UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

SISTEMA DE CLASIFICACIÓN AUTOMÁTICA SOBRE STREAMS DE TWEETS

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

Autor: Alberto Godino Martínez

Tutor: Julio Villena Román

Julio 2014

Título: SISTEMA DE CLASIFICACIÓN AUTOMÁTICA SOBRE STREAMS DE TWEETS

Autor: Alberto Godino Martínez

Tutor: Julio Villena Román

EL TRIBUNAL

Presidente:

Carlos Jesús Bernardos Cano

Secretaria:

Iria Estévez Ayres

Vocal:

Harold Molina-Bulla

Realizado el acto de defensa del Proyecto Fin de Carrera el día 2 de Julio de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Secretario Fdo: Vocal

Agradecimientos

A Julio por ser un excelente profesor, por haberme permitido desarrollar este proyecto, por su gran disposición para atenderme siempre que lo he necesitado y por haber incrementado mi interés por la inteligencia artificial.

A Francisco Javier González Serrano, por haber sido sin duda el mejor profesor a lo largo de estos 5 años, por haberme ayudado a realizar mi Erasmus en Suecia, por haberme despertado el interés en el aprendizaje máquina, la clasificación automática y la inteligencia artificial en general, que han servido de base para el desarrollo de este proyecto y en definitiva, por haberme ayudado en todo momento. Muchas gracias!

A Luis por su disposición a ayudarme en todo momento!

A mis padres por haberme apoyado y ayudado siempre que lo he necesitado.

A mis amigos por hacer estos 5 años más divertidos.

Resumen

El crecimiento de la red Social Twitter desde su aparición en el año 2006 ha sido sorprendente. En la actualidad millones de tweets son escritos y publicados al día, desde cualquier país y en cualquier idioma. El poder detectar los temas más populares (*trending topics*) de entre todos estos tweets nos permite conocer qué está ocurriendo en cualquier lugar del mundo, instantáneamente. De este modo, Twitter se ha convertido en una de las fuentes de información más poderosas.

El presente Proyecto de Fin de Carrera tiene como objetivo conocer la importancia que están adquiriendo las técnicas de *data mining* (conjunto de técnicas que permiten extraer información relevante y desconocida de manera automática dentro de grandes volúmenes de información), estudiar en profundidad algunas de las técnicas de clasificación automática supervisadas y no supervisadas más importantes y finalmente diseñar, desarrollar y estudiar un clasificador automático de textos que haga uso de algoritmos diferentes pertenecientes al campo de aprendizaje no supervisado. Dicho clasificador se aplicará sobre una colección de miles de tweets con el objetivo de encontrar los temas más importantes o *trending topics* de dicha colección. Por último, se compararán las prestaciones de cada algoritmo utilizado en el desarrollo del clasificador.

El sistema empleado se basa fundamentalmente en encontrar el grado de similitud entre los tweets tras procesarlos usando técnicas propias del Procesamiento del Lenguaje Natural para posteriormente y usando dos algoritmos de *clustering* diferentes (KMeans y DBSCAN) obtener la clasificación.

El conseguir un sistema automático de clasificación para esta tarea es muy importante puesto que evitará la intervención humana y hará factible el procesamiento de la inmensa cantidad de información que la red social Twitter genera a diario.

Abstract

Since Twitter appeared in 2006, it has experienced a huge growth. Today millions of tweets are written and posted every day. Detecting trending topics allow us to know what is happening everywhere. This makes Twitter one of the most powerful sources of information.

The aim of this final project is to know the increasing importance of data mining techniques, study deeply some of the automatic classification techniques and finally design, develop and study an automatic classifier based on two different algorithms in order to find the trending topics over thousands of tweets. The developed system relies on finding the similarity between tweets which previously were preprocessed using Natural Language Processing techniques and finally the classification will be got thanks to two different algorithms (DBSCAN and KMeans).

Getting an automatic classification, specifically for this task, is very important because it will make things easier and faster and it will avoid the problems that could appear because of the human interaction.

With this project, different techniques will be studied, compared and checked, getting a better understanding in that way in Artificial Intelligence concepts, specifically in the Automatic classification and Natural Language Processing issues.

Índice de contenidos

ANEXOS

REFERENCIAS

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 MOTIVACIÓN

Desde épocas remotas la información ha sido siempre sinónima de poder. El que posee la información es capaz de anticiparse, de controlar, de dirigir. Viviendo actualmente en la conocida como Sociedad de la Información, todo ello adquiere, aún si cabe, más importancia. No obstante, uno de los principales problemas a los que se enfrenta esta Sociedad de la Información es la gestión óptima y productiva de toda la gran cantidad de documentación disponible.

La minería de datos, en inglés conocida como *data mining*, comprende una serie de técnicas, algoritmos y métodos cuyo objetivo final es la explotación de grandes volúmenes de datos con vistas al descubrimiento de información previamente desconocida y que pueda servir de ayuda en el proceso de toma de decisiones tácticas y estratégicas, generando tanto modelos descriptivos (permitiendo comprender datos, identificar patrones y relaciones que pueden influir en resultados finales) como descriptivos. Las definiciones formuladas para este término son muchas:

I: "La tarea fundamental es encontrar modelos inteligibles a partir de los datos".

[J. Hernández Orallo. Introducción a la minería de datos. Prentice-Hall,2004.]

II: "Se trata del proceso no trivial de descubrir patrones válidos, nuevos, potencialmente útiles y comprensibles

dentro de un conjunto de datos ".

[G. Piateski-Shapiro, W.J. Frawley and C.J Matheus. Knowledge discovery in databases : an overview AAAI-MIT Press 1991]

III: "Es el proceso de analizar datos desde diferentes perspectivas y resumirla en información útil – información que puede ser usada para incrementar ingresos, reducir costes o ambas–. "

[Concepts and Techniques Second Edition Jiawei Han And Micheline Kamber University of Illinois at Urbana-Champaign]

Una de las principales fuentes de información en la actualidad son las redes sociales. Es indudable que durante los primeros años del siglo XXI su crecimiento ha sido enorme. Tuenti, Facebook, Twitter, Linkedin, etc. diferentes pero fundamentadas en pilares similares, cuentan con millones de seguidores en todo el mundo. Y este crecimiento continúa imparable.

De entre todas ellas, sobresale Twitter por su evolución en los últimos 5 años. Twitter se ha convertido en una de las páginas más solicitadas por los usuarios de Internet tal y como revela un reciente estudio (año 2013) de **Alexa.com** (subsidiaria de la compañía Amazon que provee información acerca de la cantidad de visitas que recibe un sitio web y los clasifica en un ranking):

1. Facebook
2. Google
3. Youtube
4. Yahoo
5. Amazon
6. Baidu
7. Wikipedia
8. W.Live
9. QQ
10. Taobao
11. Google Ind
12. Blogspot
13. **Twitter**
14. Yahoo Jap

Twitter fue creada en Marzo de 2006. Permite a sus usuarios intercambiar mensajes de texto plano con una longitud máxima de 140 caracteres, denominados *tweets* a través de cuadros de diálogo como el de la **figura 1** ofreciendo la posibilidad de comunicación rápida y en tiempo real.

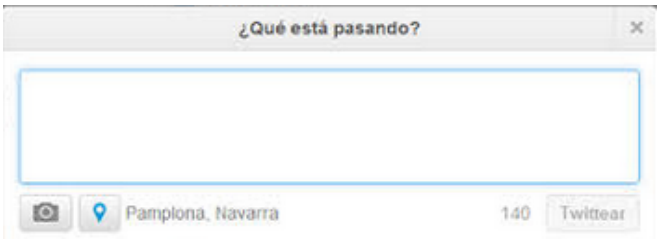


Figura 1. Cuadro de diálogo de Twitter

Una de las principales características que han popularizado Twitter ha sido su capacidad para acercar las noticias más relevantes casi al instante. De hecho, esta red social proporciona una herramienta en la que se muestran los 10 temas más populares del momento, de aquello de lo que más se está hablando: los *trending topics.* Es decir, se establece una clasificación por temas y posteriormente se eligen aquellos más importantes. De este modo es muy fácil enterarse prácticamente al instante de cualquier suceso relevante que ocurra en cualquier parte del mundo. Pero este proceso no es una tarea fácil y numerosos estudios han sido y se siguen llevando a cabo al respecto [1].

Figura 2. Imagen página principal de Twitter

1.2 OBJETIVOS

El objetivo del presente proyecto se centra en los siguientes aspectos:

- Conocer la importancia que están adquiriendo las técnicas de *data mining*.

- Estudiar en profundidad algunas de las técnicas de clasificación no supervisadas más importantes tales como el algoritmo del líder, el algoritmo de Markov (*Markov Cluster Algorithm*), DBSCAN o KMeans, así como hacer una revisión de las técnicas de clasificación supervisada más relevantes.

- Diseñar y desarrollar un clasificador automático estudiando el comportamiento de dos de estos algoritmos (DBSCAN y KMeans) sobre una colección de miles de tweets con el objetivo de encontrar los temas más importantes o *trending topics*.

Las razones que motivaron la elección del algoritmo KMeans fueron tanto su popularidad, muy extendido en el mundo de la clasificación automática y del que existen multitud de variantes, como su rapidez de ejecución y su "sencillez" de implementación. Este algoritmo basa la clasificación en la distancia de los puntos de entrada a los centroides escogidos. Se pretende comprobar su funcionamiento en un ámbito tan novedoso como el de las redes sociales: ver cómo afecta la elección aleatoria de los centroides semilla a la clasificación, ver como el número de clústeres que a priori debe ser determinado afecta a la calidad de los resultados, … La principal razón que motivó la elección del algoritmo DBSCAN fue que a diferencia del anterior se trata de un algoritmo que basa la clasificación en la densidad y además no necesita establecer a priori un número de clústeres determinado.

De este modo se pretende comparar dos algoritmos que se basan en un principio de funcionamiento bastante diferente pero que persiguen un mismo objetivo.

Para comprender y ver la evaluación de los resultados obtenidos se prestará especial atención a los parámetros configurables tanto de KMeans (número de clústeres y centroides iniciales) como de DBSCAN (radio de la vecindad y grado de densidad necesario), todos ellos explicados en las siguientes secciones.

Previamente será necesario procesar los tweets uno a uno para extraer de ellos la información que contienen (seleccionando las palabras que verdaderamente aportan dicha información) y convertirlos en un representación "amigable" para ambos

algoritmos.

El tratamiento de la información en Twitter exigirá por tanto hacer frente a un enorme volumen de información, lo que hace que se convierta en una tarea extremadamente complicada.

1.3 ESTRUCTURA

El presente documento se estructura en diversos capítulos.

Capítulo 1: *Introducción*
En este primer capítulo se lleva a cabo una pequeña introducción sobre los conceptos más generales que se van a desarrollar a lo largo del documento. A continuación se aportan los motivos que han justificado y que han despertado el interés por desarrollarlo. Por último, se presenta la estructura que tendrá el documento, los capítulos que lo componen y qué se presenta en cada uno de ellos.

Capítulo 2: *Estado del arte*
En este capítulo primeramente se resalta la importancia que está adquiriendo en los últimos años la clasificación automática de información. Se hace un repaso a las técnicas de clasificación automática más importantes: clasificación supervisada y clasificación no supervisada. Sobre las técnicas de clasificación supervisada se procede a hacer una breve enumeración sobre sus diferentes tipos y se introducen algunos conceptos básicos de las mismas. Sobre las técnicas de clasificación no supervisada se lleva a cabo un análisis mucho más exhaustivo, ya que será en ellas en las que se centre el presente documento. Se repasan tanto conceptos básicos como cuatro de los algoritmos más importantes que se utilizan: KMeans, DBSCAN, algoritmo del líder y MCL (*Markov Clustering Algorithm*). Se comentan a su vez las principales aplicaciones que tienen las técnicas de clasificación automática. Por último, se hacen una serie de observaciones sobre la manera en la que Twitter presumiblemente clasifica los tweets y obtiene así los famosos *trending topics*.

Capítulo 3: *Arquitectura de un clasificador automático de textos (no supervisado)*
Se lleva a cabo una descripción general sobre la arquitectura empleada para el tipo de clasificador de textos que se desea implementar. Se detallan las fases generalmente necesarias para el diseño de un clasificador automático de textos, haciendo referencia a algunas de las técnicas que pueden emplearse en cada una de las fases.

Capítulo 4: *Diseño del sistema*
Se describen detalladamente todas las decisiones de diseño tomadas. Se describen a su vez todos los pasos llevados a cabo durante la implementación de los dos sistemas estudiados: KMeans y DBSCAN.

Capítulo 5*: Estudio y evaluación de los resultados*
En este capítulo se presentan los resultados tras la aplicación de ambos algoritmos. Se llevan a cabo diferentes tipos de experimentos que ponen de manifiesto los puntos fuertes y débiles tanto de KMeans como de DBSCAN comentados de manera teórica en los **capítulos 2 y 3**.

Capítulo 6: *Conclusiones y trabajos futuros*
En este capítulo se hace un breve repaso sobre el esquema seguido a la hora de desarrollar el clasificador. Se analizan las ventajas y desventajas tanto de KMeans como de DBSCAN y se establece una comparativa entre ambos.
Se presentan a su vez posibles líneas de investigación en las que seguir trabajando con el fin de obtener mejoras en el sistema y profundizar más en el ámbito de la clasificación automática de textos.

Capítulo 2

Estado del arte

2.1 INTRODUCCIÓN: CLASIFICACIÓN AUTOMÁTICA DE TEXTOS

El creciente aumento de todo tipo de documentos en formato electrónico, aptos por consiguiente, para ser procesados de manera automática, ha dado un gran auge e impulso en los últimos años a la clasificación automática de textos. Es decir, continuamente se generan grandes cantidades de datos y es imprescindible establecer técnicas que nos permitan localizar, lo antes posible, la información que resulte relevante según las necesidades.

Es en este punto donde entran en juego los sistemas de clasificación automática de documentos, empleados para optimizar el tratamiento (obtención, filtrado, clasificado y extracción) de la información (en cualquier idioma), a fin de poder proporcionar al usuario, de forma eficaz y eficiente, exclusivamente los datos que necesita.

Se está por tanto ante uno de los problemas tradicionales dentro de la Inteligencia Artificial en el ámbito del Aprendizaje Automático: la clasificación automática.

Se puede definir la clasificación automática de textos de dos maneras diferentes. Por un lado se podría considerar como la tarea que consiste en asignar de manera automática un conjunto de documentos a una o varias categorías previas, creadas a su vez a partir de un conjunto predefinido de textos categorizados de antemano, a partir de los cuales el sistema lleva a cabo un proceso de clasificación supervisado [8].

Por otro, se podría definir como la asignación de documentos a una o varias categorías siendo esta vez las categorías generadas de forma automática por el clasificador. Es decir, no se necesitará de ningún corpus para crear una serie de categorías iniciales, las agrupaciones surgirán automáticamente. Se presenta por tanto un proceso de clasificación no supervisado [8].

2.2 REVISIÓN DE ALGUNAS DE LAS TÉCNICAS MÁS IMPORTANTES DE APRENDIZAJE SUPERVISADO

2.2.1 APRENDIZAJE SUPERVISADO: INTRODUCCIÓN

Parte de una serie de categorías diseñadas a priori. Estas categorías requerirán de elaboración manual. El principal objetivo de las técnicas supervisadas es por tanto aprender a generalizar y clasificar la nueva información basándose en la información previamente clasificada.

La primera fase es la denominada fase de entrenamiento: se tiene una información inicial, es decir se dispone de una serie de datos de entrada y se conoce a priori a qué grupo pertenecen - información etiquetada **(figura 3)**. A partir de esta información se lleva a cabo la fase de aprendizaje. Una vez finalizada, el clasificador debería haber aprendido a generalizar y por tanto poder clasificar información de la que a priori no se conoce a qué grupo pertenece [8].

Su principal desventaja reside en que nunca llegarán a mejorar a aquél del que aprenden (nunca serán mejores que su "profesor").

A continuación se va a hacer un repaso de algunas de las técnicas de clasificación supervisada más importantes.

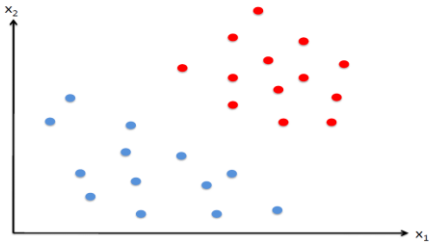


Figura 3. Aprendizaje supervisado, información previamente "etiquetada" durante la fase de entrenamiento.

2.2.2 ALGORITMOS DE APRENDIZAJE SUPERVISADO

2.2.2.1 K-NEAREST NEIGHBOUR

Es uno de los algoritmos más sencillos de implementar [6]. La idea en la que se basa reside en comparar un documento con sus *k* vecinos, de modo que este documento será clasificado del tipo del que haya más documentos entre esos *k* vecinos. Se trata de un algoritmo sencillo y eficaz especialmente cuando existen un gran número de categorías diferentes. El parámetro *k* deberá ser decidido a priori, y en función de la decisión que se tome se obtendrán unos resultados u otros.

En la **figura 4** se comprueba como la elección de dicho parámetro influye de manera notoria en el resultado del algoritmo. Si *k* = 3, el punto a clasificar en el caso de la **figura 4** (el círculo verde), pasaría a formar parte del grupo de triángulos rojos puesto que hay dos de este tipo por tan solo uno del grupo de cuadrados azules; no obstante, si *k* = 5 el círculo verde pasaría a formar parte en esta ocasión del grupo de los cuadrados puesto que de entre sus cinco vecinos más cercanos existen tres elementos de este tipo por tan solo dos de los triángulos.

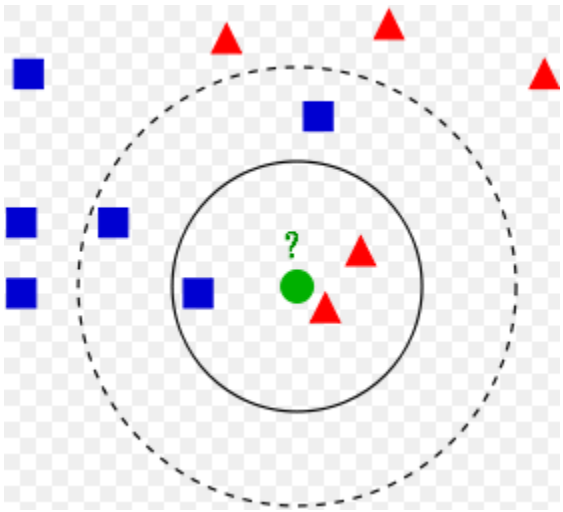


Figura 4. Algoritmo KNN, K=3 Vs Algoritmo KNN, K=5

2.2.2.2 REDES NEURONALES:

Algoritmo Perceptrón:

El Perceptrón [10] es un modelo de red neuronal que se basa en una colección de

neuronas de *McCulloch and Pitts* **(figura 5).** Las redes neuronales se inspiran en modelos biológicos y representan una simplificación matemática basada en el estímulo/respuesta de las neuronas. Son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc [16].

El modelo del Perceptrón se basa en una serie de entradas y una serie de pesos asociados a cada entrada, tal y como se muestra en la **figura 5**. Además se dispone de una función de activación que proporciona la salida en función de un umbral preestablecido (existiendo distintos tipos de funciones de activación) que puede tomar un valor binario 0 (neurona no activada) ó 1.

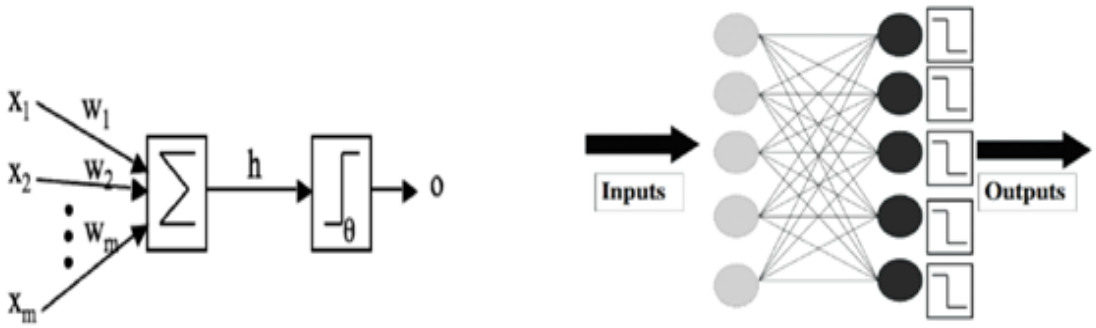


Figura 5. Neurona (izquierda), modelo de red neuronal (derecha)

El algoritmo de aprendizaje basado en el Perceptrón se basa en la actualización de los pesos en cada iteración hasta que se consigan clasificar correctamente todos los datos del set de entrenamiento. Por tanto, el algoritmo se podría describir en una serie de sencillos pasos (método online) [10]:

1. Inicializar los pesos a un número pequeño

2. Iterar T veces siendo T el número vectores de entradas diferentes:

2.1 Calcular la función de activación para cada neurona:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} 1 \text{ if } w_{ij}x_i > 0 \\ 0 \text{ if } w_{ij}x_i \leq 0 \end{cases}$$

(ECUACIÓN 1)

2.2 Actualizar los valores de los pesos

$$w_{ij} \leftarrow w_{ij} + \eta(t_j - y_j) \cdot x_i$$

(ECUACIÓN 2)

2.3 Volver a empezar de nuevo con las nuevas entradas

La actualización de los pesos, llevada a cabo por la **ecuación 2** es un paso clave en el desarrollo del algoritmo. El segundo factor de dicha ecuación se ve multiplicado por un parámetro conocido como *learning rate* o ratio de aprendizaje, η. Este factor indicará con que velocidad es capaz de aprender el algoritmo (cuanto se modifica el valor del peso en cada iteración). No obstante, debe ser diseñado con cuidado puesto que tanto valores muy elevados como demasiado pequeños afectarán negativamente. Valores típicos suelen estar en el rango: $0.1 < \eta < 0.4$

Por ejemplo, estas redes neuronales se pueden aplicar sobre la tabla lógica de una puerta OR **(figura 6, izquierda)** para ver más claramente como ocurre la actualización de estos pesos. La red neuronal a tratar **(figura 6, derecha)** se compone de una única salida y dos nodos de entrada más una entrada conocida como *Bias* (las entradas *Bias* son un tipo de entradas especiales que permiten desplazar la función de activación hacia la derecha o la izquierda, sin necesidad de ir teniendo que variar el umbral de la función de activación, lo que sería mucho más costoso. Esta entrada *Bias* suele definirse con un valor típico y constante de -1).

Entrada A	Entrada B	Salida $A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

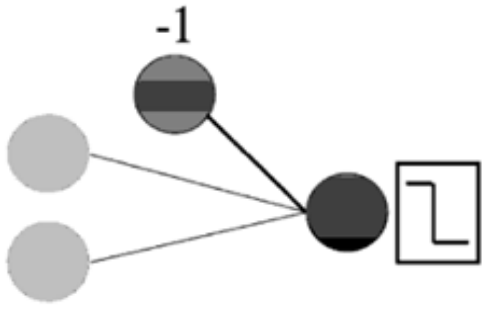


Figura 6. Tabla lógica puerta OR (izquierda), modelo de red neuronal equivalente (derecha)

- Se inicializan los pesos a un número pequeño:

$W_0 = -0.05 \quad W_1 = -0.02 \quad W_2 = 0.02$; Entradas = (0,0)

- Aplicando la fórmula de la **ecuación 1**:

-0.05x-1+-0.02x0+-0.02x0=0.05

Se obtiene un resultado que está por encima de 0 con lo que la salida, según la **ecuación 1** sería 1, lo cual es incorrecto (mirando la tabla de la **figura 6** se comprueba que debería obtenerse 0). De este modo se necesitaría aplicar la **ecuación 2** para actualizar cada uno de los pesos:

$W_0 = -0.05+0.25*(0-1)*-1 = 0.2$

$W_1 = -0.02+0.25*(0-1)*0 = -0.02$

$W_2 = 0.02+0.25*(0-1)*0 = 0.02$

Ahora se procedería a seguir con la siguiente entrada, (0,1) y se calcularía la salida de igual manera pero usando los pesos actualizados. Se vería que de nuevo vuelve a producirse un error, y los pesos se volverían a actualizar tal y como se ha hecho en el paso anterior.

El algoritmo finaliza cuando los pesos alcanzan estabilidad y deja de ser necesario que se actualicen para obtener los resultados correctos.

Además, las redes neuronales pueden estar constituidas desde una (las más simples) hasta varias capas. Las de varias capas permitirán realizar clasificaciones no lineales mientras que las de una sola capa son mucho menos flexibles, tal y como se aprecia en la **figura 7**.

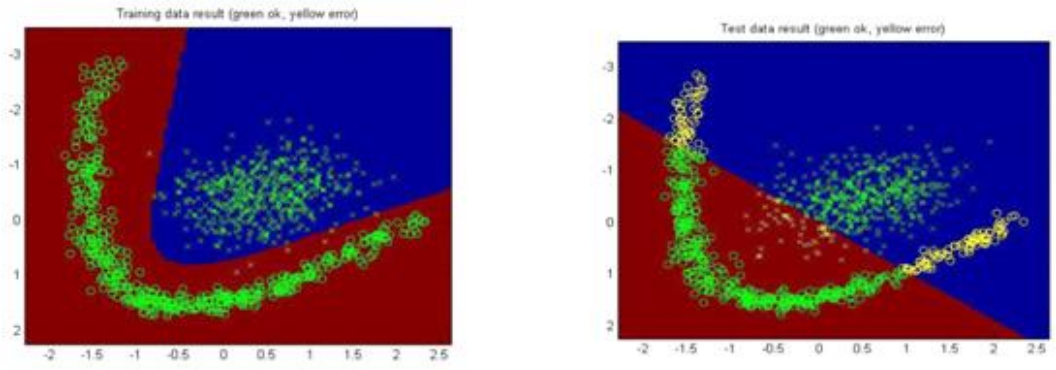


Figura 7. Red neuronal multicapa (izquierda) Vs unicapa (derecha)

2.2.2.3 ALGORITMOS DE BOOSTING

Se basan en el uso de un conjunto de clasificadores sencillos [10].

Cada uno de estos clasificadores va aportando algo a la clasificación final y finalmente, teniendo en cuenta a todos ellos, se puede llegar a obtener buenos resultados.

El más común es el conocido como el algoritmo de *Ada Boost*.

2.2.2.4 ÁRBOLES DE DECISIÓN

Se trata de algoritmos de aprendizaje por inducción supervisada que pretende modelar los datos de ejemplo mediante un árbol, que es llamado árbol de decisión; dentro de este árbol, cada *nodo interior* contiene una pregunta sobre un atributo concreto (con un hijo por cada posible respuesta) y cada *nodo hoja* se refiere a una decisión (clasificación) [16].

Los dos aspectos más importantes a la hora de crear un árbol son:

- Cómo se decide la división en un nodo
- El criterio de parada en el desarrollo del árbol

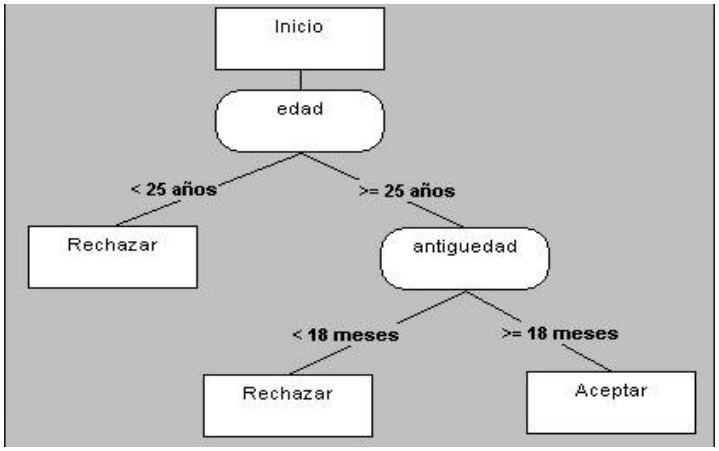


Figura 8. Árbol de decisión

Existen una gran cantidad de algoritmos basados en el aprendizaje supervisado, que aparecen claramente detallados y explicados en una extensa bibliografía dedicada a ellos como [2] o [10].

2.3. APRENDIZAJE NO SUPERVISADO

2.3.1 APRENDIZAJE NO SUPERVISADO: INTRODUCCIÓN

No existen categorías previas. El principal objetivo se centra en encontrar una cierta estructura entre los datos que se están intentando clasificar, a menudo en espacios dimensionales muy grandes. La categorización y el agrupamiento de objetos basada en sus propiedades y características similares se conoce como *clustering* [8].

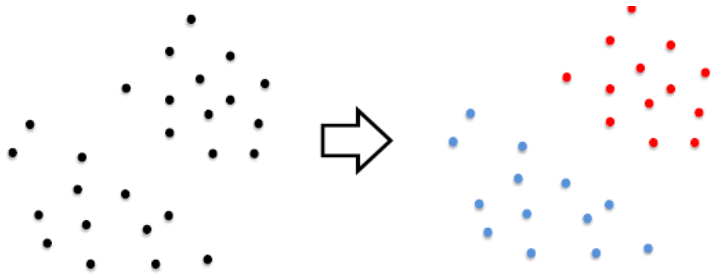


Figura 9. Descubriendo información: aprendizaje no supervisado

Por lo tanto se denomina clúster a aquellos agrupamientos de objetos que comparten ciertas características comunes tales como cercanía o conectividad de sus puntos en el espacio.

Figura 10. Clústeres

Los requisitos deseados por cualquier algoritmo de agrupamiento se pueden resumir en:

- Escalabilidad
- Posibilidad de trabajar con diferentes tipos de atributos.
- Posibilidad de descubrir grupos de distintas formas.

- Capacidad de detectar o encontrar la manera de no verse afectados por los *outliers* (aquellos datos que se apartan o que son muy diferentes respecto del resto de los datos de entrada).
- Capacidad de manejar datos de gran dimensionalidad.

Además, hay tres aspectos muy importantes a la hora de conseguir una buena clasificación:

a) Una buena selección de las variables que van a describir a los individuos. Esta tarea es muy importante puesto que de ella depende en gran medida el grado de calidad y rapidez del algoritmo.

b) Escoger cuidadosamente el criterio de similitud a emplear.

c) Seleccionar adecuadamente el algoritmo de clasificación de entre la gran cantidad existente, ya que dependiendo de los datos sobre los que se trabajen uno de estos algoritmos será más útil y adecuado que los otros.

2.3.2 CONCEPTO DE SIMILITUD

Una vez se haya hecho una adecuada selección de las variables a considerar sobre los datos de entrada, proceso que será de vital importancia para obtener buenos resultados, cada uno de los individuos a analizar vendrá representado por los valores que tomen estas variables en cada uno de ellos.

Este es el inicio de la clasificación. Posteriormente, para clasificar adecuadamente los individuos se deberá determinar lo similares o diferentes que son entre sí.

Para medir lo similares (o diferentes) que son los individuos existen una gran cantidad de técnicas diferentes. Todos ellas tienen propiedades y utilidades distintas y habrá que ser consciente de ellas a la hora de llevar a cabo el estudio que se esté realizando. A continuación se presentan algunas de las más importantes [14].

2.3.2.1 MEDIDAS DE SIMILITUD

PRODUCTO INTERNO:

$$\mathrm{sim}(\mathrm{d_j}, \mathrm{q}) = \sum_{i=1}^{t} \mathrm{w_{ij}} * w_{iq} \qquad \textbf{(ECUACIÓN 3)}$$

siendo w_{ij} es el peso del término i en el documento j y w_{iq} es el peso del término i del vector consulta q.

SIMILARIDAD DEL COSENO:

Medida de la similaridad que hay entre dos vectores que pertenecen a un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Devuelve un valor igual a 1 si el ángulo comprendido es cero, esto es, si los 2 vectores apuntan a un mismo lugar. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. Para el resto de posibles ángulos los valores obtenidos serán mayores a -1 y menores que 1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1.

$$\mathrm{CosSim}(d_j, q) = \frac{\sum_{i=1}^{t} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 * w_{iq}^2}} \qquad \textbf{(ECUACIÓN 4)}$$

SIMILITUD DE JACCARD:

La similitud de Jaccard es otra de las técnicas empleadas para medir el grado de semejanza entre vectores. Según la **ecuación 5**, el numerador representa lo que es común a los dos vectores de entrada P1 y P2, y el denominador aquello que es diferente a ambos. Una distancia grande indicará que se trata de vectores muy similares entre sí, y una distancia baja que son muy diferentes.

$$J(P1, P2) = \frac{P1 \cap P2}{P1 \cup P2} \qquad \textbf{(ECUACIÓN 5)}$$

2.3.2.2 MEDIDAS DE DISIMILITUD

Para expresar por el contrario la disimilitud entre elementos existen también numerosas técnicas, entre las que se podrían mencionar las siguientes:

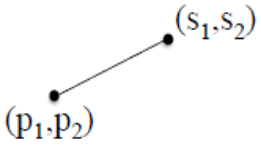
DISTANCIA EUCLÍDEA:

Se trata quizás de uno de los métodos más empleados y conocidos. No obstante, presenta

un inconveniente principal.

Es una distancia sensible a las unidades de medida de las variables: las diferencias entre los valores de variables medidas con valores altos contribuirán en mucha mayor medida que las diferencias entre los valores de las variables con valores bajos. Como consecuencia de ello, los cambios de escala determinarán, también, cambios en la distancia entre los individuos. Una posible vía de solución de este problema es la normalización previa de las variables, o la utilización de la distancia euclídea normalizada.

$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{p}-\mathbf{s}\|_2 = \sqrt{\sum_i (p_i - s_i)^2}$$

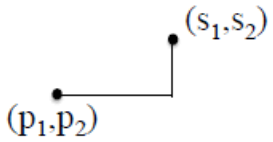


(ECUACIÓN 6)

DISTANCIA MANHATTAN:

La distancia Manhattan calcula la distancia que se recorrería desde un punto P hasta un punto S si se siguiese un camino en forma de "rejilla".

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

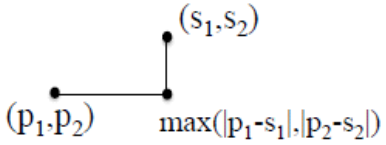


(ECUACIÓN 7)

DISTANCIA MÁXIMA:

La distancia máxima representa la distancia máxima entre las componentes x e y.

$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{p}-\mathbf{s}\|_\infty = \max |p_i - s_i|$$



(ECUACIÓN 8)

2.3.3 ALGORITMOS DE CLUSTERING

Dentro de los algoritmos de aprendizaje no supervisado se pueden encontrar algoritmos de muy diferentes tipos, como se describe a continuación:

2.3.3.1 ALGORITMOS JERÁRQUICOS

Dentro de este grupo se puede hacer referencia tanto a los métodos aglomerativos como a los disociativos [15].

- Los métodos aglomerativos son también conocidos como ascendentes, comenzándose el análisis con tantos clústeres como elementos haya. Poco a poco se van formando grupos entre estos elementos (en función de las distancia que los separa) hasta que al final del proceso todos los elementos tratados forman parte del mismo bloque.
- Los disociativos: son conocidos como descendentes, y llevan a cabo un proceso totalmente inverso al anterior, esto es, se irán formando grupos cada vez más pequeños.

Estos métodos jerárquicos se pueden representar mediante la construcción de un árbol de clasificación, denominado dendograma, representado en la **figura 11**, el cual muestra el proceso de unión seguido a la hora de ir formando los diferentes clústeres.

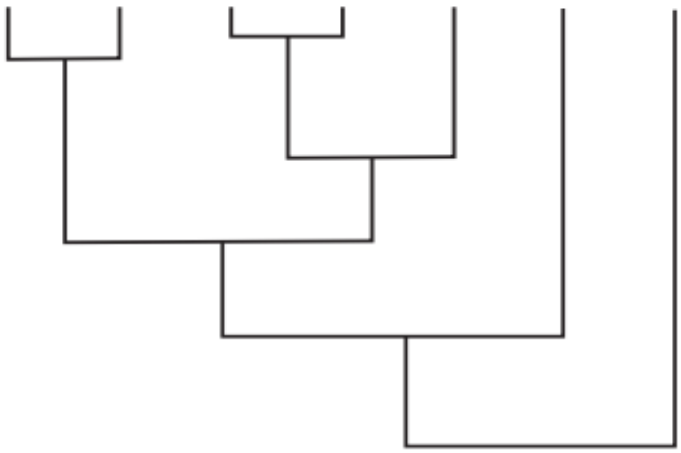


Figura 11. Ejemplo de dendograma

2.3.3.2 ALGORITMOS DE PARTICIONADO

Quizás el más famoso sea el KMeans [8].

KMEANS

Este algoritmo permite llevar a cabo agrupamientos de un modo fácil y rápido. No obstante, se debe conocer a priori el número de clústeres. Se basa en lo siguiente:

- Se elige la posición en el espacio donde se vayan a colocar "los centroides" de los clústeres, tradicionalmente de manera aleatoria, cuyo número ha sido elegido de antemano.
- Se va asignando cada dato a un clúster en función de su distancia al "centroide" del clúster (el centroide del clúster que este más cercano a ese dato será el clúster elegido).
- Se recalculan los centroides de cada clúster en función de la distancia media de cada dato a su centroide.

Con lo cual, resumiendo, el desarrollo del algoritmo sería:

Cada vector de entrada => X_j

Cada centroide => P_i

1. Se empieza con un número arbitrario de centroides
2. Se itera hasta que no se produzcan nuevos movimientos:

    2.1 Se asigna cada dato al centroide más cercano que tengan.

    2.2 se actualiza la posición de cada centroide en el espacio:

    2.2.1 Para uno de los centroides y todos sus elementos:

$$\sum_{x_j \in S_i} ||x_j - p_j||^2 \quad \textbf{(ECUACIÓN 9)}$$

    2.2.2 Para todos los clústeres:

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - p_j||^2 \quad \textbf{(ECUACIÓN 10)}$$

2.2.3 Se debe minimizar la **ecuación 10** (derivada parcial respecto a cada centroide). Finalmente, aplicando técnicas estadísticas, se obtiene que para recalcular los centroides se deberá proceder del siguiente modo:

$$p_i = \frac{1}{|S_i|}\sum_{x_j \in S_i} x_j \qquad \textbf{(ECUACIÓN 11)}$$

El número de iteraciones hasta alcanzar la convergencia del algoritmo podrá variar para las diferentes simulaciones. Las siguientes imágenes muestran un ejemplo de las iteraciones que se van sucediendo durante el desarrollo del algoritmo.

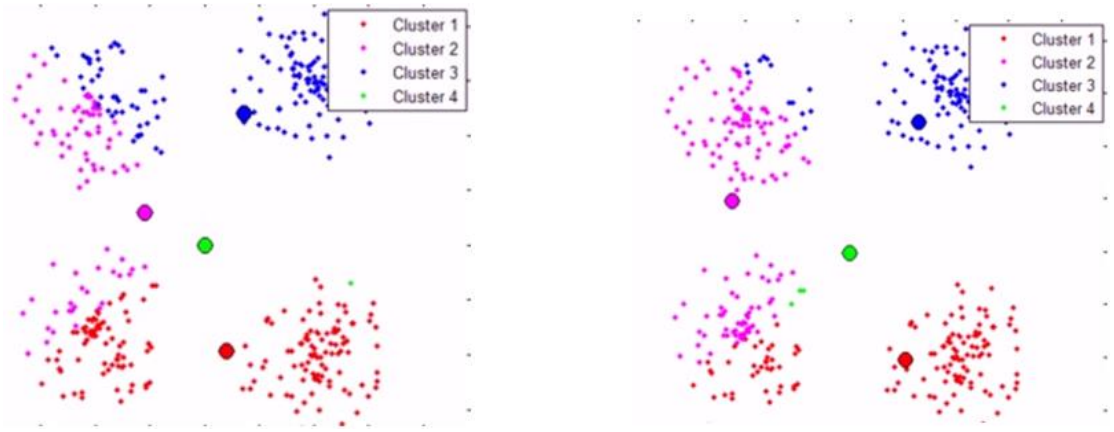


Figura 12. Primera iteración (izquierda) y segunda iteración (derecha) KMeans

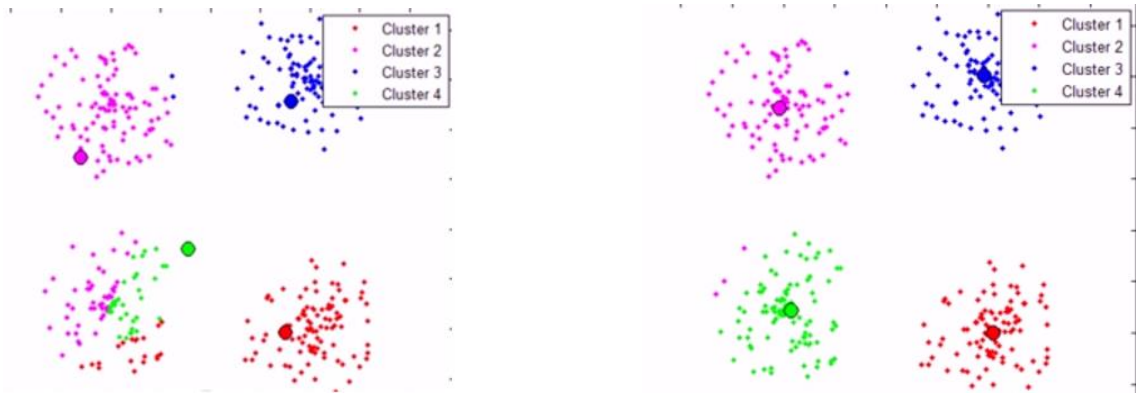


Figura 13. Tercera iteración (izquierda) y cuarta iteración (derecha) KMeans

Algunos inconvenientes típicos de este algoritmo se describen a continuación:

- No existe un método universal para identificar las particiones iniciales y para determinar el número de grupos K.
- El procedimiento óptimo iterativo de KMeans no puede garantizar la convergencia a un óptimo global.
- El algoritmo KMeans es sensible al ruido y a valores atípicos (*outliers*).

ALGORITMO DEL LÍDER

Éste es un algoritmo perteneciente a la familia de los algoritmos de partición rápida (Quick Partition Algorithms) [7]. Se caracterizan por ser algoritmos particionales y por tener la propiedad de necesitar una única pasada sobre los datos de entrada para devolver un resultado, lo que les confiere la cualidad de la rapidez.

Como cualquier algoritmo de clustering parte de tener que repartir los datos de entrada en distintos grupos o clústeres.

A diferencia que el KMean*s* no necesita predecir el número de grupos a priori ya que será el propio algoritmo el que fije el número de clústeres.

El algoritmo del líder se basa en el grado de similitud entre los datos de entrada y se apoya en un parámetro fundamental: una similitud mínima que actúe de limite *(smin).* Su mecanismo básico es el siguiente: obtiene un conjunto de clústeres disjuntos entre sí en los que la similitud de cada instancia de entrada al "líder" del clúster al que ha sido asignado es mayor que *smin* y, a su vez, la similitud entre los líderes de los distintos clústeres es menor que ese límite. En cada iteración este algoritmo comprueba si la entrada actual se asemeja a cada líder menos de *smin*. En este caso esa entrada se convierte instantáneamente en un nuevo líder. En caso contrario será asignado al clúster de ese líder en concreto. Algo particular de este algoritmo es que si durante las comparaciones se encuentra un líder lo suficientemente similar (mayor que *smin*), la instancia de entrada se asigna al clúster de ese líder sin comprobar el grado de similitud con el resto de líderes que quedaban.

El principal problema del algoritmo reside por tanto en que se ve demasiado afectado por el orden en el que se van comparando los parámetros de entrada, lo que puede dar lugar a la creación de unos lideres u otros. Además también puede darse el caso de que un dato de entrada sea asignado a un determinado líder por compararse primero a él, aunque tenga un mayor grado de similitud con alguno de los otros líderes con los que se ha quedado sin compararse.

2.3.3.3 ALGORITMOS DE CLUSTERING GRÁFICOS

MCL (MARKOV CLUSTER ALGORITHM)

Este algoritmo [4] se basa en la siguiente consideración: imaginando gráficamente un conjunto de agrupaciones en el espacio como el que sigue **(figura 14),** se puede afirmar que habrá mayor cantidad de uniones o links (mayor cantidad de relaciones) dentro de los

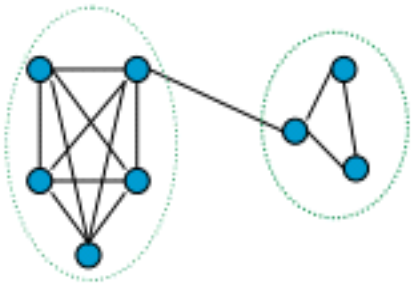


Figura 14. Agrupaciones en el espacio

elementos en un clúster que entre clústeres. Esto significa en cierto modo que si se empezase en un nodo del clúster al azar y se iniciase un movimiento también al azar a través de una de estas uniones, lo más probable es que se finalizase el "paseo" dentro del mismo clúster.

Pues bien, estos movimientos o paseos al azar en un gráfico son calculados a partir de la denominada *cadena de Markov (Markov chains).*

Matriz estocástica: Una matriz estocástica es una matriz usada para describir las transiciones de una *cadena de Markov*. Existen dos tipos de matrices estocásticas:

- La matriz estocástica por columna: se trata de una matriz cuadrada donde cada entrada debe ser un número real positivo y la suma de cada columna es igual a uno.
- La matriz estocástica por fila es una matriz cuadrada donde cada entrada es un número real positivo y la suma de cada fila es igual a 1.

Así pues, un ejemplo de matriz estocástica por columna sería la siguiente:

$$\begin{pmatrix} 0 & 1/2 & 1 & 3/5 \\ 1/3 & 0 & 0 & 2/5 \\ 1/6 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix}$$

Partiendo del grafo de la **figura 15** se va a llevar a cabo un ejemplo del algoritmo MCL, desglosado en diferentes pasos.

Antes de todo, se normalizará la matriz de similitudes para convertirla en una matriz estocástica. La columna j de dicha matriz se corresponderá con la probabilidad de ir desde el nodo j hasta cualquier otro nodo. De esta manera, se consigue que la probabilidad de ir de un nodo a otro sea proporcional al grado de similitud entre ellos.

Existen dos tipos de acciones diferentes dentro del algoritmo MCL: una de ellas es conocida como expansión (*expansión step*) y consiste en calcular el cuadrado de la matriz estocástica. Entre cada expansión se produce el otro tipo de acción, conocida como inflación (*inflación step*), que consiste en elevar la matriz estocástica al cociente de *Hadamard*, parámetro siempre mayor que 1. Esta potencia lo que permite es aumentar las probabilidades elevadas y decrementar las bajas. Dicho parámetro deberá ser elegido, siempre mayor que 1 como ya se ha comentado e influirá en el resultado notablemente y en la rapidez en llegar a la convergencia del algoritmo. Como siempre, se escogerá en función de los datos de entrada con los que se trabaje.

Hay que tener en cuenta que al finalizar cada *inflation step*, cada valor de la matriz menor que un cierto umbral, parámetro que también deberá ser elegido, será actualizado a cero. *Expansion step* e *inflation step* se alternan uno tras otro hasta que se alcanza una situación de equilibrio.

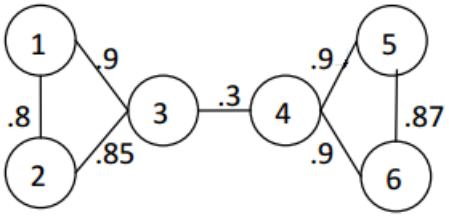


Figura 15. Grafo de ejemplo para algoritmo MCL

En el grafo de la **figura 15**, cada enlace y su correspondiente valor representan las similitudes entre nodos. Se calcula primeramente la matriz de similitud:

1	0,8	0,9	0	0	0
0,8	1	0,85	0	0	0
0,9	0,85	1	0,3	0	0
0	0	0,3	1	0,9	0,9
0	0	0	0,9	1	0,87
0	0	0	0,9	0,87	1

Se normaliza por columna para convertirla en matriz estocástica, tal y como se ha comentado. La matriz de similitudes se convierte ahora en una matriz de probabilidades ya que es directamente proporcional a la probabilidad que tiene un nodo en saltar de uno a otro. El primer valor de la esquina superior izquierda representa la probabilidad de permanecer en el nodo 1 desde el nodo 1, existiendo una probabilidad de 0,33 por ejemplo, en saltar hacia el nodo 3 desde el nodo 1.

0,37	0,3	0,3	0	0	0
0,3	0,38	0,28	0	0	0
0,33	0,32	0,33	0,1	0	0
0	0	0,1	0,31	0,32	0,32
0	0	0	0,29	0,36	0,31
0	0	0	0,29	0,32	0,36

Se eleva la matriz al cuadrado para simular un “paseo aleatorio” por el grafo *(expansión step).*

0,32	0,32	0,29	0,03	0	0
0,31	0,33	0,28	0,03	0	0
0,33	0,32	0,3	0,06	0,03	0,03
0,03	0,03	0,06	0,3	0,33	0,33
0	0	0,03	0,29	0,32	0,32
0	0	0,03	0,29	0,32	0,32

Se utiliza el coeficiente de *Hadamard*, *(inflation step)* el cual eleva a 1.5 cada elemento de la matriz con lo cual hace que las probabilidades más altas crezcan y las más bajas se hagan aún más pequeñas. Se ha elegido *Hadamard* = 1.5 en este caso particular.

0,32	0,32	0,32	0	0	0
0,28	0,28	0,28	0	0	0
0,39	0,39	0,39	0	0	0
0	0	0	0,38	0,38	0,38
0	0	0	0,33	0,33	0,33
0	0	0	0,29	0,29	0,29

Después de un cierto número de iteraciones se llega a una situación de equilibrio en la que se pueden distinguir dos clústeres: uno que engloba a los nodos 1, 2 y 3 y otro al 4, 5 y 6.

0	0	0	0	0	0
0	0	0	0	0	0
1	1	1	0	0	0
0	0	0	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0

2.3.3.4 METODOS BASADOS EN DENSIDAD

DBSCAN

DBSCAN es otro de los algoritmos que no necesitan conocer a priori el número de clústeres. DBSCAN hace referencia a *Density-Based Spatial Clustering of Applications with Noise* [11]; este algoritmo fue desarrollado en el año 1996 por M. Ester et al., en la Universidad de Múnich.

La idea es que se necesitará tener una densidad suficiente de nodos en un determinado radio para conseguir un clúster.

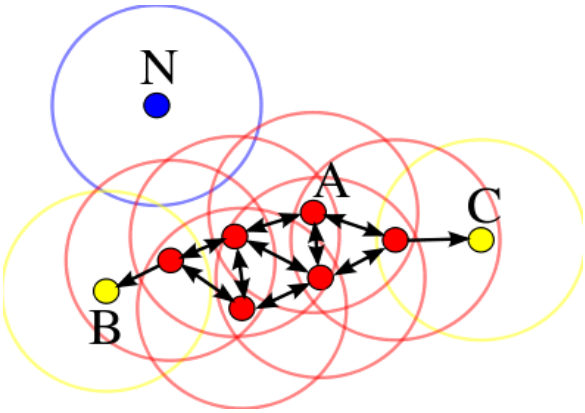


Figura 16. DBSCAN

En este algoritmo se tendrá por tanto que tener en cuenta dos parámetros:

- *Épsilon*: La distancia de dicho radio
- *MinPoints*: el mínimo número de puntos necesarios que debe haber dentro de ese radio para poder conformar un clúster.

Así pues, si en el vecindario de un punto 'p' hay más puntos que el valor establecido en el parámetro *Minpoints*, se creará un nuevo clúster, cuyo núcleo será el punto 'p'. A su vez este clúster puede ser expandido por parte de cualquier punto que forma parte del clúster. Se comprobará iterativamente por tanto si cada uno de esos puntos vuelve a tener un determinado número de puntos dentro de su vecindad, y si así fuere se incorporarán al clúster.

El proceso terminará cuando no se puedan añadir nuevos puntos al clúster: se recibirán como parámetros un conjunto de objetos D, el tamaño del radio de la vecindad, *Épsilon*, y el mínimo número de elementos que debería haber dentro de ese radio para formarse un clúster, *Minpoints.*

Con todo esto, se puede diferenciar tres tipos de puntos diferentes a la hora de aplicar el DBSCAN, tal y como se ven en la **figura 17**.

- Puntos nucleares: aquellos que están en un radio-vecindario y contienen el mínimo número de puntos dentro de ese vecindario.
- Puntos fronterizos (*border point*): Son los puntos que tienen menos de *MinPoints* vecinos dentro de su vecindario, pero están en la vecindad de un punto nuclear.

- Puntos de ruido (*noise point*): Son aquellos puntos que no caen en ninguna de las dos categorías anteriores. Los que están fuera de los círculos y, por lo tanto fuera, de los grupos.

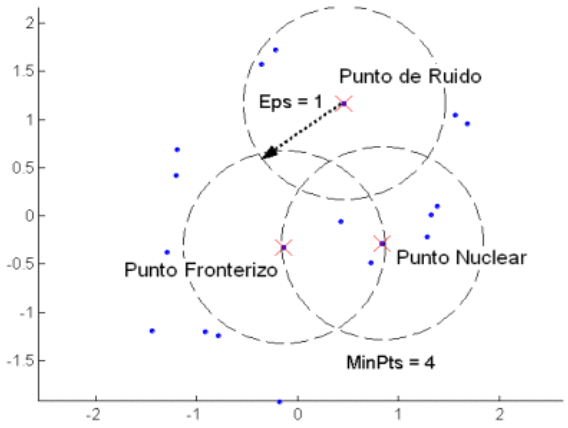


Figura 17. Puntos notables DBSCAN

Así pues y a diferencia de otros algoritmos como KMeans (que se basaba de manera exclusiva en la distancia que separaba a unos puntos de otros a la hora de formar clústeres), ahora se podrán llegar a distinguir grupos de muy diferentes formas, tal y como se aprecia en la **figura 18** gracias al concepto de densidad.

Figura 18. Ejemplos de clústeres formados a partir de DBSCAN

Uno de los principales problemas que presenta el algoritmo DBSCAN es que si el parámetro *Épsilon* para los elementos de entrada no es lo suficientemente pequeño pueden llegar a juntarse agrupaciones que no deberían pertenecer al mismo clúster o si no es lo suficientemente grande puede tener como resultado que dos conjuntos de datos,

similares entre sí acaben formando dos clústeres diferentes.

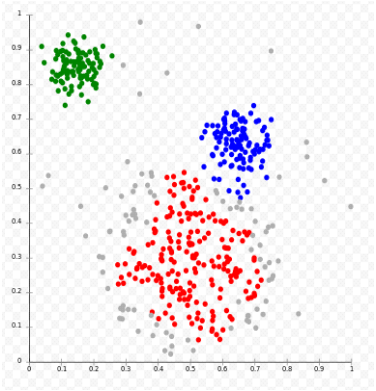
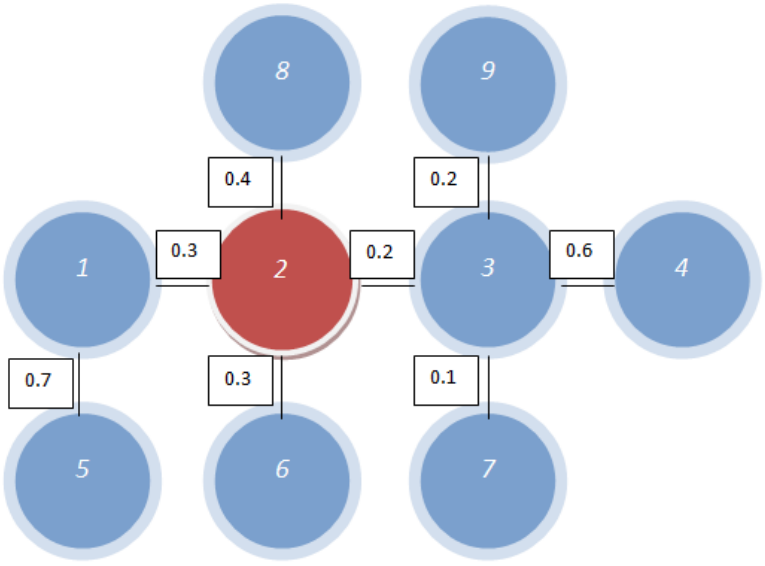


Figura 19. Problemas de DBSCAN a la hora de separar clústeres

La selección tanto del parámetro *Minpoints*, y el radio de la vecindad, *Épsilon* se llevará a cabo de manera experimental puesto que dependerán del conjunto de datos a clasificar y del tipo de clasificación que se quiera llevar a cabo.

Un ejemplo básico de este algoritmo con parámetros *Épsilon*=0.35 y *Minpoints*=2 sería el siguiente:



Se empieza de manera aleatoria por un nodo cualquiera del grafo (nodo 2 por ejemplo). Este se convertirá en punto nuclear y podrá formar clúster si dispone en su radio-vecindad *(Épsilon*) de al menos 2 nodos vecinos.

Figura 20. Grafo DBSCAN (I)

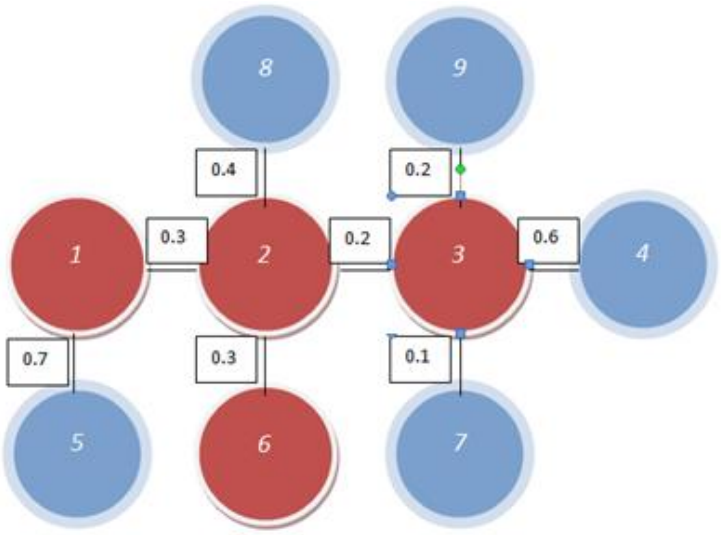


Figura 21. Grafo DBSCAN (II)

Se ve como el nodo 2 verdaderamente dispone de más de dos vecinos en su radio vecindad. Entonces se forma un clúster con 2 como punto nuclear. Se comprueba ahora si se puede expandir este clúster a través de los vecinos.

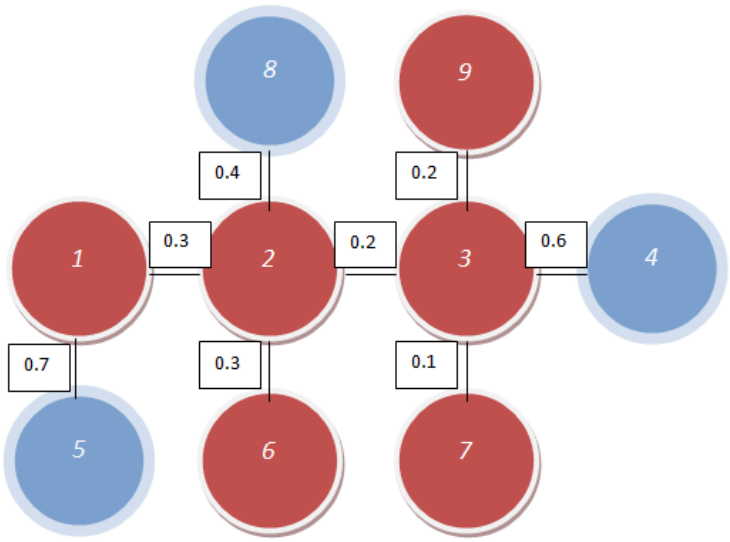


Figura 22. Grafo DBSCAN (III)

Se ve como se ha podido expandir el clúster debido a que el nodo 3 también poseía 2 vecinos en su vecindad (nodos 9 y 7). Sin embargo, el resto de nodos no pudieron expandir el clúster debido a la falta de vecinos. Los nodos 4, 5 y 8 se consideran ruido y no pasan a formar parte de ninguno de los clústeres existentes

2.4 APLICACIONES DE LA CLASIFICACIÓN AUTOMÁTICA

Algunas de las aplicaciones de la clasificación automática de información más empleadas son [5]:

- ✓ **En la red**: principalmente en la clasificación de documentos y del gran volumen de información en general.
- ✓ **En mercadotecnia**: en comercio y marketing para obtener todo tipo de información de los clientes (quejas, gustos, …), selección de personal, formación de equipos competitivos, … Está demostrado que se obtienen grandes resultados tras la aplicación de estas técnicas.
- ✓ **En política**: Estudios y estimaciones de las intenciones de voto, estudio de la valoración de la ciudadanía frente a una propuesta, un candidato, …
- ✓ **En el campo de la salud**: Están teniendo un gran desarrollo (agrupación de genes con funciones asociadas, identificación de patrones en células, …) . Facilitan el trabajo de los profesionales y se ahorra tiempo, dinero y además evitan posibles fallos humanos.
- ✓ **En climatología:** predicción de tormentas u otro tipo de fenómeno meteorológico

El desarrollo de todo tipo de estudios, aplicaciones informáticas, … para satisfacer cada una de ellas ha crecido exponencialmente en los últimos años, paralelamente a la evolución de Internet; de nuevo, información es poder.

2.5 TRENDING TOPICS EN TWITTER

2.5.1 OBSERVACIONES SOBRE LA MANERA DE OBTENER LOS TRENDING TOPICS POR TWITTER

La manera de obtener los *trending topics* por parte de Twitter sigue un algoritmo que no es totalmente trasparente a los usuarios. Primeramente, la definición que hace Twitter de que es una tendencia es la siguiente:

" Twitter Trends are automatically generated by an algorithm that attempts to identify topics that are being talked about more RIGHT NOW *than they were previously. The Trends list is designed to help people*

discover the 'most breaking' breaking news from across the world, in real-time. The Trends list captures the hottest emerging topics, not just what's most popular. "

Wednesday, December 8, 2010 | By Twitter (@twitter) [23:57 UTC]

Es decir, considera un *trending topic (TT)* a aquel asunto más importante (del que más se habla) pero reciente al mismo tiempo. De este modo las noticias de última hora son grandes candidatas a convertirse en los temas estrella de Twitter al mismo tiempo que se evita que por ejemplo, palabras como Leganés, Getafe, fútbol, España, pelota, mesa, silla… se conviertan en muchas ocasiones en *trending topics* a pesar de que son palabras muy comunes y que atendiendo a la frecuencia con la que se producen muy probablemente siempre lo serían.

Esto lleva a pensar directamente en que la asignación de pesos que lleva a cabo Twitter es a partir de la conocida combinación *Term frequency - Inverse Term Frequency* [9] la cual proporciona un resultado para cada palabra, que se incrementa proporcionalmente al número de veces que la palabra aparece en los post más recientes pero se decrementa si la misma aparece muy frecuentemente en el conjunto de entrada (el conjunto de entrada puede hacer referencia a la lista de tweets recopilada los últimos 3 meses por ejemplo).

Otra evidencia de que Twitter usa este algoritmo o al menos se basa en él es que palabras sin mucho sentido llegan a convertirse rápidamente en *trending topics* si por casualidad se repiten un mínimo de veces. Esto es porque al ser palabras sin sentido, ya sea porque estén mal escritas, porque procedan de otro idioma y es la primera vez que se usan o por cualquier otro motivo similar, no tienen una frecuencia de aparición grande en los últimos meses (es posible que no hubieran aparecido nunca en Twitter) con lo cual al aplicar el algoritmo *Term frequency - Inverse Term Frequency*, el resultado del mismo para esas palabras es muy alto.

Por tanto, algunos de los factores que Twitter posiblemente evalúa al considerar una palabra como *trending topic* o no son:

- La velocidad, es decir la concentración del uso del término en un momento determinado.
- El número de usuarios diferentes que están usando el término.
- El incremento de usuarios. Si la cantidad de usuarios que hablan de ese tema se mantiene constante, Twitter lo interpreta como un signo de que el interés no es

creciente.

- La permanencia. Cuanto más tiempo lleve una palabra o *hashtag* siendo TT, más difícil es que permanezca en esa lista.
- El volumen general de tweets que contienen dicha palabra.

Parece como conclusión que Twitter más que basarse en ningún tipo de algoritmo de *clustering*, se basa en las frecuencias de aparición de cada palabra.

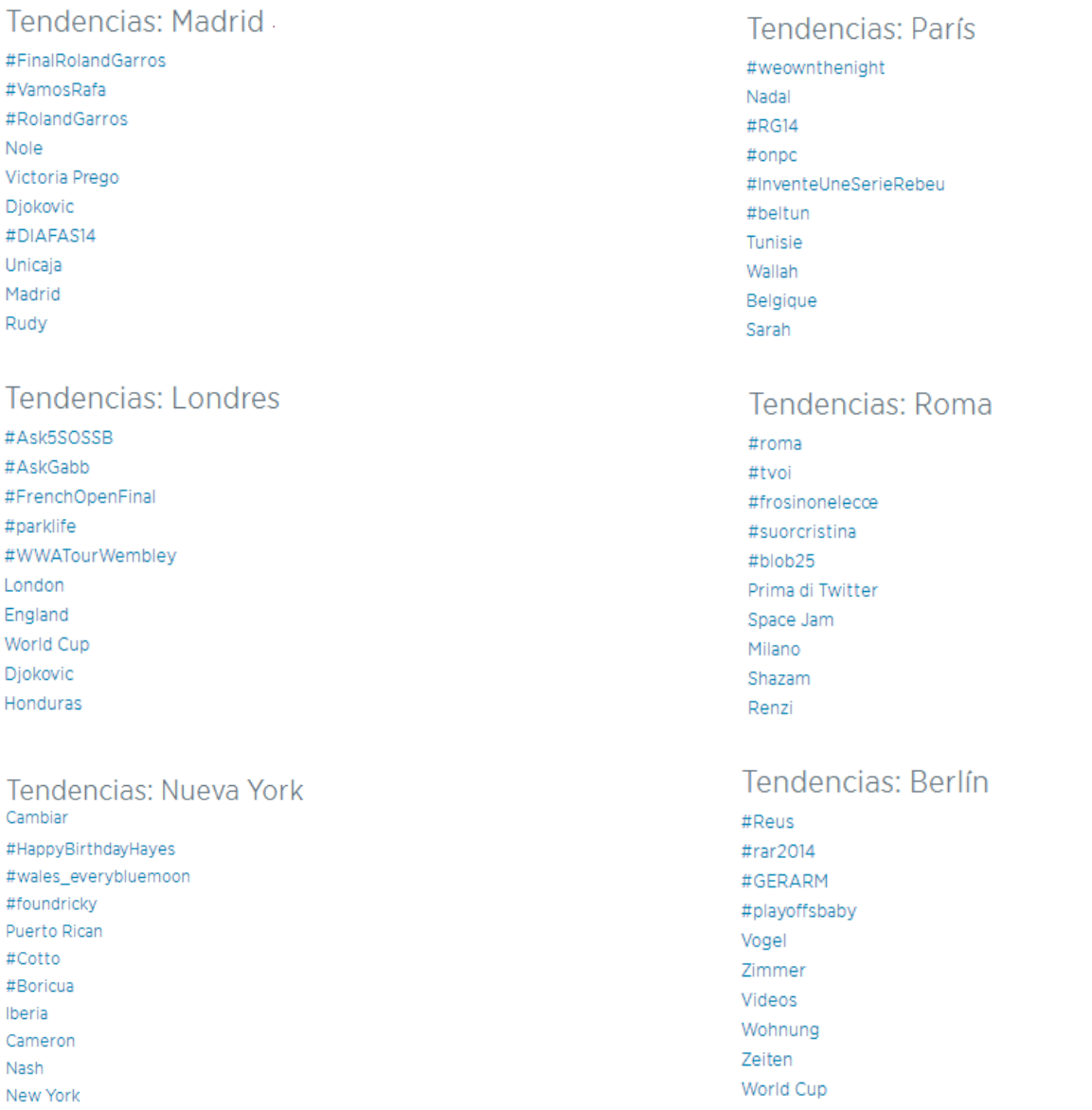


Figura 23. Listado de trending topics en diferentes ciudades (8 de Junio de 2014, 17:38)

Capítulo 3

Arquitectura de un clasificador automático de textos (no supervisado)

3.1 INTRODUCCIÓN

La arquitectura del sistema a desarrollar tiene como finalidad la construcción de un clasificador automático de textos basado en dos algoritmos de agrupamiento diferentes: DBSCAN y KMeans.

Para ello se desarrollan una serie de fases, comunes a todos los clasificadores automáticos de textos no supervisados, que tienen como principal objetivo obtener la máxima información posible de los documentos en cuestión (tweets en esta ocasión).

La obtención de la máxima cantidad de información posible es un proceso complejo ya que es necesario realizar un filtrado de todos aquellos términos que apenas aporten información. Por ejemplo, pronombres, artículos, preposiciones, … no suelen aportar nada relevante que vaya a ayudar al clasificador final y suelen aparecer con una elevadísima frecuencia en el conjunto de entrada. Otras técnicas como la aplicación de la lematización (stemming), la discriminación de aquellas palabras que se repiten prácticamente en cada documento (tweet) o que aparecen con muy baja frecuencia en los mismos, ... serán

llevadas a cabo también para extraer la máxima información de cada tweet.

Posteriormente se adoptará un modelo de representación vectorial del conjunto de entrada, asignando un determinado peso a cada dimensión. Esta asignación podrá seguir diversos esquemas existentes, que serán descritos a continuación.

Finalmente, se procederá a llevar a cabo el proceso de clasificación.

3.2 FASES DEL DISEÑO DE UN CLASIFICADOR DE TEXTOS

3.2.1 INTRODUCCIÓN

Por consiguiente, las fases imprescindibles en las que puede dividirse el diseño de un clasificador no supervisado de textos son las siguientes [13].

PREPROCESADO: esta primera etapa de un clasificador consiste en la toma de los documentos y la obtención de una representación adecuada de los mismos para posteriormente poder ser utilizados por el algoritmo clasificador, en este caso KMeans y DBSCAN. Se empleará un modelo de representación vectorial en el que cada palabra del tweet representará una dimensión de dicho espacio.

REDUCCIÓN DE DIMENSIONES: los espacios vectoriales creados suelen ser de grandes dimensiones, a pesar de eliminar los *stopwords* (comentados en la siguiente sección) y llevar a cabo procesos de lematización. Será necesaria por tanto una reducción de las dimensiones de estos espacios para conseguir clasificadores eficaces y eficientes.

ASIGNACIÓN DE PESOS: en el modelo de representación vectorial cada posición de vector representa como ya se ha visto a una palabra del documento (o del tweet en el caso que ocupa) y se le asigna un peso que hace referencia a la importancia de la palabra en dicho documento.

CLASIFICACIÓN: durante esta fase se lleva a cabo el algoritmo de clasificación que intenta conseguir una agrupación óptima de los vectores de entrada. Esta fase necesita de todas las anteriores y los resultados finales se ven influidos en gran medida por ellas.

En la **figura 24** se esquematiza la arquitectura del clasificador automático que se va a desarrollar.

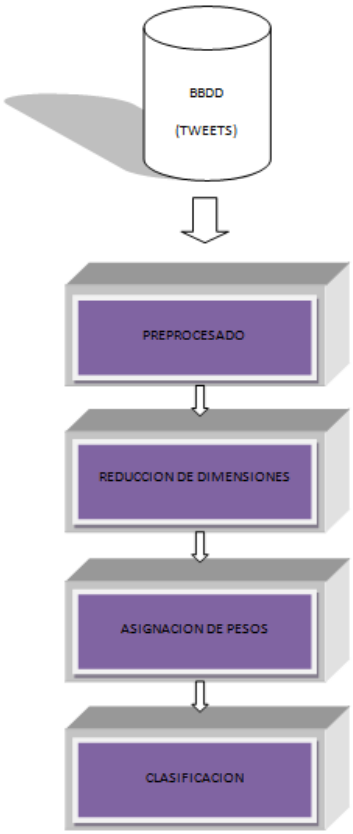


Figura 24. Fases del diseño de un clasificador

La preparación de datos por tanto suele generar un conjunto de datos más pequeño que el original, lo cual puede y debe mejorar la eficiencia del proceso de clasificación.

3.2.2 PREPROCESADO

Se trata de la primera fase de todas. Se parte de documentos (tweets recopilados directamente de Twitter en este estudio) de los que se extraen las palabras más importantes y con mayor contenido informativo. Posteriormente, cada documento o tweet se representa de manera vectorial, formándose un espacio vectorial de tantas dimensiones como palabras elegidas para representar los documentos (o tweets) de la colección.

Para llevar a cabo este proceso se realizan diversos pasos.

3.2.2.1 STOPWORDS

Primeramente se eliminan las conocidas como palabras de parada o *stopwords*, término acuñado por A Hans Peter Luhn.

Se trata de un conjunto formado fundamentalmente por preposiciones, adverbios artículos, … que no aportan ningún tipo de información relevante para la clasificación y que paradójicamente serían las más frecuentes y las que más peso tendrían en la clasificación final si no fuesen eliminadas [13].

No existe una lista definitiva de palabras vacías que sea incorporada por todas las herramientas de procesamiento de lenguajes naturales. En el **Anexo 2**, al final del documento, se puede observar la lista de *stopword*s que fueron eliminadas a la hora de llevar a cabo el presente estudio.

3.2.2.2. LEMATIZACIÓN

Formalmente se define la lematización como: “el neologismo que se aplica al proceso de eliminación automática de partes no esenciales de los términos (sufijos, prefijos) para reducirlos a su parte esencial (lema)” [La lematización en español: una aplicación para la recuperación de información; Gómez Díaz, Raquel].

Los algoritmos de *stemming*, o lematización para el castellano más importantes son Lovins (1968), Porter (1980) y Paice (1990) aunque originalmente todos fueron hechos para el inglés [12].

Los métodos de lematización y las reglas que estos aplican para llevar a cabo su misión son muy dependientes del idioma.

A modo general un lematizador suele seguir el siguiente esquema [12].

- hace pasar la palabra por varios conjuntos de reglas, cada conjunto formado por "n" reglas y cada regla está constituida por:
    - Un identificador de la regla.
    - Un sufijo a identificar.
    - El texto por el que se reemplaza el sufijo.

- El tamaño del sufijo.
- El tamaño del texto de reemplazo.
- El tamaño mínimo que debe tener la raíz resultante luego de aplicar la regla (para no procesar palabras demasiado pequeñas).
- Una función de validación (verifica si se debe aplicar la función una vez encontrado el sufijo)

El algoritmo empleado para el presente estudio se basa en el algoritmo de Porter [12]; se irán quitando sufijos de las palabras por etapas, eliminando las tildes en la etapa final.

En conclusión, se procede a reducir las palabras a su raíz de modo que todas las formas verbales del mismo verbo, todos los sustantivos, ya sean masculinos o femeninos, singulares o plurales , … sean representadas por la misma forma consiguiendo una representación más compacta del conjunto y consiguiendo posteriormente una clasificación más eficiente.

3.2.2.3. CONVERSIÓN DE LAS MAYÚSCULAS A MINÚSCULAS

Se procede a convertir todas las letras mayúsculas en minúsculas para conseguir una representación más compacta de los términos de la colección.

3.2.3 REDUCCIÓN DE DIMENSIONES

A pesar de haber conseguido reducir los documentos (tweets) a sus palabras más significativas gracias a los pasos anteriores (eliminación de stopwords y lematización fundamentalmente) no se pueden considerar todas las palabras que aún integran la colección, ya que en el modelo de representación vectorial que se va a usar en el que cada palabra representa una dimensión, la dimensión sería muy grande.

Existen por tanto diversas maneras para intentar reducir la dimensión del documento de entrada. Una de ellas es la conocida como frecuencia de documentos, que ni considera aquellas palabras que se repiten muy poco ya que no están aportando una información esencial relativa al conjunto, ni aquellas que se repiten con una frecuencia muy alta

porque apenas aportarían información discriminatoria. Por tanto se van a establecer unos umbrales, tanto superior como inferior con el objetivo de reducir las dimensiones pero a su vez intentando conseguir con ello mejores resultados.

El rango de estos umbrales será diferente en función del conjunto de entrada y se elegirán experimentalmente buscando compromiso entre eficiencia y efectividad.

3.2.4 MODELOS DE REPRESENTACION VECTORIAL; ASIGNACIÓN DE PESOS

Según el modelo de espacio vectorial, un documento (o un tweet) puede ser considerado como un vector $d_j = \{w_{1j},\ldots,w_{kj}\}$, donde w_{kj} es un valor numérico que expresa la importancia de la palabra k en el documento j.

Cada palabra o término de un documento representa por tanto una posición dentro del vector creado. Además, es necesario ponderar el valor de dicha palabra dentro del vector. Para ello existen diversas técnicas [9].

3.2.4.1 BINARIA

En cada posición del vector se indica la presencia o no presencia de una palabra correspondiente a esa posición mediante un 1 (presente) o un 0 (no presente).

3.2.4.2 FRECUENCIA DE PALABRA (TF, TERM FREQUENCY)

A cada palabra se le asigna una importancia proporcional al número de veces que aparece en el documento.

$$TF(k,j) = \frac{\text{frecuencia del termino k}}{\text{numero de palabras en doc.j}} \qquad \textbf{(ECUACIÓN 12)}$$

En algunas ocasiones la normalización se produce respecto a la palabra de la colección de entrada con mayor frecuencia.

3.2.4.3 FRECUENCIA INVERSA DEL DOCUMENTO (IDF, INVERSE DOCUMENT FREQUENCY)

Puesto que existen palabras que aparecen en gran cantidad de documentos, su relevancia será mínima y deberán ser eliminadas del vector. Por ello, serán más importantes las palabras que tengan menor presencia en los documentos analizados. La importancia de cada palabra es inversamente proporcional al número de documentos que la contienen. El factor IDF de la palabra k viene dado por:

$$IDF(k) = \log\left(\frac{\text{numero de documentos(tweets)de la coleccion}}{1+\text{numero de documentos(tweets)que contienen la palabra k}}\right) \quad \textbf{(ECUACIÓN 13)}$$

3.2.4.3 TF·IDF

Es posible obtener mejores prestaciones cuando se combinan los dos mecanismos anteriores: TF-IDF es el producto de dos medidas: frecuencia de término (TF) y frecuencia inversa de documento (IDF), vistas ambas anteriormente.

El valor TF-IDF aumenta por tanto proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras.

3.2.5 CLASIFICACIÓN

Es el apartado final que a menudo se le otorga mayor importancia, pero que necesita de los pasos anteriores para poder llegar a resultados óptimos. Forma las diferentes agrupaciones en las que se clasifican los documentos. Existen multitud de algoritmos distintos (en el **capítulo 2** se vieron algunos de los más importantes o más utilizados en la actualidad) y dependiendo del tipo de documentos sobre los que se trabaje otorgan mejores o peores resultados. En este caso se llevará a cabo una clasificación a partir de aprendizaje no supervisado, sin crearse categorías previas y sin llevar a cabo ninguna fase de entrenamiento.

Capítulo 4

Diseño del sistema

4.1 INTRODUCCIÓN: ALGORITMOS DBSCAN Y KMEANS

Se ha llevado a cabo la implementación de dos algoritmos diferentes para la clasificación de tweets según su temática. Los escogidos han sido KMeans y DBSCAN, ambos algoritmos de clasificación no supervisada.

A la hora de construir el clasificador automático se han seguido las mismas fases que fueron descritas en el **capítulo 3:** preprocesado de la información, reducción de dimensiones, asignación de pesos y clasificación.

La diferencia entre ambos clasificadores y por tanto lo que determina la clasificación final que cada uno de ellos hace de los tweets se halla en el apartado de clasificación.

El lenguaje de programación elegido para llevar todo ello a cabo ha sido PHP [http://www.php.net/] (sobre otros como Java o Matlab) debido a que cuenta con grandes ventajas. Se trata de un lenguaje totalmente libre y abierto, cuyo aprendizaje es muy rápido si se conocen otros lenguajes de programación (especialmente Java) y presenta una de las comunidades en Internet más grandes en la que existe una gran cantidad de documentación.

Como se describirá en la evaluación de resultados, como datos de entrada a clasificar se ha utilizado una amplia colección de tweets previamente recopilada y clasificada por temas (Madrid, Sevilla, iPhone y UC3M). El objetivo de utilizar estas colecciones es poder comprobar más fácilmente al final del algoritmo si los resultados obtenidos se ajustan a los resultados teóricos más o menos esperados y poder así establecer medidas y porcentajes aproximados en cuanto al éxito o fracaso de la clasificación. No obstante, y al tratarse de algoritmos de clasificación no supervisada no existe una fase de aprendizaje previo y la clasificación se producirá sin necesidad de asistencia manual y sin definir categorías previas. Al usarlos se conseguirá una ayuda para evaluar la calidad de los algoritmos.

4.2 PREPROCESADO: KMEANS Y DBSCAN

4.2.1 INTRODUCCIÓN

Tal y como se ha descrito en el apartado anterior, se ha desarrollado una primera fase de preprocesado de la información de entrada. Se demostrará posteriormente, en la fase de análisis y evaluación de los resultados que esta etapa ha sido clave a la hora de obtener una correcta clasificación.

También como ya se ha visto, la implementación de la misma se ha llevado a cabo de igual manera tanto para KMeans como para DBSCAN.

En este primer apartado por tanto se necesitará representar los tweets en un formato que pueda ser procesado posteriormente por los algoritmos. Un tweet consiste en un mensaje corto, de 140 caracteres como máximo. El resultado final buscado en esta fase será obtener una representación para cada uno de los tweets en formato vectorial, representando cada elemento de dicho vector a una determinada palabra.

4.2.2 MANIPULACION INICIAL: FORMATO JSON

Lo primero es hacerse con la colección de tweets que se va utilizar.

Existen multitud de librerías creadas en Internet para poder, usando la API de Twitter, conseguir tweets cada cierto periodo de tiempo de manera automática de cuentas abiertas, de los conocidos como “timelines” (por ejemplo de periódicos, de empresas, de

negocios, ….). Para poder usar todos los servicios para desarrolladores que ofrece Twitter es necesario estar registrado en Twitter y acceder a través de: https://dev.twitter.com. Posteriormente y aprovechando la gran comunidad de desarrolladores y aplicaciones de Twitter es muy fácil crear colecciones de tweet.

Se podría aprovechar por ejemplo la API de la siguiente figura:

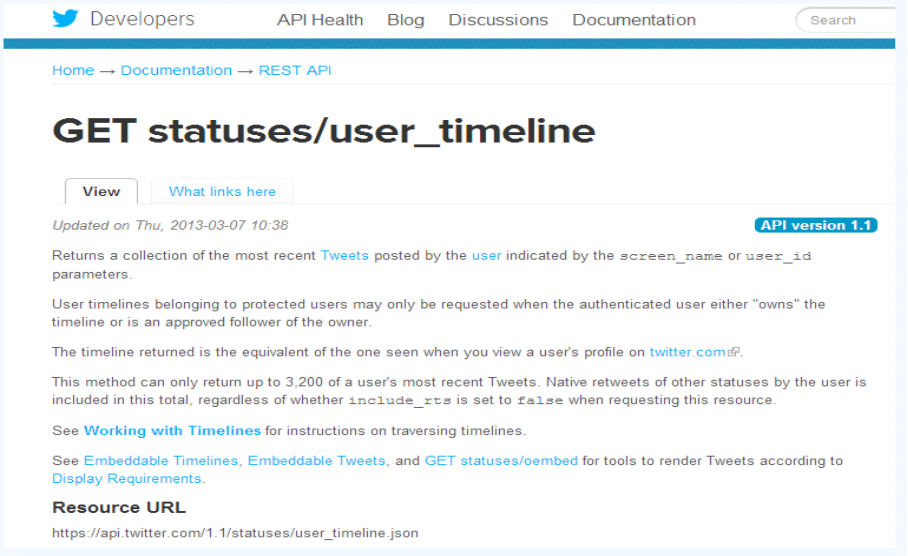


Figura 25. GET statuses/user_timeline de Twitter

Su uso permitirá obtener fácilmente el *"timeline"* de la cuenta deseada, en este caso del diario La Razón.

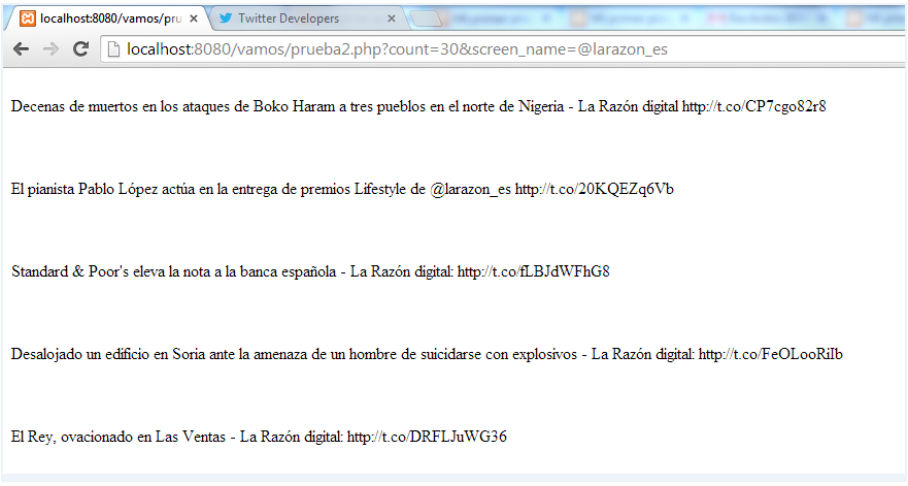


Figura 26. Ejemplo de adquisición de tweets de un timeline a partir de GET user_timeline

No obstante, en esta ocasión se trabajará con una colección de tweets ya dada que permitirá poder evaluar, estudiar y analizar el comportamiento de los algoritmos a la hora de clasificar más fácilmente, como ya se había comentado.

Es frecuente que los tweets de cualquier colección vengan formateados en JSON [www.json.org/json-es.html]. JSON es un acrónimo de *JavaScript Object Notation*, un formato ligero originalmente concebido para el intercambio de datos en Internet. Toda la información del mismo está disponible en el RFC4627. Brevemente, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor y los tiempos de procesamiento de dichos datos son de vital importancia (de aquí su uso por *Yahoo*, *Google*, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar. JSON optimiza los tiempos de respuesta debido a su leve peso.

Por tanto, y lo primero es hacerse con cada uno de los tweets. PHP proporciona una función muy útil para poder hacerse con ellos: *json_decode*. A modo de ejemplo:

Formato JSON :

```
<?php

$json = '{"a":1,"b":2,"c":3,"d":4;}';

var_dump(json_decode($json));

?>
```

Traducido a PHP:

```
object(stdClass)#1 (5) {

   ["a"] => int(1)

   ["b"] => int(2)

   ["c"] => int(3)

   ["d"] => int(
```

Así pues, para hacer uso de la colección a tratar, en PHP:

$tweets = json_decode(file_get_contents('./sevilla.json'), true);

$tweets2 = json_decode(file_get_contents('./madrid.json'), true);

$tweets3 = json_decode(file_get_contents('./iphone.json'), true);

$tweets4 = json_decode(file_get_contents('./uc3m.json'), true);

4.2.3 FORMATO DE UN TWEET

Al recoger un tweet, no se recoge solo el tweet en sí, el mensaje de 140 caracteres. Se recibe junto a él una gran cantidad de información que hace referencia al usuario, al contenido de texto, al idioma, lista de seguidores, …

array(13) { ["id"]=> float(3.9374806640441E+17) ["user"]=> string(9) "ismaelubo" ["name"]=> string(6) "ismael" ["description"]=> string(0) "" ["image"]=> string(63) "http://pbs.twimg.com/profile_images/1747694000/image_normal.jpg" ["retweet_count"]=> int(0) ["text"]=> string(118) "@cristobalsoria yo en la vida privada de la gente no me meto, viva el sevilla y a seguir dándole caña a los vikingos" ["lang"]=> string(2) "es" ["geo"]=> string(0) "" ["coordinates"]=> string(0) "" ["place"]=> string(0) "" ["date"]=> string(19) "2013-10-25 16:37:09" ["raw"]=> array(23) { ["metadata"]=> array(2) { ["result_type"]=> string(6) "recent" ["iso_language_code"]=> string(2) "es" } ["created_at"]=> string(30) "Fri Oct 25 14:37:09 +0000 2013" ["id"]=> float(3.9374806640441E+17) ["id_str"]=> string(18) "393748066404409344" … Ver completo (**Anexo 2)**

Se deberá por tanto seleccionar, la parte que verdaderamente interesa, los 140 caracteres (a lo que ahora en adelante se llamará simplemente tweet). Esto será posible gracias a que toda esta información que incluye el tweet viene organizada por etiquetas, siendo ['text'] la etiqueta que interesa para el estudio de clasificación.

$save = $tweet['text'];

De este modo, la variable $save contendría ya el tweet:

"@cristobalsoria yo en la vida privada de la gente no me meto, viva el Sevilla y a seguir dándole caña a los vikingos".

4.2.4 CONVERSIÓN DE LAS MAYÚSCULAS A MINÚSCULAS

Se procede además a convertir todas las letras mayúsculas en minúsculas para conseguir una representación más compacta de los términos de la colección.

$save = strtolower($save);

"@cristobalsoria yo en la vida privada de la gente no me meto, viva el sevilla y a seguir dándole caña a los vikingos".

4.2.5 DESCOMPOSICIÓN DEL TWEET EN LAS PALABRAS QUE LO FORMAN

Una vez que se ha conseguido el tweet será necesario dividirlo en las distintas palabras que lo forman. Para ello se deberá tener en cuenta que la separación de dichas palabras se puede llevar a cabo por:

- Espacios en blanco
- Signos de puntuación
- Otros símbolos

La función de php *preg_split()* permitirá llevar a cabo esta tarea con relativa facilidad.

$pieces = preg_split();

Por tanto, una vez conseguido recuperar cada tweet y descomponerlo en los distintos términos que lo forman se debe proceder de una vez por todas al análisis de su contenido.

4.2.6 ELIMINACIÓN DE LAS PALABRAS DE PARADA O *STOPWORDS*

Conseguir reducir las dimensiones de la amplia colección con la que se trabaja permitirá mejorar tanto la eficiencia como la eficacia posterior de los algoritmos de clasificación implementados.

Como se ha comentado en el **capítulo 3** existen ciertas palabras que apenas aportan ningún tipo de significado y que por tanto no ayudan nada en el proceso de clasificación. Se trata de las ya conocidas *stopwords* (**apartado 3.2.2.1**).

```
$stopwords = array(
        0 => 'el',
        1 => 'la',
        2 => 'los',
        3 => 'les',
        4 => 'las',
        5 => 'de',
        6 => 'del',
        7 => 'a',
        8 => 'ante',
        9 => 'con',
        10 => 'en',
        11 => 'para',
        12 => 'por',
        ...
```

Por tanto, se lleva a cabo un filtrado de todas ellas (recogidas en el **Anexo 1).** A partir de ahora los tweets pasan a estar constituidos por un conjunto de palabras menor.

```
$pieces = array_diff ($pieces, $stopwords);
```

Además de eliminar estas stopwords de cada tweet, pueden existir otro conjunto de palabras que tampoco sean necesarias durante fases posteriores. En este sentido, se decidió eliminar todas aquellas *URLs* que incluyen y referencian imágenes o páginas web, muy frecuentes en algunos tweets, puesto que va a ser muy complicado que dos tweets redirijan exactamente a la misma fotografía o al mismo vídeo y en el fondo no aportan una información esencial ya que lo que se procesan son palabras y no imágenes o links externos.

4.2.7 LEMATIZACION O STEMMING

Otro apartado clave consistirá en hacer un *stemming* **(apartado 3.2.2.2)** de las palabras restantes. Con ello se conseguirá seguir reduciendo las dimensiones de cada uno de los tweets sin perder apenas información. En el estudio llevado a cabo se ha usado un algoritmo basado en la lematización de Porter para el idioma castellano.

```
foreach ($pieces as $pieza) {
        $st[] = stemm_es::stemm($pieza);
```

Así pues, un ejemplo de este algoritmo se muestra en la **figura 27**.

tórax	torax
torcer	torc
toreado	tor
toreados	tor
toreándolo	tor
torear	tor
toreara	tor
torearlo	tor
toreó	tore
torero	torer
toreros	torer

Figura 27. Ejemplo de aplicación del lematizador

Aplicándolo sobre los tweets de estudio, conjuntamente con la ya comentada eliminación de s*topwords*:

Sevilla ha acertado en fichar -> sevill acert fich

Se venden camisetas -> vend camiset

Las mayores variantes del Barcelona -> mayor variant barcelon

4.3 MODELO DE ESPACIO VECTORIAL Y ASIGNACION DE PESOS

4.3.1 MODELO DE ESPACIO VECTORIAL

Tras llevar a cabo la fase de preprocesado se va a proceder a representar cada tweet siguiendo un formato vectorial. Cada tweet dejará de ser una combinación de palabras para convertirse en una combinación de números, en un vector. Cada dimensión de dicho vector, representará a una palabra. Por ello es frecuente que se formen espacios vectoriales de grandes dimensiones.

Todos los elementos de la colección de entrada, es decir todos los tweets pertenecientes a una misma colección, se representarán con el mismo número de dimensiones. Para ello tras el preprocesado del conjunto de entrada se obtendrán las palabras más representativas de dicho conjunto, y a continuación se irá comprobando la existencia o ausencia de estas palabras en cada tweet, uno a uno. En el caso de que la palabra exista

en un tweet determinado se le asignará un determinado peso a esa palabra en ese tweet, y un 0 si no está presente.

Hay que tener en cuenta, tal y como se vio en el **capítulo 3**, que existen distintos tipos de ponderaciones para las palabras dentro del modelo de espacio vectorial que se ha empleado y por tanto habrá que tomar una decisión de cual implementar basándose en un equilibrio entre rapidez y efectividad. En este caso, se ha optado por usar una asignación de pesos basada en la ya comentada *Term Frequency* para ambos algoritmos. Su elección ha permitido obtener mejores clasificaciones como se verá en el c**apítulo 5** sobre otras como la ponderación binaria y simplificar los trabajos en la matriz de tweets sobre otras como la ponderación TF-ID.

4.3.2 ACONDICIONAMIENTO DEL ESPACIO VECTORIAL: REDUCCIÓN DE DIMENSIONES

Tal y como describió en el **apartado 3.2.3**, se aplica una reducción de dimensiones basada en la frecuencia de las palabras representantes de la colección de entrada.

Para ello, previamente se establece una matriz que contiene todas las palabras usadas en la colección, sin incluir las ya eliminadas durante la fase de preprocesado, así como el número de veces que dichas palabras aparecen en la colección, tal y como se ve en la **tabla 1**.

Tabla 1. Palabras representantes de la colección tras preprocesado y sus frecuencias

Palabras de la colección	Frecuencia
fich	4
delanter	4
sevill	240
sevillafc	4
cristobalsori	2
vid	6
priv	2
gent	11
met	4
…	…

Posteriormente, se establecen unos umbrales que se encargan de eliminar todas aquellas palabras que se repiten prácticamente en todos los tweets y aquellas que aparecen con muy poca frecuencia y que por tanto apenas aportan capacidad de discriminación a la hora de formar clústeres, creándose así el conjunto de palabras representantes de la colección, que puede estar constituida desde unas pocas palabras hasta centenares de ellas, en función del volumen de dicha colección.

La elección de estos umbrales se hace de manera experimental, y se aplican de manera porcentual sobre el total de palabras de todos los tweets de la colección, en rangos cercanos a [0.3% - 90%]. A modo de ejemplo, en los experimentos llevados a cabo en el **capítulo 5**, se trabajó con 1000 tweets que sumaban, tras la fase de preprocesado más de 11000 palabras totales. Por tanto, aquellas palabras que se repitan un mínimo de 33 veces a lo largo de la colección (y menos de 9900 veces) pasarán a ser representantes de esos 1000 tweets. Las que se repitan con una frecuencia menor a 33 (o mayor a 9900) no serán consideradas en las siguientes fases del proceso de clasificación.

```
function reduccion_dimensiones ($Array_de_tweets, $frec_inf, $frec_sup, &$palabras_validas) {...}
```

4.3.3 ASIGNACIÓN DE PESOS: *TERM FREQUENCY*

Tal y como se ha visto en el **apartado 4.3.1**, se asigna una ponderación de pesos en función de la frecuencia de las palabras de la colección de entrada.

En un primer momento se llevó a cabo una asignación binaria; la asignación binaria se trata de la forma más clásica que existe a la hora de asignar pesos, otorgando un 1 si la palabra aparece en un determinado tweet o un 0 si no aparece. No obstante, se comprobó que si existían un número considerable de dimensiones en el espacio vectorial formado, como suele el ser el caso al llevar a cabo este tipo de experimentos, no ofrecía grandes resultados.

Se optó finalmente por llevar a cabo una asignación basada en la frecuencia de las palabras (*Term Frequency,* explicada en el **apartado 3.2.4.2**). Para ello es necesario saber en todo momento la frecuencia de las palabras representantes de la colección de entrada. Se llevó a cabo a su vez una normalización de dichas frecuencias basada en la palabra con mayor frecuencia en el conjunto de entrada de los tweets.

A continuación, se muestra el proceso seguido para llevar a cabo la ponderación de las palabras de los tweets de la colección de entrada (**figura 30**).

```
$maximo = max($palabras_representantes_de_coleccion);
for ($i = 0; $i < count(Array_de_tweets); $i++) {
    foreach ($palabras_representantes_de_coleccion as $elemento => $frq) {
        if (array_key_exists($elemento, $Array_de_tweets[$i])) {
            $data3[] = $frq;
        }
        else{
            $data3[] = 0;
            }
    }

    foreach ($data3 as $key3) {
        if ($maximo != 0)
            $P = $key3 / $maximo;
         else
            $P = $key3;
        $data[] = $P;
    }
    $valor[] = $data;
    ...
    }//end for
```

Figura 30. Asignación de pesos sobre tweets de Twitter

4.4 CLASIFICACIÓN: KMEANS

4.4.1 PSEUDO-CÓDIGO:

1. Se empieza con un número arbitrario de centroides
2. Se itera hasta que no se produzcan nuevos movimientos:

    2.1 Se asigna cada dato al centroide más cercano que tengan.

    2.2 se actualiza la posición de cada centroide en el espacio:

        2.2.1 Recalculo de centroides

$$p_i = \frac{1}{|S_i|}\sum_{x_j \in S_i} x_j$$

Cada vector de entrada => X_j

Cada centroide => P_i

4.4.2 FUNCIONES DESARROLLADAS E IMPLEMENTADAS: ALGORITMO KMEANS

A la hora de implementar el algoritmo KMeans se ha procedido a crear una serie de

funciones que desarrollan cada una de las fases más importantes en los que se descompone este algoritmo. Finalmente se optó por escoger como medida de disimilitud entre los clústeres la distancia euclídea, por ser la más común en la implementación de KMeans y proporcionar buenos resultados. Dicha distancia euclídea estará normalizada proporcionando valores entre [0,1]; de este modo dos tweets muy parecidos estarán separados por una distancia euclídea próxima a 0 y dos tweets diferentes estarán separados por una distancia en torno a 1.

A continuación se van a explicar las funciones desarrolladas durante la implementación de KMeans.

FUNCIÓN PRINCIPAL: se seleccionan el número de clústeres que se desean, además de pasarse la colección de entrada de tweets y diversos parámetros necesarios para la configuración del algoritmo detallados a continuación.

Al finalizar cada iteración esta función lleva a cabo la comprobación de si se ha alcanzado la convergencia del algoritmo y por tanto se da por finalizado el mismo o si por el contrario es necesaria una nueva iteración. Alcanzar convergencia implica que los centroides calculados en la última iteración del algoritmo resultan ser iguales que los obtenidos en la nueva y por tanto última iteración. Para ello se realiza una copia de los centroides antiguos y se comparan con los nuevos centroides recalculados.

```
/**
* Función principal KMeans
* @param $xs es un array que representa a cada uno de los tweets de entrada al algoritmo
* @param type $k número de grupos que se van a formar, parámetro fundamental del KMeans
* @param type &$centroids almacena los centroides del algoritmo para tenerlos accesibles para futuros
  estudios .
* @ param type $iteraciones devuelve el número de iteraciones hasta alcanza la convergencia.
* @return int devuelve los clústeres formados, incluyendo los integrantes de los mismos.
*/
function kmeans ($xs, $k, &$centroids,$iteraciones)
```

SELECCIÓN DE LOS CENTROIDES INICIALES P_J: función para seleccionar los centroides iniciales; devolverá el número de centroides que se haya solicitado a través del parámetro de entrada *k*. Esta función no permite que los centroides iniciales escogidos sean iguales, puesto que esto implicaría la creación de un grupo menos de los que se había solicitado previamente. Se lleva a cabo una selección aleatoria de *k* tweets de entre todos aquellos que forman parte de la colección de entrada; estos constituirán los centroides iniciales o centroides semilla del algoritmo.

```
/**
* Selección de los centroides iniciales pj
* @param type $k número de grupos que se desean formar
* @param type $x datos de entrada
* @return devuelve los centroides semilla iniciales
*/
function init_centroids ($k, $xs)
```

CALCULAR EL CENTROIDE MÁS CERCANO A CADA PUNTO, FORMAR LOS CLÚSTERES: función que asocia cada vector de entrada (tweet) a su centroide más cercano; para ello hace uso de la función distancia euclídea, invocada desde el interior de esta función.

```
/**
* Calcula el centroide más cercano para un punto en concreto.
* @param type $x dato de entrada
* @param type $centroids centroides de cada clúster
* @return type devuelve el centroide más cercano a dicho punto
*/
function closest_centroid ($x, $centroids)
```

RECOLOCACIÓN DE LOS CENTROIDES: función que actualiza los centroides de los clústeres que se van formando. A modo de ejemplo, si se trabajase con los siguientes datos ficticios que siguen un formato vectorial (en este caso de ejemplo se usan únicamente 2 dimensiones):
[a] (61.0, 100.0)
[b] (64.0, 150.0)
[c] (70.0, 140.0)
El enfoque adoptado para calcular el centroide de dichos puntos consiste en llevar a cabo un promedio matemático de modo que:

[m] = ((61.0 + 64.0 + 70.0) / 3, (100.0 + 150.0 + 140.0) / 3)
= (195.0 / 3, 390.0 / 3)

Centroide_final = (65.0, 130.0)

La única diferencia que existe entre este caso de ejemplo y el algoritmo implementado reside en el uso de espacios vectoriales de dimensiones mucho mayores en este último.

```
/**
* Recolocación de los centroides a la media de sus miembros
* @param type $centroids los centroides actuales
* @param type $belongs_to los clúster formados
* @param type $xs vector de entrada
* @return type devuelve los nuevos centroides recalculados.
*/
function reposition_centroids ($centroids, $belongs_to, $xs)
```

FUNCIÓN AUXILIAR: Función que calcula de las distancias euclídeas normalizadas entre dos tweets. Es posible gracias a que en etapas anteriores cada tweet se ha representado de manera vectorial.

```
/**
* Calcula la distancia euclídea entre 2 puntos.
```

```
* @param type $centroid centroide

* @param type $x dato de entrada

* @return type distancia entre ambos puntos, normalizada

*/

function  distance_to_centroid ($x, $centroid)
```

Además, se han implementado diversas funciones que permitan evaluar los clústeres formados:

function distance_between_centroids(): calcula, utilizando la función distancia euclídea, la distancia entre los centroides de las agrupaciones formadas.

function see_centroids(): permite obtener los centroides finales de los clústeres formados.

function numberOfIterations(): para conocer el número de iteraciones que ejecutó el algoritmo

function more_common_words(): para conocer la frecuencia de las palabras más repetidas de cada clúster y que darán nombre a dicho clúster. Puesto que todos los tweets presentan una notación vectorial, donde cada palabra es una dimensión y todos los tweets son representados con las mismas dimensiones, basta con sumar los tweets de cada clúster entre sí (antes de haber realizado la normalización por la frecuencia de la palabra más repetida como fue comentado en el **apartado 4.4.3)**.

function numberOfIntegrants(): Permite conocer el número de elementos que componen cada clúster.

function intracluster_distance(): Conociéndose los centroides de cada clúster y sus elementos utiliza la distancia euclídea para conocer la distancia intra-clúster media de cada grupo.

4.5 CLASIFICACIÓN: DBSCAN

A la hora de implementar el algoritmo DBSCAN se ha procedido a crear una serie de funciones que desarrollan cada una de las fases más importantes en los que se descompone este algoritmo.

4.5.1 PSEUDO-CÓDIGO

Se reciben como parámetros un conjunto de objetos D, el tamaño del radio de la vecindad, *Épsilon*, y el mínimo número de elementos que debería haber dentro de ese radio para formarse un clúster, *Minpoints.*

```
DBSCAN (D, eps, MinPts)
  Para cada punto P del conjunto de entrada D
    marcar P como visitado
    vecindad = points_in_region (P, eps)
    if tamaño(vecindad) < MinPts
      marcar P como ruido
    else
      expandCluster (P, Vecindad, C, eps, MinPts)

expandCluster(P, Vecindad, C, eps, MinPts)
  añadir P al cluster C
  for each point P' de Vecindad
    if P' no ha sido visitado
```

```
        marcar P' como visitado
        Vecindad' = points_in_region (P', eps)
        if tamaño(Vecindad') >= MinPts
          Vecindad = Vecindad se junta con Vecindad'
      if P' no es miembro de ningún clúster
        añadir P' al cluster C

points_in_region(P, eps)
  return todos los puntos dentro de P's eps-vecindad
```

4.5.2 FUNCIONES DESARROLLADAS E IMPLEMENTADAS: ALGORITMO DBSCAN

Las principales funciones diseñadas para llevar a cabo el algoritmo DBSCAN han sido las siguientes.

FUNCIÓN PRINCIPAL: se establecen los parámetros del algoritmo: tanto *Épsilon* o radio de la vecindad como *MinPoints*, o el número mínimo de puntos necesarios dentro del radio de la vecindad para cada nodo para formar un clúster. Devuelve las agrupaciones formadas.

```
/**
* Función principal del DBSCAN
* @param type $data , vectores de entrada en formato binario.
* @param type $epsilon, parámetro básico I DBSCAN
* @param type $minimumPoints, parámetro básico II DBSCAN
* @param type $palabras_validas, palabras representantes de la colección de tweets
* @param type $mapeo_tweets, tweets en formato string
```

```
*/

function dbscan ($data, $epsilon, $minimumPoints, $palabras_validas, $mapeo_tweets)
```

PUNTOS DENTRO DE LA VECINDAD, RADIO ÉPSILON. Encuentra todos aquellos nodos en la vecindad de un nodo determinado. Hace uso de la distancia euclídea normalizada.

```
/**

*  Función que calcula los puntos dentro del radio-épsilon

* @param type $point

* @param type $data vectores de entrada posibles vecinos de $point

* @param type $epsilon, radio de la vecindad

* @return type array que contiene aquellos puntos del vector de entrada que están a  distancia menor
    que  epsilon de $point

*/

function points_in_region ($point, $data, $epsilon)
```

EXPANDIR CLÚSTERES: Función que lleva a cabo la expansión de un clúster en el caso de que se den las condiciones necesarias, es decir, que dicho nodo que pertenecía ya a un clúster tenga más cantidad de *MinPoints* vecinos dentro de su vecindad de radio *Épsilon*.

```
/**

* Función para la expansión de un clúster

* @param type $point punto central o nuclear

* @param type $data vectores vecinos de $point

* @param type $epsilon radio de la vecindad

* @param type $minimumPoints mínimo número necesario de puntos en una región
```

```
* @param type $visited vectores de entrada ya visitados

* @param type $data2 vectores de entrada

*/

function expand_cluster ($point, $data, $epsilon, $minimumPoints, &$visited, $data2)
```

FUNCIÓN AUXILIAR I: Función que calcula de las distancias euclídeas normalizadas entre dos tweets. Es posible gracias a que en etapas anteriores cada tweet se ha representado de manera vectorial.

```
/**

* Calcular la distancia euclídea entre dos vectores (tweets)

* @param type $point

* @param type $datum

* @return type distancia euclídea entre 2 puntos

*/

function linear_euclidian_distance ($point, $datum)
```

FUNCIÓN AUXILIAR II: función que permite la unión de dos clústeres.

```
/**

* Función para la unión de 2 clústeres

* @param type $one clúster 1

* @param type $two clúster 2

* @return type la unión de ambos clústeres

*/
```

```
static function __ll__join__clusters($one, $two) {
```

Además se han implementado diversas funciones que permitan evaluar los clústeres formados:

function more_common_words(): idéntica a la función del mismo nombre desarrollada para el algoritmo KMeans.

function numberOfIntegrants(): idéntica a la función del mismo nombre desarrollada para el algoritmo KMeans.

Capítulo 5

Estudio y evaluación de los resultados

5.1 COLECCIÓN DE TWEETS DE ENTRADA

Se va a proceder a llevar a cabo un estudio sobre los dos algoritmos de clasificación no supervisada implementados: KMeans y DBSCAN. Posteriormente se compararán el uno con el otro para intentar determinar cuál aporta mejores resultados y en qué condiciones sobre un conjunto de entrada muy particular: tweets de la red social Twitter. Para llevar a cabo el estudio de ambos algoritmos se va a hacer uso de una colección de 1000 tweets. Estos tweets han sido recopilados previamente y tratan de 4 temas principales:

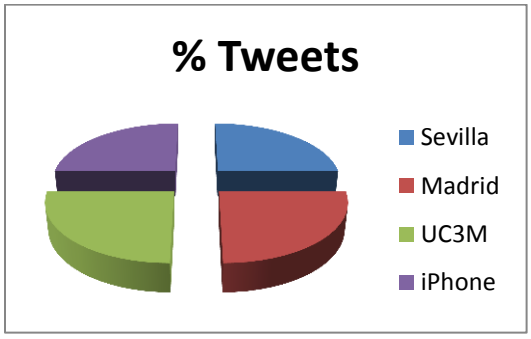


Figura 28. Colección de tweets

De este modo va resultar más fácil estudiar y analizar los resultados obtenidos puesto que se conoce de antemano los grupos principales que a priori deberían formarse, así como el número de integrantes de cada grupo, 250.

No obstante, hay que recordar que se trabaja con algoritmos de clasificación no supervisada, es decir, que no requieren de una clasificación previa de sus datos de entrada. El conocimiento previo de la colección por tanto solo será usado a posteriori con el objetivo de evaluar los resultados.

Los tweets serán representados vectorialmente, representando cada palabra una dimensión del espacio vectorial generado.

Se empleará la distancia euclídea normalizada para calcular las distancias entre tweets, calcular las distancias a los centroides y calcular las distancias intra-clúster según corresponda. Se ha decidido establecer, por experimentación, que todas aquellas palabras que aparezcan en el rango [0.3% - 90%] del total de palabras totales (más de 11000 tras el preprocesado inicial de los 1000 tweets) se conviertan en representantes de la colección.

5.2 KMEANS

5.2.1 INTRODUCCIÓN

El algoritmo KMeans presenta una serie de inconvenientes que han quedado reflejados notablemente durante la fase de análisis y evaluación de los resultados.

El primero de ellos tiene que ver con la elección del número de clústeres que debe estimarse de manera manual por el programador/usuario. Es muy difícil establecer a priori un número total de clústeres, y la elección de una cantidad u otra genera resultados totalmente diferentes, desde la agrupación de miles de documentos que poco tienen que ver en la misma agrupación, hasta la creación de decenas de clústeres diferentes que podrían perfectamente constituir uno solo.

El segundo tiene que ver con la elección de los centroides semillas de cada clúster. Es decir, el algoritmo KMeans requiere de la elección inicial de una serie de centroides sobre los que comienza a trabajar. De nuevo, dependiendo de la elección de estas agrupaciones iniciales se obtendrán unos resultados u otros. Una de las elecciones que se suele hacer a la hora de elegir estos clúster semillas, y la más simple, es que la elección sea aleatoria. No obstante, esta puede no ser la mejor solución puesto que los mejores resultados se

obtienen cuando más distancia haya habido entre los centroides iniciales elegidos.

Una vez llevada a cabo la clasificación, uno de los resultados deseables es que las distancias entre los centroides de cada clúster final se hallan maximizado y que cada grupo formado sea lo más compacto posible, esto es, que la media de las distancias de los elementos de un grupo respecto al centroide de dicho grupo sean lo mínimas posibles.

Este algoritmo también se puede ver afectado por los "outliers", aquellos documentos que presentan una gran anomalía frente al resto de documentos de entrada y que en la fase de recálculo de los centroides pueden repercutir negativamente sobre los resultados finales. No obstante, puesto que se lleva a cabo una normalización a la hora de asignar pesos a las palabras en la representación vectorial de cada tweet, al mismo tiempo que se establecen unos rangos de frecuencia que impiden a las palabras o bien que se repiten siempre o bien que no aparecen casi nunca aparecer en la representación vectorial de la información de entrada, se evita en gran medida este problema típico del algoritmo.

En los **apartados 5.2.2, 5.2.3, 5.2.4, 5.2.5**, **5.2.6** se procede a llevar a cabo varias simulaciones de KMeans que ponen de manifiesto las características del algoritmo.

5.2.2 EXPERIMENTACION CON KMEANS: ELECCIÓN DE LOS CENTROIDES INICIALES

Como ya es conocido, uno de los principales problemas de KMeans es que necesita que le sean definidos una serie de clústeres iniciales o clústeres semillas para poder empezar a iterar. El principal objetivo, como también se ha visto es conseguir empezar por los k tweets más separados entre sí de la colección. No obstante, la forma más típica de elección en el algoritmo KMeans ha sido tradicionalmente la elección aleatoria.

KMeans, tal y como se vio en el **apartado 2.3.3.2** tenía como objetivo minimizar la función:

$$\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \mathbf{p}_i \right\|^2$$

No obstante, la aleatoriedad comentada más arriba da lugar a que en muchas ocasiones se alcancen mínimos locales, y no globales, obteniéndose diferentes resultados incluso iterando sobre el mismo conjunto de datos. Esto puede provocar malas agrupaciones en

algunas ocasiones, especialmente si se trabaja sobre conjuntos representados por espacios vectoriales de grandes dimensiones, como es el caso.

5.2.2.1 EVOLUCIÓN DE LOS GRUPOS EN KMEANS

Se procede a simular el algoritmo KMeans sobre el conjunto de tweets de entrada descritos en el **apartado 5.1**. Se pretende observar la evolución de los grupos a lo largo de las distintas iteraciones que KMeans lleva a cabo hasta que alcanza una solución final. Los resultados que se obtienen son los que siguen.

COLECCIÓN EVALUADA

Colección: 1000 tweets de entrada

Numero de clúster elegidos: 4

Centroides iniciales: elegidos aleatoriamente.

Numero de iteraciones hasta alcanzar solución final: 7

Dimensiones del espacio vectorial representadas por:

sevill, diez, provinci, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, lleg, hem, luch, present, barcelon, chic, gratis, exam, gan, 5, iphon, barç, real, clásic, huelg, 5s, uc3m, mareaverdeuc3m, profesor.

Se observa como a lo largo de las 7 iteraciones que en este caso particular han tenido lugar, los tweets van ajustándose y acomodándose a su grupo, según se puede observar en **la tabla 2**. En las **tablas 3 y 4** se representan las distancias euclídeas normalizadas entre los centroides de los grupos formados y las palabras más importantes de cada uno de ellos, respectivamente.

Tabla 2. Evolución de los clústeres- KMeans

Iteración	*Clúster*	*Número de tweets*
	Clúster 0 (Sevilla)	81
	Clúster 1 (Madrid)	158
	Clúster 2 (iPhone)	634
	Clúster 3 (UC3M)	36

Iteración II	*Clúster*	*Número de tweets*
	Clúster 0 (Sevilla)	80
	Clúster 1 (Madrid)	150
	Clúster 2 (iPhone)	311
	Clúster 3 (UC3M)	368
Iteración III	***Clúster***	***Número de tweets***
	Clúster 0 (Sevilla)	125
	Clúster 1 (Madrid)	150
	Clúster 2 (iPhone)	255
	Clúster 3 (UC3M)	379
Iteración IV	***Clúster***	***Número de tweets***
	Clúster 0 (Sevilla)	315
	Clúster 1 (Madrid)	150
	Clúster 2 (iPhone)	234
	Clúster 3 (UC3M)	210

Iteración V	*Clúster*	*Número de tweets*
	Clúster 0 (Sevilla)	258
	Clúster 1 (Madrid)	230
	Clúster 2 (iPhone)	232
	Clúster 3 (UC3M)	189

Iteración VI	*Clúster*	*Número de tweets*
	Clúster 0 (Sevilla)	222
	Clúster 1 (Madrid)	266
	Clúster 2 (iPhone)	232
	Clúster 3 (UC3M)	189

Iteración VII	*Clúster*	*Número de tweets*
	Clúster 0 (Sevilla)	220
	Clúster 1 (Madrid)	266
	Clúster 2 (iPhone)	234
	Clúster 3 (UC3M)	189

Tabla 3. Distancia entre centroides finales

	Clúster 0	Clúster 1	Clúster 2	Clúster 3
clúster 0	0.0	0.5	0.6	0.8
clúster 1	0.5	0.0	0.5	0.6
clúster 2	0.6	0.5	0.0	0.8
clúster 3	0.8	0.6	0.8	0.0

Tabla 4. Palabras con mayor frecuencia

	Palabras más usadas	Número de veces que aparecen
clúster 0	Sevilla	202
	chicas	30
clúster 1	Madrid	292
	Real	80
clúster 2	iPhone	246
	5s	31
clúster 3	UC3M	202
	Mareaverdeuc3m	30

Clúster 0: Sevilla

RT @ocondacofrade: Una representación de Hermandades de Cádiz, invitada mañana a Sevilla con motivo del Año Jubilar Macareno. http://t.co/E…
RT @AESevilla: El lunes 28 se celebrará en Cámara Sevilla Jornada sobre los primeros pasos de la Reforma de la PACC http://t.co/YHJNnWXMlk
Es posible que en TODA SEVILLA no haya una clase de pilates cuyo horario me venga bien??!!
RT @NazaretMorena95: @ismael_trianero pero calla xikillo k no se tiene k enterar nadie jajaja y tu mañana a Sevilla
direccion Sevilla.... uffff vaya mañana he tenido!!!
Que alegría se debe estar viviendo en #Sevilla hoy... y yo sin poder estar ahí y comiéndome los mocos!! Buaaa http://t.co/AYLrXj9Vr5
*Yo y mi obsesión con la semana santa de sevilla *-**
@evitaabelieve Hoy hablamos de Sevilla y todos mirandome y yo en plan: ¿Qué coño...? jskaijdisjid

RT @AlbaGarcia_162: Yo tambien quiero ir a Sevilla jo...
vamonos para sevilla a dar una vueltecita :)

Clúster 1: Madrid

RT @GolCaracol: Barcelona vs. Real Madrid: ¡el clásico español da para todo! http://t.co/VSxO8VdtoM #CalentandoelClásico
Holi, Madrid, te quiero. http://t.co/W4voCHFwOL
Las mayores variantes del Barcelona están en las bandas. Neymar, Aléxis Sánchez, Pedro, Cesc o, último recurso, Tello http://t.co/nHIiPdYZgL
RT @Real_Madrid_CF: De los 6 últimos clásicos disputados el Real Madrid ha ganado TRES, ha empatado 2 y perdido tan solo UNO.
Ole que a estas horas mañana estaré en Madrid :)
RT @Josemicod5: RT Barça, FAV Real Madrid, NADA ninguno o te la pela el fútbol xD... A ver que sale =D.
Menuda semanita mostró espera... pero mientras tanto a descansar y A VER EL CLASICOOO ÁNIMO MADRID!!!
RT @elpadrecorajede: Vista aérea de la manifestación De Madrid #24O Confirmado hemos ido 4 http://t.co/oeDTM8rf0G

Clúster 2: iPhone

Arrancó de #defiesta977 on air http://t.co/6MNLvMjcZX o http://t.co/n1TlHO8sF3 desde el Bb o iPhone
RT @EldelaPuerta: Tener un iPhone o Android y tener BBM, es como tener un TV OLED de 60" y ver películas grabadas de TV en un BETAMAX. #pin…
¡He reunido 43,260 monedas de oro! ¿A ver quien es capaz de ganarme? http://t.co/c8m5TQsFJw #iphone, #iphonegames, #gameinsight
Ese pasillo esta bien terminado "@Lewisbk09: Nice iphone RT @PostBadBitches: #body http://t.co/2aQcAi3s4v"
Mi iPhone se apagó de manera compulsiva. Prendió, por suerte. Ahora le echaré leña para que funcione.
RT @Julian_Gonzalz: @SoloUnaMortal_ Mira que hermoso se ve mi iPhone ahora. http://t.co/RaY9d5BJei
Las mejores aplicaciones de Fitness para iPhone 5s http://t.co/pGhuXiAisO"
@kamilo_velez ya gracias me hackearon mi cuenta y creo q es el q te tiene mi id! de apple ahora no puedo validar mi iphone lo voy a perder.
RT @A_iPhone: BlackBerry Messenger ya ha sido descargado más de 10 millones de veces

http://t.co/eAPVD6szzS
—¿QUÉ QUEREMOS?—¡Quitarle el auto corrector al "iPhone"—¿CUÁNDO LO QUEREMOS?—¡AHORCA!—¡AHORRO!—¡ABORTA!—¡ALBORA!—

Clúster 3: UC3M

Hoy el Sr. Miguel Muñoz, Abogado del Estado Jefe, imparte el módulo "The Legal Order of the EU: General Guidelines".
RT @meriblay: Elecciones en la Uc3m. #Lorenaforpresident @22_logen http://t.co/GCkPg2Wkt0
La II Convocatoria BECAS ALUMNI UC3M se entregaron en el campus de Getafe a 12 nuevos estudiantes de Grado #UC3M... http://t.co/eIYCPnmizu
Gracias @emprendoteca y @eltenedor por todos los RTs y difusión a nuestro evento de hoy. "El dinero no es lo primero" a las 1945 n @uc3m
RT @AlvEnDiferido: La biblioteca de la UC3M ha cerrado porque el 100% de sus trabajadores ha secundado la huelga. Imprescindibles para la u…
RT @CGTUC3M: Ayer, el rector de la #uc3m cerró las puertas a la decencia

ANÁLISIS

Por lo tanto en 7 iteraciones el algoritmo KMeans alcanza una solución final. Comprobando los resultados se observa que estos son bastante buenos ya que ha conseguido separar los 4 temas más importantes con un alto porcentaje de éxito (**figura 29).**

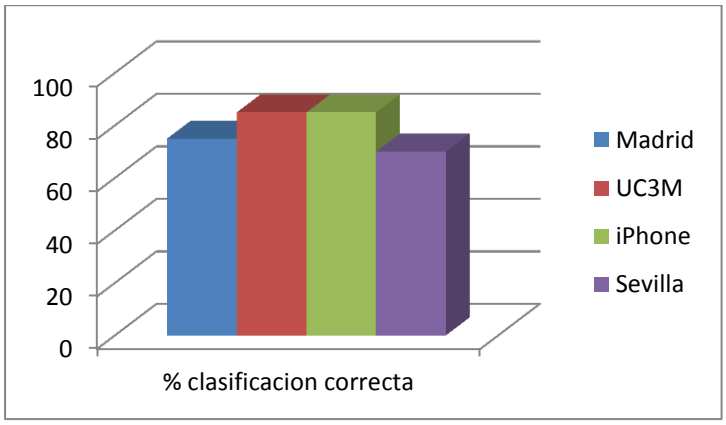


Figura 29. Porcentaje de éxito

No obstante, estos porcentajes de éxito tan elevados se deben a dos factores principales:

- De antemano se conocía que existían 4 temas principales. En iteraciones normales del algoritmo este factor no se conoce a priori y provoca que en numerosas ocasiones KMeans realice malos y muy diferentes agrupamientos.
- Los centroides semillas de los clústeres, elegidos aleatoriamente, han llegado a buenos resultados, y esto no tiene por qué ser siempre así. Como se verá en el siguiente apartado, **apartado 5.2.2.2**, los clústeres semillas juegan un papel muy importante.

Analizando cuidadosamente los grupos formados y las palabras que más se repiten en cada uno de ellos se puede observar como suele existir un tema principal, acompañado de un tema secundario. Por ejemplo en el clúster de iPhone, la palabra iPhone aparece 246 veces y 31 la palabra 5S (uno de los tipos de iPhone que existen actualmente en el mercado). Con lo cual esto evidencia que existe un claro subgrupo que tratará sobre dicho tipo de móvil. Lo mismo ocurre en el resto de clústeres. Como ya se ha comentado, conocer el número de clústeres a priori que se deben formar es una tarea complicada. En el **apartado 5.2.3** se procede a intentar descubrir dichos subgrupos a partir de aumentar el número de grupos que deben formarse.

Hay que tener en cuenta que, tras llevar a cabo el primer experimento anterior, se comprobó que de entre los 1000 tweets procesados, había 91 cuya representación vectorial fue (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), esto es, no habían quedado representados por ninguna de las palabras más generales, frecuentes y representativas de la colección. Para proceder a una mejor clasificación, se consideró a estos 91 tweets como tweets que poco o no tenían que ver con el resto y se procedió a llevar a cabo el algoritmo sobre el resto de elementos. Esta consideración será contemplada de aquí en adelante.

5.2.2.2 ELECCIÓN ALEATORIA DE LOS CENTROIDES SEMILLA

Si se sigue manteniendo la premisa de que se conocen a priori la existencia de 4 temas principales en la colección de tweets anterior y se realizan nuevas simulaciones, se observa el efecto que introduce la aleatoriedad en la elección de los centroides iniciales.

COLECCIÓN EVALUADA

Colección: 1000 tweets de entrada

Numero de clústeres elegidos: 4

Centroides iniciales: aleatorios

Dimensiones del espacio vectorial representadas por:

sevill, diez, provinci, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, lleg, hem, luch, present, barcelon, chic, gratis, exam, gan, 5, iphon, barç, real, clásic, huelg, 5s, uc3m, mareaverdeuc3m, profesor.

SIMULACIÓN I

Se calculan las distancias entre los centros de los clústeres finales.

Tabla 5. Distancia entre los centroides finales formados - KMeans

	Clúster 0	Clúster 1	Clúster 2	Clúster 3
clúster 0	0.0	0.8	0.6	0.4
clúster 1	0.8	0.0	0.5	0.7
clúster 2	0.6	0.5	0.0	0.5
clúster 3	0.4	0.7	0.5	0.0

Tabla 6. Datos de las agrupaciones formadas- KMeans

	Número de tweets	Distancia intra-clúster	Palabras más usadas	Frecuencia
clúster 0	180	0.021	UC3M	301
			Mareaverdeuc3m	80
clúster 1	234	0.029	iPhone	246
			5S	31
clúster 2	486	0.286	Madrid	292
			Sevilla	238
clúster 3	9	0.020	Madrid	9
			UC3M	9

ANÁLISIS

En esta ocasión se han formado nuevas agrupaciones, tras 4 iteraciones. Han quedado unidos los tweets que hablaban tanto de Sevilla como de Madrid formando una agrupación de grandes dimensiones (y una distancia intra-clúster mucho mayor que la del resto de grupos). El clúster 0 y el clúster 3, los cuales tratan temas relacionados con la UC3M, han quedado demasiado cerca uno del otro, tal y como se aprecia en la **tabla 5**, lo que tampoco es un buen dato.

Los resultados obtenidos no son por tanto los esperados. Un mínimo local y no global ha sido alcanzado.

SIMULACIÓN II

Se calculan las distancias entre los centros de los clústeres finales.

Tabla 7. Distancia entre los centroides finales formados- KMeans

	Clúster 0	Clúster 1	Clúster 2	Clúster 3
clúster 0	0.0	0.8	0.6	0.8
clúster 1	0.8	0.0	0.4	0.6
clúster 2	0.6	0.4	0.0	0.5
clúster 3	0.8	0.6	0.5	0.0

Tabla 8. Datos de las agrupaciones formadas- KMeans

	Número de tweets	Distancia intra clúster	Palabras más usadas	Frecuencia
clúster 0	189	0.040	UC3M Mareverdeuc3m	211 14
clúster 1	234	0.029	iPhone 5S	246 31
clúster 2	287	0.127	Madrid Real	292 81
clúster 3	199	0.020	Sevilla chicas	202 30

ANÁLISIS

En esta ocasión se ve como los resultados obtenidos son bastante precisos puesto que se forman las 4 agrupaciones mínimas que deberían, tras 6 iteraciones. Se comprueba que los grupos formados son además compactos puesto que en general poseen una distancia intra-cluster pequeña. Además, se ve como la distancia entre los centroides de los clústeres finales es por lo general grande, lo que como ya se ha visto es bueno a la hora de llevar a cabo la clasificación ya que así las agrupaciones han acabado lo más lejos posibles unas de otras.

5.2.3 TÉCNICAS PARA CONSEGUIR DETERMINAR A PRIORI UN NÚMERO DE CLUSTERES ADECUADO

Existen un conjunto de técnicas que permiten intentar averiguar de manera aproximada el número de agrupaciones finales que podrían finalmente formarse. No obstante, no se trata de mecanismos totalmente fiables, y su objetivo muchas veces es dar una idea general de lo que podría ocurrir.

Se van a presentar dos de las más usadas sobre conjuntos de documentos representados en un espacio vectorial.

5.2.3.1 NÚMERO DE CLÚSTERES EN BASES DE DATOS DE DOCUMENTOS

Este mecanismo se basa en [3]. Teniendo un conjunto de documentos D, que integran la matriz D de dimensiones m, el número total de documentos, en este caso tweets, y n, el número total de palabras que representan estos documentos, es claro que el número de clústeres va a tener una cierta dependencia tanto del número total de documentos, m, así como del número de palabras que los forman, n: a más documentos en principio más clústeres diferentes podrían formarse; lo mismo ocurre con el número de dimensiones de la representación vectorial de cada tweet: a mayor número de dimensiones (conjunto de palabras que se usan en la representación de los tweets), más posibles agrupaciones podrían formarse ya que es más probable que se esté tratando nuevos temas. A su vez, al estar usando una representación vectorial, se podría dar el caso de que a pesar de que cada documento se representase por un mayor número de dimensiones (una n mayor) como se acaba de ver, si la mayoría de estas nuevas dimensiones fuesen cero (puesto que no sean palabras con una elevada frecuencia dentro de la colección) es muy probable que

no aumentasen la posibilidad de que se estuviesen tratando nuevos temas (puesto que más bien se trataría de temas aislados y no de carácter general). Por tanto, se puede establecer una relación entre los tres parámetros que indiquen de una manera aproximada, sobre bases de datos que contienen representaciones vectoriales de centenares o miles de documentos el número de clústeres:

$$\mathbf{Número}_{\mathbf{aproximado\ clústeres}} = (\mathrm{m} * \mathrm{n})/\mathrm{t}$$

siendo *t* el número de entradas que no son ceros dentro de la representación vectorial de todo el conjunto de documentos que forman la matriz D; m es el número total de documentos; n es el número de palabras representantes de toda la colección.

No obstante en muchos casos se trata de una aproximación o de una orientación. Sobre esta colección en particular se obtienen los resultados que a continuación se detallan.

Palabras representantes de la colección de entrada:

sevill, diez, provinci, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, lleg, hem, luch, present, barcelon, chic, gratis, exam, gan, 5, iphon, barç, real, clásic, huelg, 5s, uc3m, mareaverdeuc3m, profesor.

Tabla 9. Clústeres formados a partir del estudio de Can, F.; Ozkarahan, E. A.

	Número de tweets	Palabras representativas del clúster
clúster 0	155	UC3M Mareaverdeuc3m
clúster 1	9	UC3M Huelga
clúster 2	201	Sevilla Chicas
clúster 3	11	Llegar Presentar
clúster 4	36	Alerta Madrid
clúster 5	9	UC3M
clúster 6	2	Barcelona
clúster 7	3	Madrid Chicas

clúster 8	118	Madrid Ver
clúster 9	232	iPhone 5S
clúster 10	8	Barça Real
clúster 11	80	Madrid Real
clúster 12	7	Llegar Haber
clúster 13	9	UC3M Madrid
clúster 14	10	Madrid Barcelona
clúster 15	19	Barça Madrid

Clúster 0: UC3M/Mareaverdeuc3m

Defendiendo nuestra exigencia al Rector de la UC3M que aplique lo que propugna de puertas afuera: http://t.co/eIpQoHAqAL @MareaVerdeUC3M
RT @CharlesDio: Algunas de las pancartas de la comisión del #EncierroUC3M por la @uc3m. Buen trabajo! @MareaVerdeUC3M http://t.co/NiWmgIBoWl
RT @MareaVerdeUC3M: COMPAÑEROS Y COMPAÑERAS El MIÉRCOLES Y EL JUEVES LA UC3M HA DEMOSTRADO QUE CUANDO SE QUIERE MOVER, LO HACE Y MUY BIEN. …
RT @yolandatricolor: Que un profesor en la #uc3m hoy pusiera un examen y quien no se presentara suspendia es democracia no? #andaalamierda

Clúster 1: UC3M /Huelga

RT @AlvEnDiferido: La biblioteca de la UC3M ha cerrado porque el 100% de sus trabajadores ha secundado la huelga. Imprescindibles para la u…
RT @MareaVerdeUC3M: Se ha tenido un seguimiento cercano al 80% de la huelga en la UC3M, #sísepuedeysísemueve #HuelgaUC3M
Alumnos y padres tiran de la huelga http://t.co/FSK3B1xAYe #uc3m

Clúster 2: Sevilla/Chicas

RT @RBb_betis1907: El Betis, el equipo español mas visto en Europa League en #Cuatro y #GolTV, con el Valencia como segundo y el Sevilla co…
VIA @raphaelnet_com @RAPHAELartista presenta hoy y mañana Disco Tour #migrannoche en Sevilla LOCALIDADES AGOTADAS http://t.co/eJ2eDfHWzb
#SIDHARTA LISTA DE LUISMA HERVÁS DE 12 A 2H COPA PARA CHICAS y GRUPO 5 CHICAS LAMBRUSCO GRATIS INF 687.169.173 @Noches_Sevilla
@mariiagallardo7 jajajaja pues me parece a mi que poco negocio vas a hacer tu conmigo... Y del Sevilla no tienes na?
--> HOY VIERNES, FREEWAY SEVILLA , CHICAS GRATIS + COPA Y CHICOS GRATIS EN PUERTA LISTA ALE GAMEZ DE

Clúster 3: Llegar/Presentar

Keynote Remote: para controlar presentaciones remotamente, esta GRATIS en App Store http://t.co/kmR2EzCyKb vía @MovidaApple
Cómo narices se adjunta un archivo a Gmail desde la app de android? :s
sobre analíticas de los datos que tenéis (trabajo en algo parecido). Podemos hablar? No apunté el mail :S @mcanaleta (2/2)
Hola @mcanaleta! Después de la participación en el #awssummit me he quedado con ganas de compartir algunas impresiones contigo (1/2)

Clúster 4: Alerta/Madrid

#mn Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/Ai6eKVq14B
#jh Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/09zb9LDMzg
#am Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/kg6iFQeeUh
#dg Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/eiWZkfvxIK

Clúster 5: UC3M

RT @Barralta: No es por nada pero hoy hemos demostrado que la uc3m también se mueve (y muy bien). Enhorabuena compañer@s! @CarlosMarxUC3M @…
RT @Barralta: No es por nada pero hoy hemos demostrado que la uc3m también se mueve (y muy bien). Enhorabuena compañer@s! @CarlosMarxUC3M @…

RT @RafaEscudero1: Muy orgulloso de estudiantes y trabajador@s de la UC3M encerrados esta noche en el rectorado http://t.co/iN0vdHdNas vía …

Clúster 6: Barcelona

Las mayores variantes del Barcelona están en las bandas. Neymar, Aléxis Sánchez, Pedro, Cesc o, último recurso, Tello http://t.co/6NrjP3EqTV

Clúster 7: Madrid/Chicos

313 : Lo mejor de @metro_madrid los dos chicos que han cantado en el vagón. Han despertado la sonrisa de todos y se han llevado su buen tributo
371 : Madrid se nos presentaba gris, hasta que @BOAMISTURA lo llenó de color. Gracias chicos! #viveencolor http://t.co/HbKmQ6iQj1

Clúster 8: Madrid/Ver

@CarmeloHP Si vas a visitar BroadcastIT o BitamShow nos veremos por Madrid RT.
@ERNESTOVICARIO: RT a los que no vamos a ver mañana el Barsa-Madrid @Mullingations hablando de madrid ems isa te quiero ver ya u know.
RT @laurapgradus: De camino a Marbella ! Qué ganas dejar unos días este tiempo asqueroso de lluvia en Madrid e ir a ver el solete wiii ☀□ ☺□.
Menuda semanita mostró espera... pero mientras tanto a descansar y A VER EL CLASICOOO ÁNIMO MADRID!!!

Clúster 9: iPhone/5S

@Eugeniabusso ya me voy a comprar el iphone 5s y voy a tener TODO
No se pero ... le quitare el Iphone a wendy :$ o el blackberry
Así son los iPhone y sus predecesores http://t.co/CG52jzhYQ8
RT @ClaroRD: El iPhone 5s estará llegando a @ClaroRD el 8 de noviembre. Reserva el tuyo hoy. http://t.co/a4sjGBu0xj
RT @untalweroh: Dicen que si le instalas BBM al iPhone le da cáncer a Siri.
Más ordinario que Iphone con PIN
RT @EdyStreets: El iPhone negro se roba el wi-fi mejor que cualquier otro celular.
iOS 7.0.3 soluciona el problema del acelerómetro en el iPhone 5s http://t.co/h0woMH6aPd
RT @Technomexico: Precios del iPhone 5S y 5C en México http://t.co/L24qca7i1y
El nuevo #iPhone 5s analizado por Applesfera: http://t.co/bEVe7wsNLd
¡He reunido 31,747 monedas de oro! ¿A ver quien es capaz de ganarme? http://t.co/c8m5TQsFJw #iphone, #iphonegames, #gameinsight

Clúster 10: Barça/Real

"Todos dan favorito al Barça y eso es bueno" http://t.co/lAt1Qu02JX
RT @FutboolFichajes: En @puntopelota dicen que uno de los equipos interesados provoca miedo al Barça. Con lo cual, tiene que ser el Real Ma…
#LigaBBVA "Todos dan favorito al Barça y eso es bueno" http://t.co/Xr2D5ALfR4

Clúster 11: Madrid/Real

RT @jbalsera5: mañana Real Madrid - Barcelona aver si ganamos ::) HALA MADRID
RT @RealMadridxCR7: RT: REAL MADRID FAV: BARCELONA
RT @Real_Madrid_CF: ¡¡Mañana Clásico!!! ¿Quién crees que va a ganar? RT REAL MADRID FAV BARCELONA
RT @quediario: Clásico Barcelona-Real Madrid: El 'Tata' y Ancelotti, primer 'cara a cara' http://t.co/maF2By3emX
RT @MBonafiglio: JOSE LUIS CHILAVERT: "LOS JUGADORES TIENEN QUE SABER QUE ESTAR EN VELEZ, ES COMO ESTAR EN EL BARCELONA O REAL MADRID"

Clúster 12: Haber/Llegar

RT @ESN_UPF: Hemos llegado al #esnmadness !!! Empieza el finde de locura con @ESNUAM y @esn_uc3m :)
RT @ESN_UPF: Hemos llegado al #esnmadness !!! Empieza el finde de locura con @ESNUAM y @esn_uc3m :)
Hemos llegado al #esnmadness !!! Empieza el finde de locura con @ESNUAM y @esn_uc3m :)

Clúster 13: UC3M/Madrid

El próx martes 29 celebramos la Mesa Redonda "Ecologismo en la Universidad" en la UC3M @greenpeace_madr @EeAmadrid https://t.co/8WFyo0DcvV
RT @MareaVerdeUC3M: ...más de 100 personas de la UC3M en la manifestación de Getafe y 300 en la manifestación de Madrid SIN PARAR DE CANTAR…
RT @BeusMorada: La uc3m en la mani de madrid, luchando por la publica!! http://t.co/WffQoCYIYc

Clúster 14: Madrid/Barcelona

RT @Real_Madrid_CF: ¿Quien ganará el clásico? RT: Real Madrid FAV: F.C Barcelona http://t.co/ejGGbEvDMy

RT @aaroncas93: Se viene el único partido que paraliza al mundo entero, Real Madrid vs Barcelona!! Nervios desde hoy!!
RT @FCBarcelona: SPECIAL CONTENTS - The Clásico FC Barcelona v Real Madrid http://t.co/OSZ1kPs6eO
Mañana se le hará un homenaje a Manolo Escobar en el partido F.C.Barcelona - Real Madrid.
Real Madrid OR Barcelona
@luchosm: HOY...LOS CULES SUEÑAN. MAÑANA...BARCELONA SERA BLANCA! PODER VIKINGO! SOMOS EL MADRID!" • #HalaMadrid!! ☺.
Marathon de cine de terror Halloween 2013 en Barcelona y Madrid http://t.co/fCUWjKtwRV vía @wordpressdotcom.
Clásico sin favorito. Mucha incertidumbre por el nivel bajo de ambos a dia de hoy. Mas confianza para el Madrid. Mas fútbol Barcelona.
RT @Gareth_BaleRM: Ancelotti\|"Jugaremos con coraje y personalidad" EL ENTRENADOR DEL MADRID APELA AL "CARÁCTER" PARA GANAR EN BARCELONA htt…

Clúster 15: Barça/Madrid

RT @MessiBrasil2014: Messi se consagró en un Barça-Madrid y Mañana le toca a Neymar.Y Messi marcará minimo un Golito para ser el maximo art…
@Kokismo @ElMejorCule Jajajaja ilusos que sois.. Tengo mejor delantera que Madrid y barça juntos.
RT @TwiHits: RT si gana el Barça. FAV si gana el Madrid. ¿Quien ganara el clásico?
RT @PenyaBB: Tito Vilanova podría asistir al Barça-Madrid de este sábado http://t.co/jfqLwmFpg8
No soy ni del Barça ni del Madrid, pero simplemente ver el Clásico es un espectaculo
@nostrapacuss Ah bueno, creía que te referias a: "pues en el curro llevan 1 semana hablando del Madrid-Barça". Excuse me.

ANÁLISIS

Se comprueba que siguen existiendo 5 clústeres principales que son los que concentran mayor cantidad de tweets y aparecen una gran cantidad de pequeños nuevos grupos muchos de ellos constituidos por *retweets* (reenvío de un mismo un tweet). Algunos de estos pequeños grupos son interesantes, pero mucho otros están constituidos por un número muy pequeño de elementos, lo que demuestra que dicho grupo, lo más lógico es

que perteneciese a un grupo de mayores dimensiones.

5.2.3.2 MIDIENDO LA DISTANCIA INTRA-CLÚSTER

Se va probando con un número diferente de clústeres iniciales de modo que se ve como lógicamente se irá reduciendo la distancia intra-clúster. No obstante llegará un punto en el que dicha reducción será prácticamente nula; será en ese momento cuando se haya obtenido el número óptimo de clústeres que representan dicho conjunto de documentos, en este caso de tweets. El principal problema de esta técnica es que debido a la aleatoriedad con la que KMeans trabaja a la hora de terminar los clúster semilla puede producir variaciones a la hora de ir mostrando las distancias intra-cluster, modificando en ocasiones la tendencia que en principio debería seguirse. Por tanto, para aplicar esta técnica, sería aconsejable combinarla con otra que permita llevar a cabo una selección de los centroides semilla basada en algún objetivo concreto, como por ejemplo conseguir que dichos centroides se encuentren siempre lo más separado posibles.

5.2.4 SIMULACIONES ATENDIENDO A LA ETAPA DE PREPROCESADO DE LOS TWEETS

Se va a analizar el comportamiento que hubiera tenido KMeans en el caso de que no se hubieran eliminado las *stopwords* o si no se hubiera aplicado la lematización. Para ello se va a seguir utilizando la colección de 1000 tweets empleada en los anteriores experimentos.

5.2.4.1 SIN STOPWORDS

Si no se eliminan las palabras de parada o *stopwords* se obtienen los resultados mostrados en las siguientes tablas.

Tabla 10. Datos de las agrupaciones formadas – sin eliminar *stopwords*

	Distancia intra-clúster	Palabras más repetidas	Frecuencia
clúster 0	0.24	De	356
		A	303
clúster 1	0.23	Y	458
		La	238
clúster 2	0.25	En	169
		El	123
clúster 3	0.26	De	153
		Rt	119

Tabla 11. Distancias entre los centroides de los clústeres formados – sin eliminar *stopwords*

	Clúster 0	Clúster 1	Clúster 2	Clúster 3
clúster 0	0.0	0.3	0.4	0.3
clúster 1	0.3	0.0	0.3	0.3
clúster 2	0.4	0.3	0.0	0.2
clúster 3	0.3	0.3	0.2	0.0

ANÁLISIS

A partir de estas tablas se puede ver la importancia de eliminar las palabras vacías o *stopwords*. Estas palabras, si no se eliminasen serían las más repetidas y las que por tanto guiarían al algoritmo de clasificación. Obviamente, lo conducirían a malos resultados, tal y como se está viendo: la distancia intra-clúster (normalizada en todo momento como ya se había comentado pudiendo tomar valores entre 0 y 1) aumenta mucho respecto al mismo caso en el que sí se estuviesen eliminado las *stopwords* (simulaciones de apartado anteriores); la distancia final entre los centroides de los clústeres es muy pequeña, y las palabras más usadas son preposiciones, que no aportan ningún tipo de información discriminatoria.

5.2.4.2 SIN LEMATIZACIÓN

Si no se lleva a cabo lematización sobre las palabras de la colección se obtienen los resultados mostrados en las siguientes tablas.

Tabla 12. Datos de las agrupaciones formadas – sin lematización

	Distancia intra-clúster	Palabras más repetidas	Frecuencia
clúster 0	0.03	UC3M	211
		Mareaverdeuc3m	14
clúster 1	0.01	Sevilla	202
		Gratis	29
clúster 2	0.02	iPhone	245
		5S	31
Clúster 3	0.10	Real	81
		Madrid	292

Tabla 13. Distancias entre centroides finales– sin lematización

	Clúster 0	Clúster 1	Clúster 2	Clúster 3
clúster 0	0.0	0.6	0.8	0.8
clúster 1	0.6	0.0	0.7	0.7
clúster 2	0.8	0.7	0.0	0.8
clúster 3	0.8	0.7	0.8	0.0

ANÁLISIS

Se observa que para esta colección de datos, la lematización verdaderamente no ha ejercido una gran influencia pues los resultados obtenidos prácticamente son idénticos al caso en el que sí se usaba lematización. Una de las posibles razones es porque en esta ocasión las palabras con mayor peso como Madrid, iPhone, Barcelona, Sevilla, clásico, UC3M, etc. no admiten femenino o masculino, plural o singular. Además, muchas de las formas verbales usadas proceden de verbos que no aportan gran información como el verbo "ser" o "estar" con lo que no son tenidas en consideración.

5.2.5 KMEANS: AGRUPACIONES Y DISTANCIA INTRA-CLUSTER

Por último, para acabar el estudio de KMeans sobre la colección completa descrita en el **apartado 5.1** se va a intentar mejorar el algoritmo: se desea paliar el efecto de la aleatoriedad en la elección de los centroides y que a menudo genera no muy buenos resultados debido a que al final se alcanzan mínimos locales y no globales. Dos de los parámetros más importantes en la creación de clústeres a partir de KMeans son la distancia entre los centroides finales (distancia inter-clúster, **figura 30 derecha**), y el grado que indica lo compacto que es cada clúster (distancia intra-clúster **figura 30, izquierda**).

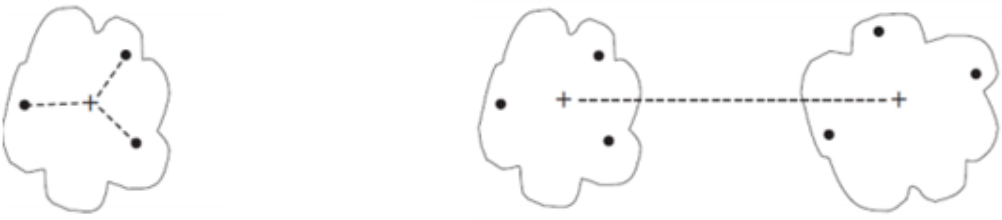


Figura 30. Distancia intra-clúster (izquierda) y distancia inter-clúster (derecha)

Es decir, KMeans tiene como objetivo formar clústeres lo más alejados posibles unos de otros y lo más compactos posibles **(figura 31).** No obstante, tal y como se vio en el **capítulo 2,** el algoritmo KMeans no garantiza que los centroides finales obtenidos sean los que minimizan globalmente la función objetivo (función que pretende minimizar la cohesión y maximizar la separación, **ecuación 10**) puesto que los resultados finales alcanzados dependen de los centroides semillas iniciales.

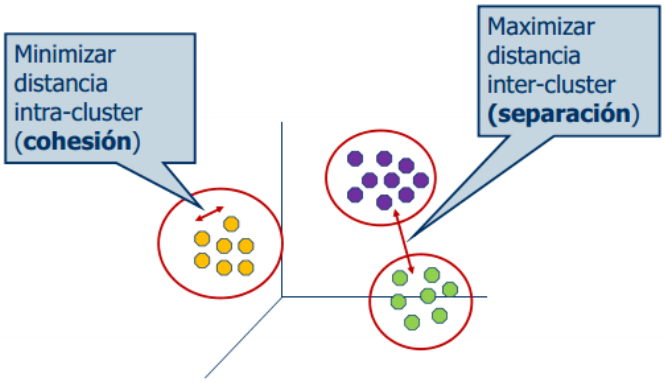


Figura 31. Objetivo final de KMeans: distancias intra-clúster e inter-clúster (Fernando Berzal, Universidad de Granada)

Aprovechando todo ello, se va a imponer al algoritmo una nueva condición. Si al llevar a cabo las agrupaciones, estas no son todo los compactas que debieran, se vuelve a llevar a cabo el algoritmo sobre la misma colección de entrada. De este modo y gracias a un umbral establecido de manera experimental sobre lo compacto que debe ser cada clúster, se llega a mejores resultados como se verá a continuación. Dicho umbral queda establecido en 0.2; esto quiere decir que cada vez que el algoritmo KMeans llegue a su fin y entre los grupos formados halla alguno con una distancia intra-clúster superior a ese valor, se lanza una nueva ejecución que previsiblemente partirá de nuevos centroides iniciales, más apropiados, permitiendo llegar al algoritmo a mejores resultados.

Se conoce de partida que se van a formar 4 clústeres y que, por la colección de entrada con la que se trabaja, estos deberían estar formados por un número muy parecido de tweets, en torno a 220-250; se llevan a cabo una serie de iteraciones y se obtienen los siguientes resultados.

Palabras representantes de la colección de entrada:

sevill, diez, provinci, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, lleg, hem, luch, present, barcelon, chic, gratis, exam, gan, 5, iphon, barç, real, clásic, huelg, 5s, uc3m, mareaverdeuc3m, profesor.

SIMULACIÓN I

Tabla 14. Datos de las agrupaciones formadas – KMeans, colección completa, simulación I

	Número de tweets	Distancia intra clúster	Palabras más usadas	Frecuencia
clúster 0	199	0.021	Sevilla Chicas	202 30
clúster 1	234	0.029	iPhone 5S	246 31
clúster 2	287	0.127	Madrid Real	292 80
clúster 3	189	0.041	UC3M Mareaverdeuc3m	211 14

SIMULACIÓN II

Tabla 15. Datos de las agrupaciones formadas – KMeans, colección completa, simulación II

	Número de tweets	Distancia intra clúster	Palabras más usadas	Frecuencia
clúster 0	220	0.124	Sevilla Chicas	202 30
clúster 1	234	0.029	iPhone 5S	246 31
clúster 2	266	0.099	Madrid Real	292 80
clúster 3	189	0.040	UC3M Mareaverdeuc3m	211 14

SIMULACIÓN III

Tabla 16. Datos de las agrupaciones formadas – KMeans, colección completa, simulación III

	Número de tweets	Distancia intra clúster	Palabras más usadas	Frecuencia
clúster 0	189	0.040	UC3M Mareaverdeuc3m	211 14
clúster 1	234	0.029	iPhone 5S	246 31
clúster 2	199	0.020	Sevilla Chicas	202 30
clúster 3	287	0.127	Madrid real	292 81

De nuevo de la colección de 1000 tweets, 91 han quedado sin estar representados por ninguna de las palabras de la colección de entrada.

Con este nuevo algoritmo, por tanto, se forman mucho mejores agrupaciones a pesar de seguir existiendo una elección aleatoria de los centroides semilla iniciales.

5.2.6 KMEANS: AGRUPACIONES SOBRE SUBCOJUNTOS DE LA COLECCIÓN

Una manera fácil de encontrar subgrupos dentro de cada bloque de la colección de entrada es aplicar KMeans directamente sobre dichos bloques. De este modo, se pueden obtener subtemas dentro de los temas principales. A modo de ejemplo se va a aplicar KMeans sobre el subconjunto de tweets referidos a Sevilla.

5.2.5.1 SUBCONJUNTO DE TWEETS DE SEVILLA

Aplicando KMeans sobre la colección de 250 tweets de Sevilla se obtienen los siguientes resultados:

Tabla 17. Datos de las agrupaciones formadas - KMeans, subconjunto Sevilla

	Número de tweets	Palabras más usadas	Frecuencia
clúster 0	21	Chicas	30
		Gratis	29
clúster 1	38	Alerta	76
		Sevilla	38
clúster 2	165	Ver	168
		Sevilla	14
clúster 3	12	Sevilla	12
		Betis	12

Clúster 0: Chicas/Gratis

--> HOY VIERNES, FREEWAY SEVILLA , CHICAS GRATIS + COPA Y CHICOS GRATIS EN PUERTA LISTA ALE GAMEZ DE 12;00 A 2;00 + INFO 625394147 =)
SIDHARTA LISTA DE LUISMA HERVÁS DE 12 A 2H COPA PARA CHICAS y GRUPO 5 CHICAS LAMBRUSCO GRATIS INF 687.169.173 @Noches_Sevilla
*ESTA NOCHE REVIENTA #FREEWAY SEVILLA! Solo di en puerta LISTA DE ARANTXA de 12-2 **ENTRADAS HALLOWEEN YA A LA VENTA** +INFO 651407827....*

Clúster 2: Alerta/Sevilla

#es Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/kKmcq3LEsb
#ao Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/pmvC5qxFuO
#sl Diez provincias, en alerta por lluvia y viento - Alerta naranja en Sevilla y Córdoba, y amarilla en Madrid... http://t.co/lbpq938lCP

Clúster 2: Ver/Sevilla

@izah_sevilla si tu. tu mi novia tu a la ke amo<3
Que reze sevilla para verla por las calles... #AngustiasCoronada #ReinaGitana #SanRoman
RT @forrestcedrick: A ver a cuántos llegamos: RT si el Sevilla es el equipo que más ASCO te da.
RT @kike284: Que ganas tengo de ver a mi virgen de las angustias en la calle y a mis niños perfumando sevilla con sus sones OLE MIS NIÑOS
RT @DavidCarrasco98: 4 horitas para que juegue nuestro Sevilla Atlético!!! #VamosSAT!!!
Tren con destino Sevilla,se presenta un buen finde ;)

Si la ciudad no va al campo. El campo va a la ciudad #sevilla http://t.co/j9dvTqpthQ
@MiRondo: Celebración anoche del gol de Joaquín en #UEL http://t.co/JjVWZHrVfs" Olé Sevilla!

Clúster 3: Sevilla/Betis

RT @RBb_betis1907: El Betis, el equipo español mas visto en Europa League en #Cuatro y #GolTV, con el Valencia como segundo y el Sevilla co…
RT @UnLocoDeGolSur: El Betis, el equipo español mas visto en Europa League en #Cuatro y #GolTV, con el Valencia como segundo y el Sevilla c…
RT @UnLocoDeGolSur: El Betis, el equipo español mas visto en Europa League en #Cuatro y #GolTV, con el Valencia como segundo y el Sevilla c…

5.3 DBSCAN

5.3.1 INTRODUCCIÓN

Como ya se ha visto, el algoritmo DBSCAN depende de dos parámetros fundamentales:

- Número de nodos vecinos (*MinPoints*)
- Radio de la vecindad (*épsilon*)

En función de ambos, los resultados del algoritmo de clasificación variarán mucho. Como se ha visto también, las distancias euclídeas entre los tweets de la colección están normalizadas, es decir si dos tweets no se parecen en nada tendrán un distancia = 1 y si son idénticos tendrán una distancia = 0.

5.3.2 PARÁMETROS DE DBSCAN SOBRE COLECCIÓN COMPLETA

Por lo tanto, la calidad de las clasificaciones se van a ver afectadas directamente por los dos parámetros anteriores: si se define el parámetro *Épsilon* a un valor muy próximo a 1 lo lógico es que se formen muy pocos grupos, y aparezca solo 1 que englobe a todos. A medida que se vaya decrementando este valor deberían ir formándose nuevos clústeres ya que al ser este radio menor es capaz de alcanzar y englobar bajo el mismo clúster a un menor número de tweets. No obstante, habrá que tener en cuenta en todo momento al número de tweets necesarios para formar un clúster, es decir, *MinPoints.*

En esta ocasión se vuelven a realizar experimentos sobre la colección descrita en el **apartado 5.1**. Tras el análisis de los resultados que se han ido obteniendo tras la modificación continua del valor de estos parámetros se puede llegar a una conclusión aproximada sobre cuál es la combinación óptima de los mismos para el conjunto de entrada. No obstante, esta combinación óptima depende de los resultados que se deseen conseguir. En este apartado se ha entendido como óptima aquella clasificación que como mínimo permita clasificar a los tweets en los 4 grupos bien diferenciados: UC3M, Madrid, Sevilla, iPhone, con la probabilidad mínima de error. Como se comprueba a continuación, un valor de *Épsilon* de 0.25 y un valor de *MinPoints* de 4 han llevado a obtener dicha clasificación.

Durante los siguiente experimentos, de la colección de 1000 tweets, 91 tweets fueron representados en notación vectorial únicamente por 0s; esto es el 9.1% de los tweets no estaban representados por la palabras que fueron elegidas durante la fase del preprocesado. Este grupo es incorporado al grupo denominado “tweets ruido”, es decir, aquellos que no guardan una gran semejanza con el resto y no pueden formar grupo con nadie.

Palabras representantes de la colección de entrada:

sevill, diez, provinci, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, lleg, hem, luch, present, barcelon, chic, gratis, exam, gan, 5, iphon, barç, real, clásic, huelg, s, 5s, uc3m, mareaverdeuc3m, profesor.

SIMULACIÓN I

Radio de la vecindad *(Épsilon): 0.25*

Mínimo número de nodos vecinos *(MinPoints): 4*

Numero de clústeres: 5

Tabla 18. Datos de las agrupaciones formadas – DBSCAN, *Épsilon*=0.25, *MinPoints*=4

	Número de tweets	Palabras más usadas	Frecuencia
clúster 0	266	Madrid	292
		Real	80
		Alerta	64
clúster 1	232	iPhone	244
		5S	31
		5	27
clúster 2	180	UC3M	202
		Mareaverdeuc3m	13
clúster 3	221	Sevilla	204
		Chicas	30
		Gratis	30
clúster 4	9	Madrid	9
		UC3M	9

SIMULACIÓN II

Radio de la vecindad *(Épsilon): 0.10*

Mínimo número de nodos vecinos *(MinPoints): 10*

Numero de clústeres: 6

Tabla 19. Datos de las agrupaciones formadas – DBSCAN, *Épsilon* =0.10, *MinPoints* =10

	Número de tweets	Palabras más usadas	Frecuencia
clúster 0	80	Madrid	101
		Real	80
		Barcelona	42
clúster 1	231	iPhone	244
		5S	31
		5	27
clúster 2	180	UC3M	202
		Mareaverdeuc3m	13
clúster 3	193	Sevilla	196
		Chicas	30
		Gratis	29
clúster 4	32	Alerta	64
		Sevilla	32
		Diez	32
clúster 5	150	Madrid	154
		Barça	21
		Ganar	11

SIMULACIÓN III

Casos extremos:

Radio de la vecindad *(Épsilon): 0.010*

Mínimo número de nodos vecinos *(MinPoints): 2*

Numero de clústeres: 57

Radio de la vecindad *(Épsilon): 1*

Mínimo número de nodos vecinos *(MinPoints): 10*

Numero de clústeres: 1

5.3.3 SIMULACIONES DBSCAN SOBRE SUBCONJUNTOS DE LA COLECCIÓN

El diferente grado de densidad que pueden presentar los conjuntos de entrada es un factor limitante de este algoritmo. De este modo, habrá que ajustar manualmente en muchas ocasiones el valor de dichos parámetro para obtener unos resultados adecuados.

Se va a proceder a comprobar los grupos que se formarían aplicando solo el algoritmo a cada uno de alguno de los bloques de la colección, es decir, aplicándolo independientemente a los 250 tweets de Sevilla o a los 250 tweets de iPhone, por ejemplo.

5.3.3.1 SUBCONJUNTO DE TWEETS DE IPHONE

Manteniendo los mismos umbrales que en ocasiones anteriores, pero al trabajar sobre una nueva colección constituida por tan solo 250 tweets, las palabras más representativas en esta ocasión son:

iphon, 5s, tweetbot, actualiz, versión, app, total, adapt, ios, hay, bbm, csrclassics, ahor, gratis, lleg, españ, nuev, 5c, cre, preci, oficial, méxic, quier, 5, blackberry, andro, apple , mejor, aplic, sal, pantall, premiasorte, 10, minut, rifará, deb, dar, tuit, segu, whatsappchist, ¡particip, gan, compr, descarg, dieg, sopita2, messeng, irte, put, acab, travel, pin, reun, moned ,oro, ver, capaz, iphonegam, gameinsight

Radio de la vecindad *(Épsilon): 0.25*

Mínimo número de nodos vecinos *(MinPoints): 4*

Numero de clústeres: 1

Que solo se halla formado un grupo empleando la misma combinación de parámetros que en experimentos anteriores es razonable. Trabajar solo con los tweets del subgrupo iPhone implica trabajar con un grupo de datos mucho más homogéneos, mucho más similares entre sí.

No obstante, modificando los parámetros se podrían obtener grupos más interesantes como los siguientes.

Radio de la vecindad *(Épsilon): 0.07*

Mínimo número de nodos vecinos *(MinPoints): 4*

Numero de clústeres: 7

Tabla 20. Datos de las agrupaciones formadas - DBSCAN, subconjunto iPhone

	Número de tweets	Palabras más usadas	Frecuencia
clúster 0	46	iPhone	53
		5s	29
		5c	19
clúster 1	78	iPhone	79
		Gratis	5
		5	5
clúster 2	10	Tweetbot	10
		Actualizar	10
		Versión	10
clúster 3	58	iPhone	58
		Bbm	24
		Blackberry	22
clúster 4	11	iPhone	11
		5	11
		Premiasorteo	11
clúster 5	14	iPhone	18
		5	16
		Pantalla	2
clúster 6	14	iPhone	14
		iPhonegame	14
		Gameinsight	14

5.3.3.2 SUBCONJUNTO DE TWEETS DE SEVILLA

Manteniendo los mismos umbrales que en ocasiones anteriores, las palabras más representativas de la colección en esta ocasión son:

sevill, part, betis, equip, call, tard, diez, provincia, alert, lluvi, vient, naranj, córdob, amarill, madr, ver, valenci, lleg, español, vist, europ, leag, goltv, segund, da, viern, freeway, chic, gratis, cop, puert, list, ale, gamez, info, unlocodegolsur

Radio de la vecindad *(Épsilon): 0.25*

Mínimo número de nodos vecinos *(MinPoints): 4*

Numero de clústeres: 1

Se trata de un resultado lógico, pues el grado de densidad de los tweets perteneciente al bloque Sevilla es mucho mayor que el de un conjunto de entrada formado por cientos de tweets de diferentes temas (la distancia entre los tweets de un mismo subconjunto de la colección debe ser siempre mucho menor puesto que se está ante un grupo mucho más homogéneo).

Modificando los parámetros y ajustándolos para el nuevo grado de densidad se obtendrían nuevos grupos. Así por ejemplo,

Radio de la vecindad *(Épsilon): 0.1*

Mínimo número de nodos vecinos *(MinPoints): 4*

Numero de clústeres: 2

Tabla 21. Datos de las agrupaciones formadas - DBSCAN, subconjunto Sevilla

	Número de tweets	Palabras más usadas	Frecuencia
clúster 0	196	Sevilla	199
		Chicas	30
		Gratis	29
clúster 1	38	Alerta	76
		Sevilla	38
		Córdoba	38

Capítulo 6

Conclusiones y trabajos futuros

6.1 CONCLUSIONES

En el presente trabajo se presenta un primer análisis de diversos métodos de clasificación automática de información sobre una amplia colección de tweets. En concreto, los algoritmos diseñados y desarrollados han sido KMeans, uno de los algoritmos tradicionales en el campo de la clasificación automática y DBSCAN, caracterizado por su capacidad para formar grupos basándose en el concepto de densidad y por su gran resistencia frente al "ruido".

El desarrollo de un clasificador automático ha implicado seguir una serie de fases, que se han desarrollado en cascada: preprocesado, reducción de dimensiones, asignación de pesos y clasificación. Todas ellas han sido muy importantes a la hora de conseguir una buena clasificación final, como ha quedado demostrado en los diferentes experimentos que se han ido desarrollando.

Uno de los primeros objetivos fue conseguir representar cada tweet en una notación vectorial que pudiera ser fácilmente utilizada por los algoritmos KMeans y DBSCAN.

Por tanto, y primeramente, se obtuvieron los tweets de la colección. Un tweet no se compone exclusivamente de un mensaje de 140 caracteres. Tal y como se ve en el **Anexo 2**, un tweet está conformado por numerosos campos. Para llevar a cabo la clasificación solo el campo *"text"* fue necesario y fue el que se extrajo. Posteriormente se aplicaron

técnicas propias del Procesado del Lenguaje Natural; con ellas se pretendía conseguir una representación del tweet mucho más compacta, pero evitando siempre perder información; se cambiaron las letras mayúsculas por minúsculas para obtener una representación estándar, se eliminaron las palabras de parada (generalmente preposiciones, adverbios, …) y se llevó a cabo una lematización de las palabras restantes. A partir de las simulaciones realizadas **(simulaciones 5.2.4)** se comprueba como verdaderamente todos estos procesos son estrictamente necesarios; si no se llevan a cabo, preposiciones adjetivos, adverbios, … se convierten en las palabras clave que guían el proceso clasificatorio y que conducen a resultados muy deficientes.

A continuación, con vistas a obtener una representación vectorial de cada tweet y con el objetivo de reducir las dimensiones del futuro espacio vectorial, se procedió a seguir reduciendo el número de palabras que formaban cada tweet. Para ello se tuvo en cuenta la frecuencia de las mismas: se consideró que aquellas que se repetían muy poco en la colección o que por el contrario aparecían muy frecuentemente no aportaban ningún tipo de información discriminatoria que fuese a ayudar en el proceso de clasificación posterior. Por tanto, se establecieron unos umbrales experimentales, tanto inferior como superior, que permitieron llevar a cabo dicha reducción. Con esta reducción se comprueba en las simulaciones llevadas a cabo a lo largo de todo el **capítulo 5** que se obtienen mejores resultados, además de conseguir una importante reducción en el tiempo de procesamiento de los algoritmos posteriores. Se demuestra que trabajar con espacios vectoriales de enormes dimensiones no tiene por qué proporcionar mejores resultados, sino que precisamente puede ocurrir lo contrario. Con la reducción de dimensiones se busca obtener el mejor compromiso tiempo-eficacia en el proceso de clasificación.

Los tweets, por tanto, a partir de este momento estuvieron constituidos por una menor cantidad de palabras, pero la información que contenía cada uno de ellos se mantenía. Se procedió posteriormente a realizar una ponderación de las palabras que integraban cada tweet. Cada uno de estas palabras se convirtió en una dimensión del nuevo espacio vectorial, integrado por todas aquellas que habían quedado tras realizar las transformaciones anteriores. Existen distintos esquemas de ponderación; el usado por proporcionar la mejor combinación entre tiempos y eficacia fue finalmente el esquema de ponderación por frecuencia: *Term Frequency*.

Por último se procedió a programar en PHP los algoritmos KMeans y DBSCAN. Cada uno de ellos se basan en mecanismos diferentes y ofrecen por tanto diferentes resultados.

6.1.1 KMEANS

KMeans tiene que hacer frente a dos problemas principales tal y como se ve en las simulaciones realizadas: han de ser escogidos a priori tanto el número de clústeres o agrupaciones que se van a formar así como los centroides semilla sobre los que empieza a iterar el algoritmo. KMeans suele llevar a cabo diversas iteraciones, en las que se ve claramente la progresiva formación de los grupos hasta alcanzar finalmente convergencia **(simulaciones 5.2.2.1).**

Cuando los centroides iniciales seleccionados son los adecuados se consiguen porcentajes de clasificación con éxito de hasta un 80%.

Además, gracias a los estudios realizados se comprueba que conseguir maximizar la distancia inter-clúster (distancia entre los centroides de las distintas agrupaciones), y minimizar la distancia intra-cluster (distancia de los tweets al centroide de la agrupación) permite obtener los mejores resultados **(simulaciones 5.2.5).** A pesar de todo, KMeans converge en algunas ocasiones a mínimos locales y no globales, poniéndose de manifiesto la clara influencia de la elección de los centroides semillas iniciales (**simulaciones 5.2.2.2**). Una elección aleatoria puede generar malas o muy malas agrupaciones.

La elección del número de agrupaciones que se van a crear es otra de las principales complicaciones que ofrece el algoritmo. La mayoría de los experimentos que se han realizado han supuesto conocido dicho dato, no obstante se han comentado y explicado dos técnicas que pueden ayudar a obtener una aproximación que indique este valor (la primera técnica se deriva del estudio llevado a cabo por Fazli Can*, Miami University* y Esen A. Ozkarahan, *The Pennsylvania State University* y la segunda se deriva del uso de la distancia intra-clúster). Se ha comprobado como a menudo estas técnicas (**simulaciones del apartado 5.2.3**) llevan a cabo una ligera estimación de dicho valor, formándose en ocasiones o bien grupos muy generales, o bien grupos demasiado específicos (se acabaron formando grupos de incluso dos únicos elementos).

6.1.2 DBSCAN

El algoritmo DBSCAN también ha de escoger a priori a su vez dos parámetros: DBSCAN basa su funcionamiento en la densidad, con lo cual es necesario definir tanto el radio de la vecindad (*Épsilon*) de cada tweet así como el número de tweets necesarios en dicha vecindad para formar una agrupación (*MinPoints*). En los experimentos llevados a cabo se

comprueba cómo se debe alcanzar un cierto compromiso entre ambos parámetros para llegar a buenas agrupaciones **(simulaciones 5.3.2).** Una *Épsilon* de alrededor de 0.25 y *MinPoints* en torno a 4 han proporcionado sobre la colección de tweets buenos resultados **(simulaciones 5.3.2).** Es de destacar que habiendo elegido unos parámetros adecuados para el DBSCAN, su grado de efectividad es muy alto ya que apenas comete errores a la hora de clasificar en un grupo u otro y a diferencia del algoritmo KMeans, siempre llega a los mismos resultados, evitando los efectos introducidos por la aleatoriedad que se dan en este último.

Su mayor problema es que a veces puede dividir un grupo en varios o no conseguir separar clústeres diferentes en el caso de que los datos de entrada presenten diferente grado de densidad **(simulaciones 5.3.3**).

6.1.3 RESUMEN

Los resultados con ambos algoritmos han sido analizados exhaustivamente en el **Capítulo 5**. Las conclusiones más importantes de ambos algoritmos se pueden resumir en la siguiente tabla.

Tabla 22. Resumen comparativo entre KMeans y DBSCAN

	K-MEANS	DBSCAN
BASICOS	Es utilizado para la clasificación de datos basada en sus atributos formando 'k' grupos diferentes.	Se basada en la densidad de los datos de entrada a la hora de formar los grupos de clasificación.
	Usa la recolocacion iterativa para dividir a los datos de entrada en 'k' grupos.	Se basa en dos parametros: radio de la vecindad y número de vecinos mínimos que ha de haber dentro de dicho radio para poder crearse un nuevo clúster.
	Para cada iteracion el algoritmo intenta mejorar la homogeneidad de cada grupo.	
A FAVOR	Facil de implementar.	Es resistente frente al "ruido".
	Trabaja bien sobre conjuntos de entrada con formas convexas.	Puede encontrar clústeres de formas arbitrarias pues se basa en seguir la conectividad de cada punto con sus vecinos.
	Trabaja bien con la distancia euclidea.	Muy alto porcentaje de acierto de clasificacion sobre el conjunto de entrada de 1000 tweets (cercano al 80%).
	Muy rapido en la ejecución.	
	Buenos resultados sobre el conjunto de entrada de 1000 tweets si la elección de los clúster semillas es adecuada.	
EN CONTRA	Se debe conocer el número de grupos k a formar a priori.	Se ve afectado negativamente si los datos de entrada tienen diferente grado de densidad (lo que puede originar problemas a la hora de separar clústeres).
	Se debe llevar a cabo una selección adecuada de los centroides semilla, punto de comienzo del algoritmo.	
	Puede converger a mínimos locales y no globales obteniendo resultados no deseados.	
	Muy sensible a los outliers (es necesario eliminarlos para mejorar la calidad de los resultados tal y como se ha hecho en nuestro conjunto de entrada).	

La realización de este proyecto supone por tanto un acercamiento a la clasificación automática de tweets, un acercamiento que pretende hacer ver de manera práctica a través del desarrollo de dos sistemas reales basados en los algoritmos KMeans y DBSCAN las dificultades que entraña conseguir obtener información de un gran volumen de información. El uso de técnicas de procesamiento del Lenguaje Natural es también clave a la hora de desarrollar esta tarea.

6.2 TRABAJOS FUTUROS

En la realización del presente proyecto se ha diseñado y desarrollado un clasificador automático basado tanto en KMeans cono en DBSCAN. No obstante, podrían ser realizadas modificaciones en ambos algoritmos para adaptarlos mejor a la información de entrada con la que se trabaja, los tweets.

Se podrían estudiar mejores y nuevos algoritmos para determinar el número de clústeres en KMeans o estudiar una elección automática más adecuada de los centroides iniciales; serían vitales a la hora de conseguir mejores resultados.

Respecto a DBSCAN, el mayor problema que presenta y contra el que se podrían realizar investigaciones al respecto sería conseguir actualizaciones automáticas de los dos parámetros sobre los que basa (*Épsilon* y *MinPoints*) con el objetivo de ir adaptándose en el caso de encontrar muy diferentes densidades en la colección sobre la que trabaje.

Sobre ambos algoritmos se podrían también buscar formas que permitan mejorar sus tiempos de repuesta para poder trabajar con colecciones de incluso millones de tweets.

Además se podrían analizar el resto de algoritmos comentados a lo largo del presente trabajo para estudiar su rendimiento sobre este conjunto de entrada tan especial como son los tweets.

El empleo de estos algoritmos podría llegar a reemplazar la manera de clasificación de aquellos que como Twitter deciden basar únicamente su clasificación en la frecuencia de las palabras. Esa manera de clasificación puede conducir a situaciones en las que palabras en diferentes contextos y por tanto con diferentes significados acaban apareciendo en el mismo clúster. Sería recomendable por tanto, estudiar como los algoritmos de aprendizaje no supervisado consiguen paliar este problema.

Por otra parte el trabajar con tweets ha permitido conocer más y mejor su estructura

interna, viéndose que numerosas investigaciones se pueden hacer con ellos en lo referente a toda la información que incluyen.

Por ejemplo, gracias a la información que aportan los tweets acerca de parámetros como la localización, se podrían llevar a cabo estudios conjuntos de clasificación y zona geográfica para obtener información más acotada, y servir por tanto mejor para cualquier tipo de estudio, ya sea de marketing, política, salud, etc.

Anexo 1:

Listado de *stopwords*

a	aquello	conseguir	dejar
acá	aquellos	consigo	del
ahí	aqui	consigue	demás
ajena	aquí	consiguen	demas
ajenas	arriba	consigues	demasiada
ajeno	asi	contigo	demasiadas
ajenos	atras	contra	demasiado
al	aun	cual	demasiados
algo	aunque	cuales	dentro
algún	bajo	cualquier	desde
alguna	bastante	cualquiera	donde
algunas	bien	cualquieras	dos
alguno	cabe	cuan	el
algunos	cada	cuán	él
allá	casi	cuando	ella
alli	cierta	cuanta	ellas
allí	ciertas	cuánta	ello
ambos	cierto	cuantas	ellos
ampleamos	ciertos	cuántas	empleais
ante	como	cuanto	emplean
antes	cómo	cuánto	emplear
aquel	con	cuantos	empleas
aquella	conmigo	cuántos	empleo
aquellas	conseguimos	de	en

encima	hasta	mucho	primero
entonces	incluso	muchos	primero desde
entre	intenta	muy	puede
era	intentais	nada	pueden
eramos	intentamos	ni	puedo
eran	intentan	ningun	pues
eras	intentar	ninguna	que
eres	intentas	ningunas	qué
es	intento	ninguno	querer
esa	ir	ningunos	quien
esas	jamás	no	quién
ese	junto	nos	quienes
eso	juntos	nosotras	quienesquiera
esos	la	nosotros	quienquiera
esta	largo	nuestra	quiza
estaba	las	nuestras	quizas
estado	lo	nuestro	sabe
estais	los	nuestros	sabeis
estamos	mas	nunca	sabemos
estan	más	os	saben
estar	me	otra	saber
estas	menos	otras	sabes
este	mi	otro	se
esto	mía	otros	segun
estos	mia	para	ser
estoy	mias	parecer	si
etc	mientras	pero	sí
fin	mio	poca	siempre
fue	mío	pocas	siendo
fueron	mios	poco	sin
fui	mis	pocos	sín
fuimos	misma	podeis	sino
gueno	mismas	podemos	so
ha	mismo	poder	sobre
hace	mismos	podria	sois
haceis	modo	podriais	solamente
hacemos	mucha	podriamos	solo
hacen	muchas	podrian	somos
hacer	muchísima	podrias	soy
haces	muchísimas	por	sr
hacia	muchísimo	por qué	sra
hago	muchísimos	porque	sres

sta	tiempo	un	verdadera
su	tiene	una	vosotras
sus	tienen	unas	vosotros
suya	toda	uno	voy
suyas	todas	unos	vuestra
suyo	todo	usa	vuestras
suyos	todos	usais	vuestro
tal	tomar	usamos	vuestros
tales	trabaja	usan	y
también	trabajais	usar	ya
tambien	trabajamos	usas	yo
tampoco	trabajan	uso	k
tan	trabajar	usted	tb
tanta	trabajas	ustedes	rt
tantas	trabajo	va	hoy
tanto	tras	vais	dl
tantos	tú	valor	tu
te	tu	vamos	dsd
teneis	tus	van	m
tenemos	tuya	varias	d
tener	tuyo	varios	http
tengo	tuyos	vaya	
ti	ultimo	verdad	

Anexo 2

Formato de los tweets

Tweet de ejemplo: "@cristobalsoria yo en la vida privada de la gente no me meto, viva el sevilla y a seguir dándole caña a los vikingos".

Algunos campos:
["id"]=> 3.9374806640441E+17

["user"]=>"ismaelubo"

["name"]=>"ismael"

["description"]=>""

["image"]=>http://pbs.twimg.com/profile_images/1747694000/image_normal.jpg

["retweet_count"]=>””

["text"]=>"@cristobalsoria yo en la vida privada de la gente no me meto, viva el sevilla y a seguir dándole caña a los vikingos"

["lang"]=>"es"

["geo"]=>""

["coordinates"]=>""

["place"]=>""

["date"]=>"2013-10-25 16:37:09"

["raw"]=>{

["metadata"]=>{

["result_type"]=>"recent"["iso_language_code"]=>"es"

}

["created_at"]=>"Fri Oct 25 14:37:09 +0000 2013"

["id_str"]=>"393748066404409344"

….

["profile_link_color”]=>"0084B4"

["profile_sidebar_border_color"]=>"C0DEED"

["profile_sidebar_fill_color"]=>"DDEEF6"

["profile_text_color"]=>"333333"

["profile_use_background_image"]=>bool(true)

["default_profile"]=>bool(true)

….

Referencias

[1] Aiello, Luca Maria; Georgios, Petkos; Martin, Carlos; Corney, David; Papadopoulos, Symeon; Skraba, Ryan; Goker, Ayse; Kompatsiaris, Yiannis; Jaimes, Alejandro.“Sensing trending topics in Twitter”. IEEE Transactions on Multimedia 15(6): 1268-1282 (2013).

[2] Bishop, Christopher M. Machine Learning: “Pattern Recognition and Machine Learning”. (Springer Science+Business Media, LLC, 2006).

[3] Can, F.; Ozkarahan, E. A. (1990). "Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases". ACM Transactions on Database Systems. Vol. 15, No. 4 (December, 1990), pp. 483-517, Sección 2.7.

[4] Dongen, S. van. A new cluster algorithm for graphs. Amsterdam: CWI (Centre for Mathematics and Computer Science), 1998.

[5] Fernández, César. Aprendizaje automático y Data Mining. Universidad Miguel Hernández, 2012.

[6] Figuerola, Carlos G.; Alonso Berrocal, José L. y Zazo Rodríguez, Ángel F. “Algunas

Técnicas de Clasificación Automática de Documentos," Cuadernos de documentación multimedia, ISSN 1575-9733, Nº. 15, 2004 , pp. 1-2.

[7] Hartigan, John. Clustering Algorithms (Wiley, 1975), capítulo 3 "Quick Partition Algorithms", punto 3.2 "Leader Algorithm".

[8] Friman, Ola. Neural networks and learning systems, Linkoping University ,2013.

[9] Manning, Christopher D.; Raghavan, Prabhakar and Schütze ,Hinrich. Introduction to Information Retrieval (Cambridge University Press. 2008.)

[10] Marsland, Stephen. Machine Learning: An Algorithmic Perspective (USA: Chapman & Hall/CRC, 2009).

[11] Martin, Ester; Hans-Peter, Kriegel; Jörg Sander and Xiaowei Xu. A density-based algoritm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996.

[12] Panessi, Walter y Bordignon, Fernando Raúl. Procesamiento de Variantes Morfológicas en Búsquedas de Textos en Castellano. Universidad Nacional de Luján, Departamento de Ciencias Básicas, División Estadística y Sistemas.

[13] Pérez Abelleira, M. Alicia y Cardoso, Carolina A. Minería de texto para la categorización automática de documentos. Cuadernos de la Facultad n. 5, 2010. Universidad Católica de Salta.

[14] Urbano, Julián; Morato, Jorge; Marrero, Mónica; Sánchez-Cuadrado Sonia.

Recuperación y acceso a la información, capítulo 1. Universidad Carlos III de Madrid.

[15] Vicente Villardón, José Luis. Introducción al análisis de clúster. Universidad de Salamanca, departamento de Estadística.

[16] Villena Román, Julio. Inteligencia en redes de comunicaciones. Universidad Carlos III de Madrid, 2013.