Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática
con Mención en Ciencias de la Computación

Trabajo Fin de Grado

Análisis de clusterización en R

Autor: Aarón Casado Monge

Tutor/es: Juan José Cuadrado Gallego

2020

0.1 Índice

Análisis de clustering en R

Aarón Casado Monge[1], Juan José Cuadrado Gallego[1]

[1]University of Alcala, Polytechnic School, Computer Science Department, Scientific and Technological Campus, Politechnic Building. Office: O243, 28805, Alcala de Henares, Madrid, Spain

25 de noviembre de 2020

Resumen

Resumen en español

Abstract

English abstract

Keywords: Data Mining, Text Mining, Clustering, R, COVID-19.

1 Introducción

Desde finales del sigo XX se ha considerado que vivimos en la "era de la información", una etapa caracterizada por el incremento, desarrollo y propagación de emergentes tecnologías de la información y comunicación que han permitido al ser humano romper las barreras de la distancia, el tiempo y el lugar a la hora de compartir información y comunicarse; actividades que han sido decisivas en nuestra historia [1]. Sin embargo, la era en la que realmente vivimos es la "era de los datos", donde cada día se generan más de veinticinco mil petabytes[1] de datos provenientes de comercios, ciencias, Internet y casi cualquier actividad del día a día [2] que acaban volcados en redes de ordenadores, sitios web, bases de datos y otros medios de almacenamiento.

Esta explosión de datos, a la que se ha denominado Big Data, se debe al alto grado de computarización de la sociedad y el avance de herramientas de recolección y almacenamiento de datos. Negocios en todo el mundo generan grandes cantidades de datos derivados de transacciones, stock de productos, platillas de empleados, etc. Las ramas de la ciencia producen datos de manera constante fruto de experimentos, observaciones, recogida de muestras, etc. Y más recientemente, Internet y las redes sociales han sido

[1]Un Petabyte es una unidad de información o almacienamiento de datos equivalente a un cuadrillon de bytes, mil terabytes o un millón de gigabytes. En este caso, es el equivalente a 2.5 trillones de bytes.

las principales responsables del aumento excesivo de datos, siendo usadas por millones de personas simultáneamente.

Y, aunque esto ha supuesto una considerable mejora para la humanidad, pues la información nunca había sido tan accesible y abundante, también ha ocasionado consecuencias negativas y problemas como el almacenamiento y organización de los datos, datos no estructurados que entorpecen su acceso y procesamiento, dificultades a la hora de analizar los datos apropiadamente pudiendo generar desinformación y complicaciones para mostrar los resultados de forma apropiada y aplicarlos de manera eficiente y útil en el mundo real [3].

Como resultado, ha surgido una nueva ciencia que se ha posicionado rápidamente como una de las disciplinas más influyentes de la actualidad: Data Science (Ciencia de los Datos), que debido a su reciente aparición, carece de una definición consensuada, pero podríamos concretarla como "*Ciencia que usa Estadística, Inteligencia Artificial, Programación y Bases de Datos para posibilitar la extracción de conocimiento a partir de datos*" [4]. A su vez, dentro de esta ciencia se han desarrollado otras tres ramas: Data Warehousing, Data Mining y Visualization; cada una de ellas enfocada a resolver o afrontar uno o varios de los problemas mencionados previamente: organización y agrupación de datos, análisis de los mismos y presentación de los resultados, respectivamente.

De entre estas nuevas disciplinas, Data Mining es la que se centra en el procesamiento de los datos, procedimiento por el cual se obtiene la información, y que se podría definir como "*Proceso de descubrimiento de patrones interesantes y conocimiento a partir de grandes volúmenes de datos*" [16], donde dentro de la misma podemos encontrar diferentes técnicas para encontrar patrones y relaciones y dependiendo de cuál se aplique se puede obtener un resultado totalmente diferente incluso con el mismo conjunto de datos, por lo que es fundamental emplear la técnica apropiada en función del objetivo a conseguir, los datos con los que se pretende trabajar y el ámbito de aplicación.

Una de las numerosas formas con las que podemos obtener información de los datos es mediante la clasificación. Esta es una de las actividades más primitivas del ser humano y que ha sido fundamental en nuestro desarrollo, dado que en pos de entender un nuevo fenómeno o asimilar un nuevo objetos solemos buscar las características principales que lo defininen y compararlo con otros objetos o fenómenos conocidos en función de la similitud o disparidad que exista entre ellos. Este método comunmente aplicado en algunos de los campos más importantes del mundo moderno como la analítica de negocios, el reconocimiento de imágenes, las búsquedas web, seguridad, biología y ciencias de la salud. Sin embargo, existen dificultades a la hora de agrupar aquellos conjuntos de datos que no dispongan de una etiqueta o valor conocido que sirva como criterio de clasificación, ya sea porque dicho valor no existe o porque todavía no ha sido definido, y que son muy comunes en las disciplinas mencionadas previamente.

Para poder clasificar este tipo de datos y afrontar este problema, se utiliza la técnica de *clustering*, un método que precisamente permite generar valores y etiquetas para un determinado conjunto de datos realizando agrupaciones denominadas *clusters* o grupos, donde los datos de un mismo cluster sean muy similares entre ellos y a su vez tengan diferencias claras con datos de otros clusters, permitiendo que cada grupo resultante pueda ser etiquetado y tratado como una clase propia, generando una clasificación de los datos que puede ser analizada y de la que se puede obtener conocimiento útil.

De esta manera, el objetivo de este Trabajo de Fin de Grado (TFG) es realizar un estudio sobre el método de clustering, exponiendo de manera teórica qué es este método y qué tipo de utilidades tiene, así como el desarrollo de las diferentes técnicas que existen y las aplicaciones que estas ofrecen. Posteriormente se explorará este método dentro de la Bioinformática, un campo centrado en desarrollar técnicas y programas software para analizar datos biológicos, viendo qué aporta a dicha disciplina y cuáles de las técnicas expuestas previamente se emplean y por qué.

Una vez realizado el marco teórico previo, se pretende hacer una parte práctica en la que se analizarán los diferentes paquetes que ofrecen técnicas de clustering tanto de carácter general como dentro de Bioinformática para el lenguaje de programación R [5], uno de los más relevantes dentro de Data Science.

2 Clustering

Clustering o cluster analysis, llamado en español agrupamiento o análisis de grupos y adaptado al término clusterización, es un método de Data Mining que se basa en el Aprendizaje Automático (Machine Learning), una rama de la Inteligencia Artificial (Artificial Intelligence) que pretende desarrollar sistemas que sean capaces de resolver problemas basándose en los resultados de experiencias previas, aprendiendo de sus errores. Concretamente, se fundamenta en uno de los métodos de aprendizaje explorados en esta disciplina: Aprendizaje No Supervisado, enfocado en el desarrollo y descubrimiento de nuevos conocimientos. Este, a diferencia de otros métodos de aprendizaje, no dispone de conocimiento o experiencias previas sobre las que poder aprender, por lo que su objetivo principal es discernir patrones y relaciones entre los datos para poder agruparlos y trabajar sobre los resultados de la clasificación resultante [6].

En esencia, el proceso de clustering es el mismo, hasta el punto en que básicamente podríamos considerarlos sinónimos, puesto que la clusterización se aplica principalmente sobre conjuntos de datos que se desean organizar en diferentes grupos pero los valores que delimitan cada cluster son desconocidos y por lo tanto se definen en el propio proceso de clasificación.

De una manera más técnica, podríamos decir que clustering busca definir

para una determinada característica[2] o Suceso Elemental[3] (SE), un conjunto de grupos de observaciones (suceso) [4] con valores cercanos, donde los clusters permiten, dados los diferentes sucesos elementales que configuran un suceso, asignar cada SE al mismo cluster [7].

Para poder decidir si dos datos deben ser agrupados en el mismo cluster o por el contrario, separados, es necesario introducir los conceptos de similitud y disparidad. Cuanto más similares sean los datos, más tendrán en común y por lo tanto es más probable que terminen bajo el mismo cluster; y por otra parte, si los datos son dispares entre sí, estos terminarán separados en clusters diferentes. Este criterio se basa en la proximidad de dichos objetos. Si midiéramos la similitud entre dos objetos i y j, esta devolvería 0 si ambos fueran totalmente distintos, y cuanto más elevado fuera el valor, más semejantes serían, siendo 1 el valor más alto, indicando que ambos datos son idénticos. De la misma manera, medir la disparidad entre los objetos daría resultados opuestos, con un 0 indicando que son idénticos y con un 1 que no tienen nada en común.

Esta forma de calcular la proximidad no solo ayuda a la hora de agrupar los datos en clusters y formar subconjuntos a partir de los resultados, sino que con este proceso también somos capaces de detectar *outliers*, datos anómalos que pueden ser datos erróneos procedentes de equivocaciones en mediciones o fallos que deben ser eliminados; o, datos correctos sumamente importantes que son diferentes al resto y deben ser analizados detenidamente. Podemos detectar este tipo de datos a la hora de agrupar los datos, pues puede que queden varios objetos sueltos sin pertenecer a ningún cluster, siendo esos los datos anómalos.

Además, clustering también sirve como primera aproximación a la hora de procesar los datos, porque aunque de por sí, este proceso puede aportar información útil agrupando datos similares y clasificándolos, permitiendo un análisis de los resultados que puede dar lugar al descubrimiento de nuevos conocimientos, también se usa como método de preprocesamiento de datos para otros algoritmos de Data Mining que trabajarán sobre los clusters generados y los atributos seleccionados como criterio a la hora de crearlos, considerando cada grupo resultante como una nueva clase de objetos.

Es gracias a estas características que la clusterización es empleada en muchos ámbitos del mundo actual, siendo la técnica de Aprendizaje No Supervisado más extendida. Campos como ingeniería (aprendizaje automático, inteligencia artificial, reconocimiento de imágenes), ciencias de la salud (genética, biología, microbiología, paleontología, psiquiatría, patología), ciencias sociales (sociología, psicología, educación) o economía (marketing, negocios) [8] hacen uso habitual de esta técnica para clasificar gran parte de

[2]Dicho de una cualidad que da carácter o sirve para distinguir a algo o alguien de sus semejante.

[3]Cada uno de los resultados más simples que se pueen obtener de la realización de un experimento. Obtener el número 3 al tirar un dado.

[4]Se denomina así al subconjunto total de resultados posibles al realizar un experimento. Obtener un 3 o sacar par al tirar un dado.

los datos con los que trabajan y de los que se desconoce un atributo concreto por el que agruparlos. Dentro de estos sectores algunos de los ejemplos más comunes de clustering consisten en clasificar tipos de consumidores que comparten preferencias, aunar bajo un mismo subconjunto muchas formas diferentes de escribir el mismo caracter para facilitar el reconocimiento de textos escritos a mano, agrupar resultados similares de una consulta en internet y mostrar los más relevantes dentro de cada grupo o el estudio de la taxonomía[5] de las especies.

También se puede hacer uso de su capacidad para detectar datos anómalos con la finalidad de revelar posibles fraudes o transacciones financieras sospechosas, incluso ayudar en la disminución de crímenes analizando los resultados obtenidos tras clusterizar datos de detenciones y delitos; incluso aplicada a datos geofísicos[6] para interpretarlos y obtener resultados significativos. Asimismo, se utiliza a la hora de comparar comunidades en redes sociales permitiendo recomendar a los usuarios contenido de su agrado, o dentro del mundo sanitario, ayudando en la identificación y control de diversos tipos de enfermedades, e incluso puede usarse como apoyo en la gestión de edificios públicos como bibliotecas, agrupando a los lectores por sus preferencias de manera similar a los consumidores de un negocio [9, 10, 11, 12, 13].

Comprimiendo toda esta información, podemos concretar los tres principales objetivos por los que clustering es empleado: clasificación natural de individuos, objetos, etc., compresión de información para mayor facilidad a la hora de mostrar resultados y generación de una estructura base para el posterior tratamiento de los datos [14].

La utilidad de este método y la flexibilidad que ofrece con las diversas técnicas y formas de aplicarlo de las que dispone es también parte fundamental de que sea tan comúnmente utilizado y con objetivos tan dispares en gran parte de las áreas del conocimiento. Sin embargo, también es un método frágil, pues depende en gran medida de los datos con los que se trabaje y el criterio escogido para determinar si dos objetos deben agruparse bajo el mismo cluster o separarlos, siendo necesario seguir una serie de pasos a la hora de realizar un análisis de grupos de manera correcta [8, 15]:

1. Primero, hay que seleccionar cuidadosamente los datos con los que se va a trabajar, puesto que tanto el tipo de dato como la cantidad de los mismos y el número de atributos[7] influyen directamente en los resultados. Si se utilizan demasiados datos el procesamiento puede tener un coste computacional alto y es difícil ofrecer una visualización elegante de los resultados. Pero si se escogen pocos datos se pierde información útil y puede dar lugar a equivocaciones. Asimismo, datos

[5] Ciencia que trata de los principios, métodos y fines de la clasificación. Se aplica en particular, dentro de la biología, para la ordenación jerarquizada y sistemática, con sus nombres, de los grupos de animales y de vegetales.

[6] La Geofísica es la ciencia que estudia la Tierra desde el punto de vista de la física.

[7] Dicho de una característica cualitativa de un individuo o dato usada para distinguirlo de una variable o característica cuantitativa.

con gran cantidad de atributos representan una dificultad añadida, elevando el coste computacional aún más y complicanto la toma de decisiones. Es por ello que este paso suele realizarlo un experto en el sector o con su ayuda.

2. Segundo, hay que optar por uno de los diversos algoritmos de clusterización que existen, pues los resultados pueden cambiar considerablemente al variar la estrategia con la que se realiza la agrupación. Este paso suele venir acompañado por la elección del procedimiento por el que se va a calcular la similitud entre los datos y clusters, y del criterio en el que se basa la decisión de agrupar en el mismo cluster dos datos. Esta última elección sinifica escoger el umbral de similitud a partir del cual dos datos pasan a formar parte del mismo cluster y qué característica o conjunto de ellas van a ser utilizadas para realizar la clasificación. La elección del umbral suele ser complicada, y es necesaria mucha experiencia o varias iteraciones de ensayo y error para encontrar un valor correcto.

3. Tercero, una vez generada la clasificación fruto del proceso de clustering es necesario validar el resultado obtenido, pues cualquier técnica de clusterización generará una agrupación de un conjunto de datos independientemente de si la solución aporta o no información útil. Es por ello necesario evaluar que la clusterización cumple con ciertos criterios y estandares para afirmar que el resultado es óptimo.

4. Por último, una vez obtenidos los resultados hay que interpretarlos, pues el objetivo final de este proceso es aportar información útil de los datos analizados para resolver el problema inicial que ha llevado a este proceso. Este proceso suele ser realizado por expertos.

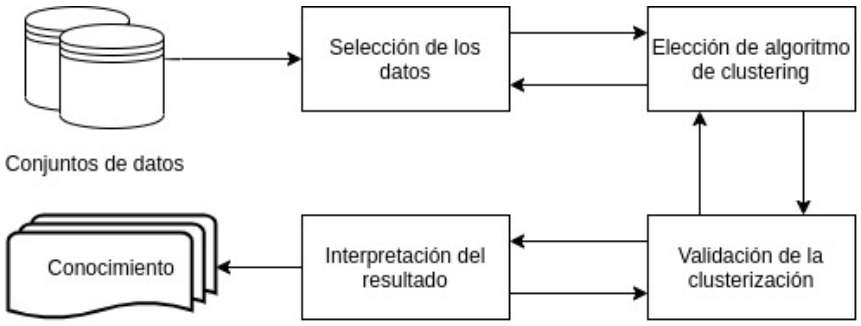


Figura 1: Procedimiento para el análisis de grupos. Este consta de cuatro pasos fundamenteales con realimentación [8].

Como se puede aprecian el la figura 1, hay una vía de realimentación, pues la clusterización de los datos precisa de iteraciones de prueba y error, puesto que aunque la validación de la clusterización pueda aportar información acerca de la calidad del resultado, a veces es complicado encontrar el ajuste óptimo.

Asimismo, tanto el paso inicial como el último requieren de la ayuda de personas cualificadas si pretenden realizarse correctamente, y es en los pasos intermedios donde la intervención de computadoras y programas informáticos son más útiles y están más desarrollados, pues permiten aligerar considerablemente la carga de trabajo y como consecuencia, se puede analizar una mayor cantidad de datos. Pero no deja de ser una tarea complicada que ha ocasionado una gran demanda de expertos en Data Mining y Data Science en diversas áreas del conocimiento durante los últimos años.

La utilidad que aporta clustering permitiendo crear grupos con datos similares entre sí y dispares con respecto a otros grupos con el fin de clasificarlos y las ventajas que esto presenta queda reflejada en la cantidad de usos que recibe, por lo que vamos a profundizar en las diferentes técnicas que son empleadas para la generación de clusters que lo hacen tan importante.

3 Técnicas de clustering

Hay una gran variedad de técnicas de clustering que cumplen con el propósito de clasificar un conjunto de datos, pero no todas son iguales. Existen una serie de requisitos típicos que debe cumplir un algoritmo de clusterización para satisfacer las expectativas y realizar un buen análisis. Sin embargo, cumplir todos estos requerimientos de forma simultánea es una tarea casi imposible, haciendo que el desarrollo de nuevas técnicas y algoritmos o la modificación de los ya existentes sea constante. Analizar cuáles ofrece cada algoritmo nos ayuda a identificar sus puntos fuertes y débiles y por ende, facilitar la elección del algoritmo apropiado para cada ocasión. Veamos cuáles son dichos requisitos y qué criterios se usan para comparar algoritmos.

3.1 Requisitos de un algoritmo de clustering

[16] El proceso de clustering es ubícuo siendo utilizado en gran cantidad de entornos, motivo por el que se han desarrollado numerosos algoritmos y técnicas orientadas a resolver diversos problemas en campos y disciplinas específicos. Sin embargo, no existe un algoritmo que pueda aplicarse uiversalmente a cualquier conjunto de datos y esperar que el resultado sea inmejorable.

Existen trabas que complican este procedimiento tales como la cantidad de datos a analizar o el tipo de datos. Algunos de los algoritmos de clustering iniciales no tenían en cuenta este tipo de problemas, porque no existían en su momento, pero ahora es necesario afrontarlos para que el análisis de los datos sea eficiente y produzca resultados completos y veraces en cualquier circunstancia.

A continuación, se exponen los requisitos típicos que se exigen dentro de Data Mining a un algoritmo de clustering, así como debilidades que presentan algunos de los existentes, pues es imprescindible investigar el problema a tratar y saber escoger el algoritmo más adecuado para resolverlo.

- **Escalabilidad**: Muchos algoritmos de clusterización trabajan bien con conjuntos de datos pequeños que contienen solo varios cientos de objetos; sin embargo, la gran mayoría de bases de datos contienen millones y miles de millones de datos. Analizar solo un pequeño porcentaje de esos datos puede resultar en conocimiento parcial. Que los algoritmos sean capaces de procesar pequeños y grándes volúmenes de datos es necesario para obtener resultados óptimos.
- **Habilidad para lidiar con diferentes tipos de atributos**: Coloquialmente se suele entiende por atributo un dato cuantitativo[8], es decir, un número. Sin embargo, existen otros tipos de datos: nominales[9], binarios[10], ordinales[11], imágenes, gráficos, documentos y mezclas de varios de estos tipos. Por lo que un algoritmo de clustering debe estar preparado para trabajar con todos o la gran mayoría de ellos.
- **Descubrimiento de clusters con formas arbitrarias**: Gran parte de los algoritmos utilizan formas de calcular la similitud entre los datos basadas en la distancia entre los mismos, lo que suele dar lugar a clusters con formas redondas o esféricas. En determinadas ocasiones esto puede resultar contraproducente y erróneo, por lo que también es preciso que los algoritmos sean capaces de identificar diferentes formas de clusters, no solamente formas circulares o geométricas.
- **Requerir información al usuario**: En algunos algoritmos de clustering es común pedir cierto tipo de información al usuario a la hora de trabajar con los datos, como el número de clusters que se quieren generar. Este tipo de práctica puede ocasionar problemas dada la complejidad de esta decisión, sobre todo en escenarios en los que los datos cuentan con múltiples atributos. Por lo que es recomendable que los algoritmos eviten este comportamiento tanto para aliviar a los usuarios como para no comprometer la calidad a la hora de formar clusters, pues el algoritmo se adaptaría al valor introducido por el usuario, aunque fuese erróneo, generando también, un resultado errado.
- **Capacidad para trabajar con ruido en los datos**: Aunque es común tratar de antemano los datos para eliminar cualquier tipo de dato no deseado, recordemos que esta técnica también se aplica como método de preprocesamiento para otros algoritmos de Data Mining, por lo que es necesario que no sean sensibles hacia estas impurezas, puesto que podría resultar en una clasificación pobre. Asimismo, es posible que existan datos anómalos dentro de la muestra que pueden ser de gran utilidad y detectarlos es súmamente importante.
- **Clusterización incremental e insensibilidad al orden de entra-**

[8] Valor obtenido para una característica en una observación.

[9] Dicho de un dato cualitativo que proporciona suficiente información para nombrar a una característica y poder diferenciarla de otra.

[10] Datos que solo pueden tomar los valores cero y uno (0, 1).

[11] Datos cualitativos que proporcionan suficiente información para ordenar las observaciones.

da: Cuando se aplica clustering en entornos en movimiento constante como el de los negocios, es común que se introduzcan datos nuevos junto a los ya existentes. Ciertos algoritmos no son compatibles con este incremento de información, y es necesario volver a iniciar un nuevo proceso de clusterización. Mientras que otros que sí toleran este cambio resulta que son sensibles al orden con el que se han introducido los datos y pueden generar agrupaciones erróneas. Estos problemas han dado lugar a la necesidad de algoritmos que puedan lidiar con la incorporación de nuevos datos y que no se vean afectados por el orden en el que estos son introducidos.

- **Capacidad de Clusterizar datos con muchos atributos**: Aunque de manera común se suele trabajar con datos que no contienen mucho más que una decena de atributos, hay ciertos tipos de datos como documentos, en los que cada palabra clave puede ser considerada un atributo, dando lugar a cientos de atributos por cada dato, haciendo complicado trabajar con ellos y también clasificarlos, haciendo necesarios algoritmos que puedan lograr un buen resultado con este tipo de datos.
- **Clustering basado en restricciones**: En el mundo real podemos encontrar ciertas barreras a la hora de aplicar un algoritmo de clustering, en los que, por ejemplo, sí se tenga que tener en cuenta un límite a la cantidad de clusters que se pueden generar o de decidir si dos datos o clusters deben unirse. Aunque es complicado realizar este tipo de agrupaciones, es indispensable que los algoritmos obtengan resultados decentes dadas estas limitaciones.
- **Interpretabilidad y usabilidad**: De nada sirve clasificar los datos y cumplir con los requisitos previos si no se puede interpretar el resultado. Los algoritmos, por lo tanto, deben ofrecer conclusiones comprensibles, coherentes y que puedan usarse, sometiéndose a ciertas limitaciones semánticas y visuales con la finalidad de acomodar el resultado al objetivo inicial por el que se ha llevado a cabo la clusterización que el usuario pueda enternder.

3.2 Criterios de comparación

[16] Cada algoritmo opera de forma distinta en diferentes aspectos como el criterio de división, la separación de clusters, el cálculo de la similitud y el espacio de clusterización. Observando los pasos a seguir y la toma de decisiones de un algoritmo, podemos compararlo con otro y escoger cuál es más adecuado para el problema a afrontar.

- **Criterio de división**: Principalmente existen dos tipos de métodos respecto a este criterio, aquellos que fraccionan todos los datos para que no exista ningún tipo de orden o diferencia de nivel entre ellos, y los que, por el lado opuesto, dividen los datos de manera jerárquica permitiendo formar clusters en diferentes niveles semánticos. Cada uno aporta una utilidad distinta, con el primer tipo de criterio po-

demos abordar problemas como la clasificación de clientes o lectores, conjuntos de datos en los todos reciben el mismo trato; mientras que con criterio de partición jerárquico podemos organizar documentos en varios campos como “deportes”, “política” o “comida”, y dentro de cada campo tener subconjuntos como “postres”, “pasta” o “carne”.

- **Separación de clusters**: Otro criterio en el que hay dos enfoques: si un dato forma parte de un cluster, este no puede formar parte de otro a no ser que ambos clusters se unan; o permitir que un dato pueda estar en dos clusters a la vez, un concepto denominado “Fuzzy clustering”. La primera aproximación ofrece un enfoque más determinista que es quizás el más aplicado entre ambos métodos, pero existen disciplinas en las que permitir que un dato forme parte de más de un cluster, como se aprecia en la Figura 2, puede ser realmente útil, por lo que cada vez recibe más usos [17].

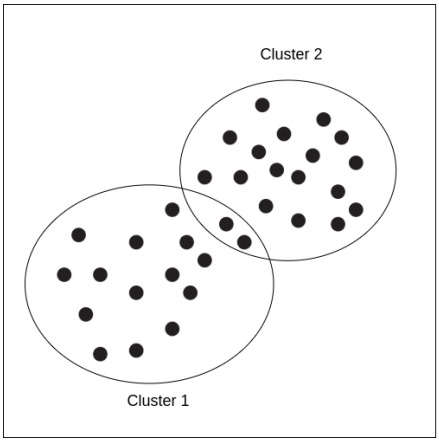


Figura 2: Fuzzy clusterering.

- **Cálculo de similitud**: Una de las pautas que más determina un algoritmo de clusterización es la forma en la que calculan la similitud entre los datos, y que exploraremos detalladamente más adelante. Gran parte de los algoritmos utilizan la distancia que hay entre dos objetos para calcular su similitud aplicando métodos como la distancia Euclídea o espacios vectoriales. Este tipo de cálculo se ve beneficiado de métodos de optimización, pero suele generar clusters con formas esféricas o cilíndricas, por lo que cada vez más se utilizan algoritmos que miden la similitud entre objetos mediante la densidad o contigüidad del espacio para formar clusters, lo cual permite descubrir grupos con formas arbitrarias, aunque no son los únicos dos tipos.
- **Espacio de clusterización**: Muchos algoritmos de clustering observan todo el espacio de datos para buscar grupos, que es un método totalmente factible y válido para datos con pocos atributos. Sin embargo, cuando se tratan de procesar datos con gran cantidad de atributos,

es mejor centrarse en pequeñas partes del total, en subconjuntos del espacio de datos y buscar clusters dentro de ellos de forma que se pueda obtener información valiosa a un menor coste computacional.

3.3 Clasificación general de técnicas

[16] Una vez que hemos visto los requisitos de los algoritmos de clustering y los criterios para compararlos entre sí, podemos agrupar la mayor parte de las técnicas bajo cuatro grandes grupos basados principalmente en el cálculo de la similitud y el criterio de división. Hablaremos brevemente sobre estos a continuación y posteriormente, precisaremos de manera individual sobre cada uno de ellos en las secciones 4, 5, 6 y 7, señalando los principales algoritmos dentro de cada una de ellos.

- **Métodos basados en particiones**: Este tipo de métodos trata a cada objeto de manera equitativa situándolos a todos en el mismo nivel. Esto se consigue haciendo que en cada cluster haya al menos un objeto, es decir, si tenemos un conjunto de datos con n objetos, los métodos basados en particiones generarán k clusters siendo $k \leq n$.

  Normalmente las técnicas de este grupo generan clusters exclusivos, donde un datos solo puede pertenecer a un cluster, pero cada vez se desarrollan más algoritmos de tipo fuzzy que relajan la agrupación. Asimismo, muchos de estos métodos calculan la proximidad entre los objetos como medida de similitud y generan de manera inicial k particiones para posteriormente, de forma iterativa, tratar de reagrupar los datos y clusters para mejorar la clasificación, basándose en que los datos dentro del mismo cluster estén realmente cerca entre sí, y que con los datos de otros clusters se encuentren lejos.

  Estos métodos sirven tanto para búsquedas en todo el espacio de datos, como clusterización de espacios reducidos, pero cuando la cantidad de atributos es muy alta, es preferible utilizar esta segunda opción y aplicarlos en subconjuntos, pues es complicado optimizar computacionalmente este tipo de algoritmos a la hora de generar las particiones iniciales. Para lidiar con este inconveniente, se suelen usar métodos heurísticos como *k-means* que intentan mejorar la calidad de clustering progresivamente. Sin embargo, como se ha mencionado previamente, esta aproximación de clustering genera grupos con formas esféricas y circulares, por lo que uno de los principales focos de mejora de estos algoritmos es evitar ese tipo de comportamiento cuando se trabaja con grandes conjuntos de datos.

- **Métodos jerárquicos**: Las técnicas de este grupo crean, como su propio nombre indica, una separación jerárquica de los datos. De forma diferente a los métodos basados en particiones donde todos los datos están en el mismo nivel, aquí se busca generar varios rangos para la clusterización.

  Existen dos enfoques principales dependiendo de cómo se generen los

clusters: *aglomerativo* y *divisivo*. Las técnicas aglomerativas parten de la separación de todos los objetos en clusters individuales que progresivamente se van uniendo hasta que solo queda uno; mientras que las técnicas divisivas parten de un solo cluster inicial en el que se encuentran todos los objetos y va formando clusters más pequeños hasta que cada objeto es en sí un cluster. Es por eso que también se les conoce por métodos de "abajo a arriba" y de "arriba a abajo" respectivamente.

Esta técnicas pueden utilizar proximidad, densidad y continuidad como medida de calcular la similitud entre los datos y son útiles tanto en búsquedas de espacios completos como de subconjuntos.

- **Métodos basados en densidad**: Este tipo de métodos surgieron como solución a los métodos basados en particiones y su tendencia a encontrar únicamente clusters con formas esféricas o circulares, puesto que las técnicas que utilizan la densidad del espacio a la hora de formar clusters permiten descubrir clusters con formas más arbitrarias.

  El concepto general de estos métodos consiste en agrandar un cluster mientras la densidad en el *vecindario* esté por encima de un umbral. La densidad se calcula por la cantidad de objetos que hay en el espacio de datos y el vecindario hace referencia a todos los datos que se encuentran a cierta distancia de un objeto determinado.

  El concepto general es que para cada objeto dentro de un cluster, en su vecindario de radio r de distancia debe haber un mínimo de x objetos. Este tipo de aproximación resulta muy conveniente a la hora de detectar datos anómalos.

  Los métodos basados en densidad pueden clasificar los datos en clusters exclusivos o de manera jerárquica, y aunque normalmente consideran que un objeto solo puede pertenecer a un cluster, también existen técnicas de fuzzy clustering. Asimismo, se pueden aplicar tanto en espacios completos como en regiones puntuales.

- **Métodos basados en cuadrícula o rejilla**: Estos métodos dividen el espacio de objetos en una cantidad determinada de celdas que forman una estructura de cuadrícula o rejilla, de ahí su nombre. Todas las operaciones con los datos se realizan dentro de la rejilla, lo que permite un procesamiento mucho más rápido, pues no depende de la cantidad de datos, sino de la de celdas existentes.

  Esto permite que estos métodos puedan ser integrados con otros métodos de clustering pertenecientes a otros grupos como jerárquicos y basados en densidad para mejorar su eficiencia.

En la tabla 1 se han resumido estos grupos y las principales características de los mismos. Cabe mencionar que esta clasificación no es perfecta, pues existen algoritmos que integran conceptos de diferentes métodos.

Sin embargo, el mundo de Data Mining y de los datos no deja de cre-

cer, implicando la aparición de nuevos tipos de problemas que afrontar con clustering. Esto ha dado lugar al desarrollo de nuevos algoritmos de clusterización que son lo suficientemente diferentes a los ya mencionados como para poder considerarlos otro tipo de métodos. Algunos de estos nuevos enfoques de clusterización se han convertido muy populares debido a sus nuevas características:

- **Métodos de clusteriación relajada**: Como se ha mencionado previamente, lo más común a la hora de formar cluster es que estos sean exclusivos, es decir, que un objeto solo puede pertenecer a un cluster. Los métodos de clusterización relajada o fuzzy clustering permiten que los objetos puedan estar en varios clusters a la vez. Esto se consigue utilizando el grado de pertenencia, un valor entre 0 y 1 que cuantifica la probabilidad o confianza de que un objeto pertenezca a un determinado cluster. La suma del grado de pertenencia de un objeto debe ser 1.

  Este enfoque es realmente útil cuando las fronteras entre los clusters no están separadas de una forma clara y es ambigua, puediendo además obtener más informacion gracias al grado de pertenencia.

  Este tipo de métodos a veces se consideran modificaciones o variaciones de algoritmos, pero con la cantidad de atención y aplicaciones que ha recibido recientemente es adecuado tratarlos como un grupo distinto.

- **Métodos basados en modelos probabilísticos**: Uno de los objetivos de clustering es encontrar una estructura subyacente en los datos, algo oculto que no se vea a simple vista pero que esté latente en la muestra a analizar. Este tipo de métodos se aprovechan de esta característica para utilizar funciones de distribución y/o densidad como el origen de los datos a analizar, pues desde el punto de vista probabilistico, se asume que los objetos se generan de acuerdo a diversas distribuciones de probabilidad.

  Normalmente se utilizan distribuciones conocidas y estudiadas en profundidad como la Gausiana, eliminando así tener que buscar el modelo generativo y limitando la identificación de clusters a estimar los parámetros de dichas funciones probabilísticas.

- **Métodos basados en la teoría de grafos**: Podemos utilizar la teoría de grafos [18] para representar la clusterización mediante grafos ponderados, donde los nodos representan los objetos y las aristas que los conectan reflejan la proximidad entre pares de datos.

  Para determinar los clusters se cortan las aristas más débiles, dejando juntos aquellos objetos que estén bien conectados entre sí, considerando esto un cluster, tal y como se aprecia en la Figura 3, donde hay tres clusters claramente diferenciados por la falta de conexiones entre los objetos.

  Estos métodos surgieron como variante de los métodos jerárquicos

y sirven tanto para clusterizar datos con muchos atributos o para analizar grafos per se.

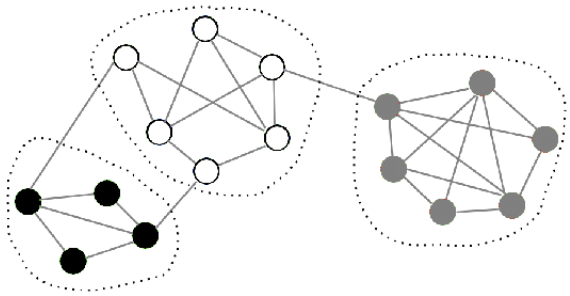


Figura 3: Clusterización basada en teoría de grafos [19].

- **Métodos basados en redes neuronales**: Las redes neutonales artificiales son un área del aprendizaje automático que representan modelos matemáticos de la actividad cerebral. Su objetivo inicial era reproducir el procesamiento de información humano, y ahora este enfoque se ha aplicado en mucho campos, entre ellos clasificación.

  Estos modelos consiste en un conjunto de unidades denominadas neuronas que están conectadas entre sí para trasmitir señales, de forma similar al cerebro humano. La información que recibe la red de forma inicial atraviesa las distintas neruonas donde se realizan ciertas operaciones y finaliza dando un valor de salida.

  Este tipo de métodos utilizan redes neuronales para localizar los clusters o en su defecto, sus centros, para facilitar el posterior desarrollo.

A continuación, exploraremos los cuatro primeros grupos (métodos basados en particiones, jerárquicos, basados en densidad y basados en rejilla), puesto que son los más populares y comunes, y tambien veremos los principales algoritmos empleados en cada uno de ellos y otros pertenecientes a los nuevos tipos de métodos.

Métodos	**Características generales**
Basados en particiones	- Encuentran clusters de forma esférica mutuamente exclusivos - Utilizan distancia/proximidad - Usan la media o medoid para representar el centro del cluster - Efectivos en conjuntos de datos pequeños y medianos
Jerárquicos	- Clasifican en múltiples niveles - No pueden deshacer agrupaciones o divisiones erróneas - Pueden incorporar otras técnicas como microclustering y tener en cuenta vínculos entre los objetos
Basados en densidad	- Pueden encontrar clusters con formas arbitrarias - Los clusters son regiones con gran densidad de objetos separados por zonas con poca densidad - La densidad queda definida por un mínimo de objetos cercanos dentro del vecindario - Sirve para detectar datos anómalos
Basados en rejilla	- Utiliza una estructura de rejilla o cuadrícula - La velocidad de procesamiento es alta, pues no influye el número de objetos

Cuadro 1: Resumen de métodos de clusterización

4 Métodos basados en particiones

[16] Los métodos pertenecientes a este grupo son quizás los más simples y básicos de clustering. Estos organizan los datos de un conjunto en diversos clusters exclusivos, y es común que soliciten al usuario el número de grupos o clusters finales que debe haber, lo que sirve como punto de partida . Esto permite simplificar su funcionamiento, pero a su vez, supone el incumplimiento de uno de los requisitos de los que se ha hablado previamente en la sección 3.1.

De forma técnica, dado un conjunto de datos D con n objetos y siendo k el número de clusters a formar, un algoritmo basado en particiones organizará los objetos en k divisiones siendo $k \leq n$, donde cada una de ellas representa un cluster.

Dichos clusters son generados intentando optimizar el criterio de similitud, siendo el del cálculo de proximidad uno de los más utilizados. Este intenta agrupar bajo el mismo cluster aquellos objetos que están cerca entre sí en el espacio de datos mientras hacen que todos los datos de un cluster estén alejados de los objetos de otro cluster, asegurando de esta manera un alto grado de semejanza entre los miembros del mismo cluster y de diferencia con los datos de otros clusters.

A continuación se exponen los dos algoritmos más utilizados de este grupo y de los más conocidos: k-media (k-means) 4.1 y k-medoids 4.3.

4.1 *k*-Means

El objetivo de los métodos basados en particiones es, dado un conjunto de datos D con n objetos en un espacio Euclídeo[12], formar k clusters, $C_1, ..., C_k$, donde cada cluster es un subconjunto del conjunto de datos $C_i \subset D$ y la intersección entre clusters es nula $C_i \cap C_j = \emptyset$ para $i \leq j$ y $j \leq k$. Sin embargo, estos requieren de una gran capacidad computacional, y es por ello que algoritmos como k-Means son tan utilizados, pues disminyen este coste.

k-Means representa cada cluster con su *centroide*, que es básicamente el punto central del cluster y que se calcula como la media de los puntos que lo forman. Para un conjunto de datos con dos atributos cuantitativos, la fórmula para calcular el centroide c_i es la siguiente:

$$c_i = \left(\frac{\sum_{i=1}^{n} x_i}{n}; \frac{\sum_{j=1}^{m} y_j}{m} \right). \tag{1}$$

Asimismo, otro de los métodos más comunes para medir la distancia entre puntos y clusters, y que también utiliza k-Means, es la distancia Euclídea. La fórmula de la distancia Euclídea entre dos puntos cualesquiera $dist(p, q)$ de

[12] Espacio geométrico donde se satisfacen los axiomas de Euclides.

un espacio Euclídeo p y q, con coordenadas cartesianas[13] (x_1, y_1) y (x_2, y_2) es como sigue:

$$dist\,(p,q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2}$$

De esta manera, se utilizan sendas fórmulas para comprobar la calidad de un cluster y garantizar que los objetos del mismo sean similares entre ellos y diferentes con los objetos de otros clusters. Para ello, los métodos basados en particiones suelen utilizar la *varianza* dentro del cluster, que es la suma del error cuadrático[14] entre todos los objetos del cluster C_i y el centroide c_i, definida como:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2, \tag{3}$$

donde E es la suma del error cuadrático de todos los objetos pertenecientes al cluster $p_i \in C_i$. La finalidad de esta operación es asegurar que los k cluster resultantes sean lo más compactos y distantes posible.

Es aquí, en el cálculo de la varianza donde la optimización de este método se vuelve computacionalmente complicada, pues en el peor de los casos, el total de posibles clasificaciones depende de manera exponencial del número total de clusters y posteriormente habría que comprobar los valores de la varianza dentro de cada cluster, siendo un problema de complexión NP-complejo[15]. Para resolverlo, se utilizan algoritmos voraces[16] como k-Means, simples y usados habitualmente.

Este algoritmo sigue una serie de pasos para formar los clusters y evaluar su calidad. Primero, selecciona n objetos dentro del conjunto de datos D para representar los clusters iniciales; y puesto que son los únicos integrantes de ellos, el cálculo del centroide da como resultado el propio objeto. Posteriormente, asigna cada uno de los objetos restantes al cluster con el que es más similar basándose en el cálculo de la distancia Euclidiana entre el objeto y representante del cluster. A partir de aquí, el algoritmo intenta mejorar los resultados de la varianza, volviendo a calcular el centroide de cada cluster con los nuevos objetos añadidos en la iteración previa, y reagrupa los objetos ahora con los representantes de cada cluster actualizados. El proceso continua hasta que los centroides, y por ende, los clusters de la iteración actual sean idénticos a los de la iteración previa. El funcionamiento de k-Means queda reflejado en el Algoritmo 1 y ejemplificado en la Figura 4.

Sin embargo, este método no garantiza converger hacia un máximo global, sino que muchas veces termina en un máximo local por la utilización del

[13] Tipo de coordenadas ortogonales usadas en espacios Euclídeos que tienen como referencia los ejes ortogonales.

[14] Técnica para calcular la diferencia entre el estimador y lo que se estima.

[15] No se definir esto bien

[16] Estrategia de búsqueda en la que se sigue una heurística que consiste en escoger la opción más optima en cada paso de manera local, esperando llegar a una solución general óptima.

Algoritmo 1: k-Means, basado en la media

Entrada: k: número de clusters,
D: conjunto de datos con n objetos.
Salida: Un conjunto de k clusters.

```
1 inicio
2     selección arbitraria de k objetos de D como representantes de los
        clusters iniciales
3     repetir
4         calcular la distancia de cada objeto al centroide de cada cluster
5         (re)asignar cada objeto al cluster con el que es más similar en
            función de la distancia obtenida
6         actualizar los centroides de los clusters con los nuevos objetos
            añadidos
7     hasta no hay cambios
8 fin
```

algoritmo voraz. Esto hace que el resultado dependa de la selección de clusters iniciales sobre los que se trabaja, por lo que es es común realizar varias pruebas con diferentes clusters iniciales hasta encontrar un resultado óptimo.

Por otro lado, la complejidad temporal de k-means es considerablemente baja $O(nkt)$, pues depende del número de objetos n, del de clusters k y de la cantidad de iteraciones t y normalmente encontramos que $k \ll n$ y $t \ll n$. Lo cual sugiere que escalar este algoritmo a grandes conjuntos de datos es asequible.

Asimismo, este algoritmo solo puede ser utilizado cuando podemos calcular la media de los objetos para encontrar el centroide, por lo que en casos donde los datos tengan atributos nominales, no se puede aplicar de manera directa. Para ello existen modificaciones y versiones de k-means que solucionan algunos de estos problemas, como k-modas, que en vez de utilizar la media para calcular el centroide, utiliza la moda, el atributo que más se repite para una característica y si permite trabajar con datos nominales.

Otra de las desventajas que presenta y que comentábamos al inicio de la sección, es el hecho de que no cumple uno de los requisitos de los métodos de clustering, y es que k-means pide al usuario el número k de clusters a generar como dato de entrada indispensable. Se han realizado estudios para solventar este problema intentando encontrar el mejor valor para k ejecutando varias veces el algoritmo con distintos valores para esta medida, pero este paso extra supone un mayor coste computacional.

Además, recordemos que este algoritmo flaquea a la hora de encontrar clusters con formas poco convencionales y de tamaños dispares, siendo realmente sensible a datos anómalos e impurezas, que pueden alterar los cálculos de los centroides y del resultado final.

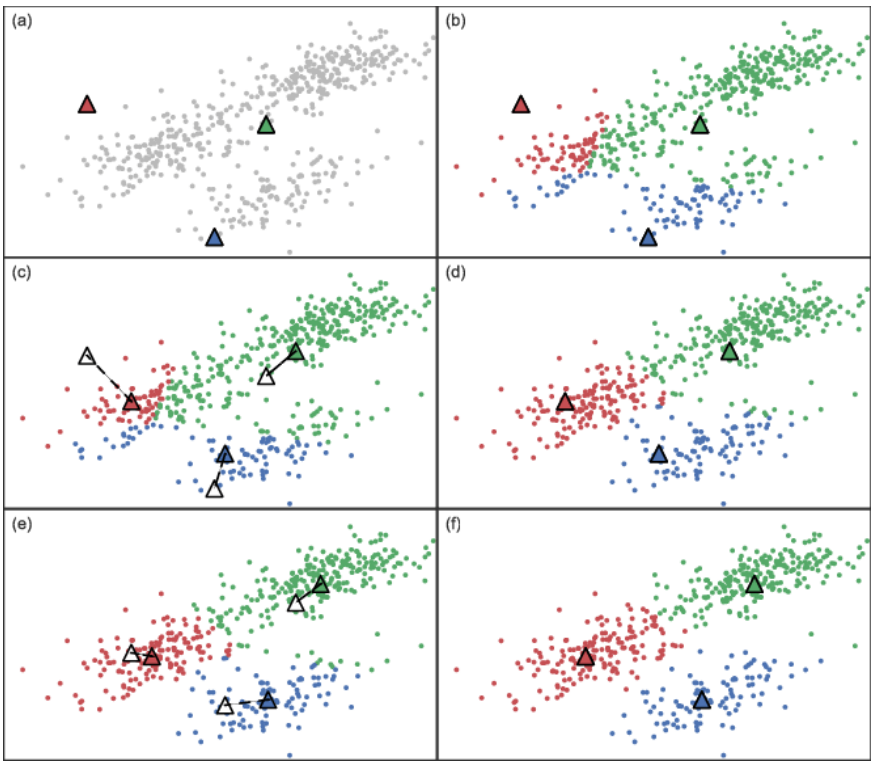


Figura 4: Ejemplo de k-means en dos dimensiones. (a) Los centroides de los clusters se inicializan en posiciones aleatorioas. (b) Primer paso: cada objeto se añade al cluster cuyo centroide es el mas cercano. (c) Segundo paso: Cada centroide se mueve a la media de los puntos que conforman el cluster. (d, e, f) Estos pasos se repiten hasta que el resultado de dos iteraciones sea el mismo. [20].

4.2 Fuzzy *k*-Means

Este algoritmo, también conocido como cuzzy c-means (FCM) [21] es uno de los métodos de clusterización relajados más populares y utilizados [35]. Utilizar algoritmos exclusivos en conjuntos de datos donde existen clusters superpuestos queda lejos de ser óptimo, por lo que aplicar una clusterización relajada es el método recomendado.

En FCM al igual que muchos otros métodos de clusterización relajada la pertenencia a diferentes clusters varía entre 0 y 1 y tiene un procedimiento similar al de k-means puesto que pretende reducir de forma iterativa el error cuadrático actualizando y modificando los valores del grado de pertenencia a los clusters utilizando los centroides de los mismos como representantes. El proceso termina cuando los centroides de una iteración no cambian con respecto a la anterior.

Al igual que k-means, este también es susceptible a las impurezas y datos

anómalos de los datos. Muchos de los algoritmos de clusterización relajada se han basado en este para su desarrollo.

4.3 *k*-Medoids

Como se ha mencionado en el apartado previo 4.1, k-means es realmente susceptible a datos anómalos e impurezas, pues como estos objetos se suelen encontrar alejados de todos los datos, cuando se asignan a un cluster y se recalcula el valor del centroide, este valor se ve altamente alterado, lo cual también afecta a la formación del resto de clusters.

Con el objetivo de ser menos sensible con estos datos, k-medoids utiliza uno de los objetos del cluster en vez de su centroide como representante del mismo, y el resto de objetos se añaden al cluster cuyo objeto representativo sea más similar. Por lo que el método para comprobar la calidad del cluster ya no es la varianza, sino el error absoluto, cuya fórmula es la siguiente:

$$E = \sum_{i=1}^{k} \sum_{\in C_i} dist(p, o_i)^2, \tag{4}$$

siendo E la suma del error absoluto de todos los puntos p_i pertenecientes al cluster C_i ($p_i \in C_i$) con respecto al objeto representativo de dicho cluster o_i. De esta manera, k-medoids agrupa n objetos en k clusters minimizando el error absoluto.

Los algoritmos más conocidos que utilizan esta aproximación para formar clusters son Partioning Around Medoids(PAM) y Clustering LARge Applications (CLARA), cada uno con sus propias características, que veremos a continuación.

4.3.1 PAM

Este algoritmo, que podría traducirse al español como Particionamiento Sobre Medoids, es una de las aplicaciones más extendidas del método k-medoids, y tiene un funcionamiento similar a k-means en el sentido de que utiliza también una heurística voraz de forma iterativa para la formación de clusters y disminuir la complejidad computacional.

Y de la misma forma, el paso inicial también es seleccionar de manera aleatoria k objetos para formar los clusters iniciales, denominados "semillas", y asignar los objetos restantes al cluster cuyo objeto representativo (medoid) es el más similar. Posteriormente, con los nuevos clusters, se comprueba si cambiando el objeto representativo del cluster actual o_i por otro objeto cualquiera dentro del cluster $o_{cualquiera}$ se mejora la calidad del cluster, y en caso de que sea así, se intercambia uno con otro. Esta comprobación se realiza para todos los objetos dentro del cluster que no sean el representante $o_{cualquiera} \neq o_i$ y para todos los clusters, hasta que no se pueda mejorar la situación actual. Una vez ha terminado este proceso, se vuelve a calcular la

similitud entre todos los objetos y los representantes de los clusters y se reagrupan si es necesario. El algoritmo termina cuando los clusters resultantes de la iteración actual son idénticos a los de la previa.

Para determinar la calidad del cluster y decidir si un representante del cluster debe ser cambiado por otro objeto dentro del cluster como se ha mencionado anteriormente, se calcula la distancia entre todos los puntos del cluster y su objeto representante actual o_i mediante la fórmula 4, cuyo resultado es almacenado. Luego, de forma iterativa cada objeto perteneciente al cluster se convierte en su representante o_j temporal y se vuelve a calcular la distancia entre el resto de puntos y el representante temporal, y su resultado también se almacena, de forma que, una vez todos los puntos han sido representantes temporales y se ha aplicado la fórmula, se escoge como representante aquel objeto cuyo resultado ha sido el más bajo, es decir, el objeto más cercano a la mayoría de puntos del cluster, y si este nuevo punto es distinto al representante actual $o_i \neq o_j$, o_j se convierte en el nuevo representante del cluster.

Cuando un intercambio de representante ocurre en un cluster, se debe comprobar si es necesario reestructurar los clusters y el coste que esto conlleva, pues puede haber puntos cuyo cluster más cercano haya variado. Por lo que para cada punto del espacio de objetos se calcula la distancia a los nuevos representantes y se selecciona el que esté más cerca, pudiendo ser un nuevo cluster o el mismo en el que se encontraba pero con el nuevo representante. Una vez seleccionado el cluster más cercano para cada objeto, se calcula el coste con la misma función 4, y si el resultado de esta reestructuración es negativo, es decir, si en la nueva situación los clusters representan mejor a los objetos porque el representante está más cerca de los puntos, se acepta este cambio y se considera que ha aumentado la calidad de los clusters.

Este procedimiento, que queda reflejado en el algoritmo de PAM 2, es más robusto que k-means al ser más tolerante a datos anómalos e impurezas, porque el medoid está menos influenciado por estos que la media. Sin embargo, esta aproximación es mucho más compleja computacionalmente para cada iteración, siendo esta $O\left(k\left(n-k\right)^2\right)$ y además, sigue solicitando el número de clusters k, por lo que para grandes volúmenes de datos resulta ineficiente, dando origen a otra versión enfocada precisamente a este tipo de trabajo: CLARA.

4.3.2 CLARA

CLARA es una variación de PAM y que también aplica k-medoids a la hora de formar clusters, pero que en lugar de tener en cuenta todos los datos n del conjunto de datos D, este algoritmo escoge una muestra aleatoria m de datos y posteriormente aplica PAM para calcular los mejores representantes de la muestra. Esta muestra idealmente representa de una manera precisa al conjunto de datos original D siempre y cuando cada objeto p_i tenga las mismas posibilidades de ser escogido para la muestra. De esta manera y tras varias iteraciones seleccionando muestras arbitrarias, CLARA devuelve la

Algoritmo 2: PAM, basado en representantes

Entrada: k: número de clusters,
D: conjunto de datos con n objetos.
Salida: Un conjunto de k clusters.

```
1  inicio
2      selección arbitraria de k objetos de D como representantes o_i de los
         clusters iniciales
3      repetir
4          calcular la distancia de cada objeto al representante de cada
             cluster
5          (re)asignar cada objeto al cluster con el que es más similar en
             función de la distancia obtenida
6          seleccionar un objeto cualquiera del cluster que no sea el
             represente actual o_cualquiera
7          calcular el coste T de intercambiar el representante del cluster
             o_i con o_cualquiera
8          si T<0 entonces
9              intercambiar o_i con o_cualquiera como representante del cluster
10     hasta no hay cambios
11 fin
```

mejor clusterización obtenida de todas ellas para aplicarla, ahora sí, en todo el conjunto de datos D.

Esto permite que la complejidad computacional disminuya, pues la comprobación de la calidad de los clusters y el intercambio de representantes que tanto costaba en PAM es reducido de manera drástica con este enfoque, permitiendo de esta manera que pueda trabajar con grandes volúmenes de datos. La complejidad para calcular los medoids de una muestra aleatoria es de $O\left(k \cdot s^2 + k\left(n-k\right)\right)$, siendo s el tamaño de la muestra.

Sin embargo, este algoritmo depende totalmente de las muestras escogidas y la probabilidad de que el mejor representante de cada cluster se encuentre en ella, puesto que si tan solo uno de ellos no aparece en alguna de las muestras, CLARA nunca encontrará la mejor clusterización posible.

5 Métodos jerárquicos

[16] Mientras que los métodos basados en particiones cumplen con el requisito básico de agrupar los datos en clusters exclusivos, hay ocasiones en las que clasificar los datos en diferentes niveles, es decir, de forma jerárquica, puede proporcionar grandes ventajas. De esta manera, los métodos jerárquicos organizan los datos en una jerarquía o árbol de clusters.

Útil en escenarios donde los datos tengan de por sí cierta estructura jerárquica (*p. ej.* estructura de empleados de una empresa) o en aquellos en los que se pretenda descubrir si existe una oculta(*p. ej.* teoría de la evolución),

estos métodos se clasifican a su vez en dos grupos: aglomerativos y divisivos.

- **Métodos jerárquicos aglomerativos**: Estos métodos utilizan una estrategia de "abajo a arriba", partiendo de que cada objeto compone su propio cluster, de forma iterativa los va uniendo en clusters más grandes hasta que solo queda un único cluster, que se conoce como "raíz". En cada iteración se mergen los dos clusters más cercanos entre sí para formar uno solo.
- **Métodos jerárquicos divisivos**: Este tipo de métodos utilizan una estrategia de "arriba a abajo", donde se sitúa a todos los objetos en un cluster inicial, la raíz, y de manera progresiva se divide la raíz en subclusters más pequeños sucesivamente hasta que cada cluster queda formado únicamente por un objeto o los objetos de cada cluster son prácticamente idénticos entre sí.

Como se puede ver, son métodos opuestos pero que operan bajo el mismo principio. Sin embargo, es mucho más complicado dividir los datos que juntarlos, pues hay muchas más posibilidades, en concreto, existen $2^{n-1} - 1$ formas de dividir un conjunto de n datos en dos clusters; por lo tanto, hay má muchas más comprobaciones que realizar, dando lugar a que los métodos aglomerativos sean más comunes y los divisivos dependan de métodos heurísticos que pueden dar resultados inexactos.

Los métodos jerárquicos pueden sufrir dificultades a la hora de dividir o juntar clusters, una parte crítica en su procedimiento, puesto que, a diferencia de los métodos basados en particiones donde en cada iteración se reagrupaban los puntos hacia el cluster más cercano, si un método jerárquico divide o junta dos clusters, no hay vuelta atrás y los siguiente pasos trabajarán con el resultado obtenido previamente sin poder cambiarlo. Por lo que decidir cómo y por qué motivo se genera un cluster es una tarea complicada y que de no ser correcta, puede ocasionar clasificaciones pobres. Otra desventaja principal de este enfoque es que no escalan bien hacia grandes volúmenes de datos, puesto que deben realizar muchas comprobaciones cuando se quieren unir o separar clusters aumentando exponencialmente el coste computacional.

Para lidiar con estos problemas se han desarrollado técnicas como BIRCH y Chameleon de las que se habla en las secciones 5.4 y 5.5, que han resultado en otro tipo de clustering conocidos como multi fase y microclustering.

Antes de explicar algunos de los algoritmos más utilizados dentro de este grupo, AGNES 5.2, DIANA 5.3, BIRCH 5.4 y Chameleon 5.5, analizaremos brevemente las diferentes maneras que existen en los métodos jerárquicos para medir distancias entre clusters.

5.1 Medidas de distancia

Independientemente del tipo de método que se utilice, aglomerativo o divisivo, se necesita una forma de medir la distancia entre dos clusters. Hay

principalmente tres formas diferentes de realizar este cálculo como se puede apreciar en la figura 5, aunque existen otras variaciones en las que, por ejemplo, se consideran los centroides como en k-means, estas utilizan puntos de clusters.

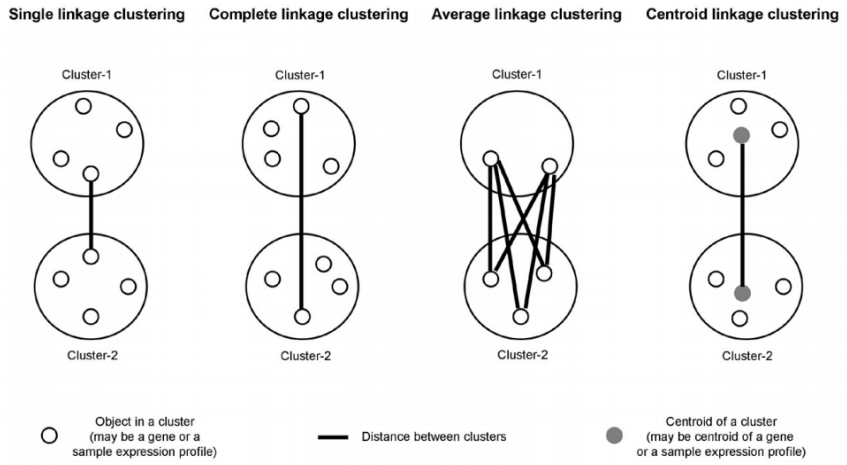


Figura 5: Diferentes métodos para calcular la distancia entre clusters [22].

- **Distancia mínima**: Esta define la proximidad entre dos clusters como la distancia entre los puntos más cercanos entre ambos, también conocida como "single link".

$$dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\} \tag{5}$$

- **Distancia maxima**: Define la proximidad entre dos clusters como la distancia entre los puntos más alejados de ellos, también conocida como "complete link".

$$dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\} \tag{6}$$

- **Distancia media**: Define la proximidad entre dos clusters como la distancia media entre todos los puntos de ambos clusters.

$$dist_{avg}(C_i, C_j) = \frac{1}{n_i \cdot n_j} \sum_{p \in C_i, p' \in C_j} |p - p'| \tag{7}$$

Las dos primeras son medidas que se van a los extremos y son susceptibles a datos anómalos e impurezas, por lo que el uso de la media es un compromiso entre ambas para solventar este problema.

5.2 AGNES

[23] AGglomerative NESting (AGNES) es quizás el algoritmo jerárquico aglomerativo más básico y extendido, pues aplica de forma exacta el concepto mencionado previamente.

Parte de una separación de todos los objetos, cada uno formando un único cluster y compara la proximidad entre todos ellos utilizando el método que más convenga de los mencionados en la sección previa 5.1. Selecciona aquellos dos con el resultado más pequeño, es decir, los dos más cercanos y los fusiona en un solo cluster, y se vuelven a calcular la similitud entre todos los clusters. Así sucesivamente hasta que todos los datos quedan agrupados en la raíz [24].

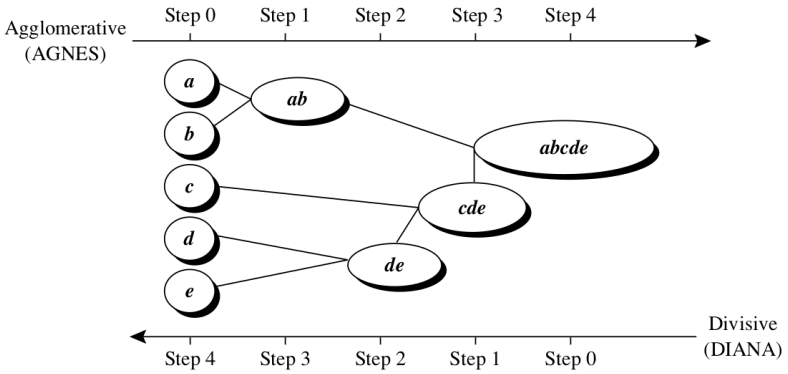


Figura 6: Clusterización jerárquica algomerativa (AGNES) y divisiva (DIANA) sobre el conjunto de datos $\{a,b,c,d,e\}$ [16].

5.3 DIANA

[23] El algoritmo DIvisive ANAlysis (DIANA) es la contraparte a AGNES, siendo también el método jerárquico divisivo más simple y utilizado. En este caso, al ser divisivo, parte de un solo cluster en el que se encuentran todos los datos y que de forma iterativa los va dividiendo en función de un criterio preestablecido, como por ejemplo la distancia Euclidiana mínima entre los objetos más cercanos del vecindario.

Tanto AGNES como DIANA generan una estructura denominada dendograma que se puede ver en la figura 6, donde se compara el proceso de estos dos algoritmos.

5.4 BIRCH

[26, 27] El algoritmo BIRCH, proveniente del nombre Balanced Iterative Reducing and Clustering using Hierarchies, es uno de los que mencionábamos previamente que combina el método jerárquico con otros métodos para

solucionar los problemas de escalabilidad e imposibilidad de deshacer las divisiones o fusiones de clusters.

Para ello, BIRCH utiliza un concepto denominado *característica de agrupamiento*, en inglés clustering feature (CF), que sirve para resumir un cluster, y otro llamado *árbol de características de agrupamiento*, en inglés clustering feature tree (CF-tree), para representar la jerarquía de clusters. Estos dos atributos ayudan en la escalabilidad y rapidez del algoritmo y hacen posible utilizarlo en conjuntos de datos en los que se añaden nuevas muestras. Veamos brevemente cómo funciona.

Consideremos un cluster C_i con n objetos en él, CF es un vector de tres dimensiones que pretende resumir la información estadística del cluster de la siguiente manera

$$CF = (n, LS, SS)\,, \tag{8}$$

donde LS es la suma de los n objetos y SS es la suma de los n objetos elevados al cuadrado. Esto permite obtener datos como el centroide c_i, el radio R y diámetro D del mismo, donde estos dos últimos representan lo ajustado que está el cluster alrededor del centroide.

Usando esto, no hace falta guardar todos los datos del cluster, sino únicamente esta variable. Y a la hora de fusionar dos clusters C_i y C_j, sus CF respectivos se suman tal que

$$CF_i + CF_j = (n_i + n_j, LS_i + LS_j, SS_i + SS_j)\,. \tag{9}$$

Por ejemplo, suponiendo que tenemos un cluster C_i con los puntos (2,5), (3,2) y (4,3), CF_i se calcula de la siguiente forma:

$$\begin{aligned} n_i &= (3)\,, \\ LS_i &= (2+3+4, 5+2+3) = (9, 10)\,, \\ SS_i &= \left(2^2+3^2+4^2, 5^2+2^2+3^2\right) = (29, 38)\,, \\ CF_i &= (n_i, LS_i, SS_i) = (3, (9, 10)\,, (29, 38))\,. \end{aligned}$$

Y si ahora, consideramos que hay que fusionar C_i con C_j para formar otro cluster nuevo C_z, siendo $CF_j = (3, (6, 7)\,, (36, 49))$, la operación sería la siguiente:

$$\begin{aligned} &CF_z = CF_i + CF_j = \\ &= (3, (9, 10)\,, (29, 38)) + (3, (6, 7)\,, (36, 49)) = \\ &= (6, (15, 17)\,, (65, 87))\,. \end{aligned}$$

El otro concepto que se ha mencionado previamente, CF-tree, tiene una estructura en forma de árbol que contiene el CF de cada cluster. Esta estructura de árbol, como se aprecia en la Figura 7 está formada por nodos y hojas. Un nodo debe tener al menos un descendiente u hoja, mientras que

las hojas son nodos terminales. Las hojas almacenan el CF de un cluster, y su nodo padre o la raíz de dicha hoja almacena la suma de los CF de todos sus hijos. Este utiliza dos parámetros para controlar el desarrollo del árbol: el factor de ramificación B y el umbral T o L; estos indican cuántos hijos u hojas puede tener un nodo y cuántos nodos puede tener un cluster respectivamente. El tamaño del umbral determina el tamaño del árbol y su elección supone cierta dificultad.

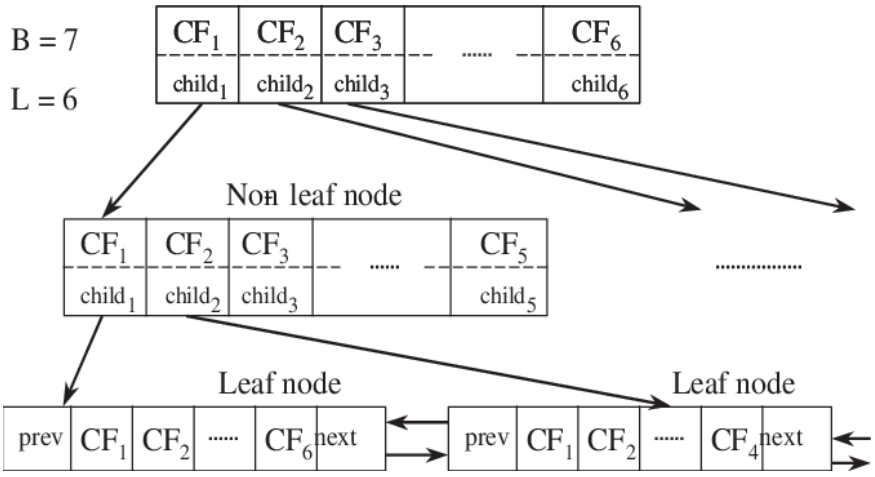


Figura 7: Estructura CF-tree del algoritmo BIRCH [25].

BIRCH es un método que consiste de dos fases principales, motivo por el que es considerado un algoritmo multifase:

1. BIRCH escanea todo el conjunto de datos con la intención de formar un CF-tree inicial para comprimir todos los datos intentando mantener la estructura inherente de los datos. Esta fase es dinámica y el CF-tree es generado conforme se van insertando más datos, que se van añadiendo en las hojas siempre y cuando estas no superen ya el umbral T de objetos permitidos, en cuyo caso, el cluster se divide en dos. Si llegase un punto en que tantos nodos no cupiesen en la memoria disponible, se aumenta el valor de T, permitiendo que los nodos alberguen más objetos y haya menos información que almacenar, provocando una reconstrucción de CF-tree.

2. Posteriormente, se aplica un algoritmo de clusterización a las hojas del CF-tree que elimina clusters con poca calidad o pocos datos, que suelen ser datos anómalos y fusiona grupos de datos densos en clusters aún más grandes. Típicamente son algoritmos basados en particiones.

Esta aproximación de clusterización ha probado ser rápida con una complejidad temporal de $O(n)$, escalable y fiable con los resultados generados, sin embargo sigue teniendo problemas puesto que el umbral T determina en gran medida la formación de clusters y puede que se generen grupos que no son precisamente "naturales", y por otro lado, al utilizar el radio como

criterio para definir clusters tiene peores resultados si los clusters no son esféricos o circulares.

5.5 Chameleon

[28] Chameleon es un algoritmo jerárquico y basado en grafos que utiliza un modelado dinámico para calcular la similitud entre pares de clusters. Esto es determinado por el grado de conexión entre los objetos dentro del cluster y la proximidad entre clusters; es decir, dos clusters se fusionarán si su interconectividad es alta y están muy cerca. Una de las principales ventajas de esta aproximación es que puede adaptarse a casi cualquier forma que pueda tomar un cluster para casi todos los tipos de datos, siempre que a estos se les pueda aplicar una función para calcular la similitud.

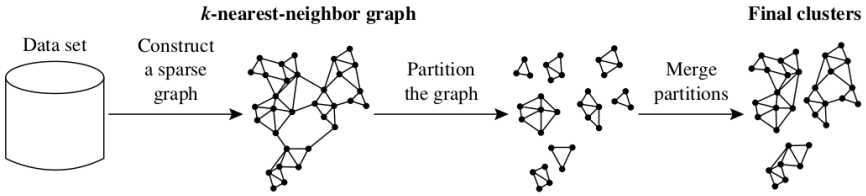


Figura 8: Chameleon: clusterización jerárquica basada en k-vecinos y modelado dinámico [16].

Este usa la aproximación de *divide y vencerás*, donde primero particiona los datos en subclusters para posteriormente ir fusionándolos [14].

En la imagen 8 podemos observar el funcionamiento de este algoritmo. Primero utiliza k-vecinos para construir un grafo disperso, donde cada vértice representa un objeto y estos están conectados mediante aristas si dichos objetos se encuentran dentro del rango de k-vecinos y son similares. Las aristas, además, están ponderadas en función de la similitud de los objetos.

Posteriormente, se aplica un algoritmo basado en particiones para dividir el grafo generado por k-vecinos en clusters, eliminando aquellas aristas con menos peso, es decir, las que unen clusters menos similares. Y una ver termina este proceso, se vuelve a aplica un algoritmo jerárquico aglomerativo que itera sobre los clusters generados y los fusiona basándose en la *interconectividad relativa* y la *cercanía relativa* de los clusters.

Este algoritmo ha demostrado ser capaz de identificar clusters con formas arbitrarias de mejor manera que otros algoritmos, pero su coste computacional sigue siendo elevado cuando se trabaja con conjuntos de datos con muchos atributos, siendo su complejidad $O\left(n^2\right)$.

5.6 Métodos jerárquicos probabilísticos

Todos los métodos jerárquicos vistos hasta ahora utilizan algoritmos y medidas de proximidad para clusterizar, lo cual hace que sean fáciles de entender

y eficientes en muchos ámbitos. Sin embargo, tienen varios problemas con los que todavía se intenta lidiar:

- Establecer una distancia de proximidad para los métodos jerárquicos es complicado en la gran mayoría de los casos.
- Para aplicar un algoritmo, es necesario que los objetos tengan todos los atributos si se desea obtener un resultado de alta calidad.
- Muchos de estos métodos utilizan heurísticas que buscan optimizaciones locales fusionando o separando clusters, pudiendo resultar en un resultado global inexacto.

De esta manera, los métodos jerárquicos probabilísticos pretenden solventar algunos de estos problemas utilizando modelos probabilísticos para calcular la proximidad entre clusters tal.

Estos métodos, que también pertenecen al grupo de métodos basados en métodos probabilísticos de los que hablábamos al principio en la Sección 3.3, tratan de ver el conjunto de datos como el resultado de un mecanismo subyacente que ha generado los datos, denominado *modelo generativo*, y lo que pretende esta forma de clustering es intentar estimar de la manera más precisa el modelo generativo a traves de los datos que se van a procesar.

Normalmente se asume que estos modelos generativos siguen funciones de distribución como la de Bernoulli o la Gausiana, por lo que se reduce mucho el campo de trabajo y solo queda probar valores hasta dar con los que más se ajustan a los objetos del conjunto de datos.

6 Métodos basados en densidad

[16] Los métodos jerárquicos y basados en particiones son muy útiles y eficaces encontrando clusters con formas esféricas y circulares, y aunque algunos de ellos sí puedan detectar en cierta medida otras formas, sigue siendo insuficiente en conjuntos de datos con datos anómalos e impurezas, donde es muy probable que no sean capaces de generar clusters de calidad.

Una solución a este problema es modelar los clusters como regiones densas de objetos separadas por regiones de baja densidad, la estrategia principal de los algoritmos basados en densidad.

A continuación exploraremos los tres algoritmos más populares dentro de este grupo, DBSCAN 6.1, OPTICS 6.2 y DENCLUE 6.3.

6.1 DBSCAN

El algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise), propuesto por Martin Ester, Hans-Peter Kriegel, Jorg Sander y Xiaowei Xu en 1996 [29], se basa fundamentalmente en encontrar los objetos en cuyos alrededores haya más puntos y conectarlos con aquellos de iguales

características para formar clusters, dejando como separación entre ellos áreas con pocos datos.

Este algoritmo solicita dos valores al usuario, el primero *Espsilon* $\epsilon > 0$ delimita el radio del vecindario de todos los objetos, y el segundo *MinPts* es utilizado para determinar la densidad del vecindario de un objeto o. Cuando el vecindario de un objeto o, definido como $N_\epsilon(o) = \{o \in D | dist(o, q) \leq \epsilon\}$, contiene más de *MinPts* objetos, o se considera un objeto central o núcleo, que es el centro o pilar de un cluster.

Sin embargo, si solo utilizáramos esto para formar los clusters, tendríamos problemas con interferencias y datos anómalos, pues los puntos en los bordes tienen menos objetos a su alrededor y se podría dar al cluster una forma errónea. Para evitar eso, se requiere que, para cada punto p de un cluster C, haya otro punto q en C tal que p esté dentro del vecindario de q ($p \in N_\epsilon(q)$) y que en el vecindario de q haya al menos $MinPts$ objetos ($N_\epsilon(q) > MinPts$). Si se cumple esta condición, se considera que p es *directamente alcanzable por densidad* desde q.

Esta condición suele ser recíproca entre núcleos, pero no entre núcleos y puntos fronterizos, tal y como se aprecia en la figura 9

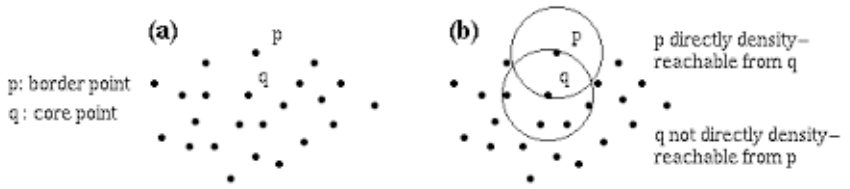


Figura 9: DBSCAN: diferencia entre núcleos y puntos fronterizos [29].

De esta definición se puede extrapolar otra, y es que un punto p es *alcanzable por densidad* desde q, si hay una cadena de puntos $p_1, \ldots, p_n$ donde el primer punto es q ($p_1 = q$) y cada punto sucesivo p_{i+1} es directamente alcanzable por densidad desde el anterior p_i. Al igual que la anterior, no suele ser recíproca con núcleos y puntos fronterizos, pero sí entre núcleos. Sin embargo, puede que existan dos puntos fronterizos del mismo cluster que no sean alcanzables por densidad entre ellos porque no tengan $MinPts$ objetos en su vecindario, pero que si lo sean desde un núcleo del cluster, que da lugar a otra definición.

Se dice que un punto p está *conectado por densidad* a un punto q si hay otro punto o desde el cuál, ambos, p y q, son alcanzables por densidad, y esta propiedad si es recíproca siempre.

Ejemplos de estas tres definiciones pueden observarse en la figura 10.

Con estos conceptos, DBSCAN es capaz de formar clusters de forma que todos sus objetos estén conectados por densidad y detectando como impurezas y datos anómalos aquellos puntos que no pertenezcan a ningún cluster.

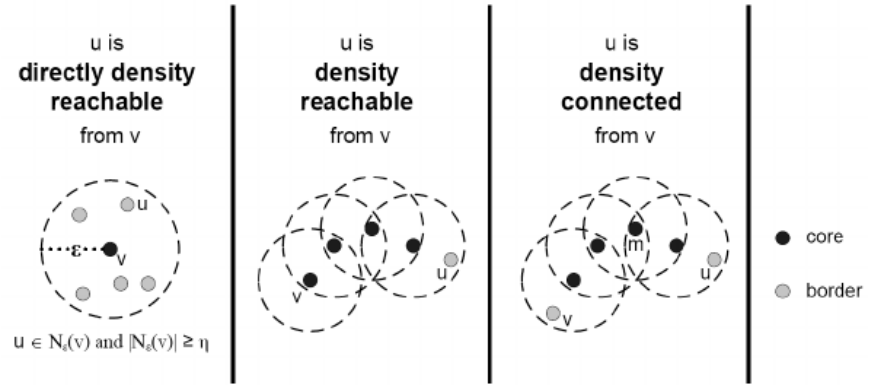


Figura 10: DBSCAN: términos *directamente alcanzable por densidad*, *alcanzable por densidad* y *conectado por densidad* [30].

Así pues, el primer paso de este algoritmo es seleccionar un punto cualquiera p del conjunto de datos D, y comprueba todos los puntos alcanzables por densidad con respecto a ϵ y $MinPts$. Si p es un núcleo, se forma un cluster con todos los objetos alcanzables por densidad desde p que no pertenezcan a otro cluster, y si es un punto fronterizo, no hay puntos que sean alcanzables por densidad desde p, por lo que es marcado como ruido y selecciona otro objeto.

Una vez ser forma el cluster, se selecciona otro objeto que no se haya visitado y se repite la operación. Asimismo, DBSCAN puede fusionar dos cluster si se encuentran muy cerca entre sí. Este procedimiento queda reflejado en el algoritmo 3.

6.2 OPTICS

Sin embargo, DBSACN sigue solicitando al usuario ϵ y $MinPts$, valores cruciales en la formación de clusters y de difícil elección que incumplen uno de los requisitos de los algoritmos de clustering vistos en la sección 3.1. Como solución a este problema y sin la necesidad de introducir parámetros de entrada, surgió OPTICS (Ordering Points to Identify the Clustering Structure), que no devuelve una clusterización, sino un orden de clusters (cluster ordering); una lista lineal de todos los objetos del conjunto de datos que representa la estructura de clusterización basada en la densidad de los datos. Aquellos objetos que sean próximos entre sí estarán más cerca en la lista de ordenación.

Con este proceso podemos extraer la información básica de la clasificación, como los centros de los clusters o grupos con formas raras, una visualización de la clusterización y también se puede derivar la estructura intrínseca de la agrupación.

Para ello, OPTICS necesita obtener dos datos adicionales para cada objeto,

Algoritmo 3: DBSCAN, basado en densidad

Entrada: D: conjunto de datos con n objetos,
ϵ: radio del vecindario,
$MinPts$: umbral de densidad del vecindario.
Salida: Un conjunto de clusters basados en densidad.

```
1  inicio
2      marcar todos los objetos como no visitados
3      repetir
4          seleccionar un objeto p no visitado
5          marcar p como visitado
6          si el ε vecindario de p tiene al menos MinPts objetos entonces
7              creamos un nuevo cluster C y le añadimos p
8              sea N el conjunto de objetos en el ε vecindario de p
9              para cada punto p' ∈ N hacer
10                 si p' no ha sido visitado entonces
11                     marcamos p' como visitado
12                     si el vecindario de p' tiene al menos MinPts
                        entonces
13                         añadimos los puntos del vecindario de p' a N
14                     si p' no pertenece a ningún cluster entonces
15                         añadimos p' a C
16             fin
17             devolvemos C
18         en otro caso
19             marcamos p como ruido
20     hasta quden objetos por visitar
21 fin
```

cuyos conceptos se reflejan en la Figura 11:

- **Distancia al núcleo**: La *distancia al núcleo* de un objeto p es el valor más pequeño para ϵ' de manera que en el vecindario delimitado por dicho valor haya al menos $MinPts$, y si p cumple con esta condición, se le considera núcleo. En caso de que que no haya $MinPts$ en el ϵ' vecindario, se considera que la distancia al núcleo es indefinida.

- **Distancia de alcance**: La *distancia de alcance* hacia un objeto p desde otro objeto q es el radio mínimo que hace que p sea alcanzable por densidad desde q (6.1), por lo que q debe ser un núcleo y p debe estar en su vecindario. Si q no es un núcleo, se considera que la distancia de alcance de p a q es indefinida.

  Es posible que un objeto p tenga más de una distancia de alcance, es decir, que sea alcanzable desde varios puntos. En este caso, la medida que más nos interesa es la distancia de alcance más pequeña. Se puede formalizar el concepto de distancia de alcance como máx$\{distancia\ al\ nucleo(q), dist(p,q)\}$.

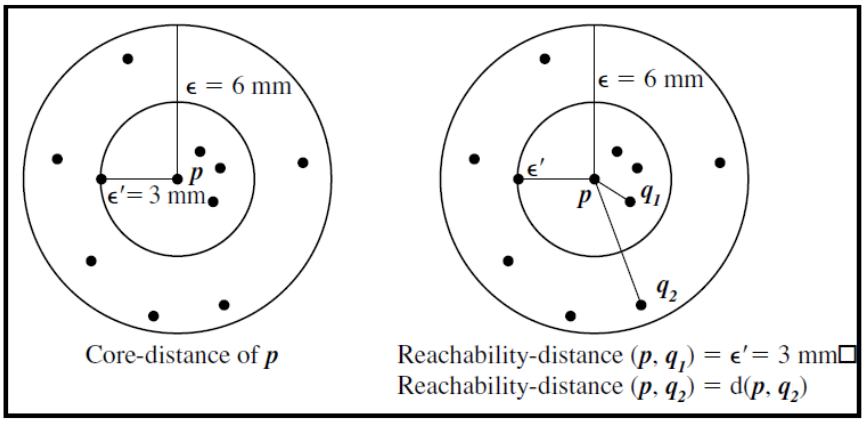


Figura 11: OPTICS: conceptos de distancia al núcleo (core-distance) y distancia de alcance (reachability-distance).

OPTICS almacena para cada objeto del conjunto de datos la distancia al núcleo y una distancia de alcance apta, y en función de la distancia de alcance de cada objeto desde su núcleo más cercano, genera una lista llamada OrdenSemillas (OrderSeeds).

Posteriormente, se escoge un objeto cualquiera como punto actual p del que se obtiene su ϵ vecindario y se calcula su distancia al núcleo, estableciendo a priori su distancia de alcance como indefinida. Si p no es un núcleo, simplemente se pasa al siguiente objeto de OrdenSemillas, pero si p es un núcleo, se actualiza la distancia de alcance para cada objeto q de su ϵ vecindario y se inserta en OrdenSemillas si todavía no está en la lista. Esta iteración continúa hasta que la lista OrdenSemillas está vacía.

Tanto DBSCAN como OPTICS tienen una complejidad de $O\left(n^2\right)$ para n número de objetos.

6.3 DENCLUE

Los dos algoritmos previos, DBSACN y OPTICS (6.1, 6.2) presentan un inconveniente, ϵ. Pequeñas variaciones en el radio del vecindario de los objetos puede resultar en clusters totalmente distintos. DENCLUE [31] (DENsity based CLUstEring) pretende solventar este problema utilizando un enfoque de estimación de densidad no paramétrica llamado *estimación de densidad nuclear*.

el concepto principal detrás de este algoritmo está basado en la idea de que la influencia de cada punto, es decir, el impacto que este tiene en su vecindario, puede ser modelada formalmente utilizando una función matemática denominada *función de influencia*, con la que podemos obtener la densidad media del espacio de datos sumando la función de influencia de todos los puntos.

De esta manera, los clusters pueden ser formados encontrando los denominados *atractores de densidad*, puntos locales máximos de la función de densidad media. Estos se pueden hallar mediante el algoritmo de Escalada Simple[17] guiado por el gradiente de la propia función. El uso de la función de densidad media también facilita encontrar clusters de formas arbitrarias.

DENCLUE implementa de manera eficaz este planteamiento y lo hace más eficaz, pues no todos los puntos del espacio de datos contribuyen de forma notable en la función de densidad media, por lo que en su lugar, usa una función local de densidad, teniendo en cuenta solo aquellos puntos que influyen en la función. En el primer paso de este algoritmo, se preprocesan los datos, generando un mapa de los objetos relevantes del espacio de datos que sirve para aumentar la velocidad de cálculo de la función de densidad. El segundo paso es el propio clustering, donde se identifican los atractores de densidad y todos los puntos a los que estos influyen.

Este algoritmo tiene varias ventajas, y es que es tan versátil como otros algoritmos como DBSACN y k-means, pero además tolera extremadamente bien datos anómalos e impurezas.

7 Métodos basados en rejilla

[16] Todos los métodos expuestos hasta ahora tienen se enfocan principalmente en los datos, es decir, se basan en la distribución de los objetos en el espacio de datos para formar los clusters. Sin embargo, los métodos basados en rejilla se centran en el propio espacio; no dividen los datos, sino el espacio de datos, que distribuyen en celdas equitativas, independientemente de cómo estén repartidos los puntos.

Estos métodos dividen el espacio de datos en un número finito de celdas que forman una estructura de rejilla o cuadrícula, donde se realizan las operaciones de clusterización, aumentando considerablemente la velocidad de clustering, hasta el punto que la cantidad de datos es irrelevante y solo depende del número de celdas. Es esta característica lo que hace estos métodos ideales para combinarlos con otro tipo de métodos y mejorar su eficiencia.

En esta sección veremos dos de los métodos más conocidos dentro de este grupo, STING 7.1 y CLIQUE 7.2.

7.1 STING

[32] STING (STatistical INformation Grid) es una técnica de clusterización en la que el espacio de datos es dividido en celdas rectangulares o cuadradas, con la opción de poder descomponerlo de forma jerárquica. Cada capa de celdas forma un nivel distinto, done cada celda de un nivel superior está compuesta por varias celdas del nivel inferior, lo cual permite obtener esa estructura jerárquica de los datos tal y como se aprecia en la Figura 12.

[17] Técnica de optimización matemática utilizada para encontrar óptimos locales.

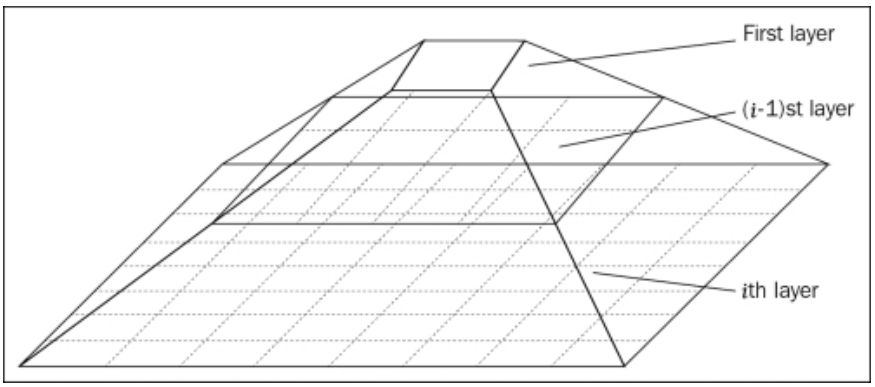


Figura 12: STING: división del espacio de datos en celdas en diferentes niveles [33].

De cada celda se calcula de antemano información estadística clave de los datos que esta contiene, tales como la media, máximo, mínimo; lo que facilita la obtención de paramentos para las celdas superiores: el número de objetos dentro de la celda, la media, desviación típica[18], máximo, mínimo y si los datos siguen algún tipo de distribución (normal, uniforme, exponencial o ninguna).

STING selecciona una de las capas superiores que se han generado, no necesariamente la primera, sino una con pocas celdas; examina todas ellas y descarta aquella celdas que tras analizarlas, se considere que tienen poca relevancia con respecto a la clasificación deseada. Posteriormente se desciende un nivel y se examinan las celdas inferiores agrupadas bajo las celdas que han superado la prueba realizando el mismo proceso de manera iterativa hasta que se alcanza la última capa. Si en esta capa se considera que se cumple con los requisitos de la clusterización, se devuelven las celdas relevantes, es decir, todas las que se han analizado en este último nivel; por el contrario, se realiza un análisis más en profundidad de las mismas. Independientemente del caso, todas las celdas descartadas durante este proceso son analizadas posteriormente.

Este comportamiento queda reflejado en el Algoritmo 4.

Este enfoque hace que la complejidad de este algoritmo para generar los clusters sea de $O(n)$ para n objetos, pero si se genera una estructura jerárquica el tiempo de procesamiento disminuye hasta $O(g)$, siendo g el número de celdas, que es mucho menor que n.

Sin embargo, uno de los principales problemas de STING es que, dependiendo del tamaño de las celdas, el resultado puede ser más o menos preciso, pues devuelve el polígono resultante de las celdas finales. Cuanto más pequeñas sean las celdas, menos se notarán las uniones bruscas y rectas entre

[18]Medida estadística que sirve para cuatificar la dispersión de un conjunto de datos numéricos.

Algoritmo 4: STING, basado en rejilla

Entrada: D: conjunto de datos con n objetos
N: nivel inicial.
Salida: Un conjunto de clusters.

```
1  inicio
2      desarrollar rejilla jerárquica
3      mientras no sea la última capa hacer
4          para cada celda del nivel actual c ∈ N hacer
5              obtener información clave
6              calcular la probabilidad de que sea una celda relevante
7          fin
8          descender un nivel
9          seleccionar las celdas de este nuevo nivel que componen las
             celdas relevantes del nivel anterior
10     fin
11     si se cumplen los requisitos de clustering entonces
12         devolver todos los datos de las celdas relevantes
13     en otro caso
14         volver a procesar los datos de las celdas relevantes y devolver
             los resultados
15 fin
```

celdas, y si el tamaño de las mismas fuera de 0, el resultado sería idéntico a DBSCAN 6.1, por lo que a veces se considera STING como un algoritmo basado en densidad.

7.2 CLIQUE

En gran cantidad de ocasiones nos encontramos con situaciones en las que algunos de los atributos de los datos no aportan información a la hora de clasificar los datos. Por ejemplo, si tratásemos de clusterizar un conjunto de pacientes de gripe, atributos como “trabajo”, “género” o “edad” varían drásticamente entre los pacientes, lo que genera dificultades a la hora de clusterizar todo el conjunto de datos. En estos casos es más conveniente realizar búsquedas en subespacios, donde se seleccionan los datos relevantes para la clasificación. En el caso de la gripe, podría ser pacientes con síntomas similares en un rango de edad entre los tres y diez años.

CLIQUE (CLustering In QUEst) es un método simple de clusterización basada en rejilla para encontrar clusters basados en densidad dentro de subespacios. Esta técnica particiona cada atributo de los datos en una dimensión diferente que posteriormente divide en celdas y cataloga como densas o no si la cantidad de objetos dentro de una celda supera cierto umbral, como se puede obeservar en la Figura 13.

El primer paso de este algoritmo consiste precisamente en esto, dividir los datos en diferentes dimensiones, generar las rejillas y encontrar las celdas

que superen el umbral. Para encontrar estas celdas, se utilizan los ejes cartesianos. Se delimitan ciertos intervalos en los valores de cada atributo, y si un intervalo supera el umbral, es decir, hay más de X puntos dentro de ese intervalo ($X > umbral$), se seleccionan las celdas de ese intervalo en las que haya objetos. Si hay varios intervalos contiguos que superan el umbral, estos se unen. Este proceso se repite para cada dimensión. Y en el segundo paso, CLIQUE utiliza las celdas densas para formar clusters, permitiendo de esta manera detectar aquellos con formas arbitrarias.

Algunas de las principales ventajas que presenta es que no es sensible al orden de entrada de los datos, escala linealmente con el tamaño del conjunto de datos y automáticamente encuentra aquellos subespacios con tantos atributos (dimensiones) posibles en los que hay clusters de alta densidad.

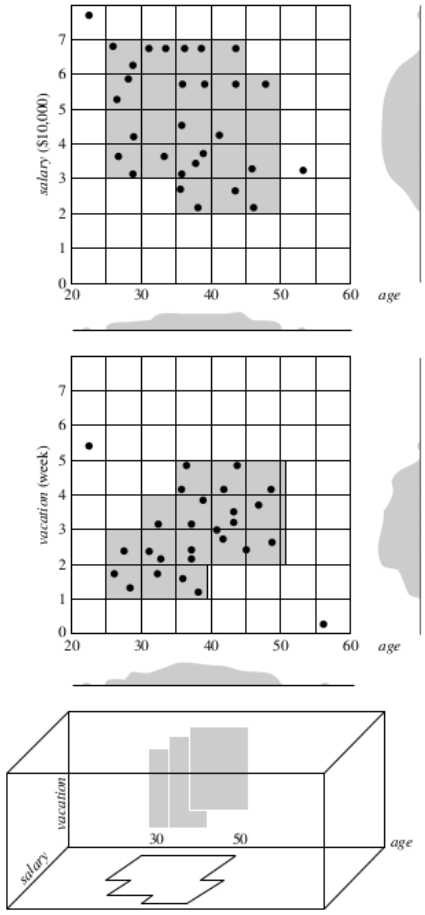


Figura 13: CLIQUE: las celdas densas encontradas con respecto a *edad* (*age*) para las dimensiones *salario* (*salary*) y *vacaciones* (*vacation*) se intersectan para formar un espacio de búsqeda de unidades densas de mayor dimensionalidad [16].

8 Tipos de datos

Otro de los factores más importantes en el proceso de clusterización es conocer el tipo de dato con el que se esté trabajando, pues esto influye en gran medida a la hora de seleccionar el algoritmo de clusterización a aplicar. Muchos de los algoritmos originales de clustering fueron diseñados con la idea de tratar únicamente con datos numéricos [35], pero como se ha mencionado previamente, el imparable desarrollo de las tecnologías de la información y la comunicación ha desembocado en la generación de otro tipos de datos como imágenes, documentos, audios, vídeos, etc.

Esto a su vez ha provocado el aumento de datos estructurados, semiestructurados y no estructurados. Los datos estructurados son todos aquellos que se suelen encontrar en bases de datos, donde cada campo tiene su descripción, restricciones y relaciones con otros campos y estructuras. Los datos semisupervisados provienen de información estructurada pero que no está representada con modelos de datos estructurados y bases de datos relacionales. Y por último, los datos no estructurados no guardan ningún tipo de formato ni está organizada de una forma estructurada. Estos dos últimos tipos de datos dificultan su procesamiento y hacen más costoso obtener información de ellos [14].

Los diferentes categorías de datos con los que se suele trabajar de forma común dentro del mundo de la clusterización son las siguientes:

- **Datos categóricos**: Se consideran datos categóricos todos aquellos que pueden ser asignadas en un número finito de categorías, tales como sexo, edad, etnia, etc. Estos son muy comunes en conjuntos de datos reales, donde además es probable que estén mezclados con otro tipo de datos numéricos. Un ejemplo típico de este tipo de datos se da en encuestas en las que se te da a elegir entre un número limitado de opciones (*p. ej.* Tipo de educación: pública o privada).

  Clusterizar este tipo de datos ha supuesto numerosos problemas puesto que las medidas comunes de similitud quedan obsoletas, haciendo necesario definir nuevos criterios para el cálculo de la semejanza; y por otro lado, al no tratar datos numéricos, métodos que utilizan la media o mediana deben ser modificados apropiadamente para estos datos discretos. Si a esto le sumamos que puede haber atributos mezclados en el conjuntos de datos, aumentamos la complejidad del problema.

- **Datos textuales**: Los datos de texto o documentos son un tipo de dato que siempre ha sido común, pero hasta hace pocos años predominaba en formato físico. Con el desarrollo de editores de texto, la informatización de los trabajos y el todavía creciente mundo de Internet y las redes sociales estos datos abundan en su forma digital.

  Normalmente estos documentos se representan como un amalgama de palabras, donde cada una de ellas se considera un atributo, sin tener en cuenta el orden de aparición. Esto implica que la cantidad de atributos por cada documento es muy alta, que es el principal

obstáculo que presenta la clusterización de estos datos, así como la ambigüedad de las palabras.

- **Datos multimedia**: De la misma manera, las redes sociales e Internet han favorecido la creación de contenido multimedia que se compone de audio, imágenes y vídeos principalmente, que normalmente se encuentra acompañado de otros tipos de datos.

  Aplicar clusterización en este tipo de datos se ha convertido en una de las herramientas más eficaces para la obtención de información en este ámbito, pero también tiene sus complicaciones, pues en el fondo, todos ellos son tipos de datos diferentes que necesitan ser procesados de forma distinta.

- **Datos de emisiones continuas**: Este tipo de datos, también denominados datos de "streaming"[19] puede considerarse como un subconjunto de datos multimedia, pues estos datos suelen ser audio, vídeos o animaciones pero que se retransmiten en tiempo real. En los últimos años este tipo de datos se ha visto fuertemente incrementado gracias a plataformas como YouTube o Twitch[20] y diversas redes sociales que permiten emitir en directo a más personas.

  Los principales problemas que presentan este tipo de datos son el gran volumen de datos y su retransmisión en tiempo real. Esto implica que, métodos vistos en los que se necesitan varias iteraciones para clusterizar los datos no sirven, puesto que es necesario hacerlo de una sola pasada. Esto implica que la velocidad de los algoritmos debe ser elevada y con poca complejidad. Asimismo, los patrones de los datos cambian constantemente, obligando a actualizar la estructura subyacente de los mismos.

- **Series temporales**: Los datos temporales o cronológicos son obtenidos en mediciones realizadas en momentos concretos y que están ordenadas de forma secuencial, según el tiempo en el que se realizó la prueba. Algunos ejemplos sencillos son muestras de temperatura que se recogen en intervalos de tiempo constante, recuento del número de afectados por una enfermedad o incluso mediciones del control de peso de una persona.

  Entre los ejemplos planteados se puede diferenciar entre datos divididos en intervalos constantes o dispersos y suelen contener un tipo de valor asociado con un atributo contextual: el tiempo.

  La aplicación de clustering en ámbitos con este tipo de datos ayuda con la detección de entidades con tendencias similares, aunque también existen complicaciones, puesto que es difícil definir la similitud

[19] Este término inglés es cada vez más utilizado de forma coloquial sobre todo entre los jóvenes para referirse a retransmisiones en vivo principalmente emitidas en Internet o redes sociales.

[20] Plataforma que permite realizar transmisiones en vivo propiedad de Amazon, siendo una de las principales fuentes de retransmisión de videojuegos, compitiendo con YouTube

entre distintas secuencias temporales debido al hecho de que esto puede suponer grandes cambios conforme el tiempo avanza.

- **Secuencias discretas**: Cada vez más son los ámbitos que generan secuencias discretas en vez de categóricas, y que en vez de diferenciarse por el contexto temporal, lo hacen por la posición o emplazamiento. Estos datos tiene una estructura normalmente lineal y algunos ejemplos de este tipo de datos son los registros de acceso a Internet o las secuencias de comandos de un ordenador.

  Al contrario que datos numéricos, este tipo de datos requiere de un alto nivel computacional para su procesamiento, puesto que es complicado encontrar representantes de los datos.

- **Datos biológicos**: Los datos biológicos son el principal tipo de dato de las secuencias discretas y se centran sobretodo en datos referentes al genoma y proteínas. Clustering se aplica en este caso con la intención de agrupar las secuencias biológicas que están relacionadas para así poder comprender mejor el funcionamiento del genoma y las tareas que realizan las células.

- **Redes y grafos**: Este tipo de datos son de los más comunes entre las representaciones de datos, puesto que virtualmente cada tipo de dato puede ser representado como un grafo de similitud con aristas ponderadas en función de la semejanza.

  La clusterización de estos datos es una de las formas más fiables e importantes a la hora de obtener información valiosa los grafos y redes, y muchos otros problemas pueden ser afrontados si se transforman en este tipo de datos y aplicando métodos de clusterización, sirviendo como una forma de abstracción de los datos.

- **Datos difusos**: Muchos conjuntos de datos suelen tener cierto grado de incertidumbre en los mismos, y esto puede ser medido y recolectado para mejorar los resultados de los algoritmos de Data Mining. Esto se debe al hecho de que la incertidumbre de estos datos ofrece una medida probabilista de la importancia que tiene los atributos, haciendo más efectivo su procesamiento.

  Por lo tanto, es una práctica común disminuir la calidad de los datos para utilizar algoritmos que se benefician de lo mencionado previamente o simplemente utilizar herramientas de medidas imprecisas. Este puede ser el caso de sensores donde la imprecisión puede medirse de antemano o datos obtenidos de previsiones donde siempre hay cierta incertidumbre que puede ser inferida.

  Uno de los problemas que presentan estos datos ese que la incertidumbre se puede confundir con la de Fuzzy clustering. En la clusterización relajada la incertidumbre proviene del diseño de los métodos, mientras que en los datos difusos esta es inherente a los mismos.

En la Tabla 2 se han comparado los diferentes métodos que se han expuesto

en este documento en función del tipo de dato para el que son utilizados según [14, 35]. Como se puede ver, los métodos probabilísticos soportan todos los tipos de datos debido a su gran abstracción, pues al basarse en la estructura subyacente de los mismos, puede aplicarse de más formas. El siguiente tipo de algoritmos que más datos puede procesar son los basados en particiones, puesto que su gran sencillez permite modificarlos fácilmente y adaptarlos a las necesidades pertinentes.

Métodos /Tipo de dato	Particiones	Jerárquicos	Densidad	Rejilla	Fuzzy	Probabilísticos	Grafos	Redes nuronales
Categórico	✓	✓	✓	✓	✓	✓		
Texto	✓	✓	✓		✓	✓	✓	✓
Multimedia	✓	✓				✓	✓	
Streaming	✓		✓	✓		✓	✓	
Temporales	✓	✓	✓	✓		✓		
Discretos	✓					✓		
Biológicos	✓	✓		✓		✓	✓	✓
Red		✓			✓	✓	✓	✓
Difusos	✓		✓	✓		✓		

Cuadro 2: Comparativa entre los métodos de clusterización y los tipos de datos con los que suelen trabajar.

9 Evaluación del clustering

Las técnicas y algoritmos que se han expuesto son tan solo los más populares, una pequeña parte de todos los que existen, pues existen múltiples variaciones, nuevas versiones y combinaciones de cada uno de ellos enfocados en mejorar el rendimiento, cubrir flaquezas y permitir nuevas aplicaciones y funcionalidades.

Sin embargo, ¿cómo sabemos que el resultado de una clusterización es preciso y correcto? Es necesario evaluar la clasificación final de un proceso de clustering y y los pasos iniciales para cerciorarnos de que el método utilizado y los valores asignados a las variables pertinentes son los adecuados. Gracias a esto, se pueden determinar los puntos fuertes de los algoritmos e identificar aquellas condiciones, valores y tipos de datos bajo los que ofrece mejores resultados.

Algunas de las comprobaciones que se realizan para evaluar el proceso de clustering son las siguientes:

- **Evaluar la tendencia de la clasificación**: Intentar determinar si existe una estructura no aleatoria en la distribución de los datos antes de custerizarlos es crucial para obtener buenos resultados. Aplicar un método de clustering a un conjunto de datos aleatorio no devolverá resultados útiles y con significado, pues los clusters que se pueden for-

mar serán también aleatorios y por lo tanto no aportarán información. Para esto, se suele usar un procedimiento similar al de los métodos jerárquicos probabilísticos 5.6 en el que se calcula la probabilidad de que el conjunto de datos con el que se quiere trabajar haya sido generado por una distribución de datos uniforme. Uno de los procedimientos más comunes es la Estadística de Hopkins.

- **Determinar el número de clusters**: En algunos algoritmos como k-means 4.1 el número final de clusters es solicitado al usuario y por lo tanto debe aproximarse de antemano. Sin embargo, es una buena práctica calcular este valor antes de utilizar cualquier método para contrastar el resultado final. Esta tarea aunque ciertamente complicada, pues los factores de los que depende son muchos (distribución de los datos, número de atributos, cantidad de datos...) y para la que existen numerosas formas de calcularla, puede ser simplificada para un mayor entendimiento o fácil aplicación. Uno de los métodos más sencillos es aproximar el número de clusters a $\sqrt{\frac{n}{2}}$, donde en cada cluster debería haber $\sqrt{2 \cdot n}$ objetos, para un conjunto de n datos. Otro métodos es Elbow Method, en el que se repite de forma iterativa el proceso de clusterización variando el valor de k (número de clusters) y calculando la varianza de los puntos dentro de los clusters. Una vez termina el proceso, se muestran los resultados en un gráfico y se escoge como valor para k el punto de la gráfica en el que esta se curva de forma severa tal y como se puede observar en la Figura 14, donde el valor escogido para k es tres. Este proceso también es utilizado para determinar más parámetros dentro de otros algoritmos como *Epsilon* pada DBSCAN.

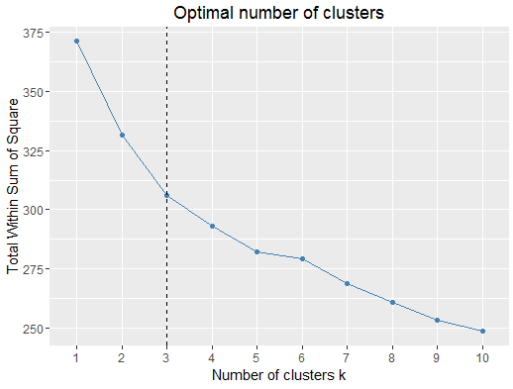


Figura 14: Elbow Method para encontrar el número de clusters óptimo [34].

Sin embargo, la forma más importante de evaluar la clusterización es comprobar el resultado final, la clasificación de los datos.

9.1 Calcular la calidad de la clusterización

Una vez los datos han sido clasificados, se debe verificar que el resultado es bueno. Para ello se utilzan métodos que miden la calidad de los clusters. Estos se dividen principalmente en dos grupos dependiendo de si se usa información externa para la validacióno no. Los métodos intrínsecos o internos comprueban si los clusters acogen bien los datos, mientras que los métodos extrínsecos o externos comprueban si los clusters se aproximan a una realidad existente que suele ser una clasificación ideal realizada por expertos en la materia o información externa que no está presente en los datos.

Ambos tipos de validaciones son curciales para muchos escenario. Dado que los métodos externos conocen el "verdadero" número de clusters de antemano, se pueden usar para determinar qué algoritmo es el más óptimo para determinados conjuntos de datos. Por ejemplo, si los métodos extrínsecos muestran que un algoritmo de clusterización de documentos genera una clasificación similar a una realizada por expertos, se puede afirmar que dicho algoritmo tiene gran impacto en el análisis de textos o documentos. Asimismo, la validación interna también permite escoger el mejor algoritmo y el número de clusters óptimos sin ningún tipo de información adicional. Sin embargo, es frecuente que no sea posible aplicar métodos de validación extrínsecos por la falta de información extra sobre el conjunto de datos, haciendo que la única forma posible de medir la calidad sea mediante métodos intrínsecos [35].

Veamos algunas de las principales formas de validación entre ámbos métodos:

9.1.1 Métodos extrínsecos

En aquellos casos en los que se disponga de información externa con la que comparar el resultado de la clusterización podemos usar métodos extrínsecos. Para afirmar si la clusterización ha sido efectiva, los métodos extrínsecos utilizan los siguientes criterios para su evaluación [16]:

- **Homogeneídad del cluster**: Cuanto más puros sean los clusters, mejor es la clusterización. Por ejemplo, si tenemos un conjunto de datos del que sabemos que extisten varias categorías gracias a la información externa utilizada, un algoritmo, A_1, que genere un cluster C_1 en el que hay datos de más de una categoria será puntuado de peor forma que otro algoritmo, A_2, que, en vez de generar un solo cluster C_1, genere diferentes subclusters en los que agrupe aquellos datos de la misma categoría, haciendo que en cada grupo solo exista un tipo de dato.
- **Integridad del cluster**: Si dos objetos pertenecen a la misma categoría, deben estar en el mismo cluster. De una forma similar al criterio anterior, si un algoritmo A_1 genera un solo cluster en el que agrupa los datos de la misma categoría C_1 mientras que otro, A_2, divide C_1 en

dos clusters diferentes, ambos teniendo datos de la misma categoría, el primer algoritmo A_1 recibirá una mejor puntuación.

- **Misceláneo**: En muchos escenarios se genera una categoriá de datos denominada "misceláneo", "varios", "otros", etc., donde se agrupan aquellos objetos que no se pueden fusionar o unir con otros. Se penalizará a aquellos algoritmos que en vez de añadir un objeto heterogéneo a al conjunto de "misceláneo" lo agrupe dentro de un cluster.
- **Preservación de clusters pequeños**: Si una categoría que contiene pocos datos es dividida en muchas partes durante la clusterización es probable que acabe convirtiéndose en ruido y por lo tanto, poco probable que se acabe descubriendo. Se dará más puntuación a aquellos algoritmos que en vez de separar en diferentes clusters los objetos de la categoría con pocos datos, separen en clusters categorías con más datos para permitir así que los objetos de la categoría pequeña estén en el mismo cluster.

La mayoría de métodos no inspeccionan todos los criterios sino que suelen explorar uno o dos, aunque sí que existen aquellos que implementan los cuatro. En la Figura 15 se muestran algunos de los métodos extrínsecos más utilizados para el algoritmo k-means 4.1,

	Measure	Definition	Range
1	Entropy (E)	$-\sum_i p_i(\sum_j \frac{p_{ij}}{p_i}\log\frac{p_{ij}}{p_i})$	$[0,\log K']$
2	Purity (P)	$\sum_i p_i(\max_j \frac{p_{ij}}{p_i})$	$(0,1]$
3	F-measure (F)	$\sum_j p_j \max_i[2\frac{p_{ij}}{p_i}\frac{p_{ij}}{p_j}/(\frac{p_{ij}}{p_i}+\frac{p_{ij}}{p_j})]$	$(0,1]$
4	Variation of Information (VI)	$-\sum_i p_i\log p_i-\sum_j p_j\log p_j-2\sum_i\sum_j p_{ij}\log\frac{p_{ij}}{p_ip_j}$	$[0,2\log\max(K,K')]$
5	Mutual Information (MI)	$\sum_i\sum_j p_{ij}\log\frac{p_{ij}}{p_ip_j}$	$(0,\log K']$
6	Rand statistic (R)	$[\binom{n}{2}-\sum_i\binom{n_i}{2}-\sum_j\binom{n_{\cdot j}}{2}+2\sum_{ij}\binom{n_{ij}}{2}]/\binom{n}{2}$	$(0,1]$
7	Jaccard coefficient (J)	$\sum_{ij}\binom{n_{ij}}{2}/[\sum_i\binom{n_i}{2}+\sum_j\binom{n_{\cdot j}}{2}-\sum_{ij}\binom{n_{ij}}{2}]$	$[0,1]$
8	Fowlkes & Mallows index (FM)	$\sum_{ij}\binom{n_{ij}}{2}/\sqrt{\sum_i\binom{n_i}{2}\sum_j\binom{n_{\cdot j}}{2}}$	$[0,1]$
9	Hubert Γ statistic I (Γ)	$\frac{\binom{n}{2}\sum_{ij}\binom{n_{ij}}{2}-\sum_i\binom{n_i}{2}\sum_j\binom{n_{\cdot j}}{2}}{\sqrt{\sum_i\binom{n_i}{2}\sum_j\binom{n_{\cdot j}}{2}[\binom{n}{2}-\sum_i\binom{n_i}{2}][\binom{n}{2}-\sum_j\binom{n_{\cdot j}}{2}]}}$	(-1,1]
10	Hubert Γ statistic II (Γ')	$[\binom{n}{2}-2\sum_i\binom{n_i}{2}-2\sum_j\binom{n_{\cdot j}}{2}+4\sum_{ij}\binom{n_{ij}}{2}]/\binom{n}{2}$	$[0,1]$
11	Minkowski score (MS)	$\sqrt{\sum_i\binom{n_i}{2}+\sum_j\binom{n_{\cdot j}}{2}-2\sum_{ij}\binom{n_{ij}}{2}}/\sqrt{\sum_j\binom{n_{\cdot j}}{2}}$	$[0,+\infty)$
12	classification error (ε)	$1-\frac{1}{n}\max_\sigma\sum_j n_{\sigma(j),j}$	$[0,1)$
13	van Dongen criterion (VD)	$(2n-\sum_i\max_j n_{ij}-\sum_j\max_i n_{ij})/2n$	$[0,1)$
14	micro-average precision (MAP)	$\sum_i p_i(\max_j\frac{p_{ij}}{p_i})$	$(0,1]$
15	Goodman-Kruskal coeff (GK)	$\sum_i p_i(1-\max_j\frac{p_{ij}}{p_i})$	$[0,1)$
16	Mirkin metric (M)	$\sum_i n_{i\cdot}^2+\sum_j n_{\cdot j}^2-2\sum_i\sum_j n_{ij}^2$	$[0,2\binom{n}{2})$

Note: $p_{ij}=n_{ij}/n,\ p_i=n_{i\cdot}/n,\ p_j=n_{\cdot j}/n.$

Figura 15: Medidas externas para la validación de clusters [35].

9.1.2 Métodos intrínsecos

Como se ha mencionado previamente, el método de clusterización se aplica cuando no se dispone de etiquetas o información por la que clasificar dichos datos de una forma supervisada, por lo que disponer de información extra para la realización de una validación externa sucede en raras ocasiones. De

esta manera, los métodos intrínsecos, también denominados no supervisados, evalúan la calidad de la clasificación examinando cuán compactos son los clusters y lo bien que están separados.

Para calificar lo compactos que son los clusters se suele utiliza la varianza, donde un menor resultado de esta medidad indica una mejor clusterización. Por otro lado, para medir la separación entre clusters se suelen utilizar cálculos de distancias o densidad.

El proceso mediante el cuá se suele determinar la mejor partición y el número óptimo de clusters mediante validación interna es el siguiente [35]:

1. Seleccionar el conjunto de algoritmos que van a ser aplicados al conjunto de datos.
2. Para cada algroritmo utilizar diferentes combinaciones de los parámetros para obtener diferentes resultados finales.
3. Calcular utilizando el método de validación interna requerido los resultados del paso anterior.
4. Escoger en función de los valores obtenidos la mejor partición y el número de clusters adecuado.

En la Figura 16 se muestran los 12 métodos de cálculo de calidad intrínsecos de entre los cuales cabe destacar el coeficiente de silueta (silhouette coefficient).

Este supone que, dado un conjunto de datos D con n objetos, este está dividido en k clusters, $C_1, \ldots, C_k$. Para cada objeto perteneciente al conjunto de datos $o \in D$ calculamos la distancia media entre dicho objeto o y todos los otros objetos dentro del cluster al que pertenece o, con lo que obtenemos una representación de lo compacto que es el cluster al que pertenece o. Por otro lado, también calculamos la distancia media desde o al resto de clusters a los que este no pertenece y selecionamos la más pequeña, como forma de observar cuánta separación hay entre el objeto o y el resto de clusters. Esto da como resultado la ecuación mostrada en la Figura 16, cuyo valor oscila entre -1 y 1. Cuando el valor es 1, quiere decir que el cluster de o es compacto y que está separado de otros clusters, indicando un buen resultado de clusterización [16].

10 Aplicaciones de clustering

Como se ha mencionado previamente en la Sección 2, la clusterización ha probado ser uno de los métodos más eficaces a la hora de obtener información de los datos mediante su clasificación y etiquetado por similitud, motivo por lo que es empleado de manera recurrente en muchos de los campos más influyentes e importantes del mundo moderno tales como ingeniería, ciencias de la computación, ciencias de la salud, ciencias sociales o economía.

	Measure	Definition
1	*RMSSTD*[1]	$\{\sum_i \sum_{x\in C_i} \| x-c_i \|^2 / [P\sum_i (n_i-1)]\}^{\frac{1}{2}}$
2	R-squared (*RS*)	$(\sum_{x\in D} \| x-c \|^2 - \sum_i \sum_{x\in C_i} \| x-c_i \|^2) / \sum_{x\in D} \| x-c \|^2$
3	Modified Hubert Γ statistic (Γ)	$\frac{2}{n(n-1)} \sum_{x\in D} \sum_{y\in D} d(x,y) d_{x\in C_i, y\in C_j}(c_i, c_j)$
4	Calinski-Harabasz index (*CH*)	$\frac{\sum_i n_i d^2(c_i,c)/(NC-1)}{\sum_i \sum_{x\in C_i} d^2(x,c_i)/(n-NC)}$
5	*I* index (*I*)	$(\frac{1}{NC} \cdot \frac{\sum_{x\in D} d(x,c)}{\sum_i \sum_{x\in C_i} d(x,c_i)} \cdot \max_{i,j} d(c_i,c_j))^p$
6	Dunn's indices (*D*)	$\min_i \{\min_j (\frac{\min_{x\in C_i, y\in C_j} d(x,y)}{\max_k \{\max_{x,y\in C_k} d(x,y)\}})\}$
7	Silhouette index (*S*)	$\frac{1}{NC} \sum_i \{\frac{1}{n_i} \sum_{x\in C_i} \frac{b(x)-a(x)}{\max[b(x),a(x)]}\}$ $a(x) = \frac{1}{n_i - 1} \sum_{y\in C_i, y\neq x} d(x,y), b(x) = \min_{j, j\neq i} [\frac{1}{n_j} \sum_{y\in C_j} d(x,y)]$
8	Davies-Bouldin index (*DB*)	$\frac{1}{NC} \sum_i \max_{j, j\neq i} \{[\frac{1}{n_i} \sum_{x\in C_i} d(x,c_i) + \frac{1}{n_j} \sum_{x\in C_j} d(x,c_j)] / d(c_i,c_j)\}$
9	Xie-Beni index (*XB*)	$[\sum_i \sum_{x\in C_i} d^2(x,c_i)] / [n \cdot min_{i,j\neq i} d^2(c_i,c_j)]$
10	SD validity index (*SD*)	$Dis(NC_{max}) Scat(NC) + Dis(NC)$ $Scat(NC) = \frac{1}{NC} \sum_i \| \sigma(C_i) \| / \| \sigma(D) \|$ $Dis(NC) = \frac{max_{i,j} d(c_i,c_j)}{min_{i,j} d(c_i,c_j)} \sum_i (\sum_j d(c_i,c_j))^{-1}$
11	S_Dbw validity index (*S_Dbw*)	$Scat(NC) + Dens\_bw(NC)$ $Dens\_bw(NC) = \frac{1}{NC(NC-1)} \sum_i [\sum_{j, j\neq i} \frac{\sum_{x\in C_i \cup C_j} f(x,u_{ij})}{max\{\sum_{x\in C_i} f(x,c_i), \sum_{x\in C_j} f(x,c_j)\}}]$
12	*CVNN*[2] index	$Sep(NC,k) / \max_{NC} Sep(NC,k) + Com(NC) / \max_{NC} Com(NC)$ $Com(NC) = \sum_i [\frac{2}{n_i \cdot (n_i - 1)} \Sigma_{x,y\in C_i} d(x,y)]$ $Sep(NC,k) = \max_i (\frac{1}{n_i} \sum_{j=1,2,\ldots,n_i} \frac{q_j}{k})$

Note: D: data set; n: number of objects in D; c: center of D; P: attributes number of D;
NC: number of clusters; C_i: the ith cluster; n_i: number of objects in C_i; c_i: center of C_i;
k: number of nearest neighbors; q_j: number of C_i's jth object's nearest neighbors which are not in cluster C_i;
$\sigma(C_i)$: variance vector of C_i; $d(x,y)$: distance between x and y; $\| X_i \| = (X_i^T \cdot X_i)^{1/2}$.

Figura 16: Medidas internas para la validación de clusters [35].

Durante este documento se han nombrado algunos de los dominios de aplicación más comunes en los que clustering ha demostrado ser una técnica súmamante útil. Ahora vamos a ampliar dicha información con la encontrada en [35]. Dichos dominios son los siguientes:

- **Paso intermedio para otros métodos de data mining**: El proceso de clusterización sirve tanto para etiquetar y clasificar un conjunto de datos como para sumarizar la información de los mismos, haciendo que sea un paso intermedio clave para otro tipo de métodos como clasificación supervisada[21] o detección de outliers.
- **Método de filtrado colaborativo**: Este tipo de filtros en los que múltiples usuarios puntúan o evalúan algo se vale de la clusterización para encontrar aquellos usuarios con respuestas similares, permitiendo utilizar esta información en gran variedad de formas, sobretodo en

[21] Método de Data Mining que define un modelo de clasificación que permite obtener el valor desconocido de un suceso elemental a partir del resto de valores dado un conjunto de datos sobre el que conocemos todos los posibles valores que puede tomar dicho valor desconocido.

sistemas recomendadores.

- **Segmentación de clientes**: Esta aplicación es una variación de los métodos de filtro colaborativos específica para clientes, que se agrupan por similitud de preferencias o gustos. La principal diferencia con el anterior es que en vez de usar las evaluaciones de los usuarios, aquí se utilizan atributos arbitrarios de los consumidores [13].
- **Resumir datos**: Muchas técnicas de clusterización están relacionadas con métodos de reducción de dimensiones en los datos, es decir, eliminar los atributos que no sean relevantes. Esto permite obtener cierta forma de resumen del conjunto de datos, que es muy útil a la hora de crear representaciones compactas de los datos, haciendo que el procesamiento e interpretamiento de los mismos sea más sencillo.
- **Detección de tendencias dinámicas**: Los métodos de clusterización dinámicos y fluidos son usados para detectar la dirección de los datos y encontrar patrones de cambio en ellos.
- **Análisis de datos multi media**: Los datos multiemedia están compuestos por imágenes, vídeos y audios. Clustering se aplica en este tipo de datos muchas veces con intención de detectar segmentos similares para la clasificación de tipos de música, detección de infracciones de derechos de autor, etc.
- **Análisis de redes sociales**: El continuo crecimiento de las redes sociales ha dado mucha importancia a este ámbito en el que clustering se utiliza para detectar estructuras sociales o comunidades subyacentes dentro de los datos. Encontrar comunidades sirve para entender mejor cómo funciona la red social y para recomendar a usuarios de una comunidad otras similares a sus gustos [11].
- **Análisis de datos biológicos**: Los datos biológicos se han convertido en datos importantes gracias al éxito de los análisis del genoma humano[22] y la capacidad de recolectar diferentes tipos de expresión génica[23]. Este tipo de datos suelen estructurarse como secuencias o redes, y aplicando clustering podemos detectar secuaencias inusuales y su tendencia.

Como se ha podido observar durante este documento, la clusterización ofrece un gran abanico de opciones y técnicas específicas que permiten su adaptación a casi cualquier área del conocimiento del mundo moderno, motivo por el que es uno de los métodos más utilizados a la hora de obtener información a partir de datos. Sin embargo, si tuviéramos que destacar un entorno en el que la aplicación de clustering puede ser súmamente provechosa, es en los documentos de texto.

[22] El genoma es el conjunto de genes contenidos en los cromosomas.

[23] La expresión génica es el proceso mediante el cual todos los organismos transforman la información codificada por los ácidos nucleicos en las proteinas necesarias para su desarrollo, fincionamiento y reproducción con otros organismos.

11 Clusterización de textos

En esta época dominada por la incesante cantidad de datos que se generan a diario, se estima que alrededor del ochenta por ciento de la información se encuentra en textos [36], bien en forma de noticias, artículos, libros, mensajes de correo, blogs[24], páginas web, etc. Esto hace que sea humanamente imposible el proceso manual de toda esa información, por lo que la utilización de computadores es algo imprescindible.

Por ello, surgió la Minería de Textos o Text Mining, una disciplina enfocada en la utilización de ordenadores para el descubrimiento de nueva información previamente desconocida mediante la automatización de la extracción de información de diferentes fuentes [37]. Esta es una variación de Data Mining, cuya diferencia principal es el tipo de texto con el que se trabaja, ya que la minería de textos se centra en las palabras y frases que forman parte de los documentos de texto, una fuente de datos no estructurada y no numérica. Por lo que el problema principal que esto representa es el hecho de que el lenguaje natural surgió como forma de comunicación entre seres humanos.

Como solución a este problema, se han desarrollado técnicas de Procesamiento de Lenguaje Natural (NLP), un área de estudio y aplicación dentro de la Inteligencia Artificial cuya meta principal es el estudio del lenguaje humano de manera que las máquinas puedan comprenderlo, tratarlo y manipularlo de una forma similar a como lo hace el ser humano [38]. Las principales técnicas que se han desarrollado en este campo son:

- **Extracción de la información (IE)**: Uno de los puntos de partida a la hora de analizar datos no estructurados es la extracción de información, que identifica palabras y frases clave y sus relaciones dentro de un texto. Una vez obtenida esta información, se pueden realizar técnicas de Data Mining que de otra forma no serían posibles [37, 38, 39]
- **Control de temática**: Este tipo de sistemas permiten a los usuarios seleccionar palabras claves y recibir alertas o recomendaciones de textos con temáticas similares y de sus gustos. Esto puede ser utilizado para alertar a compañías de nuevos competidores, nuevas ofertas laborales, descubrimientos recientes o simplemente noticias relacionadas [37].
- **Creación de resúmenes**: A la hora de comprobar si un documento cumple con las necesidades que buscan los usuarios, la creación de resúmenes es crucial. Mediante la identificación de palabras y frases claves, este proceso procesa y resume documentos en el tiempo que toma al lector leer el primer párrafo. Primero se obtiene una representación estructurada del texto original, posteriormente, un algoritmo transforma la estructura de texto en estructura de resumen y por último, el resumen es obtenido de la estructura anterior [37].

[24] Sitio web que se asemeja a un diario personal de su autor y donde escribe contenido de su interés sobre el que incita a sus lectores a comentar.

- **Clasificación de documentos**: La clasificación o categorización consiste en identificar los temas principales de un texto y posicionarlo de acuerdo a un conjunto de tópicos preestablecidos. Esto se suele hacer mediante métodos de aprendizaje supervisados, realizando la clasificación en base a ejemplos conocidos. Este proceso permite la aplicación de posteriores técnicas de Data Mining [37].
- **Vinculación de conceptos**: Las herramientas de vinculación de conceptos permiten conectar textos que estén relacionados identificando aquellos elementos que tienen en común ayudando a los usuarios a encontrar nueva información que quizás no hubieran encontrado por métodos normales. Este es habitual en áreas como la biología y disciplinas de la ciencia donde es difícil encontrar todas las publicaciones sobre un tema en concreto [37].
- **Visualización de la información**: Este proceso permite mostrar textos de forma visual en una estructura jerárquica o mapa, ofreciendo posibilidades de búsqueda y navegación en las que el usuario puede interactuar. Esto es útil cuando se necesita minimizar el área de búsqueda dentro de un gran abanico de opciones [37].
- **Responder preguntas**: Otra de las aplicaciones del procesamiento de lenguaje natural es responder preguntas, en las que intenta ofrecer la mejor réplica a una consulta realizada por un usuario. Esto es utilizado por diversas páginas web e instituciones para sus apartados de preguntas frecuentes (FAQ) [37].
- **Búsqueda y recuperación de información (IR)**: La búsqueda y recuperación de información es una tarea que se suele dar principalmente en motores de búsqueda donde, en función de ciertas palabras clave, se devuelve un conjunto de resultados relevantes que guarden relación con ellas. En la minería de textos se trabaja con documentos, por lo que esto sirve principalmente como forma de calcular la similitud entre documentos [39].
- **Análisis de sentimiento**: Este análisis consiste en la detección de ciertas palabras clave que permiten determinar la opinión de los usuarios con respecto a cierto tema. Estas palabras suelen estar preestablecidas y se dividen en palabras favorables o desfavorables. Al precisar de un número elevado de reseñas o comentarios sobre el tema para que el resultado sea certero, este método es realmente útil en redes sociales [39].
- **Representación del conocimiento**: Esta área de la Inteligencia Artificial se centra en la representación, mantenimiento y manipulación del conocimiento sobre un dominio de aplicación. En relación a la Minería de Textos, esto se refleja en la creación de formas intermedias que permiten representar el contenido semántico de un texto [38].

Y la otra técnica principal que falta en la lista anterior es la clusterización de textos. El funcionamiento de este proceso es el mismo que se ha comen-

tado al inicio de este texto en la Sección 2, y permite agrupar documentos similares basándose en la examinación de sus palabras y frases.

Las principales aplicaciones que recibe clustering dentro del ámbito textual son las siguientes [40]:

- **Organización y navegación de documentos**: Mayoritariamente los algoritmos de clusterización jerárquicos, aunque no son los únicos, ofrecen la posibilidad de organizar documentos y textos en categorías coherentes y similares que facilitan su búsqueda y navegación.
- **Sintetización de corpus**[25]: Las técnicas de clustering proveen un resumen congruente sobre una colección de textos en forma de clusters de palabras como se puede apreciar en la Figura 17, donde se ha realizado este mismo proceso sobre el conjunto de títulos de artículos de la muestra de datos sobre COVID-19 de la que hablaremos más adelante.
- **Clasificación de documentos**: Aunque clustering sea un método de clasificación no supervisado, es realmente útil a la hora de organizar documentos por su contenido, identificando en el mismo proceso aquellos que sean similares y situándolos en el mismo grupo, permitiendo así su clasificación.

Gran parte de los algoritmos de clustering tienen un carácter general y permiten trabajar con más de un tipo de datos tal y como se vio en la Tabla 2, pero su empleo sobre los datos textuales presenta un conjunto de dificultades específicas y que requieren de un procesamiento de los datos previo a su aplicación.

Uno de los problemas principales. y que se ha comentado previamente, es la alta dimensionalidad de los datos, puesto que se debe considerar cada palabra como un atributo único; y a su vez, la cantidad de información útil que hay en los textos es relativamente pequeña con respecto a su tamaño. Muchas de las palabras que se utilizan en un texto son meros conectores y palabras comunes que permiten aligerar y mejorar la experiencia del lector, pero son pocas las que realmente son importantes a la hora de discernir sobre qué versa dicho documento.

Otro problema es la relación entre las palabras que como humanos somos capaces de identificar pero que es una tarea ardua para las computadores y por lo que los algoritmos que se usan en este ámbito contienen funcionalidades que tienen en consideración este tipo de correlaciones. Y otra dificultad añadida, es que cada texto es diferente, y su número de palabras varía de forma considerable, por lo que normalizar los datos es una tarea imperativa previa a su procesamiento.

Es por estos motivos, y el hecho de que trabajamos con cadenas de texto y no con datos numéricos que muchas operaciones utilizadas en clustering

[25]Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación.

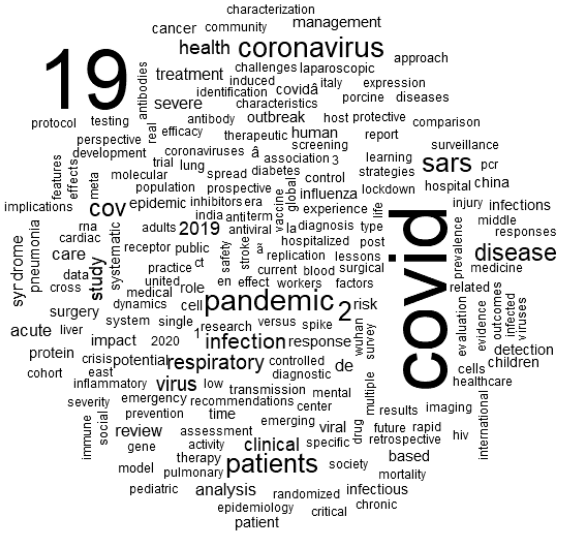


Figura 17: Cluster de palabras de los títulos de artículos científicos relacionados con COVID-19.

no pueden ser llevadas a cabo, siendo necesario transformar los datos de manera que se puedan aplicar los algoritmos que hemos visto.

11.1 Preparación de datos

Uno de los principales problemas a la hora de procesar datos de tipo texto, es la gran cantidad de atributos que estos contienen, por lo que una de las tareas principales antes de trabar con ellos es procesarlos de manera que podamos reducir dicha cantidad mediante la eliminación de todas aquellas palabras que no aportan significado o son demasiado comunes como para ser relevantes, así como la reducción de las palabras a su raiz o lexema. Estos dos procesos reciben los nombres de eliminación de *palabras vacías* y *normalización*.

- **Eliminación de palabras vacías**: Las palabras vacías o *stopwords* en inglés, son todas aquellas palabras cuya función principal sea aso-

ciar palabras o simplemente concatenar frases u oraciones. Normalmente estas son pronombres, conjunciones, signos de puntuación y demás tipos que no aportan ningún tipo de información dentro del texto y su análisis únicamente entorpecería el resultado final, puesto que su tasa de aparición en un texto es tan elevada, alrededor de un treinta o cuarenta por ciento del texto, que serían el único tipo de palabras que se analizarían, dando lugar a un resultado de clusterización o cualquier tipo de análisis de datos erróneo.

Algunas de estas palabras vacías en español son las siguientes: “a”, “además”, “casi”, “como”, “con”, “de”, “día”, “él”, “es”, “fue”, “más”, “mientras”, “muy”, etc.

- **normalización**: Se utilizan principalmente dos métodos para realizar este proceso que pretende eliminar la diversidad y variedad de palabras para resumirlas o reducirlas a su raiz o lexema: *stemming* y *lematización*. Eso nos permite reducir el espacio de términos, disminuyendo así la dimensionalidad del texto.
  - **Stemming**: Este método trata de reducir cada palabra a su raiz, un término que comparta con el resto de palabras de la misma familia o derivados, aunque a veces esto genera vocablos que no existen en el diccionario.
  - **Lematización**: Se denomina lematización al proceso que permite hayar el lema de una forma flexionada[26]. Este lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra, permitiendo así eliminar múltiples derivaciones de una misma palabra, como en el caso de las formas verbales. [41]

  Por ejemplo, considerando el siguiente grupo de palabras: “cantamos”, “cantó”,“cantaría” y “cantase”, terminaríamos con los términos: “cant”, “cant”,“cant” y “cant” si aplicamos stemming o “cantar”, “cantar”,“cantar” y “cantar” si aplicamos lematización. Si se busca una normalización rápida, stemming es la adecuada, pero el proceso de lematización es más preciso y conserva más el sentido de las palabras, por lo que es la más utilizada entre ambas.

Asimismo, los textos y documentos se suelen considerar un saco de palabras, pero normalmente su contenido se encuentra en forma de una sola cadena de texto, es decir, todo el contenido del documento es un único atributo que contiene todas las palabras. Para poder realizar operaciones con dichas palabras, es necesario separarlas individualmente mediante un proceso denominado *tokenización* (*tokenization*). Este consiste en la división de todo el texto en unidades independientes, denominadas *tokens*, utilizando como separadores los espacios en blancos y signos de puntuación. [38].

[26] La flexión es la alteración que experimtentan las palabras mediantes morfemas. Existe la flexión verbal y la nominal.

Estos tres pasos: eliminación de palabras vacías, normalización y tokenización son esenciales cuando se pretende trabajar con textos, y aunque su orden de aplicación varía en función del algoritmo o método que utilicemos, es común que la eliminación de palabras vacías vaya antes que la normalización. El motivo por el que el orden puede varíar es porque los métodos pueden trabajar con tokens o con texto plano, obligando a realizar la tokenización en primer o último lugar.

Sin embargo, esto no es una suficiente reducción de la dimensionalidad de los datos, por lo que es preciso aplicar métodos de reducción con el objetivo de acotar el número de atributos con el que se pretende trabajar al mínimo posible, escogiendo como palabras resultantes aquellas que aporten significado al texto y puedan considerarse clave a la hora de clasificarlos.

11.2 Técnicas de reducción de dimensionalidad

En el ámbito matemático, el problema de la reducción de dimensión puede ser descrito de la siguiente forma: dada una variable aleatoria de p-dimensiones $x = (x_i, \ldots, x_p)^T$, el objetivo es encontrar una representación con menor dimensión k, $s = (s_1, \ldots, s_k)^T$ donde $k < p$ y se preserve todo lo posible la información contenida en la variable original [44].

Dentro del ámbito de Text Mining existen dos tipos principales de técnicas: *Selección de Características* y *Transformación de Características*.

11.2.1 Selección de Características

La selección de características o *Feature Selection* es un proceso mediante el cual se escoge un subconjunto de los atributos originales de acuerdo a cierto criterio ordenando los términos en base a una variable numérica calculada a partir del conjunto de documentos y seleccionando aquellas palabras que sobrepasen el umbral. Los términos seleccionados mantienen gran parte de su valor original y ayudan a comprender mejor el conjunto de datos [45, 46].

Las características pueden considerarse de cuatro tipos: muy relevante, poco relevante, irrelevante y redundante. Todas aquellas palabras que se consideren muy relevantes no pueden ser eliminadas bajo ningún concepto. Las palabras relevantes no siempre son necesarias para un subconjunto óptimo, aunque depende de ciertas condiciones. Las palabras irrelevantes no tienen por qué incluirse en el subconjunto. Las palabras redundantes suelen ser palabras poco relevantes pero que pueden ser reemplazadas por otro conjunto de términos de manera que no afecte a la distribución original. De esta manera, el objetivo de este tipo de métodos es maximizar la relevancia del subconjunto y minimizar su redundancia [47].

A continuación se exponen algunos de los métodos más populares de selección de características.

- **Frecuencia de Documento (DF)**: Este método cuyo nombre en inglés es Document-Frequency, es una de las técnicas más sencillas y

que utiliza la frecuencia de aparición de las palabras como criterio. Es común ver este método reflejado en forma de matriz, que se denomina DTM (*Document-Term Matrix*) o *Matriz término-documento*. En dicha matriz cada fila representa un documento, cada columna un término y el valor de cada celda hace referencia al número de veces que aparece el término en ese documento. También se puede usar en formato binario, indicando si un término sencillamente está o no en un documento [40, 44, 45, 46].

A veces cuando se habla sobre este cambio a forma de matriz utilizando la frecuencia de las palabras se suele usar el término TF (*Term-Frequency*) o *Frecuencia de Término*, cuya estructura es idéntica a DTM, puesto que se basa en ella para el mismo concepto: contar el número de apariciones de un término en el documento. La principal diferencia es que DTM utiliza valores absolutos mientras que TF usa los relativos; es decir, divide la frecuencia de la palabra entre el número total de palabras en el documento. Esto permite aliviar o incluso eliminar el inconveniente de que en textos más largos es más probable que aparezca más veces la misma palabra que en un texto pequeño [43].

- **Frecuencia de Término - Frecuencia Inversa de Documento (TF-IDF)**: Este método es una variación de TF en la que se utiliza el concepto de *Frecuencia Inversa de Documento* (IDF). Es probable que, aunque se eliminen gran parte de palabras vacías, siempre haya ciertas palabras que se escapen a este proceso y queden como impurezas a la hora de analizar el texto, siendo palabras relativamente comunes y frecuentes en los textos. De esta manera, IDF asigna a aquellas palabras que aparecen poco un mayor peso con la siguiente fórmula:

$$IDF = \log\left(\frac{n}{F_t}\right), \tag{10}$$

siendo n el número de documentos y F_t la frecuencia total de un término en todos los documentos. Combinando el método visto previamente con este concepto, surge TF-IDF, uno de los métodos más populares entre todos los de reducción de dimensiones por su contribución eliminando palabras vacías de forma indirecta. [43, 49, 50].

- **Fuerza de Término (TS)**: Aunque inicialmente este era un método de reducción supervisado, ha sido extendido para la clusterización de textos utilizando funciones de similitud como la del coseno[27]. Este criterio intenta medir cuán importante es una palabra a la hora de identificar dos documentos relacionados. Para ello, se toma una muestra aleatoria de pares de documentos en las que se realiza la siguiente

[27] Medida de similitud entre dos vectores que poseen un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos.

operación para un término t [40, 45]:

$$s(t) = \frac{\text{Número de pares donde } t \text{ aparece en ambos documentos}}{\text{Número de pares donde } t \text{ aparece en el primer par}}. \tag{11}$$

- **Ranking basado en Entropía (EN)**: Propuesto en [48], la calidad del término se obtiene calculando la reducción de entropía cuando se elimina dicha palabra. La entropía es una forma de medir el desorden de un sistema y su fórmula para un término t en un conjunto de n documentos es la siguiente:

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n})(S_{ij} \cdot \log(S_{ij}) + (1 - S_{ij}) \cdot \log(1 - S_{ij})), \tag{12}$$

  donde $S_{ij} \in (0, 1)$ es la similitud entre los documentos i y j después de haber retirado el término t, y se define de la siguiente forma:

$$S_{ij} = 2^{-\frac{dist(i,j)}{\overline{dist}}}. \tag{13}$$

  $dist(i, j)$ es la distancia entre los documentos i y j después de haberse eliminado el término t y $\overline{dist}$ es la distancia media entre todos los documentos tras la eliminación del término [40, 45].

- **Contribución del Término (TC)**: Introducido en [45], este método llamado *Term Contribution* tiene en cuenta el peso de cada término con la intención de eliminar uno de los principales problemas de DF: considerar que cada palabra tiene la misma importancia en cada documento. Para ello, el aporte de una palabra es vista como su aporte a la similitud entre documentos, cuya fórmula se apoya en TF-IDF [40, 45].

11.2.2 Transformación de Características

La transformación de características consiste en proyectar el conjunto de datos original en un espacio dimensional más pequeño donde cada dimensión es una combinación lineal o no lineal de la muestra inicial, predominando la forma lineal [46]. Es decir, los nuevos atributos son representaciones funcionales de los atributos originales basadas en combinaciones lineales de las dimensiones de la muestra [40, 47].

Estos son los métodos más populares:

- **Indexación Semántica Latente (LSI)**: Este método es junto con TF-IDF uno de los más estudiados y aplicados en Text Mining debido a su capacidad para detectar estructuras de un alto orden semántico, ya que esto permite eliminar ambigüedades del lenguaje natural como son los sinónimos y las palabras polisémicas [46]. LSI se basa en la Descomposición en Valores Singulares (SVD) de la matriz DTM para reducir la dimensionalidad de los datos con una distorsión mínima,

puesto que las dimensiones de este nuevo espacio son ortogonales, es decir, no tienen correlación. En SVD cada factor es ordenado de forma decreciente y excepto los k primeros términos, todos los demás son reestablecidos a valor cero. Algunos estudios demuestran que el número k adecuado ronda los cien términos. Esto permite capturar las asociaciones más importantes entre los documentos a la vez que elimina las impurezas entre los datos [40, 49].

- **Proyección Aleatoria (RP)**: Esta técnica se presenta como una alternativa computacionalmente más barata a LSI. A diferencia del método anterior, aquí las dimensiones son escogidas de forma arbitraria sin ningún orden de importancia. El funcionamiento es similar, RP también proyecta las columnas de DTM en un espacio reducido utilizando una matriz aleatoria. Sin embargo, sigue habiendo muchas dudas sobre su eficacia en comparación con LSI, por lo que todavía sigue en periodo de investigación y mejora [46].
- **Análisis de componentes independientes (ICA)**: Este método de transformación surgió recientemente y ha recibido bastante atención dentro del ámbito del procesamiento de señales. ICA es una técnica estadística de caracter general que intenta tranformar de forma lineal los datos originales en componenetes que sean independientes entre sí estadísticamente hablando [46].

11.3 Algoritmos para clusterización de textos

Una vez se ha disminuido al máximo posible la dimensionalidad de los datos, podemos asegurar que los atributos restantes son aquellas palabras clave y de más importancia que permiten identificar la temática de un documento y agruparlo con otros de ándole similar. De esta manera, la gran mayoría de los proceosos de reducción de términos trabajan sobre el *Modelo de Espacio Vectorial* (VSM), en el que se representa el texto como el saco de palabras que se ha ido mencionando a lo largo de esta sección, donde un texto queda representado por un subconjunto de sus palabras, presumiblemente aquellas más relevantes. La aplicación, por lo tanto de un método de reducción prepara los datos para poder aplicar clusterización sobre ellos como si se tratase de un conjunto de datos normales, en el que por ejemplo, en el caso de TF-IDF, asociamos cada documento con sus palabras más significativas, que son consideradas atributos de dicho documento, y cada una de estas palabras toma el valor obtenido del cálculo de TF-IDF.

Muchos de los algoritmos de clusterización presentados en este proyecto son de caracter general, por lo que pueden ser aplicados a casi cualquier tipo de datos como se vio en la Tabla 2. Entre ellos, algunos de los más utilizados son k-means 4.1, Fuzzy k-means 4.2, PAM 4.3.1, CLARA 4.3.2, DBSCAN 6.1, DENCLUE 6.3 y casi cualquier algoritmo jerárquico aglomerativo (AGNES 5.2, BIRCH 5.4 o Chameleon 5.5). Estos ya han sido expuestos y estudiados en sus respectivas secciones, pero hay dos algoritmos más que no se han tratado y que son utilizados frecuentemente en la clusterización de textos:

EM y SOM. EM mezcla métodos probabilísticos con fuzzy clustering y densidad mientras que SOM (Self Organizing Maps) es un modelo basado en redes neuronales.

11.3.1 Self-Organizing Map (SOM)

El algoritmo de Mapa Autoorganizado es utilizado en Inteligencia artificial y propuesto en [51] ha sido aplicado en diversos campos como reconcimiento de voz, telecomunicacione o robtica. Su cualidad principal es que tiene la capacidad de crear representacionese internas de las características de entrada, permitiendo descubrir relaciones semánticas en las frases. Pertenece al grupo de algoritmos basados en redes neuronales, en el que intenta generar un mapa intuituvo de un conjunto de datos con alta dimensionalidad mediante aprendizaje no supervisado bien en un espacio bidimensional o tridimensional, donde gracias al producto de procesar los datos con una red neurnal de una sola capa es capaz de situar objetos similares cerca entre ellos.

Cada una de las neuronas tienen asociado un vector de pesos que tiene la misma dimensión que los vectores de entrada y una posición en el mapa, conectando entre sí aquellas neuronas que estén cerca. Esta cercanía queda determinada por los patrones encontrados en los datos de entrada. Asimismo, SOM utiliza el concepto de red neuronal competitiva, donde las neuronas avtivas refuerzan la actividad dentro de su vecindario mientras suprimen la actividad de otras neuronas [40]. Gracias a esto, este algoritmo ofrece una representación de la estructura y relación entre los datos de una forma visual, haciendo que algunos expertos lo consideren más un método para la representación de datos que un algoritmo de clusterización.

Sin embargo, SOM requiere de una etapa de aprendizaje previa. En ella, las neuronas inicializan su peso de forma aleatoria o utilizando una muestra del conjunto de datos original, siendo más eficaz esta última. Posteriormente se administran ejemplos obtenidos del conjunto de datos de forma iterativa para el que se calcula su distancia euclidiana a todos los vectores de peso Se considera como la neurona con mayor correspondencia aquella cuyo vector de peso es el más similar al de la entrada. Los pesos de dicha neurona, denominada BMU (Best Matching Unit), y los de las otras neuronas cercanas son ajustados de forma que se aproximen más al vector de la entreda. Este proceso es repetido para cada vector de entrada dentro de un número de ciclos [14]. Este proceso queda resumido en el Algoritmo 5 y en la Figura 18.

11.3.2 Expectation-maximization (EM)

Este tipo de algoritmo llamado *Esperanza-maximización* en español pertenece a los grupos de algoritmos basados en densidad y en modelos probabilísticos, siendo uno de los más populares de este último grupo. Su enfoque se basa en intentar encontrar de forma iterativa la máxima verimisilitud entre parámetros de modelos probabilísticos que dependen de variables no

Algoritmo 5: SOM, basado en redes neuronales

Entrada: D: conjunto de datos con n objetos.
Salida: Un conjunto de prototipos.

```
1  inicio
2      generar un mapa de neuronas con vectores de pesos aleatorios
3      repetir
4          administrar un vector de entrada D(t)
5          para cada neurona m hacer
6              calcular la distancia euclídea entre el vector de entrada D(t)
                y el vector de peso de la neurona m
7              escoger la neurona con menor distancia: BMU
8          fin
9          actualizar los pesos de las neuronas dentro del vecindario de
            BMU
10     hasta queden muestras que procesar
11 fin
```

observables. Este alterna entre pasos de *Esperanza* o *Expectation* (E) y *Maximización* o *Maximization* (M). En los pasos E se calcula la esperanza de la verosimilitud gracias a la incorporación de variables latentes como si fueran observables. Por otro lado, en los paso M se obtienen las estimaciones de máxima verosimilitud de los parámetros mediante la maximización de la verosimilitud esperada del paso E. Los resultados obtenidos en el paso M son usados de nuevo en otro paso E [53].

Esto sirve como entorno de trabajo donde los pasos E y M pueden ser modificados para acomodarse a otros procesos de clustering como k-means o fuzzy k-means, donde el paso E conseiste en la asignación de cada objeto al cluster cuyo centroide está más cerca y el paso M en el ajuste de los centroides tras la asignación de objetos del paso E, cuyo resultado vuelve a aplicarse a un nuevo paso E [16].

12 Consolidadción

En esta sección recapitularemos brevemente sobre lo visto hasta ahora en este proyecto, dando así por concluido el marco teórico. La técnica de clusterización y su capacidad para clasificar objetos de manera no supervisada y durante el propio proceso ha hecho que se convierta en uno de los métodos más utilizados del mundo moderno con gran variedad de aplicaciones en diversos ámbitos. Los diferentes algoritmos que existen para realizar este proceso se pueden agrupar en cuatro grandes grupos: jerárquicos, basados en densidad, basados en rejilla y basados en particiones, existiendo otros grupos menores pero de creciente relevancia como algoritmos basados en métodos probabilísticos o en redes neuronales. Cada uno de los algoritmos de estos grupos opera de manera distinta y cuentan con ventajas e inconvenientes que obligan a tomar decisiones cruciales a la hora de decir cuál usar

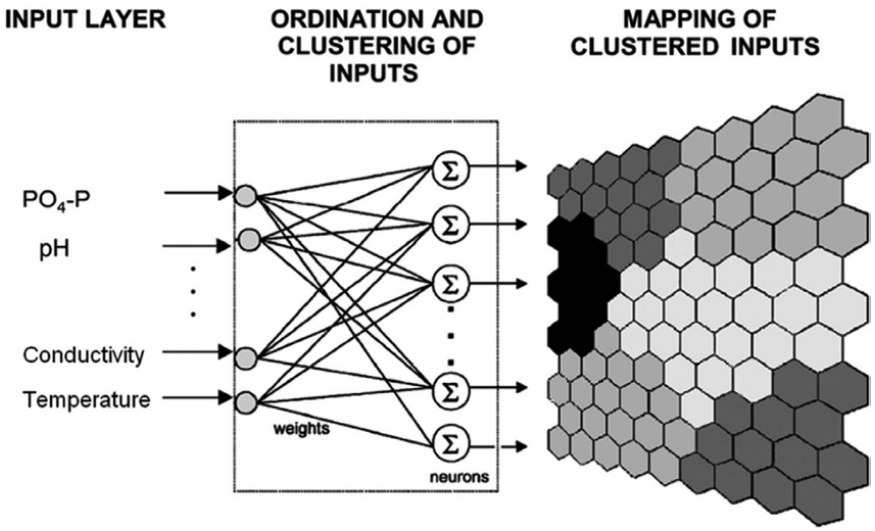


Figura 18: SOM: Esquema del proceso del algoritmo de Mapa Autoorganizado [52].

y qué valor asignar a los parámetros que nos puedan exigir.

Uno de los problemas que hacen al usuario determinar el tipo de algoritmo a utilizar es el tipo de datos con el que se pretende trabajar, siendo el texto uno de esos tipos de datos. Dentro de los textos y documentos encontramos gran parte de la información que se genera cada día, pero esto viene con la pega de que es información no estructurada y que necesita ser procesada de antemano para poder aplicar cualquier tipo de clusterización, con la que podemos agrupar documentos basándonos en su similitud repecto al tema que versan para facilitar su búsqueda y navegación.

A continuación se presenta un caso real donde la clusterización de documentos puede ser de gran importancia y que es tan relevante en estos momentos: COVID-19.

13 COVID-19

A finales del año 2019 se detecó en Wuhan, China, una nueva enfermedad cuyo origen apuntaba al Mercado Mayorista de Mariscos de dicha ciudad [54]. Los primeros casos se diagnosticaron como un principio de neumonía de origen desconocido por la similitud de sus síntomas, lo que hizo que saltasen las alamrmas y se pudiese identificar en las primeras semanas del mes de enero de 2020 al causante de la enfermedad, un nuevo tipo de *coronavirus*: *SARS-CoV-2*. Los coronavirus reciben este nombre por la forma que tiene y que se puede apreciar en la Figura 19 y que circulan principalmente en animales pero que han evolucionado de manera que también pueden infectar a humanos.

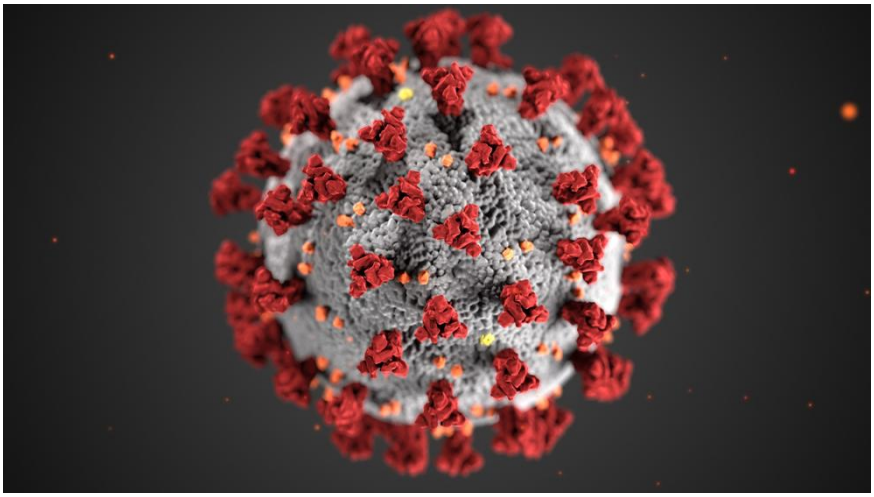


Figura 19: Ilustración del coronavirus SARS-CoV-2 realizada por Alissa Eckert y Dan Higgins [55].

Ya en diciembre la enfermedad se había propagado fuera de China, pero no sería hasta el trece de febrero de 2020 cuando la Organización Mundial de la Salud (OMS) avisó de manera oficial sobre el primer caso confirmado en Tailandia. Nuevos casos confirmaron el hecho de que este virus podía propagarse entre humanos, siendo el personal sanitario de China el más afectado y empezando a propagarse la enfermedad por otros países como Corea del Sur. No fue hasta el día treinta de enero que se declaró la existencia de un riesgo de salud pública de interés internacional y el día once de marzo se consideró pandemia debido al gran número de afectados (118.000) y muertes (4291) que habia provocado en más de ciento catorce países [56].

A fecha de la realización de este TFG (septiembre, 2020) las cifras datan más de treinta millones de personas contagiadas y casi un millón de fallecidos [58]. Se considera en este punto que se ha pasado la primera oleada de casos tras un periodo de cuarentena obligatorio que ha sido llevado a cabo en gran parte del mundo. Sin embargo, tras dicho confinamiento el número de casos volvió a aumentar generando nuevos focos de contagio, haciendo que la comunidad científica y sanitaría considere volver a tomar medidas drásccticas para preservar la salud de la población temiendo una segunda oleada.

13.1 Objetivo de la parte práctica

Desde el primer caso confirmado hasta la fecha actual han pasado nueve meses en los que las labores científicas por desarrollar una vacuna contra el virus y encontrar más información al respecto no han cesado. Sin embargo, es dificil para los expertos mantenerse actualizados con los nuevos datos y resultados obtenidos cada día sobre el virus. Gran parte de estos nuevos

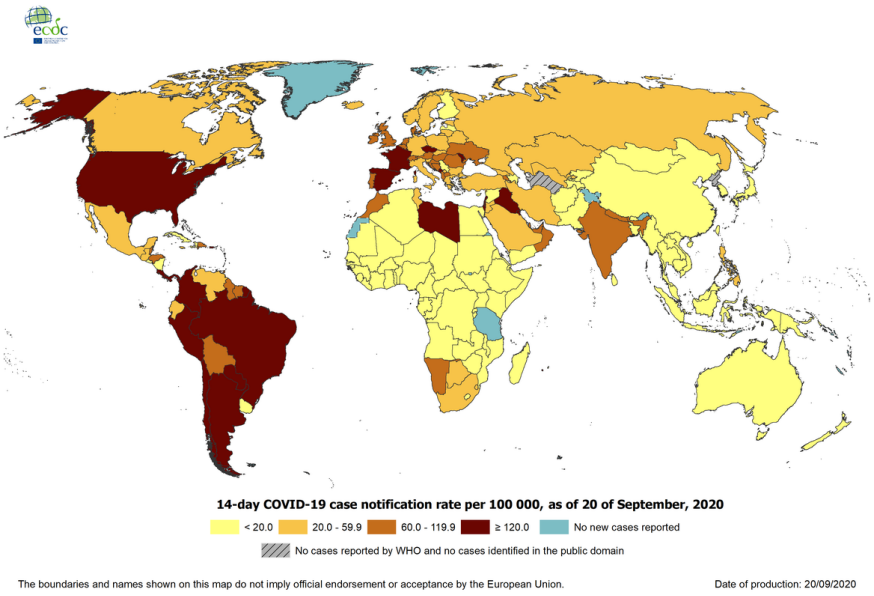


Figura 20: Distribución geográfica de catorce días acumulados de casos reportados de COVID-19 por cien mil habitantes a fecha de veinte de septiembre 2020 [57].

avances se encuentran en la forma de artículos científicos, reportes o documentos; datos de tipo texto. Aplicando clusterización a este conjunto de artículos se pretende realizar una clasificación no supervisada de los mismos en base a su contenido permitiendo a los científicos encontrar de una forma más facil aquello que buscan.

De esta manera, se aplicarán los conocimientos obtenidos a lo largo de este documento en un ámbito real con la intención de no solo ver un caso práctico de la teoría sino de ayudar a la comunidad científica en sus esfuerzos por terminar con la pandemia.

El conjunto de datos con el que trabajaremos se encuentra en la plataforma Kaggle, una comunidad online enfocada a analistas de datos y personas dedicadas al aprendizaje automático donde se pueden encontrar multitud de conjuntos de datos y publicaciones relacionadas con los mismos. De vez en cuando o en momentos cruciales, como este que estamos viviendo, Kaggle organiza competiciones remuneradas en las que incita a expertos e iniciados a participar. Con motivo de la pandemia de COVID-19 se lanzó una nueva competición en la que se pone a disposición del público más de dos cientos mil artículos sobre el virus e información relacionada con el mismo que se va actualizando periódicamente con la intención de obtener información útil [59].

Para el procesamiento de dichos datos utilizaremos R [60], un entorno len-

guaje de programación enfocado al análisis estadístico de datos, siendo uno de los más populares dentro del ámbito de la investigación científica, minería de datos, matemáticas, minería de datos, etc. Escogemos R sobre otros lenguajes por ser gratuito, de software libre, por su capacidad de alta escalabilidad y por su compatibilidad con todos los sistemas operativos. Otra alternativa que presenta las mismas características sería Python, otro de los lenguajes de programación más populares y que se caracteriza por su capacidad para emplearse en cualquier ámbito. Sin embargo, la familiaridad obtenida con R para este tipo de taréas hace que me decante por dicho lenguaje. Como entornos de trabajo se utilizarán RStudio, el IDE[28] por defecto para trabajar con R y JupyterNotebook, un entorno informático basado en web que permite crear documentos donde ejecutar código ofreciendo ayuda al usuario a la hora de mostrar gráfos y código, permitiendo exportar su contenido en diferentes formatos.

Por consiguiente, se planea aplicar diversos métodos de clusterización al conjunto de artículos relacionados con el COVID-19 ofrecidos por Kaggle utilizando R. El orden de operaciones se muestra a continuación, realizando una exploraración de los datos y reduciendo con técnicas de NLP la cantidad de atributos.

- Carga y exploración de los datos
- Preparación de los datos con NLP
  - STOPWORDS: SI USAMOS TAU SERÍA CONVENIENTE QUITAR LAS PALABRAS VACÍAS ANTES DE LA TOKENIZACIÓN.
  - TOKENIZACIÓN: CLEANNLP (SEGUN JORGE TRABAJA MEJOR CON MÁS DOCUMENTOS QUE TEXT2VEC), PERO PODEMOS COGER DIRECTAMENTE T2V PQ LA MATARIZ LA HACEMOS CON EL MISMO PAQUETE
  - LENMATIZACIÓN: USAMOS TEXTSTEM
- DTM: TEX2VEC
- TF-IDF: TEX2VEC
- EDA >TAU >TEXT2VEC >TEXTSTEM >TEXT2VEC >TEX2VEC >CLUSTERING
- FEATHER para almacenar los resultados

Lo primero que haremos será cargar los diferentes paquetes que vamos a utilizar para la realización de este proceso.

¡¡¡¡CARGA PAQUEtES!!!!

Figura 21: Conjunto de archivos del conjunto de datos ofrecido por Kaggle.

13.2 Carga y exploración de los datos

Como se ha mencionado, vamos a trabajar con el conjunto de artículos ofrecidos por Kaggle relacionados con la enfermedad COVID-19. Dentro de este paquete cuyo contenido se puede ver en la Figura 21 se encuentra un archivo .csv[29] "metadata.csv" que contiene los datos de más de 200.000 artículos tales como su título, DOI[30], resumen, autores, fecha de publicación, etc. Para cargar este archivo y poder ver sus datos utilizaremos la siguiente función:

[2]:
```
# Cargamos los datos del archivo "metadata.csv"
# stringAsFactors evita que las cadenas de texto se␣
 ↪conviertan en factores
# na.strings indica qué consideramos valores nulos. En este␣
 ↪caso celdas vacías y "NA"
metadata_df <- read.csv("D:/COVID/metadata.csv",␣
 ↪stringsAsFactors = FALSE, na.strings = c("", "NA"))

# str() permite ver la estructura del archivo: número de␣
 ↪filas, columnas y muestra el primero objeto almacenado
str(metadata_df)
```

```
'data.frame':   253454 obs. of  19 variables:
 $ cord_uid        : chr  "ug7v899j" "02tnwd4m" "ejv2xln0"␣
 ↪"2b73a28n" ...
 $ sha             : chr ␣
 ↪"d1aafb70c066a2068b02786f8929fd9c900897fb"
"6b0567729c2143a66d737eb0a2f63f2dce2e5a7d"
"06ced00a5fc04215949aa72528f2eeaae1d58927"
"348055649b6b8cf2b9a376498df9bf41f7123605" ...
 $ source_x        : chr  "PMC" "PMC" "PMC" "PMC" ...
 $ title           : chr  "Clinical features of␣
 ↪culture-proven Mycoplasma
```

[28]Se denominan así a los entornos de desarrollo integrado, aplicaciones informáticas que ofrecen servicios para ayudar a los programadores en el desarrollo de software.

[29]Tipo de archivo que se puede crear o editar en Excel que en vez de almacenar información en columnas estos guardan los datos separados por comas o puntos y comas.

[30]El DOI (Digital Objetc Identifier es una forma de identificación de objetos digitales permanente.

```
pneumoniae infections at King Abdulaziz University Hospital,␣
 ↪Jed"| __truncated__
"Nitric oxide: a pro-inflammatory mediator in lung disease?"␣
 ↪"Surfactant
protein-D and pulmonary host defense" "Role of endothelin-1␣
 ↪in lung disease" ...
 $ doi             : chr  "10.1186/1471-2334-1-6" "10.1186/
 ↪rr14" "10.1186/rr19"
"10.1186/rr44" ...
 $ pmcid           : chr  "PMC35282" "PMC59543" "PMC59549"␣
 ↪"PMC59574" ...
 $ pubmed_id       : chr  "11472636" "11667967" "11667972"␣
 ↪"11686871" ...
 $ license         : chr  "no-cc" "no-cc" "no-cc" "no-cc" ...
 $ abstract        : chr  "OBJECTIVE: This retrospective␣
 ↪chart review describes
the epidemiology and clinical features of 40 patients with"|␣
 ↪__truncated__
"Inflammatory diseases of the respiratory tract are commonly␣
 ↪associated with
elevated production of nitric oxide"| __truncated__␣
 ↪"Surfactant protein-D (SP-D)
participates in the innate response to inhaled microorganisms␣
 ↪and organic
antigens,"| __truncated__ "Endothelin-1 (ET-1) is a 21 amino␣
 ↪acid peptide with
diverse biological activity that has been implicated in num"|␣
 ↪__truncated__ ...
 $ publish_time    : chr  "2001-07-04" "2000-08-15"␣
 ↪"2000-08-25" "2001-02-22"
...
 $ authors         : chr  "Madani, Tariq A; Al-Ghamdi, Aisha␣
 ↪A" "Vliet, Albert
van der; Eiserich, Jason P; Cross, Carroll E" "Crouch, Erika␣
 ↪C" "Fagan, Karen A;
McMurtry, Ivan F; Rodman, David M" ...
 $ journal         : chr  "BMC Infect Dis" "Respir Res"␣
 ↪"Respir Res" "Respir
Res" ...
 $ mag_id          : logi  NA NA NA NA NA NA ...
 $ who_covidence_id: chr  NA NA NA NA ...
 $ arxiv_id        : chr  NA NA NA NA ...
 $ pdf_json_files  : chr
"document_parses/pdf_json/
 ↪d1aafb70c066a2068b02786f8929fd9c900897fb.json"
```

```
"document_parses/pdf_json/
 ↪6b0567729c2143a66d737eb0a2f63f2dce2e5a7d.json"
"document_parses/pdf_json/
 ↪06ced00a5fc04215949aa72528f2eeaae1d58927.json"
"document_parses/pdf_json/
 ↪348055649b6b8cf2b9a376498df9bf41f7123605.json" ...
 $ pmc_json_files  : chr  "document_parses/pmc_json/PMC35282.
 ↪xml.json"
"document_parses/pmc_json/PMC59543.xml.json"
"document_parses/pmc_json/PMC59549.xml.json"
"document_parses/pmc_json/PMC59574.xml.json" ...
 $ url             : chr  "https://www.ncbi.nlm.nih.gov/pmc/
 ↪articles/PMC35282/"
"https://www.ncbi.nlm.nih.gov/pmc/articles/PMC59543/"
"https://www.ncbi.nlm.nih.gov/pmc/articles/PMC59549/"
"https://www.ncbi.nlm.nih.gov/pmc/articles/PMC59574/" ...
 $ s2_id           : int  NA NA NA NA NA NA NA NA NA NA ...
```

Como se puede ver, contamos con 253.454 artículos y 19 atributos para cada uno de ellos. Sin embargo, dentro de este archivo no se encuentra el texto completo de los artículos, que se han almacenado de forma independiente en formato JSON (JavaScript Object Notation) en la carpeta "document_parses". Dentro de esta carpeta se hace distinción entre los JSON de artículos en formato PDF y formato PMC, donde utilizaremos los de PDF puesto que PMC puede contener más de un documento en su estructura, que se encuentra separado en PDFs. Es decir, existen PDFs distintos que aparecen en el mismo PMC. De esta forma, vamos a procesar todos los archivos JSON de los documentos en PDF y utilizar la información que estos contienen para generar nuestra propia tabla de datos con la que trabajar.

Para ello, vamos a generar una función que se encargue de sacar de la estructura JSON aquellos datos que queremos, que en este caso son el título, autores, resumen y cuerpo del texto.

[6]:
```r
# Función de ayuda  para extraer información de JSON
leer_json <- function(json){
  # Obtenemos el identificador del texto
  paper_id <- json$paper_id
  # Obtenemos sus autores
  authors <- json$metadata$authors
  author_list <- list()
  # Separamos cada autor
  for (author in authors) {
      name <- paste0(author$first," ", author$last, ";")
      author_list <- paste0(author_list, as.character(name))
  }
  # Obtenemos el título del artículo
  title_text <- json$metadata$title
```

```
  # Obtenemos todo el abstract o resumen
  abstract_text <- ''
  for (each_abstract in json$abstract) {
      abstract_text <- paste(abstract_text,␣
↪each_abstract$text)
  }
  # Obtenemos el cuerpo del artículo
  body_text <- ''
  for (each_body in json$body_text) {
      body_text <- paste(body_text, each_body$text)
  }

  # Devolvemos el resultado en forma de Data Frame
  return(tibble(paper_id, author_list, title_text,␣
↪abstract_text, body_text))
}
```

Posteriormente, debemos localizar en nuestro equipo la direccion de los archivos JSON de los PDFs y almacenarlos en una lista llamada *archivos*.

[4]:
```
# Identificamos la carpeta que contiene los JSON
directorio <- 'D:/COVID/document_parses/pdf_json/'
#Cogemos todos los archivos del directorio
archivos <- list.files(directorio)
# Mostramos el número de archivos
length(archivos)
```

106137

Como se puede apreciar, contamos con 106.137 archivos para procesar. De esta forma, iteraremos cada archivo de la lista y abriremos su contenido dento del entorno de trabajo para aplicar la función *leer_json* y extraer la información mencionada previamente. Se han tomado algunas precauciones por si el formato del archivo no es el correcto o faltasen datos en el archivo de manera que se evite cualquier problema a la hora de procesar los archivos. Asimismo, se van a añadir como campos la revista de publicación y el DOI que se obtienen del archivo "metadata.csv" buscando la información existente para el archivo que se esté procesando. Los datos extraidos de cada artículo se van almacenando en vectores que son unidos al final para formar el Data Frame final.

[7]:
```
# Inicializamos el contenido del artíuculo en formato lista
articulos <- list()
# Inicializamos el índice de artículos
indice <- 0

# Iteramos cada uno de los archivos
for (archivo in archivos) {
```

```
 # Actualizamos el puntero
 indice <- indice + 1
 # Breve comprobación para mostrar el estado de carga
 if (indice %% (length(archivos) %/% 10) == 0) {
   cat("Artículos procesados:  ", indice, " de ",␣
↪length(archivos), "\n")
 }

 # Modificamos el camino o PATH hacia el archivo actual
 ruta_archivo <- paste0(directorio, '/', archivo)
 # Convertimos el archivo JSON a objeto de R para poder␣
↪tratarlo
 json <- fromJSON(file = ruta_archivo)
 # Aplicamos la función para extraer la informaciópn del␣
↪archivo
 contenido <- leer_json(json)

 # Comprobamos si el formato del archivo es el correcto
 if(length(contenido$paper_id) > 0){
   # Si se ha obtenido información del archivo
   # Buscamos el artículo en el archivo "metadata.csv"
   meta <- metadata_df[which(metadata_df$sha ==␣
↪contenido$paper_id), ]
   # Si no hay información en metadata.csv o el cuerpo del␣
↪artículo está vacío saltamos el artículo
   if (length(meta) > 0 | length(contenido$body_text) <=␣
↪0) {
     # Si no se encuentra autor en el JSON buscamos en␣
↪metadata
     if (length(contenido$author_list) <= 0) {
       authors <- metadata_df$authors
       author_list <- list()
       # Separamos cada autor
       for (author in authors) {
           name <- paste0(author$first," ", author$last, ";
↪")
           author_list <- paste0(author_list, as.
↪character(name))
       }
       # Sustituimos los autores
       contenido$author_list <- author_list
     }

     # Si no se encuentra el título en el JSON lo buscamos␣
↪en metadata
     if (contenido$title_text == '') {
```

```
        contenido$title_text <- 'NoIncluido'
      }

      # Si no se encuentra el abstract lo dejamos como 'No incluido'
      if (nchar(contenido$abstract_text) <= 0) {
        contenido$abstract_text <- 'NoIncluido'
      }

      # Añadimos la revista de publicación
      if (length(meta$journal) > 0) {
        contenido$journal <- meta$journal[1]
      }
      else {
        contenido$journal <- 'NoIncluido'
      }
      # Añadimos el DOI
      if (length(meta$doi) > 0) {
        contenido$doi <- meta$doi[1]
      }
      else {
        contenido$doi <- 'NoIncluido'
      }

      # Incluimos el contenido al conjunto de artículos
      articulos[[indice]] <- contenido
    }
  }
}

# Enlazamos todas las filas generadas en un único Data Frame
covid_df <- bind_rows(articulos)
```

```
Artículos procesados:   10613  de  106137
Artículos procesados:   21226  de  106137
Artículos procesados:   31839  de  106137
Artículos procesados:   42452  de  106137
Artículos procesados:   53065  de  106137
Artículos procesados:   63678  de  106137
Artículos procesados:   74291  de  106137
Artículos procesados:   84904  de  106137
Artículos procesados:   95517  de  106137
Artículos procesados:   106130  de  106137
```

[10]:
```
# Observamos la primera fila del nuevo Data Frame menos el cuerpo
as.list(covid_df[1, -5])
```

```
# Comprobamos el número de artículos existentes
nrow(covid_df)
```

$paper_id '0001418189999fea7f7cbe3e82703d71c85a6fe5'

$author_list 'E Cornelissen;H Dewerchin;E Hamme;H Nauwynck;'

$title_text 'Absence of surface expression of feline infectious peritonitis virus (FIPV) antigens on infected cells isolated from cats with FIP'

$abstract_text ' Feline infectious peritonitis virus (FIPV) positive cells are present in pyogranulomas and exudates from cats with FIP. These cells belong mainly to the monocyte/macrophage lineage. How these cells survive in immune cats is not known. In this study, FIPV positive cells were isolated from pyogranulomas and exudates of 12 naturally FIPV-infected cats and the presence of two immunologic targets, viral antigens and MHC I, on their surface was determined. The majority of the infected cells were confirmed to be cells from the monocyte/macrophage lineage. No surface expression of viral antigens was detected on FIPV positive cells. MHC I molecules were present on all the FIPV positive cells. After cultivation of the isolated infected cells, 52 AE 10 % of the infected cells re-expressed viral antigens on the plasma membrane. In conclusion, it can be stated that in FIP cats, FIPV replicates in cells of the monocyte/macrophage lineage without carrying viral antigens in their plasma membrane, which could allow them to escape from antibody-dependent cell lysis. #'

$journal 'Veterinary Microbiology'

$doi '10.1016/j.vetmic.2006.11.026'

95980

Una vez realizado este proceso, podemos ver que 10.157 archivos no cumplían con los requisitos y el número total de documentos con el que se va a trabajar desciende hasta los 95.980. Con motivo de obtener información extra, calcularemos el número total de palabras para el resumen y el cuerpo de cada artículo.

[17]:
```
# Contamos el número de palabras para el abstracto ...
covid_df$words_abstract <- apply(covid_df['abstract_text'],␣
 ↪2, function(s) str_count(s, '\\w+'))
# ... y el cuerpo del texto
covid_df$words_body <- apply(covid_df['body_text'], 2,␣
 ↪function(s) str_count(s, '\\w+'))
# Mostramos el número de palabras de los primeros textos
head(covid_df[,c('title_text', 'words_abstract',␣
 ↪'words_body')])
```

title_text	words_abstract	words_body
Absence of surface expression of feline [...]	170	2120
Detection of Severe Acute Respiratory [...]	267	2022
Title: Rethinking high-risk groups in [...]	1	977
ScienceDirect ScienceDirect Effect of [...]	336	3316
Plasma inflammatory cytokines and chemokines [...]	203	2965
Journal Pre-proofs The Fire This Time[...]	1	1020

Antes de entrar más en profundidad a trabajar con los datos, veamos si hay algún duplicado. Para ello, utilizaremos una función integrada en R que detecta este tipo de situaciones.

[18]:
```
# Comprobamos cuantos objetos tienen el mismo cuerpo
duplicados <- duplicated(covid_df$body_text)
# Mostramos la información obtenida
summary(duplicados)
# Obtenemos los 5 primeros objetos detectados como␣
 ↪duplicados
head(which(duplicados==TRUE))
```

```
   Mode   FALSE    TRUE
logical   94519    1461
```

1. 2176 2. 2590 3. 3334 4. 4426 5. 5000 6. 5018

[20]:
```
# Mostramos los seis primeros duplicados
covid_df[c(2176,2590,3334,4426,5000,5018),'title_text']
```

title_text
Remdesivir in Treatment of COVID-19: A Systematic Benefit-Risk Assessment
Autoantibodies in Patients with Rheumatoid Arthritis
Initial High Viral Load Is Associated with Prolonged Shedding of Human Rhinovirus in [...]
Autoantibodies in Patients with Rheumatoid Arthritis
Extracellular superoxide dismutase, a molecular transducer of health benefits of exercise
COVID-19 pneumonia: different respiratory treatments for different phenotypes?

De este proceso se obtiene que hay más de mil objetos con el mismo contenido en el cuerpo, y como este campo no puede ser nulo por la extracción realizada a la hora de leer los JSON, la única posibilidad es que sean, en efecto, duplicados. Uno de los principales motivos por lo que esto ocurre es que en gran cantidad de ocasiones el mismo artículo es publicado en diferentes revistas.

[21]:
```
length(which(covid_df$title_text=='Autoantibodies in␣
 ↪Patients with Rheumatoid Arthritis'))
length(which(covid_df$title_text=='Remdesivir in Treatment␣
 ↪of COVID-19: A Systematic Benefit-Risk Assessment'))
length(which(covid_df$title_text=='Initial High Viral Load␣
 ↪Is Associated with Prolonged Shedding of Human Rhinovirus␣
 ↪in Allogeneic Hematopoietic Cell Transplant Recipients'))
```

```
length(which(covid_df$title_text=='Extracellular superoxide␣
 ↪dismutase, a molecular transducer of health benefits of␣
 ↪exercise'))
length(which(covid_df$title_text=='COVID-19 pneumonia:␣
 ↪different respiratory treatments for different phenotypes?
 ↪'))
```

69

2

2

2

2

Como se puede observar, de entre estos cinco primeros objetos detectados como duplicados, dos de ellos con el título "Autoantibodies in Patients with Rheumatoid Arthritis", coinciden en todos y cada uno de los campos. Y si comprobamos la cantidad de veces que aparecen estos cuatro objetos dentro del conjunto de datos, vemos que el título mencionado antes es idéntico en un total de 69 filas, siendo 68 de ellas duplicados de la primera. El resto de títulos solo tienen un duplicado. De esta manera, lo correcto es eliminar todos los duplicados para trabajar con una muestra lo más limpia posible.

[22]:
```
# Eliminamos duplicados
covid_df <- covid_df[!duplicated(covid_df$body_text),]
nrow(covid_df)
```

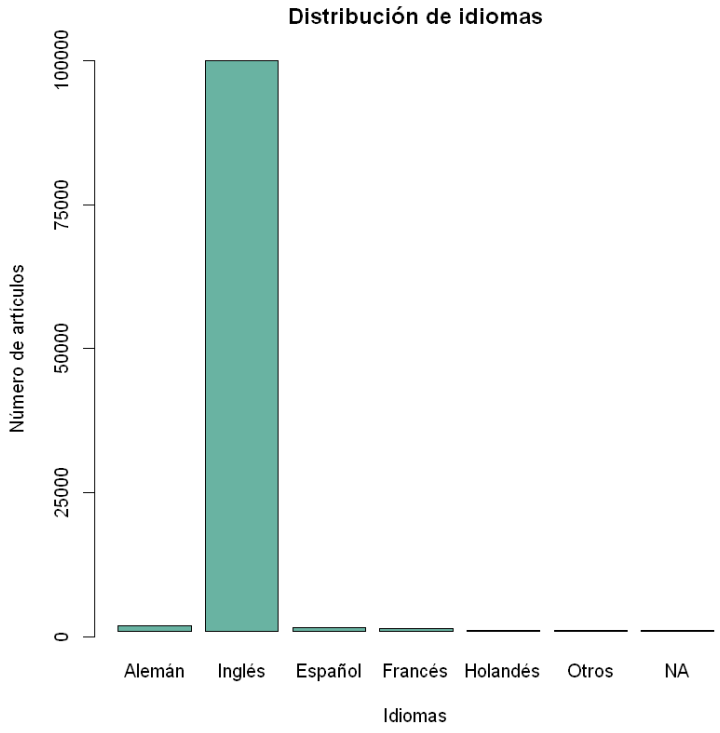
94519

A continuación, identificaremos el lenguaje principal de cada artículo comprobando los primeros 2000 caracteres del cuerpo y utilizando la librería *cld2*, que enlaza con la detección de idioma de Google para entornos Chromium, siendo capaz de detectar más de 80 idiomas.

[23]:
```
# Añadimos una nueva columna con el idioma de cada texto
covid_df$language <- apply(covid_df['body_text'], 2,␣
 ↪function(s) detect_language(substring(s, 1,  2000)))
# Información
summary(covid_df$language)
```

```
  body_text
en     :92213
de     :  842
es     :  658
fr     :  517
nl     :  130
(Other):   87
NA's   :   72
```

[24]:
```r
# Creación de matriz con los datos
idiomas <- matrix(c(842, 92212, 658, 517, 130, 71, 88),␣
 ↪ncol = 7, byrow = TRUE)
colnames(idiomas) <- (c('Alemán','Inglés', 'Español',␣
 ↪'Francés', 'Holandés', 'Otros', 'NA'))
# Generación de gráfico
barplot(idiomas, main = 'Distribución de idiomas', xlab =␣
 ↪'Idiomas', ylab = 'Número de artículos', col = '#69b3a2',␣
 ↪axes = FALSE)
usr <- par("usr")
par(usr=c(usr[1:2], 0, 100000))
axis(2, at=seq(0, 100000,25000))
```



Más del noventa y siete por ciento de los artículos están escritos en inglés, siendo el alemán el segundo idioma más utilizado, no llegando ni siquiera al uno por ciento del total. En la mayoría de los casos en que no se ha podido detectar el idioma el motivo es la falta de texto en el cuerpo. La traducción de estos artículos al inglés podría ocasionar distorsiones en el significado del documento y una pérdida de su valor, por lo que la opción más segura es

descartar cualquier archivo cuyo idioma no sea en inglés.

[27]:
```r
# Nos quedamos únicamente con textos en inglés
covid_df <- covid_df[which(covid_df$language=='en'),]
# Mostramos el número de artículos restantes
nrow(covid_df)
```

92213

Por último dentro de esta exploración de datos, sería interesante poder eliminar cualquier artículo cuya cantidad de palabras sea ínfima, puesto que sería prácticamente imposible obtener un buen concepto de la temática de un documento con menos de 200 palabras. Para ello, vamos a utilizar las columnas que se han calculado previamente que contienen la cantidad de palabras del resumen y el cuerpo del texto. Sumaremos ambos número y si el resultado es inferior, se descartará. Este proceso se realizará posteriormente, una vez eliminadas las palabras vacías, el siguiente paso.

13.3 Preparación de datos con NLP

Como se ha mencionado en la sección 11.1, el principal problema al que nos enfrentamos con la clusterización de textos es el número elevado de atributos con el que se trabaja. Para reducir dicha cantidad eliminaremos las palabras vacías y lematizaremos los términos restantes. El proceso seguido en esta sección y los paquetes elegidos para llevar esta preparación de dato a cabo han sido elegidos en base a [38] junto con ayuda de su autor, Jorge Montero Guillamón. En este trabajo se realiza un estudio intensivo sobre diferentes métodos de text mining tanto para R como para Python en los que compara su rendimiento y consumo de recursos para muestras de diferentes tamaños.

Para la eliminación de palabras vacías o stopwords se utilizará *TAU*, un paquete que ofrece una variedad de utilidades para el análisis de texto como la carga y preprocesado de documentos o el conteo de términos o patrones en documentos [38]. Este trabaja sobre texto plano, motivo por el cual no se ha realizado todavía la tokenización de los documentos. Junto con el paquete *text2vec*, TAU es el que mejor rendimiento ha obtenido en las pruebas realizadas en [38].

Posteriormente, se llevará a cabo la tokenización del texto con el paquete *text2vec*, que ofrece multitud de métodos para la minería de texto. A parte de su excelente rendimiento, el otro motivo para su utilización es precisamente la gran oferta de funciones con la que cuenta y que utilizaremos para la reducción de dimensionalidad, siendo más facil así la integracion de los resultados entre ellos.

Para el proceso de lematización se usará el paquete *textstem*, que provee un conjunto de herramientas para poder realizar stemming o lematización sobre textos. En este caso se utilizará la lematización para conservar todo lo posible la identidad y significado de las palabras originales. Este puede trabajar con texto plano o con tokens, por lo que la combinación con TAU

y text2vec es idónea. Asimismo, el rendimiento de este paquete es el que mejor resultados ha dado, otro motivo por el que se ha elegido.

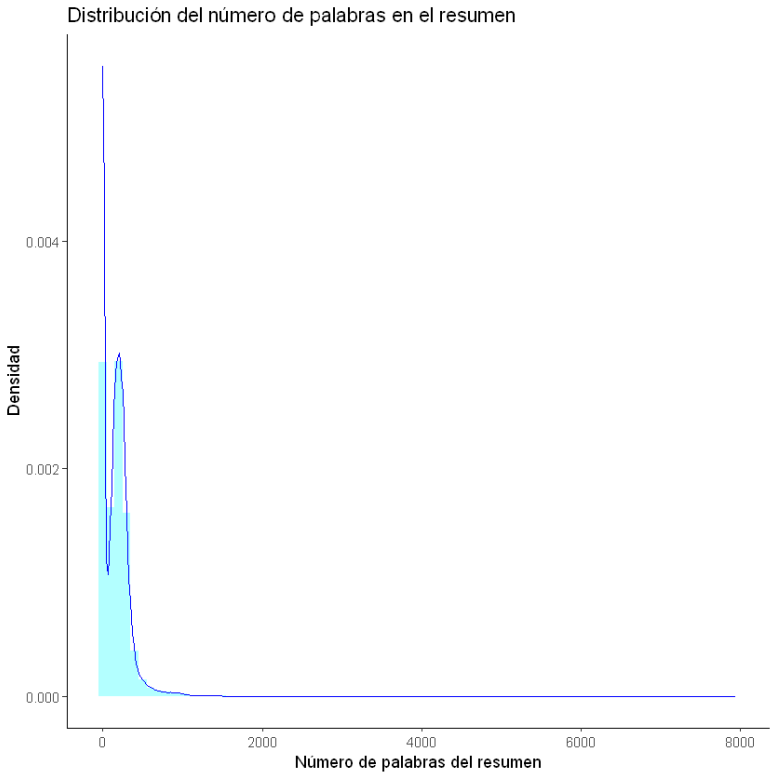
13.3.1 Eliminación de palabras vacías

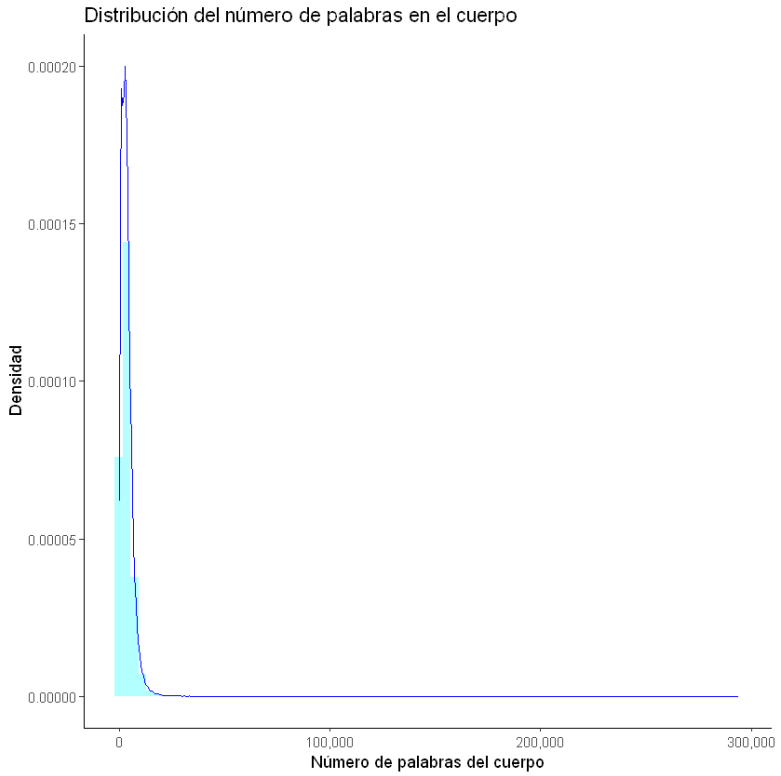
Antes de realizar este proceso, echemos un vistazo a los datos relativos al conteo de palabras que se ha realizado con la totalidad de los términos de los textos.

[28]:
```
# Información sobre el número de palabras
summary(covid_df[,c('words_abstract','words_body')])
```

```
 words_abstract.abstract_text words_body.body_text
 Min.   :   1.000             Min.   :     1.00
 1st Qu.:   1.000             1st Qu.:  1658.00
 Median : 162.000             Median :  3119.00
 Mean   : 174.521             Mean   :  4022.11
 3rd Qu.: 249.000             3rd Qu.:  4834.00
 Max.   :7922.000             Max.   :293299.00
```

[29]:
```
# Generación del grafo para el resumen con histograma y␣
 ↪densidad
palabras_abstract <- ggplot(covid_df,␣
 ↪aes(x=words_abstract)) + geom_histogram(aes(y=..density..
 ↪), alpha=.3, fill="cyan", bins=80) + geom_density(␣
 ↪colour="blue") + theme_classic() + ggtitle('Distribución␣
 ↪del número de palabras en el resumen') +  xlab('Número de␣
 ↪palabras del resumen') + ylab('Densidad')
# Misma operación para el cuerpo del texto
palabras_cuerpo <- ggplot(covid_df, aes(x=words_body)) +␣
 ↪geom_histogram(aes(y=..density..), alpha=.3, fill="cyan",␣
 ↪bins=80) + geom_density( colour="blue") + theme_classic()␣
 ↪+ scale_x_continuous(labels = comma) +␣
 ↪ggtitle('Distribución del número de palabras en el␣
 ↪cuerpo')  +  xlab('Número de palabras del cuerpo') +␣
 ↪ylab('Densidad')
# Mostramos los gráficos
palabras_abstract
palabras_cuerpo
```


Distribución del número de palabras en el resumen
Densidad
0.004
0.002
0.000
0
2000
4000
6000
8000
Número de palabras del resumen



Gracias a esta información nos hacemos una idea del volumen de datos con el que estamos trabajando, ya que la gran mayoría de los artículos tiene alrededor de 4000 palabras en el cuerpo, existiendo excepciones con casi 300000. Por otro lado, el tamaño del resumen es considerablemente menor, con unas 200 palabras de media. Además. se puede ver que existen artículos cuyo número de palabras en el resumen y el cuerpo es de 1. En el caso del resumen es muy probable que se la palabra "NoIncluido", indicando que no existe información sobre ese campo. A continuación se muestran aquellos documentos con menos de diez palabras en el resumen y el cuerpo del texto.

[34]:
```
# Cinco primeros artículos con menos de diez palabras en el␣
↪abstracto
covid_df[head(which(covid_df$words_abstract < 10 &␣
↪covid_df$abstract_text != 'NoIncluido')), c('title_text',␣
↪'abstract_text', 'words_abstract')]
```

title_text	abstract_text	words_abstract
Loss of IKK subunits limits [...]	18	1
Title: Piloting Forensic Tele-Mental [...]	Problem	1
NoIncluido	This article examines [...]	6
A Cluster-Randomized Trial of [...]	Background 49	2
Early View Sleep apnoea management [...]	on behalf of the ESADA [...]	7
Initial characterisation of ELISA [...]	Background To accurately [...]	9

[36]:
```
# Y menos de diez palabras en el cuerpo
covid_df[head(which(covid_df$words_body < 10)),␣
 ↪c('title_text', 'body_text', 'words_body')]
```

title_text	body_text	words_body
Severe SARS-CoV-2 infection in humans [...]	Supplemental Figure 15	3
Journal Pre-proof How to optimize the [...]	The authors report no [...]	7
NoIncluido	Journal Pre-proof Should	4
Journal Pre-proof Are healthcare [...]	results of a scoping review.	5
Der Chirurg Journal Club	Originalpublikation Sud	2
Journal Pre-proof THORACIC SURGERY [...]	To the Editor:	3

Para la eliminación de palabras vacías se utilizará tambíen el paquete *stopwords*, que ofrece una gran fuente palabras en varios idiomas que son utilizadas de forma común y dearlas en el texto solo serviría para aumentar los niveles de impureza, por lo que deben se detectadas y retiradas del texto. Además, se va a añadir a esta lista un conjunto extra de palabras anglosajonas que suelen ser usadas en artículos científicos. Previamente a este paso se transformarán todas las palabras a minúscula y se eliminarán los signos de puntuación con ayuda del paquete *tm*, que provee un entorno de trabajo donde permite aplicar una gran cantidad de métodod de text mining y métodos propios de R. Los números se van a dejar en este caso por su posible relevancia dentro del contexto de la enfermedad.

[37]:
```
# Cargamos la lista del paquete stopwords
p_vacias <- stopwords::stopwords("en",source =␣
 ↪"stopwords-iso")
# Añadimos más palabrasa
p_propias <- c('doi', 'preprint', 'copyright', 'org',␣
 ↪'https', 'et', 'al', 'author', 'figure', 'table',
    'rights', 'reserved', 'permission', 'use', 'used',␣
 ↪'using', 'biorxiv', 'medrxiv', 'license', 'fig', 'fig.',␣
 ↪'al.', 'Elsevier', 'PMC', 'CZI',
    '-PRON-', 'usually')
# Y las juntamos
p_vacias <- append(p_vacias, p_propias)
# Eliminamos cualquier posible duplicado
p_vacias <- unique(p_vacias)
# Vemos información al respecto
length(p_vacias)
```

```
sample(p_vacias, 10)
```

1317

1. 'trying' 2. 'grouping' 3. 'resulting' 4. 'ge' 5. 'following' 6. 'sy' 7. 'primarily' 8. 'kw' 9. 'one\'s' 10. 'arent'

[38]:
```
# Almacenamos el primer resumen para comparar
resumen <- covid_df$abstract_text[1]
# Conversión del resumen a minúsculas
covid_df$abstract_text <- apply(covid_df['abstract_text'],␣
↪2, function(s) tolower(s))
# Eliminación de signos de puntuación
covid_df$abstract_text <- apply(covid_df['abstract_text'],␣
↪2, function(s) removePunctuation(s,␣
↪preserve_intra_word_contractions = TRUE,␣
↪preserve_intra_word_dashes = TRUE))
# Eliminación de palabras vacías
covid_df$abstract_text <- apply(covid_df['abstract_text'],␣
↪2, function(s) remove_stopwords(s, p_vacias, lines =␣
↪TRUE))
# Añadimos nueva columna con número de palabras actuales
covid_df$new_word_abstract <-␣
↪apply(covid_df['abstract_text'], 2, function(s)␣
↪str_count(s, '\\w+'))

# Repetimos la operación con el cuerpo
# Conversión del cuerpo a minúsculas
covid_df$body_text <- apply(covid_df['body_text'], 2,␣
↪function(s) tolower(s))
# Eliminación de signos de puntuación
covid_df$body_text <- apply(covid_df['body_text'], 2,␣
↪function(s) removePunctuation(s))
# Eliminación de palabras vacías
covid_df$body_text <- apply(covid_df['body_text'], 2,␣
↪function(s) remove_stopwords(s, p_vacias, lines = TRUE))
# Añadimos nueva columna con número de palabras actuales
covid_df$new_word_body <- apply(covid_df['body_text'], 2,␣
↪function(s) str_count(s, '\\w+'))
```

[41]:
```
# Comparamos el resumen original con el actual
substring(resumen, 1, 100)
substring(covid_df[1,'abstract_text'], 1, 100)
```

' Feline infectious peritonitis virus (FIPV) positive cells are present in pyogranulomas and exudates'

' feline infectious peritonitis virus fipv positive cells pyogranulomas exudates

cats fip cell'

Como se aprecia, se han eliminado aquellas palabras comunes que no aportan nada al artículo como .ªnd", o ïn". Ahora veremos el número nuevo de palabras y cuántas se han eliminado.

[45]:
```
# Creamos dos nuevas columnas con la resta de palabras␣
↪originales y actuales
covid_df$abstract_comparative <- covid_df$words_abstract -␣
↪covid_df$new_word_abstract
covid_df$body_comparative <- covid_df$words_body -␣
↪covid_df$new_word_body
# Mostramos un resumen de la información
summary(covid_df[c('new_word_abstract', 'new_word_body',␣
↪'abstract_comparative', 'body_comparative')])
# Calculamos el total de palabras iniciales
inicial_resumen <- sum(covid_df$words_abstract)
inicial_cuerpo <- sum(covid_df$words_body)
inicial <- inicial_resumen + inicial_cuerpo
# Calculamos el total de palabras finales
final_resumen <- sum(covid_df$new_word_abstract)
final_cuerpo <- sum(covid_df$new_word_body)
final <- final_resumen + final_cuerpo
# Calculamos las palabras eliminadas
quitadas_resumen <- sum(covid_df$abstract_comparative)
quitadas_cuerpo <- sum(covid_df$body_comparative)
quitadas <- quitadas_resumen + quitadas_cuerpo
# Mostramos los datos
datos <- matrix(c(inicial_resumen, inicial_cuerpo, inicial,␣
↪final_resumen, final_cuerpo, final, quitadas_resumen,␣
↪quitadas_cuerpo, quitadas), nrow = 3, dimnames =␣
↪list(c("Resumen", "Cuerpo", "Total"), c("Inicial",␣
↪"Final", "Eliminadas")))
datos
```

```
new_word_abstract new_word_body    abstract_comparative␣
↪body_comparative
Min.   :   0.00   Min.   :     1   Min.   :   0.00      Min.␣
↪   :     0
1st Qu.:   1.00   1st Qu.:   848   1st Qu.:   0.00      1st␣
↪Qu.:   798
Median :  88.00   Median :  1590   Median :  73.00      ␣
↪Median :  1504
Mean   :  95.74   Mean   :  2085   Mean   :  79.24      Mean␣
↪   :  1945
3rd Qu.: 136.00   3rd Qu.:  2495   3rd Qu.: 111.00      3rd␣
↪Qu.:  2343
```

```
Max.   :4696.00   Max.   :162073   Max.   :5126.00      Max.␣
↪  :131565
```

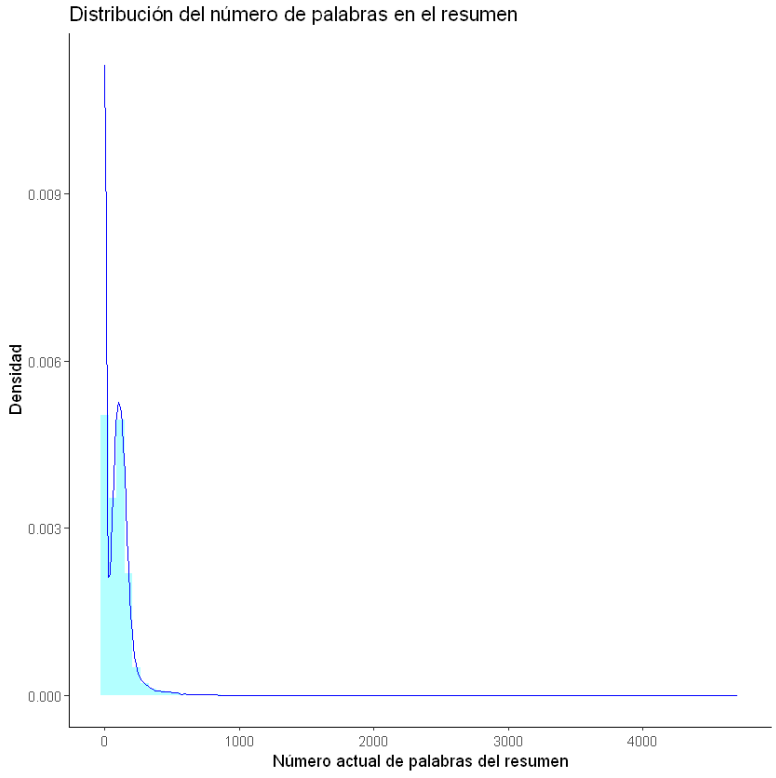
	Inicial	Final	Eliminadas
Resumen	16134914	8828313	7306601
Cuerpo	371628162	192247004	179381158
Total	387763076	201075317	186687759

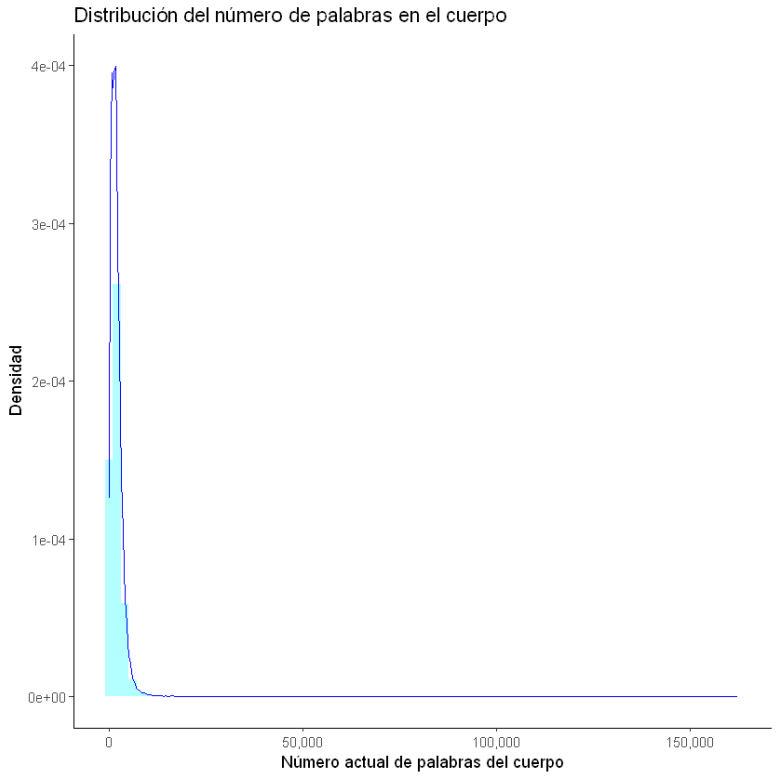
Como se puede ver, se han eliminado un gran número de palabras: 186.687.759, de las cuales el 96 % han sido retiradas del cuerpo de los artículos, como cabía de esperar. Ahora el número de palabras máximo de un cuerpo es de 162073, la mitad de lo que era antes el máximo. De media se han retirado 2.024 palabras de cada artículo haciendo que el número total de palabras se reduzca a algo más de la mitad, siendo este un muy buen resultado de eliminación de stopwords tal y como se puede ver en las siguientes gráficas.

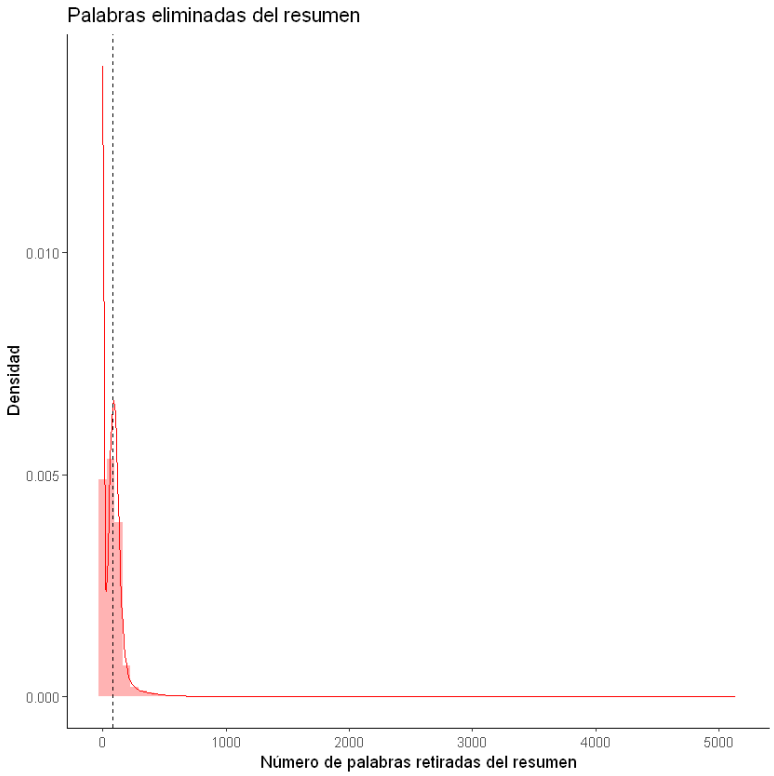
[46]:
```
# Generación del grafo para el resumen con histograma y␣
 ↪densidad
nuevas_palabras_abstract <- ggplot(covid_df,␣
 ↪aes(x=new_word_abstract)) + geom_histogram(aes(y=..
 ↪density..), alpha=.3, fill="cyan", bins=80) +␣
 ↪geom_density( colour="blue") + theme_classic() +␣
 ↪ggtitle('Distribución del número de palabras en el␣
 ↪resumen') + xlab('Número actual de palabras del resumen')␣
 ↪+ ylab('Densidad')
# Misma operación para el cuerpo del texto
nuevas_palabras_cuerpo <- ggplot(covid_df,␣
 ↪aes(x=new_word_body)) + geom_histogram(aes(y=..density..
 ↪), alpha=.3, fill="cyan", bins=80) + geom_density(␣
 ↪colour="blue") + theme_classic() +␣
 ↪scale_x_continuous(labels = comma) +␣
 ↪ggtitle('Distribución del número de palabras en el␣
 ↪cuerpo') + xlab('Número actual de palabras del cuerpo') +␣
 ↪ylab('Densidad')
# Generamos grafo para ver cuantas palabras se han quitado
comparacion_abstract<-ggplot(covid_df,␣
 ↪aes(x=abstract_comparative)) + geom_histogram(aes(y=..
 ↪density..), alpha=.3, fill="red", bins=80) +␣
 ↪geom_density( colour="red") + theme_classic() +␣
 ↪ggtitle('Palabras eliminadas del resumen') + ␣
 ↪xlab('Número de palabras retiradas del resumen') +␣
 ↪ylab('Densidad') + geom_vline(aes(xintercept=␣
 ↪mean(abstract_comparative)), linetype="dashed")
# Misma operaciópn para el cuerpo del texto
```

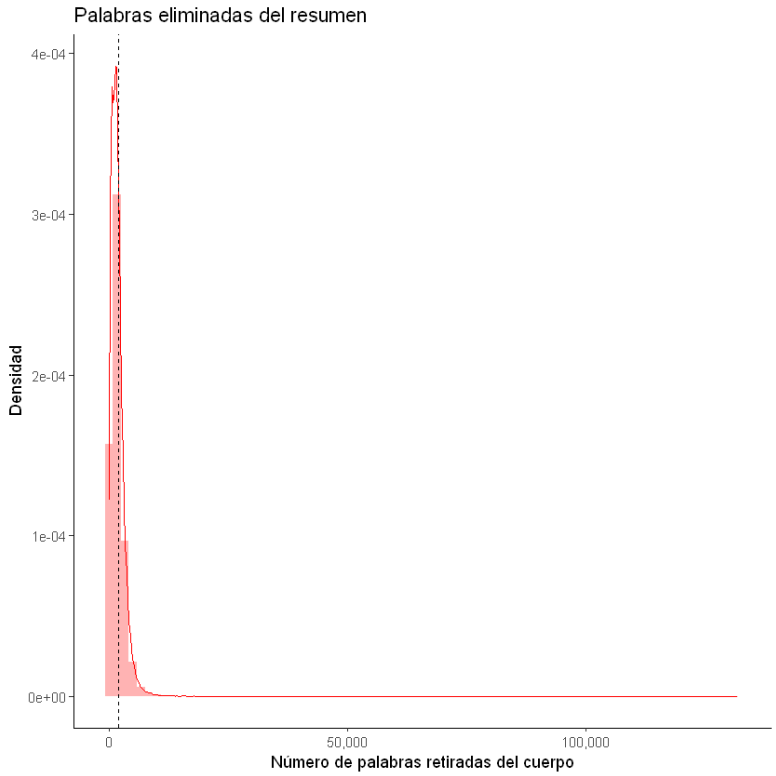
```
comparacion_cuerpo <- ggplot(covid_df,␣
↪aes(x=body_comparative)) + geom_histogram(aes(y=..density.
↪.), alpha=.3, fill="red", bins=80) + geom_density(␣
↪colour="red") + theme_classic() +␣
↪scale_x_continuous(labels = comma) + ggtitle('Palabras␣
↪eliminadas del resumen') + xlab('Número de palabras␣
↪retiradas del cuerpo') + ylab('Densidad') +␣
↪geom_vline(aes(xintercept= mean(body_comparative)),␣
↪linetype="dashed")

# Mostramos los gráficos
nuevas_palabras_abstract
nuevas_palabras_cuerpo
comparacion_abstract
comparacion_cuerpo
```




Distribución del número de palabras en el cuerpo
4e-04
3e-04
2e-04
1e-04
0e+00
Densidad
0
50,000
100,000
150,000
Número actual de palabras del cuerpo


Palabras eliminadas del resumen
Densidad
0.010
0.005
0.000
0
1000
2000
3000
4000
5000
Número de palabras retiradas del resumen



13.4 Tokenización

A continuación, recogeremos todas esas palabras restantes y las separaremos en cadenas individuales, es decir, en vez de tener un único dato de tipo String que contenga todas las palabras, tendremos una lista de todas las diferentes palabras.

[54]:
```
# Tokenizamos tanto abstracto como texto completo
covid_df['abstract_text'] <-␣
 ↪apply(covid_df['abstract_text'], 2, function(s)␣
 ↪word_tokenizer(s, xptr = TRUE, pos_keep = character('-')))
covid_df['body_text'] <- apply(covid_df['body_text'], 2,␣
 ↪function(s) word_tokenizer (s, xptr = TRUE))

# Mostramos los primeros cinco términos de los resúmenes y␣
 ↪el cuerpo
covid_df$abstract_text[[1]][1:10]
covid_df$body_text[[1]][1:10]
```

1. 'feline' 2. 'infectious' 3. 'peritonitis' 4. 'virus' 5. 'fipv' 6. 'positive' 7. 'cells' 8. 'pyogranulomas' 9. 'exudates' 10. 'cats'

1. 'feline' 2. 'infectious' 3. 'peritonitis' 4. 'fip' 5. 'fatal' 6. 'chronic' 7. 'disease' 8. 'cats' 9. 'caused' 10. 'coronavirus'

Como se puede apreciar, en efecto ahora tenemos una lista que contiene todas las diferentes palabras tanto del resumen como del cuerpo principal del texto en vez de un solo atributo con todo el contenido.

13.5 Lematización

El siguiente paso a seguir es la lematización de las palabras, que como se ha mencionado previamente es escogido por encima del proceso de stemming por su capacidad de mantener gran parte del valor semántico de las palabras. Para relizar esta tarea nos aprovecharemos del hecho de que trabajamos con una lista, a la que podemos aplicar la función *lapply*, que sirve precisamente para, en vez de ir de forma secuencial, trabaja paralelamente con todos los elementos de la lista, agilizando el proceso. Es por este motivo que se ha realizado la tokenización antes que la lematización.

[50]:
```
# Haciendo "apply <- lapply" conseguimos una paralelización␣
 ↪del trabajo gracias a la lista que genera la tokenización␣
 ↪de los textos
system.time(covid_df['abstract_text'] <-␣
 ↪apply(covid_df['abstract_text'], 2, function(s) lapply(s,␣
 ↪function(t) lemmatize_words(t))))
system.time(covid_df['body_text'] <-␣
 ↪apply(covid_df['body_text'], 2, function(s) lapply(s,␣
 ↪function(t) lemmatize_words(t))))
```

Comparando con el resultado mostrado previamente, podemos comprobar que también se ha realizado correctamente este proceso en que se han eliminado los plurales de las palabras y los tiempos verbales de palabras como çatsz çaused"

[52]:
```
# Mostramos los primeros cinco términos de los resúmenes y␣
 ↪el cuerpo
covid_df$abstract_text[[1]][1:10]
covid_df$body_text[[1]][1:10]
```

1. 'feline' 2. 'infectious' 3. 'peritonitis' 4. 'virus' 5. 'fipv' 6. 'positive' 7. 'cell' 8. 'pyogranulomas' 9. 'exudate' 10. 'cat'

1. 'feline' 2. 'infectious' 3. 'peritonitis' 4. 'fip' 5. 'fatal' 6. 'chronic' 7. 'disease' 8. 'cat' 9. 'cause' 10. 'coronavirus'

Asimismo, podemos ver que la cantidad de palabras diferentes ha disminuido drásticamente gracias al procesamiento de los datos tal y como se puede

ver en la siguiente tabla, donde se muestra la cantidad de palabras iniciales, totales y únicas de los primeros cinco artículos.

[ ]:
```
iniciales <- covid_df$words_body[1:5]
# Calculamos el total de palabras del cuerpo de los cinco␣
 ↪primeros artículos...
finales <- c(length(covid_df$body_text[[1]]),␣
 ↪length(covid_df$body_text[[2]]),␣
 ↪length(covid_df$body_text[[3]]),␣
 ↪length(covid_df$body_text[[4]]),␣
 ↪length(covid_df$body_text[[5]]))
#... y las palabras únicas de los mismos
unicas <- c(length(unique(covid_df$body_text[[1]])),␣
 ↪length(unique(covid_df$body_text[[2]])),␣
 ↪length(unique(covid_df$body_text[[3]])),␣
 ↪length(unique(covid_df$body_text[[4]])),␣
 ↪length(unique(covid_df$body_text[[5]])))
# Creamos una matriz para comparar los valores y damos␣
 ↪nombre a filas y columnas
comparacion <- matrix(c(iniciales, finales, unicas), nrow =␣
 ↪3, byrow = T)
dimnames(comparacion)<-list(c("Iniciales",␣
 ↪"Finales","Unicas"), c("Artículo 1","Artículo␣
 ↪2","Artículo 3","Artículo 4","Artículo 5"))
comparacion
```

	Artículo 1	Artículo 2	Artículo 3	Artículo 4	Artículo 5
Iniciales	2120	2024	977	3331	2968
Finales	1089	1135	474	1838	1700
Unicas	417	496	260	486	631

Con este paso realizado podemos dar casi por concluido el apartado de preparación de los datos utilizando NLP, puesto que a continuación se utilizará el método de Selección de Características *TF-IDF* visto en la sección 11.2.1, que es una parte que tienen en común tanto el procesamiento de lenguaje natural como la clusterización

14 Reducción de dimensionalidad

Como se ha mencionado a lo largo del documento, el volumen de datos con el que se trabaja en la clusterización de documentos es muy amplio y trabajar con dichos datos de forma cruda, sin tratar o minimizar tardaría exponencialmente más tiempo del que ya conlleva. De esta manera, utilizaremos una tecnia de reducción de dimensionalidad basada en la selección de caracteristicas, donde se escoge un subconjunto de los atributos originales como forma de representar al objeto. En este caso, se escogerán las palabras más relevantes del texto en función a su frecuencia inversa de aparición

en el documento mediante la fórmula vista con anterioridad en la sección referente a este método en 11.2.1.

Este es uno de los métodos más populares debido no solo a su eficacia y utilidad, sino porque también sirve en gran medida para eliminar todas aquellas palanbras vacías que hayan podido escaparse durante el pre-procesamiento de los datos. Para aplicar esta técnica utilizaremos el paquete Text2Vec que parte de la DTM para generar la matriz de pesos TF-IDF, por lo que el primer paso será obtener el DTM de cada artículo, de esta forma obtendremos una normalización más completa.

15 Anexo

15.1 Algoritmos

15.2 Tablas

15.3 Imágenes y figuras

16 Referencias

[1] Alberts, D. S., & Papp, D. S. (1997). The information age: An anthology on its impact and consequences. Office of the Assistant Secretary of Defense Washington DC Command and Control Research Program (CCRP).

[2] Becoming A Data-Driven CEO | Domo. (2018). Data never sleeps 6.0 https://www.domo.com/solution/data-never-sleeps-6

[3] Xu, Z., & Shi, Y. (2015). Exploring big data analysis: fundamental scientific problems´. Annals of Data Science, 2(4), 363-372.

[4] Definición Data Science apuntes FCD

[5] The R Project for Statistical Computing. (n.d.). Retrieved from https://www.r-project.org/

[6] Moreno, A. (1994). Aprendizaje automático. Llibre, Edicions UPC.

[7] Apuntes JJ clustering

[8] Xu, R., & Wunschii, D. (2005). Survey of Clustering Algorithms. IEEE Transactions on Neural Networks, 16(3), 645-678. doi:10.1109/tnn.2005.845141

[9] Alkhaibari, A. A., & Chung, P. (2017). Cluster analysis for reducing city crime rates. 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT). doi:10.1109/lisat.2017.8001983

[10] Song, Y., Meng, H., & Zhang, Y. (2010). Clustering analysis and its applications. 2010 Second IITA International Conference on Geoscience and Remote Sensing. doi:10.1109/iita-grs.2010.5602787

[11] Prabhu, J., Sudharshan, M., Saravanan, M., & Prasad, G. (2010). Augmenting Rapid Clustering Method for Social Network Analysis. 2010 International Conference on Advances in Social Networks Analysis and Mining. doi:10.1109/asonam.2010.55

[12] Baron, J. N., Aznar, M. N., Monterubbianesi, M., & Martínez-López, B. (2020). Application of network analysis and cluster analysis for better prevention and control of swine diseases in Argentina. PLoS ONE, 15(6), 1–26. https://doi.org/10.1371/journal.pone.0234489

[13] Li, J., & Chen, P. (2008). The application of Cluster analysis in Library system. 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop. doi:10.1109/kamw.2008.4810639

[14] Emebo, O., Aromolaran, O., Oyelade, J., Isewon, I., Oladipupo, O., Omogbadegun, Z., ... Olawole, O. (2019). Data Clustering: Algorithms and Its Applications. 2019 19th International Conference on Computational Science and Its Applications (ICCSA). doi:10.1109/iccsa.2019.000-1

[15] Carugo, O., & Eisenhaber, F. (2010). Data Mining Techniques for the Life Sciences. Humana Press.

[16] Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques (3rd ed., p. 740). 225 Wyman Street, Waltham, MA 02451, USA: Morgan Kaufmann Publishers, Elsevier.

[17] Höppner, F., Klawonn, F., Kruse, R., & Runkler, T. (1999). Fuzzy cluster analysis: methods for classification, data analysis and image recognition. John Wiley & Sons.

[18] Harary, F. (1996). Graph theory. London: Addison-Wesley.

[19] Foggia, Pasquale & Percannella, Gennaro & Vento, Mario. (2014). Graph Matching and Learning in Pattern Recognition in the Last 10 Years. International Journal of Pattern Recognition and Artificial Intelligence. doi:10.1142/S0218001414500013.

[20] Benavente, P., Protopapas, P., & Pichara, K. (2017). Automatic Survey-invariant Classification of Variable Stars. The Astrophysical Journal, 845(2), 147. doi: 10.3847/1538-4357/aa7f2d

[21] Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences, 10(2-3), 191-203.

[22] Grant, R. P., & Babu, M. M. (2004). Computational genomics: Theory and application. Wymondham: Horizon Bioscience.

[23] Struyf, A., Hubert, M., & Rousseeuw, P. (1997). Clustering in an object-oriented environment. Journal of Statistical Software, 1(4), 1-30.

[24] Sano, A. V., Imanuel, T. D., Calista, M. I., Nindito, H., & Condrobimo, A. R. (2018). The Application of AGNES Algorithm to Optimize Knowledge Base for Tourism Chatbot. 2018 International Conference on Information Management and Technology (ICIMTech). doi:10.1109/icimtech.2018.8528174

[25] A. Periklis. (2002). Data Clustering Techniques. University of Toronto.

[26] T. Zhang. (n.d.). BIRCH: An efficient data clustering method for very large databases. Proc. ACM SIGMOD Conf. Management of Data, 103-114.

[27] Lorbeer, B., Kosareva, A., Deva, B., Softić, D., Ruppel, P., & Küpper, A. (2018). Variations on the Clustering Algorithm BIRCH. Big Data Research, 11, 44-53. doi: 10.1016/j.bdr.2017.09.002

[28] Karypis, G., Eui-Hong Han, & Kumar, V. (1999). Chameleon: hierarchical clustering using dynamic modeling. Computer, 32(8), 68-75. doi: 10.1109/2.781637

[29] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

[30] Schlitter, N., Falkowski, T., & Laessig, J. (2011). DenGraph-HO: Density-based Hierarchical Community Detection for Explorative Visual Network Analysis. Research and Development in Intelligent Systems XXVIII, 283-296. doi:10.1007978-1-4471-2318-7_22

[31] Hinneburg, A., & Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise.

[32] Wang, W., Yang, J., & Muntz, R. (1997, August). STING: A statistical information grid approach to spatial data mining. In VLDB (Vol. 97, pp. 186-195).

[33] Makhabel, B. (2015). Learning data mining with R: Develop key skills and techniques with R to create and customize data mining algorithms. Birmingham, UK: Packt Publishing.

[34] Xing, Wanli. (2016). Exploring the collective knowledge curation process of online health communities´. 10.13140/RG.2.2.17964.05761.

[35] C. C. Aggarwal &. K. Reddy. (2014). Data Clustering: Algorithms and Applications. Taylor & Francis Group. LLC

[36] Sun, A. (2010). Improved SOM Algorithm-HDSOM Applied in Text Clustering. 2010 International Conference on Multimedia Information Networking and Security. doi:10.1109/mines.2010.74

[37] Gupta, V., & Lehal, G. S. (2009). A Survey of Text Mining Techniques and Applications. Journal of Emerging Technologies in Web Intelligence, 1(1). doi:10.4304/jetwi.1.1.60-76

[38] TFG Jorge

[39] Weiss, S. M., Indurkhya, N., & Zhang, T. (2015). Fundamentals of predictive text mining. Springer.

[40] Aggarwal, C. C., & Zhai, C. (2012). A Survey of Text Clustering Algorithms. Mininkg Text Data, 77-128. doi:10.1007/978-1-4614-3223-4_4

[41] Lematización. (2019, July 16). Retrieved September 18, 2020, from https://es.wikipedia.org/wiki/Lematizaci %C3 %B3n

[42] Song, W., & Park, S. C. (2009). Genetic algorithm for text clustering based on latent semantic indexing. Computers & Mathematics with Applications, 57(11-12), 1901-1907. doi:10.1016/j.camwa.2008.10.010

[43] Qaiser, S., & Ali, R. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications, 181(1), 25-29. doi:10.5120/ijca2018917395

[44] Shafiei, M., Wang, S., Zhang, R., Milios, E., Tang, B., Tougas, J., & Spiteri, R. (2007, April). Document representation and dimension reduction for text clustering. In 2007 IEEE 23rd international conference on data engineering workshop (pp. 770-779). IEEE.

[45] Liu, T., Liu, S., Chen, Z., & Ma, W. Y. (2003). An evaluation on feature selection for text clustering. In Proceedings of the 20th international conference on machine learning (ICML-03) (pp. 488-495).

[46] Tang, B., Shepherd, M., Milios, E., & Heywood, M. I. (2005, April). Comparing and combining dimension reduction techniques for efficient text clustering. In Proceeding of SIAM International Workshop on Feature Selection for Data Mining (pp. 17-26).

[47] Jović, A., Brkić, K., & Bogunović, N. (2015, May). A review of feature selection methods with applications. In 2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO) (pp. 1200-1205). Ieee.

[48] Dash, M.,& Liu, H. (2000, April). Feature selection for clustering. In Pacific-Asia Conference on knowledge discovery and data mining (pp. 110-121). Springer, Berlin, Heidelberg.

[49] Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF* IDF, LSI and multi-words for text classification. Expert Systems with Applications, 38(3), 2758-2765.

[50] Bafna, P., Pramod, D., & Vaidya, A. (2016, March). Document clustering: TF-IDF approach. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 61-66). IEEE.

[51] Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.

[52] Rosli, N. R. M., & Yahya, K. (2017). Using Non-supervised Artificial Neural Network for Determination of Anthropogenic Disturbance in a River System. Tropical Life Sciences Research, 28(2), 189.

[53] Tatiraju, S., & Mehta, A. (2008). Image Segmentation using k-means clustering, EM and Normalized Cuts. Department of EECS, 1, 1-7.

[54] OMS | Nuevo coronavirus - China. (2020, January 15). Retrieved September 21, 2020, from https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/es/

[55] Eckert, A., & Higgins, D. (n.d.). Details - Public Health Image Library(PHIL). Retrieved September 21, 2020, from https://phil.cdc.gov/Details.aspx?pid=23311

[56] Suárez, R. (2020, March 11). Atención: La OMS declara pandemia por el nuevo coronavirus. Retrieved September 21, 2020, from https://www.eltiempo.com/salud/coronavirus-ya-es-una-pandemia-declara-la-oms-471524

[57] COVID-19 situation update worldwide, as of 17 September 2020. (2020, September 17). Retrieved September 21, 2020, from https://www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases

[58] Mapa de la enfermedad por coronavirus (COVID-19). (n.d.). Retrieved September 21, 2020, from https://google.com/covid19-map/?hl=en-ES

[59] AI, A. (2020, September 17). COVID-19 Open Research Dataset Challenge (CORD-19). Retrieved September 21, 2020, from https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

[60] What is R? (n.d.). Retrieved September 21, 2020, from https://www.r-project.org/about.html

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá