su equivalente en hashtag y que antes se ignoraban, ahora aparecen entre los más frecuentes.

Por un lado, aquellas palabras que se correspondan con hashtags populares, sufrirán un mayor incremento, tal y como sucede con el término *'andalucia'* (cuya frecuencia ha aumentado en más de 400 unidades). Este comportamiento ya lo habíamos obtenido previamente en la segunda versión de detección de hashtags y palabras idénticas a ellos, en donde obteníamos el mismo valor frecuencial que en este caso.

Por otro lado, aquellos hashtags que sean muy populares y que no tengan su equivalente en palabra, se pueden ver incorporados a la lista de términos más frecuentes según esta detección, tal y como sucede con el hashtag *'elcambioandaluz'*. En la evaluación única de palabras, este hashtag no tenía su equivalente en palabra, sin embargo en la detección simultánea de hashtags y palabras ha emergido como uno de los términos más frecuentes, estando por encima de muchos otros términos.

No obstante, hay que tener en cuenta que la frecuencia de los hashtags es mucho menor que la de las palabras. Por ello es muy poco probable que aparezcan en la lista de términos más frecuentes usando la detección simultánea de palabras y hashtags. Esto implica que, de cara a detectar TT, tengamos casi el mismo comportamiento que en el caso de detección de palabras, a excepción de aquellos hashtags cuyo carácter TT sea muy acentuado.

4.4.2 N-GRAMAS CON COMBINACIONES SIN REPETICIÓN

Analizando 2-gramas de palabras y hashtagas en los instantes cercanos al pico TT de *'rajoy'*, obtenemos la siguiente tabla de los bigramas más frecuentes.

2-GRAMAS	TF_i	2-GRAMAS	TF_i
investidura rajoy	262	anuncia rajoy	42
rajoy gobierno	109	debate rajoy	41
rajoy rubalcaba	89	amaiur investidura	40
rajoy amaiur	81	mariano rajoy	40
rajoy presidente	77	buenas noches	35
discurso rajoy	62	espana rajoy	35
investidura rubalcaba	62	rajoy pensiones	35
debate investidura	60	rajoy ley	32
investidura discurso	60	rajoy zapatero	32
presidente gobierno	56	rajoy ministros	31
buenos dias	54	rajoy ahora	30
rajoy pp	48	antiguedad rajoy	29
eta rajoy	45	congreso rajoy	27
rajoy investidurarajoy	43	feliz navidad	27

Tabla 26. Frecuencia de 2-gramas en la detección de palabras y hashtags en un conjunto de tweets concreto

Este análisis nos ofrece mucha más información que en el caso de unigramas. En este caso, el hashtag TT *'investidura'* aparece en algunos de los bigramas más frecuentes. Al detectar simultáneamente hashtags y palabras, emerge el TT surgido en la detección única de hashtags. Concretamente el bigrama más popular une dos términos que individualmente eran palabras tendencia: por un lado el TT *'investidura'*, se trataba del hashtag con mayor carácter TT, y por otro el término *'rajoy'*, que en la detección única de palabras lideraba la lista de los más frecuentes, elevándose a palabra TT.

El bigrama *'rajoy rubalcaba'* aparece de nuevo como uno de los más frecuentes, con una frecuencia casi idéntica a la de la detección de palabras. Esto indica que aquellos bigramas que eran populares en la detección de palabras, seguirán apareciendo en este tipo de detección. Analizando su puntuación TF-IDF, obtenemos la siguiente tabla.

2-GRAMA	TF-IDF	2-GRAMA	TF-IDF
investidura rajoy	**174,153182**	debate investidura	47,0809271
rajoy gobierno	66,3364036	investidura discurso	55,3790889
rajoy rubalcaba	96,6283781	presidente gobierno	42,8611099
rajoy amaiur	87,9426812	buenos dias	40,3322521
rajoy presidente	50,0487329	rajoy pp	39,6515905
discurso rajoy	54,0536022	eta rajoy	50,7197159
investidura rubalcaba	79,484419	rajoy investidurarajoy	45,0607127

Tabla 27. Puntuación TF-IDF de bigramas en la detección de palabras y hashtags de un conjunto de tweets concreto

El 2-grama con mayor puntuación justo se corresponde con el bigrama cuyos dos términos eran TT individualmente. Además, aquellos bigramas que tenían una buena puntuación en la detección de palabras, siguen conservándola. Representando la evolución temporal de los bigramas que nos interesan, obtenemos la siguiente figura.

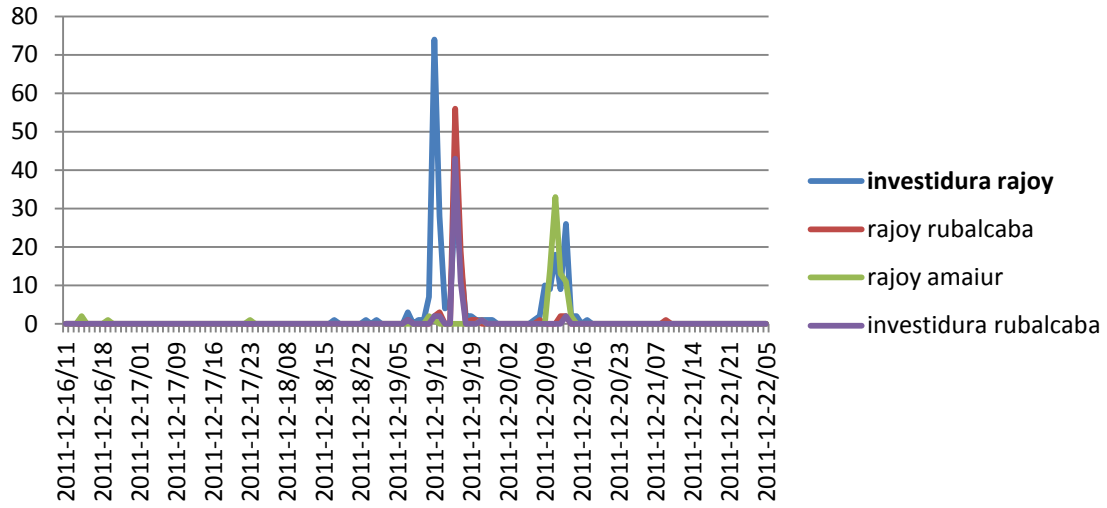

Figura 63. Puntuación TFij de los bigramas más populares en la detección de palabras y hashtags en un conjunto de tweets concreto

En este caso, el valor de la puntuación TFij está marcada por el carácter TT de cada uno de los términos que forman el bigrama. La popularidad de los 2-gramas que estén formados por una palabra y por un hashtag, dependerá del nivel TT de cada uno de ellos. En caso de que uno de los términos no adquiera un gran nivel de popularidad, el carácter TT del bigrama se verá compensado por la popularidad del otro término.

Este es el caso del bigrama *'investidura rubalcaba'* en el que el hashtag *'investidura'* compensa la menor popularidad de la palabra '*rubalcaba'* Los bigramas compuestos únicamente por palabras mantienen el mismo carácter TT en este tipo de detección, como es el caso de *'rajoy rubalcaba'* y *'rajoy amaiur'.*

Si realizamos un análisis de la frecuencia y puntuación TF-IDF de los trigramas identificados en el mismo periodo temporal, obtenemos la siguiente tabla.

3-GRAMAS	TF_i	TF-IDF
investidura rajoy rubalcaba	37	66,3184925
discurso investidura rajoy	34	40,4712777
rajoy presidente gobierno	34	25,5339661
investidura rajoy amaiur	25	44,8097922
intencion subir impuestos	20	35,8478338
debate investidura rajoy	19	19,2705683
amb leo messi	18	32,2630504
leo messi great	18	32,2630504
leo messi leo	18	32,2630504

Tabla 28. Frecuencia y puntuación TF-IDF de trigramas detectando palabras y hashtags en un conjunto de tweets concreto

El tener en cuenta además de palabras, hashtags, ha supuesto que la totalidad de los trigramas más frecuentes estén formados por términos populares relacionados con el tema político, respecto a los trigramas obtenidos en el caso de detección única de palabras. Esto ha provocado que los trigramas relacionados con '*leo messi*' hayan perdido presencia como estructuras más frecuentes, a pesar de haber obtenido la misma puntuación.

En este caso, aumenta significativamente la presencia del hashtag TT *'investidura'*. Los dos trigramas con mayor puntuación TF-IDF poseen como uno de sus términos internos al recién mencionado hashtag TT. Concretamente, el trigrama más popular unifica tres de los bigramas más frecuentes: *'rajoy rubalcaba', 'investidura rajoy' y 'investidura rubalcaba'*. Al igual que en bigramas, el grado de popularidad de los términos internos del trigrama marca la popularidad del 3-grama resultante. Esto es lo que ha originado que el trigrama *'investidura rajoy rubalcaba'* obtenga la mayor frecuencia y puntuación TF-IDF, y por lo tanto el mayor carácter TT.

4.4.3 N-GRAMAS CON VENTANA DESLIZANTE

Si analizamos 2-gramas de hashtags y palabras mediante la ventana deslizante, obtenemos la siguiente tabla de los bigramas más frecuentes.

2-GRAMAS	TF_i	2-GRAMAS	TF_i
investidura rajoy	**63**	feliz navidad	21
buenos dias	53	nuevo gobierno	20
debate investidura	51	rosa diez	20
presidente gobierno	48	rubalcaba rajoy	20
investidura discurso	38	vaclav havel	18
mariano rajoy	37	kim jong	17
rajoy anuncia	34	antiguedad amaiur	16
buenas noches	32	antiguedad rajoy	16
discurso rajoy	30	consejo ministros	16
gobierno rajoy	25	tribuna invitados	16
rajoy presidente	25	gobierno espana	15
rajoy amaiur	22	jong il	15

Tabla 29. Bigramas más frecuentes, con ventana deslizante, en la detección de hashtags y palabras en un conjunto de tweets concreto

A través de la ventana deslizante también obtenemos el hashtag TT *'investidura'* en algunos de los bigramas más frecuentes, aunque en este caso el número de bigramas que lo contienen es menor. Además, la frecuencia de éstos es bastante más baja. El bigrama *'investidura rajoy'* repite como bigrama más frecuente, sin embargo, con respecto al cálculo de n-gramas con combinaciones sin repetición, su frecuencia ha disminuido en gran medida. Asimismo, el bigrama formado por las palabras populares *'rajoy'* y *'rubalcaba'* se ha visto perjudicado con este tipo de cálculo.

Esto provoca que bigramas como *'buenos dias', 'buenas noches'* y nombres propios como *'mariano rajoy' y 'rosa diez'* adopten una mayor importancia en términos de frecuencia.

Sin embargo, los bigramas con mayor carácter TT se siguen correspondiendo con aquellos que están formados por términos individuales relacionados con el tema popular que se trata, tal y como se observa a continuación.

2-GRAMA	TF-IDF	2-GRAMA	TF-IDF
investidura rajoy	**47,0542941**	discurso rajoy	31,4377066
buenos dias	39,5853585	gobierno rajoy	25,3290362
debate investidura	41,0491607	rajoy presidente	24,5244191
presidente gobierno	38,6345042	rajoy amaiur	31,4189655
investidura discurso	38,500135	feliz navidad	23,6692008
mariano rajoy	32,2577949	nuevo gobierno	21,7142423
rajoy anuncia	45,8644206	rosa diez	26,979071
buenas noches	28,6929881	rubalcaba rajoy	28,5626959

Tabla 30. Puntuación TF-IDF de bigramas, con ventana deslizante, en la detección de palabras y hashtags de un conjunto de tweets concreto

El bigrama *'investidura rajoy'* sigue conservando el mayor carácter TT. En contrapartida *'rubalcaba rajoy'* ha desaparecido como TT. Otros como *'debate investidura'* siguen manteniéndose entre los más populares.

Analizando la puntuación TF_{ij} obtenemos la siguiente figura.

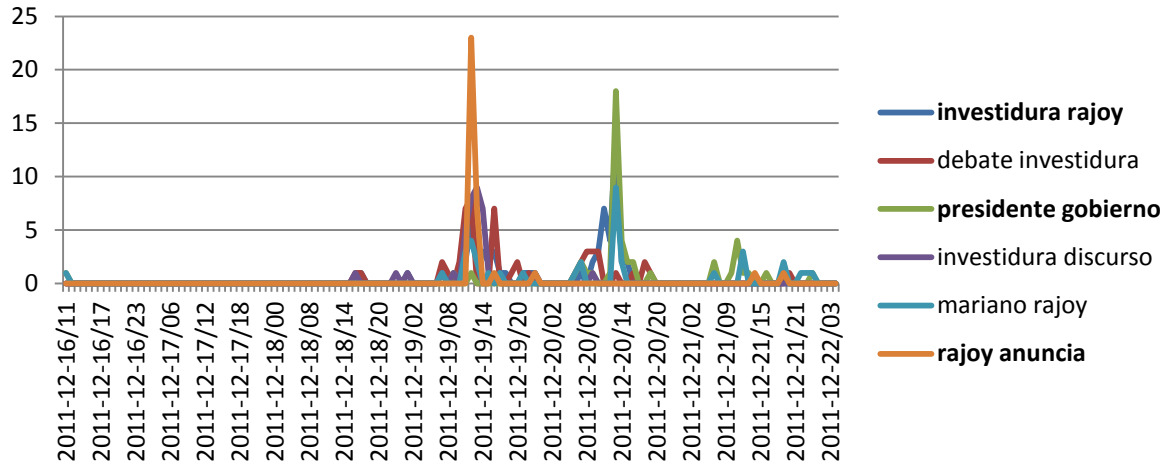

Figura 64. Puntuación TFij de los bigramas más populares, con ventana deslizante, en la detección de palabras y hashtags en un conjunto de tweets concreto

Además del bigrama *'investidura rajoy'*, que adquiere su mayor carácter TT en el mismo instante que *'presidente gobierno'*, el bigrama *'rajoy anuncia'* alcanza un pico máximo de popularidad en el mismo instante que lo alcanzadaba *'investidura rajoy'* en el cálculo de n-gramas anterior.

El cálculo de 3-gramas y 4-gramas en este caso, aporta prácticamente la misma información que en la detección única de palabras, tal y como se observa en la tabla.

3-GRAMAS	TF_i	4-GRAMAS	TF_i
kim jong il	15	xii edicion premios culturas	12
discurso investidura rajoy	14	edicion premios culturas extremadura	11
disponible historias dia	13	descombacantes disponible historias dia	8
presidente gobierno espana	13	noticias descombacantes disponible historias	8
edicion premios culturas	12	crea empleo crece libertad	7
intencion subir impuestos	12	alert real vs sevilla	6
noticias descombacantes disponible	12	carcel alava trasladado etarras	6
xii edicion premios	12	cr7 goal alert real	6
premios culturas extremadura	11	fotos nueva carcel alava	6
amaiur inaki antiguedad	9	goal alert real vs	6

Tabla 31. Frecuencia de 3-gramas y 4-gramas de palabras y hashtags (con ventana deslizante) en un conjunto de tweets concreto

La única diferencia es la inclusión del hashtag TT *'investidura'* en el segundo trigramas más frecuente, gracias a la detección adicional de hashtags. El resto de 3-gramas siguen aportando la misma información que en el caso de detección de palabras mediante ventana deslizante.

Si observamos el carácter TT de los 3-gramas, observaremos que la aparición del nuevo trigrama está relacionado con *'intencion subir impuestos'*, tal y como se observa en la siguiente figura.

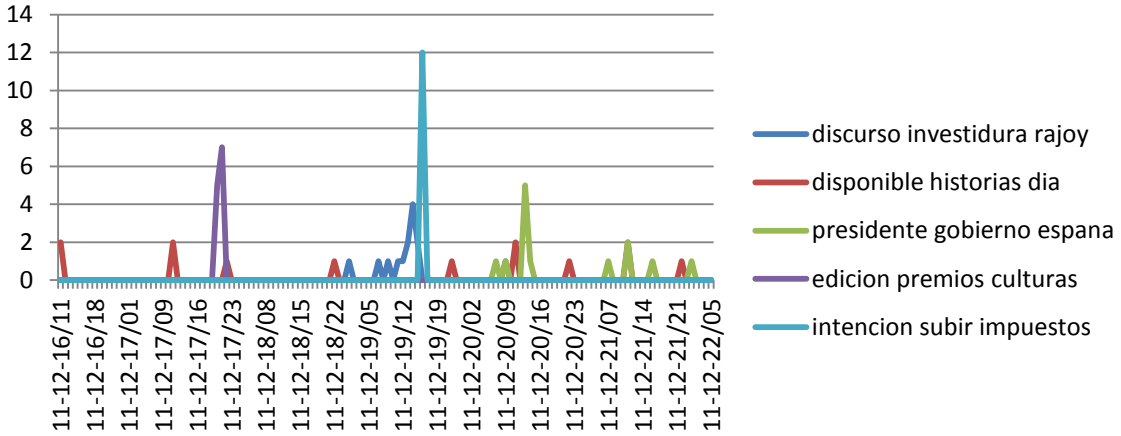

Figura 65. Puntuación TFij de los 3-gramas más populares (con ventana deslizante) en la detección de palabras y hashtags en un conjunto de tweets concreto

El trigrama *'intencion subir impuestos'* solamente adquiere un único pico de popularidad, y justo se corresponde con los instante posteriores en los que se empieza a hablar del 'discurso *investidura rajoy'.*

4.5. DETECCIÓN CON LEMATIZACIÓN

En este apartado abordaremos la detección haciendo uso del lematizador. Primero evaluaremos la detección de hashtags y posteriormente la detección de palabras. Con los resultados podremos deducir la utilidad del lematizador en el contexto de Twitter.

4.5.1 DETECCIÓN DE HASHTAGS

Aplicando el lematizador en la detección de hashtags, obtenemos la siguiente tabla de los hashtags más frecuentes. En ella se realiza una comparación de los

hashtags obtenidos con y sin lematización.

HASHTAG	HASHTAG CON LEMATIZACIÓN	TF_i	HASHTAG	HASHTAG CON LEMATIZACIÓN	TF_i
elcambioandaluz	elcambioandaluz	860	25m	25	215
ff	ff	679	eperiodico	eperiodico	177
malaga	malaga	544	losdesayunosdetve	losdesayunosdetve	176
investidura	investidura	403	grinan	grinan	142
andalucia	andalucia	405	cmin	cmin	141
congreso	congreso	385	eres	ser	135
38congresopsoe	38	290	portadaepc	portadapec	126
sevillahoy	sevillahoy	271	rajoy	rajoy	124
17congresopp	17	252	cofradiasmlg	cofradiasmlg	121
yeswespainisdifferent	yeswespainisdifferent	230	votapp	votapp	115

Tabla 32. Hashtags más frecuentes con lematización

A excepción de algunos términos, se obtienen prácticamente los mismos resultados con respecto a la detección de hashtags sin lematización.

Se puede apreciar como en aquellos hashtags que comienzan por una cifra, el lematizador no los detecta como una única entidad. Por este motivo hashtgas como *'38congresopsoe','25m'y '17congresopp'* se detectan como *'38', '17' y '25'* respectivamente.

En este caso observamos el problema que surge con la homonimia de las palabras. En ausencia del lematizador se detectaba el hashtag *'eres'*, correspondiente con las siglas de 'Expendiente de Regulación de Empleo'. En este caso obtenemos el hashtag *'ser*'. Esto se debe a que el lematizador detecta el hashtag *'eres'* como una conjugación del verbo *ser* y por lo tanto lo transforma a su forma canónica. Esto implica una pérdida de información.

4.5.2 DETECCIÓN DE PALABRAS

Aplicando el proceso de lematización a la detección de palabras, obtenemos la siguiente tabla con los términos más frecuentes.

PALABRA	TF_i	PALABRA	TF_i
bueno	6427	psoe	2013
dia	4656	ahora	2002
rajoy	3495	querer	1953
gracia	3309	seguir	1860
gracias	3233	espana	1801
gobernar	2760	noche	1746
gobierno	2625	pasar	1709
ano	2508	partir	1686
ver	2481	congreso	1638
pp	2335	publicar	1496
dar	2289	hablar	1483
nuevo	2284	andalucia	1476
manana	2213	politico	1450
hacer	2070	decir	1417
madrid	2063	esperar	1413

Tabla 33. Palabras más frecuentes con lematización

Podemos apreciar que con el uso del lematizador la lista está formada en gran medida por verbos, concretamente por verbos en su forma infinitiva. Aquellos verbos que antes se identificaban en sus distintos tiempos verbales, ahora son transformados a su forma infinitiva, provocando así que emerjan como términos más frecuentes. Una forma de evitar esto, sería incluir en la lista de stopwords todos los tiempos verbales, en sus diversas conjugaciones, de aquellos verbos que consideremos faltos de significado.

En el caso de detección única de palabras, esta lista estaba totalmente ausente de términos verbales. Sin embargo en este caso, como ya se ha visto, es totalmente lo contrario.

Esto implica que a primera vista se produzca una pérdida de información. Hay que tener en cuenta que la información que aportan los verbos es mucho menor respecto a la que ofrecen otro tipo de términos, como sustantivos, adjetivos, etc.

Sin embargo, de cara a detectar TT, aquellos términos que ya poseían un fuerte carácter TT, lo seguirán conservando. La lematización es un proceso que afecta casi en su totalidad a verbos, y a vocablos en sus distintas formas flexivas. Nombre propios como *'rajoy'* seguirán conservando su única forma, y por lo tanto su carácter TT.

Si analizamos la evolución temporal de los principales términos frecuentes, observamos el comportamiento descrito.

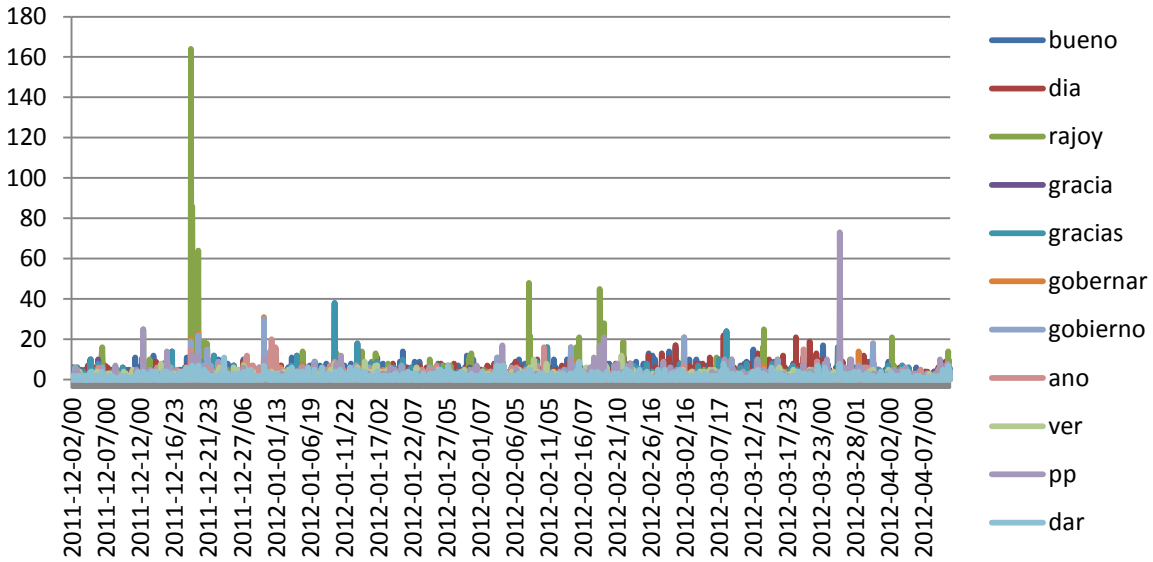

Figura 66. Puntuación TFij de las palabras más frecuentes con lematizador

Tal y como se aprecia, la aplicación de la lematización no afecta a la detección de TT. El término *'rajoy'* sigue alcanzando el mismo pico de popularidad, al igual que *'pp'*. Hay que destacar, que los verbos nunca podrán llegar a ser TT, ya que son mecanismos básicos a los que acudimos para expresarnos constantemente.

4.6. ESTUDIO DEL NÚMERO DE TÉRMINOS

En este apartado evaluaremos el número de términos que se procesarían según los distintos tipos de detección, observando cómo evolucionan a medida que aumentamos el tamaño del n-grama. El análisis se realiza sobre todo el conjunto de tweets de SEPLN, sin aplicar el proceso de lematización.

4.6.1 N-GRAMAS CON COMBINACIONES SIN REPETICIÓN

En este punto se abordará el estudio del número de términos haciendo uso de combinaciones sin repetición para el cálculo de los distintos n-gramas.

En la siguiente tabla se muestra un resumen del número de términos según los distintos tipos de detección.

COMBINACIONES SIN REPETICIÓN					
	HASHTAG	**HASHTAG + PALABRAS SIMIL (tweets con hash)**	**HASHTAG + PALABRAS SIMIL (todos los tweets)**	**PALABRAS**	**HASHTAG + PALABRAS**
1-grama	19390	19647	132810	560116	579506
2-grama	4413	5368	122610	2135765	2261403
3-grama	1395	2323	80170	5512208	5939025
4-grama	673	3692	42642	10542492	11522397

Tabla 34. Número de términos según el tamaño del n-grama y tipo de detección realizando combinaciones sin repetición para el cálculo de los n-gramas

El número de términos en aquellas situaciones en las que se detectan hashtags y hashtags y palabras idénticas a ellos, se observa que a medida que se aumenta el tamaño del n-grama el número de término se va reduciendo. El tamaño del n-grama define el número mínimo de términos que debe poseer cada tweet para que éste pueda ser calculado. Por este motivo a mayor tamaño, se necesita un mayor número mínimo de términos, y esto es algo que no todos los tweets cumplen en este tipo de detecciones.

En el caso de palabras y palabras y hashtags, se aprecia que a medida que aumenta el tamaño del n-grama el número de términos crece exponencialmente. La existencia de un gran número de términos por tweet en estos tipos de detecciones, junto con el tipo de cálculo de los n-gramas, provoca que el número de términos sea elevadísimo

6.2 N-GRAMAS CON VENTANA DESLIZANTE

En este punto se abordará el estudio del número de términos a través de la ventana deslizante para el cálculo de los distintos n-gramas.

VENTANA DESLIZANTE					
	HASHTAG	**HASHTAG + PALABRAS SIMIL (tweets con hash)**	**HASHTAG + PALABRAS SIMIL (todos los tweets)**	**PALABRAS**	**HASHTAG + PALABRAS**
1-grama	19390	19647	132810	560116	579506
2-grama	3439	4153	70907	482404	501604
3-grama	707	804	32439	406650	425291
4-grama	175	215	12917	335438	353078

Tabla 35. Número de términos según tamaño del n-grama y tipo de detección utilizando ventana deslizante para el cálculo de los n-gramas

Al igual que en el cálculo de n-gramas anterior, en la detección de hashtags y de hashtags y palabras idénticas a ellos a media que se aumenta el tamaño del n-grama, el número de términos desciende. En este caso, el número es menor debido al uso de la ventana deslizante.

La gran diferencia reside en la detección de palabras y/o hashtags. En este caso, a medida que se aumenta el tamaño del n-grama el número de términos desciende, a diferencia del caso anterior. De nuevo el uso de la ventana deslizante reduce los distintos posibles n-gramas en cada situación.

4.7 ESTUDIO DEL TIEMPO DE EJECUCIÓN

En este apartado se realizará un estudio del tiempo de ejecución de algunos de los tipos de detección en los tweets de SEPLN. Primero abordaremos el tiempo de procesamiento de los tweets para a continuación evaluar el tiempo de ejecución en el cálculo de la frecuencia total (TFi), algoritmo base a partir del que se obtienen las demás puntuaciones.

El hardware de la máquina sobre el que se han desarrollado las distintas pruebas es el siguiente: Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz (6 cores, 2 threads por core), 15M cache. RAM: 16GB. 2 Discos duros 1TB SATA (RAID1 por software).

Los resultados de todas las tablas están expresados en segundos.

4.7.1 TIEMPO DE PROCESADO DE LOS TWEETS

El tiempo de procesado engloba la lectura de cada uno de los tweets, la identificación de palabras y hashtags y el cálculo de los distintos n-gramas según las especificaciones del usuario.

4.7.1.1 Hashtags

Analizando el tiempo de procesado del todo el conjunto de tweets SEPLN en la detección de hashtags, obtenemos la siguiente tabla.

	COMBINACIONES SIN REPETICIÓN	VENTANA DESLIZANTE
1-grama	585,6	586,0
2-grama	583,0	578,3
3-grama	582,9	576,2
4-grama	581,1	573,6

Tabla 36. Tiempo de procesado de hashtags en todo el conjunto de tweets SEPLN, según los dos tipos de cálculo de n-gramas

A medida que aumentamos el tamaño del n-grama, el tiempo de procesado de los hashtags disminuye. Esto se debe a que la mayoría de tweets contienen un único hashtag. En el caso distinto a unigramas, el tiempo empleado en calcular las diferentes combinaciones se compensa con la existencia de un número menor de tweets que poseen suficientes hashtags para realizar dicho cálculo. Respecto a los distintos tipos de cálculo de n-gramas apenas existe diferencia.

4.7.1.2 Hashtags y palabras idénticas a ellos

Analizando el tiempo de procesado del todo el conjunto de tweets SEPLN en los dos tipos de detección, según la detección de hashtags y palabras idénticas a ellos, obtenemos la siguiente tabla.

	COMBINACIONES SIN REPETICIÓN		VENTANA DESLIZANTE	
	Tweets con hashtag	**Todos los Tweets**	**Tweets con hashtag**	**Todos los Tweets**
1-grama	657,8	5856,9	657,8	5856,9
2-grama	607,1	5777,1	606,4	5840,6
3-grama	611,9	4723,4	606,3	5816,5
4-grama	610,2	4410,6	609,9	5829,5

Tabla 37. Tiempo de procesado de las dos versiones de detección de hashtags y palabras idénticas a ellos, según los dos tipos de cálculos de n-gramas

La diferencia en el tiempo de procesado según los dos tipos de cálculos de n-gramas, es muy pequeña en el caso de tweets con hashtag. El tipo de cálculo no afecta sustancialmente al tiempo de procesado de los tweets en esta situación. La versión de análisis de las palabras de todos los tweets, sin embargo, se aprecia una mayor diferencia según el tipo de cálculo de n-grama

La situación en la que se examinan todas las palabras de todos los tweets, obtiene un tiempo de procesado mucho mayor. Para detectar si algunas de las palabras se corresponden con algún hashtags, primero se debe tener una lista de todos los hashtags existentes. La identificación previa de todos los hashtags, provoca ese aumento del tiempo de procesado.

La variación del tiempo de procesado a medida que se aumenta el tamaño del n-grama es muy pequeña. El mayor nivel de cálculo de cada n-grama no influye de manera acentuada en el tiempo de procesado.

4.7.1.3 Hashtags - Palabras

Analizando el tiempo de procesado en la detección de hashtags y palabras y hashtags de todo el conjunto de tweets SEPLN, obtenemos la siguiente tabla.

	COMBINACIONES SIN REPETICIÓN		VENTANA DESLIZANTE	
	Palabras	**Palabras + hashtag**	**Palabras**	**Palabras + hashtag**
1-grama	716,9	724,1	716,9	724,1
2-grama	866,3	883,3	740,5	738,4
3-grama	1113,8	1203,7	723,5	725,6
4-grama	1386,6	1497,7	700,6	710,7

Tabla 38. Tiempo de procesado en la detección de palabras y palabras y hashtags, en todo el conjunto de tweets SEPLN, según el tipo de cálculo de n-grama

En los n-gramas calculados con combinaciones sin repetición observamos que a medida que aumentamos el tamaño del n-grama, el tiempo de procesado aumenta. Esto se debe a que, como ya se ha visto, a medida que se aumenta el tamaño del n-grama el número de n-gramas a calcular crece exponencialmente, lo que se traduce en un mayor tiempo de procesado. Sin embargo, ese crecimiento exponencial, no se traduce en un incremento exponencial del tiempo de procesado.

En el caso de la ventana deslizante, el comportamiento es contrario. A medida que se aumenta el tamaño del n-grama el tiempo de procesado disminuye o se mantiene en niveles constantes. Según este tipo de cálculo, a medida que se aumenta el tamaño del n-grama, menor serán sus combinaciones posibles, y por lo tanto menor será su tiempo de procesado.

7.2 TIEMPO DE CÁLCULO DE TFi

En este punto abordaremos el tiempo de cálculo de la frecuencia global de los términos según las distintas detecciones. En el caso de palabras y/o hashtags nos centraremos en el periodo antes analizado.

7.2.1 Hashtags

Si analizamos el tiempo empleado en calcular la frecuencia de cada uno de los n-gramas en la detección de hashtags, obtenemos la siguiente tabla.

	COMBINACIONES SIN REPETICIÓN	VENTANA DESLIZANTE
1-grama	18	18
2-grama	8	6
3-grama	3	0,9
4-grama	1	0,4

Tabla 39. Tiempo de cálculo del algoritmo TFi en la detección de hashtags, según los dos tipos de cálculo de n-gramas, en todo el conjunto de tweets SEPLN

Se puede apreciar que el tiempo de cálculo en el caso de hashtags es muy pequeño. Podemos obtener la frecuencia global de cada uno de los hashtags en tan sólo 18 segundos. Para tamaños mayores, el tiempo es prácticamente despreciable.

7.2.2 Hashtags y palabras idénticas a ellos

Analizando el tiempo empleado en el cálculo de la frecuencia según las dos versiones en la detección de hashtags y palabra idénticas a ellos, obtenemos lsiguiente tabla.

	COMBINACIONES SIN REPETICIÓN		VENTANA DESLIZANTE	
	Tweets con hashtag	**Todos los Tweets**	**Tweets con hashtag**	**Todos los Tweets**
1-grama	28,6	160,1	28,6	160,1
2-grama	44,5	11394,7	6,5	4156,6
3-grama	16,4	17466,7	8,3	2731,6
4-grama	7,4	5843,9	2,8	529,9

Tabla 40. Tiempo de cálculo del algoritmo TFi de las dos versiones en la detección de hashtags y palabras idénticas a ellos, según los dos tipos de cálculo de n-gramas, en todo el conjunto de tweets SEPLN

Según los dos tipos de cálculo de n-gramas, en el caso de análisis de palabras en tweets con hashtags, a medida que aumenta el tamaño del n-grama, el tiempo empleado en el cálculo de la frecuencia global va disminuyendo. El hecho de que una gran cantidad de tweets con al menos un hashtag, no lo contengan como palabra, implica que a medida que se aumenta el tamaño del n-grama, el número de términos a evaluar sea menor. Esto provoca que el cálculo de la puntuación TFi requiera un menor coste computacional.

Sin embargo, el tiempo empleado en el análisis de todas las palabras de todos los tweets es bastante mayor. La detección de un mayor número de términos en este caso provoca que el tiempo requerido aumente.

Se puede apreciar que en esta situación, los bigramas y trigramas requieren el mayor tiempo de cálculo. Esto indica que la disminución del número de bigramas y trigramas detectados no compensa el mayor coste computacional que requiere el recuento de una estructura formada por dos y tres términos.

7.2.3 Palabras - Hashtags

Para todo el conjunto de tweets SEPLN, si analizamos el caso de unigramas obtenemos la siguiente tabla en la detección de palabras y palabras y hashtags.

	COMBINACIONE SIN REPETICIÓN		VENTANA DESLIZANTE	
	Palabras	**Palabras + hashtag**	**Palabras**	**Palabras + hashtag**
1-grama	47224,9	51399,9	47224,9	51399,9

Tabla 41. Tiempo de cálculo del algoritmo TFi de unigramas en la detección de hashtags y palabras, según los dos tipos de cálculo de n-gramas, en todo el conjunto de tweets SEPLN

Con respecto a los anteriores métodos, el tiempo de cálculo de la frecuencia global de cada término aumenta en gran medida. Tal y como se ha visto, en el caso de unigramas estos métodos de detección procesaban el mayor número de términos.

Si analizamos el tiempo en el intervalo al que nos hemos ceñido en el cálculo de los distintos n-gramas, observaremos la proporcionalidad existente en el tiempo de cálculo del algoritmo TFi.

	COMBINACIONES SIN REPETICIÓN		VENTANA DESLIZANTE	
	Palabras	**Palabras + hash**	**Palabras**	**Palabras + hash**
1-grama	121,98	128,69	122,78	127,33
2-grama	32135,58	35830,03	824,54	909,02
3-grama	227657,85	260571,7	1129,3	1277,42
4-grama	562148,64	608327,93	905,74	1024,01

Tabla 42. Tiempo de cálculo del algoritmo TFi en la detección de hashtags y palabras, según los dos tipos de cálculo de n-gramas, en un subconjunto de 4660 tweets de SEPLN

La disminución gradual del número de términos en el caso de ventana deslizante a medida que aumenta el tamaño del n-grama, no provoca una correspondiente disminución del tiempo de cálculo del algoritmo TFi. El gasto computacional requerido para evaluar la frecuencia de estructuras formadas cada vez por más elementos, no es compensado por una disminución acentuada del número de términos. Esto provoca que a pesar de obtener un menor número de términos por n-grama, el tiempo aumente.

En el cálculo de n-gramas por combinaciones sin repetición, observamos que el aumento exponencial del número de términos afecta directamente al tiempo de cálculo del algoritmo TFi. A medida que el tamaño del n-grama aumenta, el tiempo requerido crece exponcialmente. Hay que destacar, que para calcular la frecuencia global de cada uno de los distintos n-gramas es necesario recorrer por cada n-grama todos los anteriores. Esto implica, que si el número de términos crece exponencialmente, el tiempo de cálculo de la frecuencia global siga el mismo comportamiento.

Este último resultado a nivel global de todo el conjunto de tweets de SEPLN, eleva el gasto computacional requerido para n-gramas superiores a unigramas, a niveles muy altos.

CAPÍTULO 5 - CONCLUSIONES Y TRABAJOS FUTUROS

En el presente proyecto se han evaluado distintos mecanimos y métodos para la detección de Trending Topics en Twitter.

Por un lado, hemos valorado diferentes algoritmos que nos permiten identificar la popularidad y tendencia que sufre un determinado término.

La frecuencia es uno de los principales requisitos para que un término sea TT. Se trata de una característica necesaria, pero no suficiente. Un término con una frecuencia muy baja, es improbable que sea TT. Sin embargo en el extremo contrario, no se puede afirmar que un término muy frecuente siempre sea TT. Por este motivo el factor TFi proporciona un mecanismo básico para la detección de TT, aunque de manera aislada, su utilidad se reduce en gran medida.

En este contexto, el factor IDF ofrece información adicional sobre el nivel de concentración de las distintas apariciones de un determinado término. Su ponderación con la frecuencia global, a través de la puntuación TF-IDF, proporciona un mecanismo bastante completo para la identificación de TT. A pesar de ello, en algunas situaciones puede suceder que términos con una alta puntuación TF-IDF tengan un carácter TT menor que otras con una puntuación más baja. Al estar ponderado por dos componentes, en algunas ocasiones puede que uno de ellos tenga un peso mucho mayor que el otro, lo que provoca que el resultado final no plasme correctamente al 100% la tendencia de un término. Para complementar esta carencia, surge la puntuación TFij, mecanismo íntimamente relacionado con la puntuación IDF antes descrita.

Esta puntuación muestra la evolución temporal de cada término en cada uno de los distintos documentos, permitiendo apreciar cuáles alcanzan mayores picos de popularidad. A diferencia de la puntuación IDF, que sólo tiene en cuenta la existencia o no de un determinado término en cada uno de los documentos, la puntuación TFij proporciona explícitamente el número exacto de apariciones en cada uno de ellos, lo que permite identificar qué términos alcanzan cotas de popularidad más altas, y por lo tanto tienen un carácter TT mayor.

En este punto, la puntuación NTF aporta un extra de información sobre el nivel de

popularidad que alcanza un término, y en concreto, un determinado TT. Al permitir centrarse en aquellos instantes donde se producen los mayores picos de popularidad, posibilita obtener el porcentaje del total de tweets que contienen dicho término en ese periodo temporal.

Una alta puntuación NTF conlleva que la mayoría de tweets que contienen un determinado término se concentren en un periodo temporal muy concreto. Esto supone un fortísimo carácter TT. Sin embargo, una puntuación NTF baja no implica lo contrario. Términos muy frecuentes a lo largo de todo el conjunto de tweets que alcancen al menos un gran pico de popularidad, obtendrán una puntuación NTF más baja, y sin embargo, seguirán siendo TT.

Todas estas puntuaciones, permiten un análisis fiable y una identificación correcta de TT.

Por otro lado, hemos planteado distintas estrategias en cuanto a mecanismos de detección se refiere.

La detección de hashtags es el método más eficaz para detectar posibles TT. Al tratarse de una de las principales características de Twitter, la utilidad de este tipo de detección es evidente. En este caso, el cálculo de n-gramas superiores a unigramas carece de sentido. A pesar de ser computacionalmente muy ligero, no aportan ningún tipo de información adicional. Los resultados muestra la tendencia extendida por parte de los usuarios, de incorporar a sus tweets ningún o un único htashtag. La aparición de más de un hashtags por tweet, es un comportamiento menos común en Twitter.

La detección de palabras ofrece una alternativa en la detección de TT. En este caso se han distinguido dos principales tipos de términos: términos muy frecuentes de uso común y términos de frecuencia más variable pero con un fuerte carácter TT. La aparición del primer tipo depende directamente del filtrado previo de los datos, concretamente en la fase de eliminación de las stopwords. A medida que la lista sea más completa, más claros y limpios serán los resultados obtenidos. De cara a detectar TT este tipo de términos no adquiere ningún interés, por lo que su previa eliminación facilitaría el proceso de detección.

Respecto al segundo grupo de términos, en el caso de unigramas se ha visto que las palabras tendencia siempre están relacionadas con el tema que abarca el hashtag TT. Para estructuras superiores a unigramas, el tipo de cálculo utilizado condiciona en muy

pequeña medida, el resultado obtenido. Por regla general, el cálculo de todas las combinaciones sin repetición permite obtener n-gramas con un mayor detalle en relación al tema tendencia del momento, cosa que en el caso de ventana deslizante, no sucede siempre.

La detección de hashtags y palabras idénticas a ellos propone un mecanismo intermedio entre la detección única de palabras y la de palabras y hashtags. Los resultados obtenidos en este caso arrojan cierta información.

Por un lado, se concluye que el contenido de aquellos tweets que tienen al menos un hashtags, no contiene dicho hashtag como palabra. Esto se debe en gran medida al formato de los hashtags. Al tratarse de estructuras que no siguen ningún patrón definido, existe una menor probabilidad que tengan su equivalencia en palabra. Sólo aquellos hashtags que posean un formato similar al de una palabra o al de alguna expresión típica de un tema tendencia, son algo más usados en el resto del contenido del tweet. Por este motivo la lista de términos más frecuentes en la primera variante de este tipo de detección, apenas se vio modificada, por lo que no afectó a la detección de TT respecto a la detección única de hashtags.

Por otro lado, el hecho de que un tweet no contenga ningún hashtag, no indica que su contenido no esté relacionado con el tema popular del momento. Es más, en los casos en los que los hashtags adopten una estructura sencilla, aquellos tweets que no contienen ningún hash tienen una mayor probabilidad de contener palabras idénticas a esos hashtags respecto a los tweets que si contienen al menos uno. Este tipo de detección permite descubrir la posible popularidad de hashtags que en la detección única de hashtags no destacaban, sin perjudicar a aquellos hashtags que ya eran TT en su momento.

La detección simultánea de hashtagas y palabras, ofrece al mismo tiempo los resultados de ambos tipos de detección por separado.

Respecto al tipo de detección anterior, en este caso aparece entre los términos más frecuentes aquellos de uso común que no tienen su equivalente en hashtag, como consecuencia de evaluar todas las palabras. Sin embargo, de cara a detectar TT esto no afecta.

Al existir una mayor cantidad de palabras que de hashtag, existe una mayor probabilidad de que las palabras alcancen cotas superiores en las distintas

puntuaciones, lo que conlleva que puedan adquirir niveles TT mayores que el de los hashtag. Esto implica que podremos detectar nuevos términos con un mayor carácter TT respecto a la detección única de hashtag. A pesar de ello, se seguirán detectando todos los términos TT existentes, aunque las cotas alcanzadas por cada uno de ellos disten en gran medida.

Por regla general, el análisis de unigramas es el método más directo y eficaz para detectar TT. Este tipo de estructuras permiten detectar temas de carácter general. Un análisis superior, nunca va a suponer el descubrimiento de nuevas estructuras con un carácter TT más fuerte que el correspondiente a términos individuales. A medida que se aumenta el tamaño del n-grama, la probabilidad de aparición va disminuyendo proporcionalmente. Este comportamiento es algo obvio, ya que la dificultad de repetición de 2 o más términos es mayor que la de uno solo. Esto implica que aquellas estructuras que alcancen cotas de tendencia aceptables, con más motivo, se deben tener en muy cuenta.

Por una parte, el cálculo de todas las combinaciones sin repetición permite detectar un mayor número de n-gramas y con una frecuencia superior en la mayoría de los casos respecto al uso de ventana deslizante. Esto influye indirectamente en la capacidad de detección de TT, ya que al tener en cuenta un mayor número de combinaciones, el carácter TT de un determinado tema se ver acentuado.

Este tipo de cálculo de n-gramas presenta una desventaja en las situaciones en las que un determinado término se repite continua e iterativamente en un mismo o en un número pequeño de tweets. Al calcular todas las combinaciones sin repetición, la frecuencia de los distintos n-gramas aumenta a medida que aumenta el tamaño de los mismos, creando una sensación irreal de TT.

Independientemente del método usado para el cómputo de los distintos n-gramas, el cálculo de estructuras superiores a unigramas permite la detección de temas mucho más específicos. Obviando la detección única de hashtag, en donde el cálculo superior a unigramas no aporta ningún tipo de información adicional, en el resto de estrategias abordadas, el hecho de calcular estructuras cada vez mayores implica obtener temas mucho más concretos en los que se puede apreciar un nivel de detalle mucho mayor.

Por otro lado, el uso de la ventana deslizante prioriza la detección términos contiguos. Esto limita el número distinto de n-gramas detectados por tweet, lo que puede

restringir la aparición de diversos n-gramas TT. A pesar de ello, suele ofrecer estructuras que proporcionan información bastante parecida a la del cálculo anterior.

La diferencia en el número de n-gramas obtenidos según el tipo de cálculo, influye directamente en el tiempo de procesado. Se ha visto que el aumento del tamaño de los n-gramas requeridos no supone un aumento considerable en el tiempo de procesado de los tweets y cálculo de los distintos n-gramas, independientemente del tipo de cálculo. Sin embargo, el tiempo de cálculo de la frecuencia global si se ve afectada por el método usado. Mediante la ventana deslizante, a medida que se aumenta el tamaño del n-grama el tiempo de cálculo es menor, como consecuencia del menor número de n-gramas calculados. En el caso de combinaciones sin repetición, sucede lo contrario. En la detección de palabras y palabras y hashtags, el aumento exponencial del número de términos a medida que se aumenta el tamaño del n-grama, provoca un crecimiento exponencial en el tiempo de cálculo. En el resto de mecanismos de detección, al no disponer todos los tweets de un número elevado de términos válidos, ese aumento exponencial no sucede. En ambos casos, la detección de hashtag es el método que menos recursos consume, y por lo tanto, el que más rápido y eficiente es.

Respecto al comportamiento propio de Twitter, es evidente que varias de las estrategias abordadas no son las óptimas. Principalmente Twitter se centra en la detección única de hashtags y en la de palabras.

El hecho de que los hashtag sean una de las características más famosas en Twitter, provoca que su detección sea evidente. Además, según los resultados obtenidos, la detección de hashtag es un método muy rápido y eficaz en la detección de TT, por lo que es lógico su uso por parte de Twitter. En este caso, Twitter sólo se centra en la detección de unigramas de hashtag, que tal y como se ha visto, es la opción más coherente y que más información aporta.

La detección de palabras es otro método usado por Twitter. A pesar de que una gran parte de los TT se corresponden con hashtag, existen términos correspondientes a palabras que también llegan a ser TT. Por este motivo este tipo de detección es vital también.

Por regla general, las palabras TT suelen ser unigramas. Sin embargo en varias ocasiones se corresponden con n-gramas de palabras. En esta situación, los n-gramas propuestos mediante el cálculo de combinaciones sin repetición carecen de sentido

para Twitter. Este tipo de cálculo se adapta mejor a un contexto diferente a Twitter, en el que se necesite una mayor precisión sobre los temas tendencia tratados. En la mayoría de casos, los n-gramas TT mostrados por Twitter se corresponden con estructuras formadas por palabras contiguas, por lo que el cálculo mediante ventana deslizante se adapta mucho mejor a esta plataforma.

El tamaño máximo de n-grama estudiado es el correcto respecto al comportamiento de Twitter. Estructuras que contengan más de 4 términos, son muy poco probables que se conviertan en TT en Twitter.

De igual forma, el preprocesado de los datos es algo que Twitter también realiza. Respecto a nuestro previo tratamiento, puede que Twitter no elimine ciertos términos y signos de puntuación, como consecuencia de su uso extendido en típicas expresiones.

En cuanto al carácter TT de un término, está claro que Twitter no solo tiene en cuenta su frecuencia, sino cómo están distribuidas todas las apariciones de dicho término hasta el momento y qué tendencia de crecimiento siguen. Esto concuerda con los requisitos establecidos y los resultados obtenidos en la evaluación de los TT. El hecho de calcular que porcentaje sobre el total de tweets a analizar contienen dicho TT, no es algo esencial para Twitter, debido a la gran cantidad de tweets y de temas que se están tratando simultáneamente en la red social.

Hay que tener en cuenta que un requisito imprescindible de un TT es el crecimiento exponencial tanto en el número de apariciones del término como en el número de usuarios que lo utilizan. Este requisito es una posible mejora para nuestro sistema.

Para mejorar el comportamiento de nuestro sistema se podría incorporar una funcionalidad que analizase la evolución del número de usuarios que usan un determinado término y detectase cuándo este crecimiento es exponencial. Esto permitiría obtener aquellos términos que realmente fuesen TT, evitando así los términos muy frecuentes e instantáneos provocados por un número reducido de usuarios.

Por otro lado, el estudio de otro tipo de algoritmos, el desarrollo de una aplicación de detección de TT en tiempo real, la comparativa de los resultados obtenidos respecto a los TT de Twitter y la optimización en el procesamiento de los datos, entre otros, son posibles mejores que completarían la funcionalidad y eficacia del sistema.

CAPÍTULO 6 - BIBLIOGRAFÍA

[1] El origen de Internet. El camino hacia la red de redes [en línea]. Universidad Politécnica de Madrid. Nebreda Rodrigo, Iván. 3 de Junio de 2013. Disponible en: http://oa.upm.es/22577/1/PFC_IVAN_NEBREDA_RODRIGO.pdf

[2] INFORMATICAHOY. Informática, tecnología e Internet sin complicaciones. La historia de las redes sociales. Disponible en: http://www.informatica-hoy.com.ar/redes-sociales/La-historia-de-las-redes-sociales.php

[3] Gómez Plaza, Inés. Concepto 05. Estadísticas de usuarios de redes sociales en España en 2013. 10 de Julio de 2013. Disponible en: http://www.concepto05.com/2013/07/estadisticas-usuarios-redes-sociales-en-espana-2013

[4] Fernández, Gonzalo. Printernet y redes sociales. La verdadera historia de Twitter: el éxito de un fracaso [blog]. 17 de Febrero de 2014. Disponible en: http://www.prnoticias.com/index.php/internet-y-redes-sociales/1158-twitter-/20128220-la-verdadera-historia-de-twitter-el-%C3%A9xito-de-un-fracaso

[5] Redacción. Reason Why: Actualidad e Investigación sobre Marketing, Publicidad, Empresa y Tecnología. El origen de Twitter, by Evan henshaw-Plath [blog]. 11 de Noviembre de 2013. Disponible en: http://www.reasonwhy.es/actualidad/social-media/el-origen-de-twitter-evan-henshaw-plath

[6] Pérez, David. El Confidencial: El diario de los lectores influyentes. Entrevista a Evan Henshaw-Plath, confundador de la red social Twitter [blog]. 17 de Febrero de 2014. Disponible en: http://www.elconfidencial.com/tecnologia/2014-02-17/cuando-creamos-twitter-no-tenia-dinero-y-vendi-mi-parte-por-7-200-dolares_90178/

[7] Rodriguez, Manolo. Desenredando la red. Las cifras más importantes de las redes sociales en 2013 [blog]. 7 de Enero de 2014. Disponible en: http://desenredandolared.com/2014/01/07/las-cifras-mas-importantes-de-las-redes-sociales-en-2013/

[8] Escudero, Fernando. About.com, Redes sociales. Conoce qué son los hashtags en Twitter: conoce qué son, sus características y cómo usarlos en Twitter [blog]. Disponible en: http://redessociales.about.com/od/LoBasicoPrimerosPasosEnTwitter/a/Conoce-Que-Son-Los-Hashtags-En-Twitter.htm

[9] Activa Solutions. Idioma en Twitter: explicación de términos [blog]. 16 de Agosto de 2013. Disponible en: http://activasolutions.com/idioma-en-twitter-explicacion-de-terminos/

[10] Rautenstrauch, Ramón. Apasionados del Marketing. ¿Cómo funcionan los Trending Topic (TT) de Twitter? [blog]. 1 de Mayor de 2012. Disponible en: http://www.apasionadosdelmarketing.es/como-funcionan-los-trending-topic-tt-de-twitter/

[11] marketingdirecto.com, el portal para el marketing, publicidad y los medios. Social Media Marketing: ¿Cómo surgen los trending topics en Twitter? [blog]. 16 de Marzo de 2011. Disponible en: http://www.marketingdirecto.com/actualidad/social-media-marketing/%C2%BFcomo-surgen-los-trending-topics-en-twitter/

[12] Enrique de Salamanca Ros, Fernando. Técnicas de Bootstrapping en el Procesamiento del Lenguaje Natural. Sevilla, 29 de Marzo de 2007, pp. 13-15. Disponible en: https://www.lsi.us.es/docs/doctorado/memorias/Enriquez,%20Fernando.pdf

[13] Yelitza Contreras Z, Hilda. Procesamiento del Lenguaje Natural basado en una "gramática de estilos" para el idioma español. Colombia, Marzo, 2001. Disponible en: http://www.saber.ula.ve/bitstream/123456789/13157/1/hc_propuestatesis.pdf

[14] Jesús Vilares Ferro. Aplicaciones del Procesamiento del Lenguaje Natural en la Recuperación de Información en el Español. Tesis doctoral. A Coruña, Mayo, 2005.

[15] Técnicas Estadísticas para el Procesamiento de Lenguaje Natural. Capítulo 2, pp. 45-50. Disponible en: http://di002.edv.uniovi.es/~dani/downloads/blindlight-dgayo-disertacion-capitulo2-tecnicas-estadisticas-pln.pdf

[16] Garcia-Carpintero, M. Frege: Principios del contexto y composicionalidad; Sentido y Referencia. Disponible en: http://www.ub.edu/filosofia-del-llenguatge/dossiers/TEMA03.pdf

[17] M.Sakthi ganesh, CH.Pradeep Reddy, N.Manikandan, DR.P.Venkata Krishna. TDPA: Trend Detection and Predictive Analytics. International Journal on Computer Science and Engineering (IJCSE). ISSN: 0975-33917. Vol.3 No.3 Mar 2011

[18] Athena Vakali, Maria Giatsoglou, Stefanos Antaris. Social Networking Trends and Dynamics Detection via a Cloud-based Framework Design. Lyon, France: Abril, 2012.

[19] Christian Jacquemin, Evelyine Tzoukermann. NPL for Term Variant Extraction: Synergy between Morphology, Lexicon and Syntax

[20] Kowalsky, Gerald, 1945. Information retrieval systems: theory and implementation. Capítulo 5, pp. 95-122

[21] William B.Frakes and Ricardo Baeza-Yates. Information Retrieval: Data Structures & Algorithms. Capítulo 7.

[22] Snowball. Lista de stopwords en castellano. Disponible en: http://snowball.tartarus.org/algorithms/spanish/stop.txt

[23] Ian H. Witten, Alistair Moffat, and Timonthy C.Bell. Managing Gigabytes: Compresing and Indexing Documents and Images. Series Editor, Edward Fox. Second Edition. Morgan Kaufmann Publishers (1999).

[24] Gómez Díaz, Raquel. La lematización en Español: una aplicación para la Recuperación de Información. Gijón: Ediciones Trea, 2005. ISBN: 84-9704-186-0. Disponible en: http://redc.revistas.csic.es/index.php/redc/article/viewFile/301/348

[25] Carlos G. Figuerola, Ángel F.Zazo, Emilio Rodriguez Vazauez de Aldana, José Luis Alonso Berrocal. La Recuperación de Información en español y la normalización de términos. Revista Iberoamericana de Inteligencia Artifical, vol. 8, núm, 22, 2004, pp. 135-145. Disponible en: http://www.redalyc.org/articulo.oa?id=92582210

[26] Daniel Jurafsky and James H.Martin. Speech and Language Processing. An intoruduction to Natural Language Processing, Computational Linguistics and Scpeech Recognition. September, 1999.

[27]Luís Padró, Miquel Collado, Samuel Reese, Marina Lloberes, Irene Castellon. Freeling 2.1: Five years of open-source langugage processing tolos.Software Departament – TALP Research center. Universitat Poletécnica de Catalunya. Disponible en: http://nlp.lsi.upc.edu/papers/padro10b.pdf

[28] The Center for Information and Language Processing. TreeTagger: a language independent part-of-speech tagger. Disponible en: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[29] Kowalsky, Gerald, 1945. Information retrieval systems: theory and implementation. Capítulo 4, pp. 65-95

[30] Christian Jacquemin, Evelyine Tzoukermann. NPL for Term Variant Extraction: Synergy between Morphology, Lexicon and Syntax. Capítulo 2, pp. 5-11

[31] Martín Fernández, Maria Isabel. Entorno a la polisemia y la homonimia. Revista: Anuario de estudios filosóficos, ISSN 0210-8178, Vol. 13, 1990, pp. 193-206

[32] M.F.Porter. The Porter Stemming Algorithm. Disponible en: http://tartarus.org/~martin/PorterStemmer/index-old.html

[33] Enrique de Salamanca Ros, Fernando. Técnicas de Bootstrapping en el Procesamiento del Lenguaje Natural. Sevilla, 29 de Marzo de 2007, pp. 19-20

[34] Vlado Kesel, Funchu Peng, Nick Cercone, Calvin Thomas. N-gram-based Author profiles for authorship attribution. 2003, Pacific Association for Computational Linguistics.

[35] Grigori Sidorov. Consturcción no lineal de N-gramas en la lingüística computacional. N-gramas sintácticos filtrados y generalizados. México 2013

[36] P. McNamee and J. Mayfield. JHU/APL experiments in tokenization and non-word translation. volume 3237 of Lecture Notes in Computer Science, pages 85–97. 2004.

[37] Jesús Vilares, Miguel A. Alonso, Carlos Gómez-Rodriguez, Jorge Graña. RI con n-gramas: tolerancia a errores y multilingüismo. Departamento de Computación, Universidad da Coruña, 2010.

[38] Joaquin Ferreria da Silva, Gabrel Pereira Lopes. A local Maxima method and a Fair Dispersion Normalization for extracting multi-word units from corpora. Universidad Nova de Lisboa.

[39] Antonio Gabrial López Herrera, Enruique Herrera Viedma. Modelos de Sistemas de Recuperación de Información Documental Basados en Información Lingüística difusa. Granada, Marzo, 2006. EscuelaTécnica Superior de Ingeniería Informática.

[40] James Benhardus. Streaming Trend Detection in Twitter. 2010 UCCS REU for Artificial Intellingence, Natural Language Processing and Information Retrieval Final report.

[41] Chistopher D.Manning, Prabhakar Raghavan, Hinrich Schütze. Introduction to Information Retrieval, Cambridge University Press. 2008. Chapter 6, pp. 109-135

[42] Gerard Salton and Christopher Buckley. Term-Weighting approaches in Automatic Text Retrieval. Departament of Computer Science, Cornell Universisty, Ithaca, USA, 1988.

[43] Wikipedia. La enciclopedia libre. Tf-idf [en línea]. Disponible en: http://es.wikipedia.org/wiki/Tf-idf

[44] Manuel. PLN. TF-IDF: pensando las palabras [blog]. 17 de Noviembre de 2010. Disponible en: http://manuel.midoriparadise.com/2010/11/tf-idf-pesando-las-palabras/

[45] David A.Shama, Lyndon Kennedy, Elizabeth F.Churchill. Peaks and Persistence: Modeling the Shape of Microblog Conversations. 2011, Computer-supported cooperative work

[46] Luca Maria Aiello, Georgios Petkos, Carlos Martin, David Corney, Symeon Papadopoulos, Ryan Skraba, Ayse Göker, Yiannis Kompatsiaris, Alejando Jaimes Sensing trending topics in Twitter.

[47] David M.Blei, Andrew Y.Ng, Michael I.Jordarn. Latent Dirichlet Allocation. Journal of Machine Learning Research 3 (2003) 993-1022

[48] JeyHan Lau, Nigel Collier, Timothy Baldwin. On-line Trend Analysis with topic Models: #twitter trends detection topic model online.

[49] Xiaogang Wang, Eric Grimson. Spatial Latent Dirichlet Allocation. Computer Science and Artificial Intelligence Lab Massachusetts Institute of Technology, Cambridge.

[50] Wikipedia. The free Encyclopedia. Locality-sensitive hashing [en linea]. Disponible en: http://en.wikipedia.org/wiki/Locality-sensitive_hashing

[51] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng. SCAN: A Structural Clustering Algorithm for Networks.

[52] Brendam O`Connor, Michel Krieger, David Ahn. TweetMotif: Exploratory Search and Topic Summarization for Twitter. Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media

[53] Tomás Aluja. La minería de datos, entre la estadística y la inteligencia artificial. 2001, Universidad Politécnica de Cataluña.

[54] Fernando Berzal Galiano. Reglas de asociación [Association Rules]. Capítulo 2 [en línea]. Disponible en: http://elvex.ugr.es/

[55] Komal Khurana, Mrs. simple Sharma. A comparative Analysis of Association Rules Minning Algorithms. ISSN 2250-3153. International Journal of Scientific and Research Publications, Vol.3, Issue 5, May 2013

[56] Maurice Houtsma, Arun Swami. Set-Oriented Mining for Assotiation Rules in Relational Databases, 1995.

[57] Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. IBM Almaden Research Center.

[58] Jyoti Arora, Nidhi Bhalla, Sanjeev Rao. A review on Association Rule Mining Algorithms.International Journal of Innovative Research in Computer and Communication Engineering. ISSN 2320-9801, Vol. 1, Issue 5, July 2013

[59] Ron Rymon. Search Through Systematic Set Enumeration. Departament of Computer & Information Science. 1992, University of Pennsylvania

[60] Roberto J. Bayardo Jr. Efficently Mining Long Patterns from Databases. IBM Almaden Research Center.

[61] Wikibooks. Open books for an open world. Data Mining Algorithms In Frequent Pattern Mining. The FP-Growth Algorithm [en línea]. Disponible en: http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm

[62] Rakesh Verma. The data Minning Hypertextbook, 2009. Chapter 2, Association Analysis. Section 6, FP-Growth Algorithm [en línea]. Disponible en: http://www.hypertextbookshop.com/dataminingbook/public_version/contents/contents.html

[63] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, Edward Chang. PFP: Parallel FP-Growth for Query Recommendation.

[64] Sasa Petrovic, Miles Osborne, Victor Lavrenko. Streaming First Story Detection with application for Twitter.

[65] Fionn Murtagh, Pedro Contreras. Methods of Hierarchical Clustering. Departament of Computer Science, Royal Holloway, Univserity of London. May 3, 2011

[66] Droope. Stopwords para español/castellano [blog]. 28 de Febrero de 2011. Disponible en: http://droope.org/2011/02/28/stopwords-para-espanol-castellano/

[67] Snowball. A Spanigh stop Word list [blog]. 16 de Junio de 2008. Disponible en: http://snowball.tartarus.org/algorithms/spanish/

[68] Ranks NL. Default English stopwords list [blog]. Disponible en: http://www.ranks.nl/stopwords

[69] Textalytics, Meaning as a service. Lemmatization, PoS and Parsing [en línea]. Disponible en: http://textalytics.com/core/parser-info

[71] PLN. TASS 2013. Taller de Análisis de Sentimientos en la SEPLN. Disponible en: http://www.daedalus.es/TASS2013/corpus.php