UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

ALGORITMOS DE CLUSTERING PARALELOS EN SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN DISTRIBUIDOS

TESIS DOCTORAL:

Presentada por: D. Daniel Jiménez González

Dirigida por: Dr. D. Vicente Emilio Vidal Gimeno

Valencia, mayo de 2011.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Agradecimientos

Quiero darle especialmente las gracias a mis padres que me han apoyado a lo largo de todo este tiempo, y que gracias a ellos soy quien soy y he llegado a donde estoy.

Gracias Marisa, por compartir conmigo esta importante parte de mi vida. Al principio desde muy lejos y ahora desde lo más cerca posible. Me has ayudado a que a pesar de las dificultades haya podido, creo yo, ir mejorando como persona.

También le quiero dar las gracias a mi tutor que ha tenido tanta paciencia durante estos largos años que han pasado desde que el proyecto de esta tesis surgio hasta ahora que es una realidad.

Tampoco puedo olvidar a todas esas personas que durante este tiempo me han acompañado, algunas se han quedado en el camino, otras se unieron a mitad y algunas de ellas han estado desde el principio al final. Todas ellas han sido importantes, pero creo que las que lo son especialmente, y ellas lo saben, se lo demuestro continuamente, aunque a lo mejor menos de lo que debería.

Resumen

La información es útil si cuando se necesita está disponible y se puede hacer uso de ella. La disponibilidad suele darse fácilmente cuando la información está bien estructurada y ordenada, y además, no es muy extensa. Pero esta situación no es la más común, cada vez se tiende más a que la cantidad de información ofrecida crezca de forma desmesurada, que esté desestructurada y que no presente un orden claro. La estructuración u ordenación manual es inviable debido a las dimensiones de la información a manejar. Por todo ello se hace clara la utilidad, e incluso la necesidad, de buenos sistemas de recuperación de información (SRI). Además, otra característica también importante es que la información tiende a presentarse de forma natural de manera distribuida, lo cual implica la necesidad de SRI que puedan trabajar en entornos distribuidos y con técnicas de paralelización.

Esta tesis aborda todos estos aspectos desarrollando y mejorando métodos que permitan obtener SRI con mejores prestaciones, tanto en calidad de recuperación como en eficiencia computacional, los cuales además permiten trabajar desde el enfoque de sistemas ya distribuidos.

El principal objetivo de los SRI será proporcionar documentos relevantes y omitir los considerados irrelevantes respecto a una consulta dada. Algunos de los problemas más destacables de los SRI son: la polisemia y la sinonimia; las palabras relacionadas (palabras que juntas tienen un significado y separadas otro); la enormidad de la información a manejar; la heterogéneidad de los documentos; etc. De todos ellos esta tesis se centra en la polisemia y la sinonimia, las palabras relacionadas (indirectamente mediante la lematización semántica) y en la enormidad de la información a manejar.

El desarrollo de un SRI comprende básicamente cuatro fases distintas: el preprocesamiento, la modelización, la evaluación y la utilización. El preprocesamiento que conlleva las acciones necesarias para transformar los documentos de la colección en una estructura de datos con la información relevante de los documentos ha sido una parte importante del estudio de esta tesis. En esta fase nos hemos centrado en la reducción de los datos y estructuras a manejar, maximizando la información contenida. La modelización, ha sido la fase más analizada y trabajada en esta tesis, es la que se encarga de definir la estructura y comportamiento del SRI. Solamente se ha trabajado sobre el modelo vectorial, dejando a parte otros modelos como el probabilistico y el lógico. En la fase de evaluación que se encarga de determinar la calidad del SRI, se han utilizado métodos ya definidos, ampliamente usados y corroborados, todos ellos basados directa o indirectamente en la precisión (*precision*) y la cobertura (*recall*).

Por último, en la tesis no se ha abordado la fase de utilización.

Debido a la gran cantidad de métodos de *clustering* existentes en multitud de ámbitos y para una extensa variedad de sistemas de información, se ha buscado trabajar a partir de dos de los principales y más importantes métodos de la literatura: *K-Means* y DBSCAN. Y, entonces, mejorar su calidad, intentando no perder su funcionalidad ni sus prestaciones computacionales, e incluso mejorándolas. Concretamente se ha desarrollado un método menos sensible que el *K-Means* a la inicialización de sus parámetros, *α-Bisecting Spherical K-Means*. También se ha desarrollado el método VDBSCAN que obtiene los mismos *clusters* que el DBSCAN pero en casi la mitad de tiempo y eliminando la elección aleatoria de los parámetros de inicialización cuando no se tiene información suficiente sobre el SRI (fijando a un valor constante uno de sus parámetros y el otro obteniéndolo de una forma heurística también desarrollada en esta tesis). Todos estos métodos se han creado con el objetivo de trabajar en entornos distribuidos y por ello una parte importante de la tesis se centra en los aspectos de paralelización.

Tras el estudio experimental de la calidad de recuperación de información y de las prestaciones computacionales se ha concluido que el método VDBSCAN obtiene una mejor calidad respecto al método *α-Bisecting Spherical K-Mean*s. Aunque el VDBSCAN tiene una modelización claramente más costosa, responde mejor a la paralelización. El tiempo de respuesta del *α-Bisecting Spherical K-Means* siempre es un poco más rápido que el del VDBSCAN. Así y todo el VDBSCAN obtiene mejores valores de *speed up* y sensiblemente mejores resultados de eficiencia. En conclusión, el VDBSCAN será elegido siempre que se considere primordial la calidad de recuperación. Mientras que el *α-Bisecting Spherical K-Means*, cuando la fase de modelización se repita muchas veces, por su menor coste computacional.

Resum

La informació és útil si quan es necessita està disponible i es pot fer ús d'ella. La disponibilitat ocorre fàcilment quan la informació està bé estructurada i ordenada, i a més a més, no és molt extensa. Però aquesta situació no és la més comú, cada volta la tendència és més a que la quantitat d'informació oferida cresca de forma desmesurada, que estiga desestructurada i que no presente un ordre clar. L'estructuració u ordenació manual és inviable per raó de les dimensions de la informació a manejar. Per tot això es fa clara l'utilitat, i fins i tot la necessitat, de bons sistemes de recuperació d'informació (SRI). També, altra característica important és que la informació tendeix a presentarse de forma natural de manera distribuïda, la qual cosa implica la necessitat de SRI que puguen treballar en entorns distribuïts i amb tècniques de paral·lelització.

Aquesta tesi tracta tots eixos aspectes desenvolupant i millorant mètodes que permeten obtindre SRI amb millors prestacions, tant en qualitat de recuperació com en eficiència computacional, els quals a més poden treballar des de la perspectiva de sistemes ja distribuïts.

El principal objectiu dels SRI serà proporcionar documents rellevants i ometre els considerats irrellevants respecte a una consulta donada. Alguns dels problemes més destacats dels SRI són: la polisèmia i la sinonímia; les paraules relacionades (paraules que juntes tenen un significat i serapades en tenen un altre); la gran quantitat d'informació a manejar; l'heterogeneïtat dels documents; etc. De tots ells, esta tesi es centra en la polisèmia i la sinonímia, les paraules relacionades (indirectament mitjançant la lematització semàntica) i en la gran quantitat d'informació a manejar.

El desenvolupament d'un SRI comprén bàsicament quatre fases diferents: el preprocessament, la modelització, l'evaluació i la utilització. El preprocessament que inclou les accions necessàries per a transformar els documents de la col·lecció en una estructura de dades amb la informació rellevant dels documents ha sigut una part important de l'estudi d'aquesta tesi. En aquesta fase ens hem centrant en la reducció de les dades i estructures a manejar, maximitzant la informació continguda. La modelització, ha sigut la fase més analitzada i treballada en aquesta tesi, és l'encarregada de definir l'estructura i comportament del SRI. Només s'ha treballat el model vectorial, deixant a banda altres models com el probabilístic i el lògic. En la fase d"evaluació que s'encarrega de determinar la qualitat del SRI, s'han utilitzat mètodes ja definits, ampliament usats i corroborats, basats directa o indirectament en la precisió (*precision*) i la cobertura (*recall*). A la fi, aquesta tesi no es fica en la fase d'utilització.

Degut a la gran quantitat de mètodes de *clustering* que existeixen en multitud

d'àmbits i per a una extensa varietat de sistemes d'informació, s'ha buscat treballar partint de dos dels principals i més importants mètodes de la lliteratura: *K-Means* i DBSCAN. I, aleshores, millorar la seua qualitat, intentant no perdre la seua funcionalitat ni les seues prestacions computacionals, i fins i tot millorar-les. Concretament s'ha desenvolupat un mètode menys sensible que el *K-Means* a la inicialització dels seus paràmetres, *α-Bisecting Spherical K-Means*. També s'ha desenvolupat el mètode VDBSCAN que obté els mateixos *clusters* que el DBSCAN però en quasi la meitat del temps i suprimint l'elecció aleatòria dels paràmetres de inicialització quan no es té informació suficient respecte al SRI (fixant a un valor constant un dels seus paràmetres i l'altre obtenint-lo heurísticament, mètode també desenvolupat en aquesta tesi). Aquests mètodes s'han creat amb l'objectiu de treballar en entorns distribuïts i per això una part important de la tesi es centra en aspectes de paral·lelització.

Després de l'estudi experimental de la qualitat de recuperació d'informació i de les prestacions computacionals s'ha conclós que el métode VDBSCAN obté una millor qualitat respecte al métode *α-Bisecting Spherical K-Mean*s. Encara que el VDBSCAN té una modelització clarament més costosa, respon millor a la paral·lelización. El temps de resposta del *α-Bisecting Spherical K-Mean*s sempre és una mica més ràpid que el del VDBSCAN. Així i tot el VDBSCAN obté millors valores de *speed up* i sensiblement millors resultats d'eficiència. En conclusió, el VDBSCAN serà elegit sempre que es considere primordial la qualitat de recuperació. Mentres que el *α-Bisecting Spherical K-Mean*s, quan la fase de modelització es repetisca moltes voltes, pel seu menor cost computacional.

Abstract

If when we need information, it is avaiable and we can use it, then information is useful. The avaiability is easy when information has good estructure and order, and it is not very large. But this situation is unusual, every time the amount of offered information tends to grow of extreme form, to be unstructured and to show unclear order. The manual structuration or order is unviable because the size we have to handle. So goods information retrieval (IR) systems are useful and even needed. Besides, another important characteristic is that the information appears on distributed way in its natural form, so the IR systems have to work on distributed environment and with paralelization methods.

This doctoral thesis deals all these aspects developing and improving methods to obtain IR systems with improve performances, in retrieval quality and computational eficiency too. Moreover, this methods can work on distributes systems already.

The main objective of IR systems is supply relevant documents and omit irrelevant respect to gived query. IR systems have various important handicaps, emphasizing: polysemy, synonyme; related words (two join words have a concret meaning and the same words when are separated have other meaning); etc. All these ones are deal in this doctoral thesis.

The development of IR system has four different basic phases: the preprocessing, the modelization, the evaluation and the utilization. The preprocessing presents needed actions to transform documents collection to data structure with documents relevant information. One important part of the doctoral thesis studies this phase, being centered in data and structures reduction, maximizing contained information. The modelization defines the structure and behaviour of IR system and this phase has been the most analyzed and developed phase. This doctoral thesis work about vectorial model, leaving outside other models as probabilistic and boolean. The evaluation determinates IR system quality. In this doctoral thesis we use already defined methods, widely used and tested. These methods are based directly or indirectly in the precision and recall. And the fourth phase is the utilization of the system, the doctoral thesis does not raise this phase.

It exists a very large number of clustering methods in multitude of fields and for an extensive variety of information systems, so we have started from the two main and most important methods of the literature: K-Means and DBSCAN. Later, we have tried to improve their quality and not to lose their funcionality and their computational performance. Specifically, we have developed a less sensitive method than

the K-Means in respect of the parameters initialization, the α-Bisecting Spherical K-Means. Also, we have developed the VDBSCAN method, which obtains the DBSCAN same clusters, with almost double quickness and eliminating aleatory selection of initializacion parameters when we have not enough information about the IR system (it fixes the first parameter on a constant valor and the second is obtained with an heuristic, developed in this doctoral thesis). All this methods have the objective of work on distributed environment, so an important part of the doctoral thesis discusses paralelization aspects.

After the experimental study of information retrieval quality and computational performances we could conclude that VDBSCAN method obtain better quality than α-Bisecting Spherical K-Means method. VDBSCAN have a more expensive modelization, but have a better behaviour in the paralelization. With respect to the evaluation time, α-Bisecting Spherical K-Means always is a little faster than VDBSCAN, but the last one obtain better values to the speed up and efficiency. In conclusion, the VDBSCAN method will be selected always that retrieval quality would be the most important thing. And the α-Bisecting Spherical K-Means method when the modelization phase is repeated very times, because it has a smaller computational cost.

Índice general

Lista de algoritmos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Objetivos

El principal objetivo de esta tesis es desarrollar métodos eficientes de *clustering* enfocados a obtener una buena calidad de recuperación de información, concretamente en sistemas formados por colecciones de documentos. Dichos métodos deben presentar unos tiempos de respuesta, al menos en la fase de evaluación de las consultas del usuario, más que aceptables. Y, además, tienen que ser capaces de funcionar mediante técnicas de paralelización en entornos donde la información de forma natural se encuentra distribuida.

Debido a la gran cantidad de métodos de *clustering* existentes en multitud de ámbitos y para una extensa variedad de sistemas de información, se ha buscado trabajar a partir de dos de los principales y más importantes métodos de la literatura: *K-Means* y DBSCAN. Y, entonces, aprovechar las características propias de los sistemas de recuperación de información para mejorarlos, intentando no perder su funcionalidad general. Más específicamente se ha trabajado con el objetivo de mejorar la calidad de dichos algoritmos sin perder prestaciones computacionales, e incluso mejorándolas.

Concretamente se ha querido desarrollar un método menos sensible que el *K-Means* a la inicialización de sus parámetros, conseguido ello con el método desarrollado: α-*Bisecting Spherical K-Means*. También se ha buscado mejorar en los posibles tipos de *cluster* que se pueden reconocer e incluso prácticamente eliminar la elección aleatoria de los parámetros de inicialización cuando no se tiene información suficiente sobre el sistema de información. Este aspecto no se ha conseguido a partir del método α-*Bisecting Spherical K-Means*, para ello se ha desarrollado el método VDBSCAN a partir del método DBSCAN.

Una parte importante de la tesis trabaja aspectos de paralelización, ya que los sistemas de recuperación de información suelen tender a presentarse distribuidos, debido a la naturaleza distribuida que en las colecciones de documentos en general se observa. Por lo tanto, es importante que los sistemas de recuperación de información sean capaces de trabajar en entornos distribuidos, y los métodos presentados en la tesis se han desarrollado con el objetivo de que puedan trabajar en este tipo de situaciones.

1.2. Estructura de la tesis

Buscando que los objetivos descritos y que los conceptos, métodos y experimentos presentados en la tesis queden lo más claro posible y que sea fácil su consulta y seguimiento, la tesis se ha estructurado en cuatro grandes bloques, además de un apartado dedicado a la conclusiones y trabajos futuros. Los cuatro apartados principales son los siguientes:

Los sistemas de recuperación de información: En este apartado se explican lo que son los sistemas de recuperación de información, para que sirven, cual ha sido su evolución, cuales son los principales problemas a los que se enfrentan, y luego con mayor detenimiento se explican cada una de las fases que podemos encontrar en los sistemas de recuperación: preprocesamiento, modelización, evaluación y utilización (evaluación de las consultas del usuario).

Algoritmos de *clustering*: Apartado en el cual se describen los distintos tipos de métodologías de *clustering* existentes. Luego con más detalle se presentan dos de los algoritmos de *clustering* más utilizados, el *K-Means* y el DBSCAN. Y el resto del apartado se dedica al desarrollo de los nuevos métodos presentados en esta tesis: el α-*Bisecting Spherical K-Means* y el VDBSCAN, como variantes de los dos algoritmos básicos antes descritos.

***Clustering* en sistemas distribuidos:** Tras una breve introducción, se explican las distribuciones posibles y cual es la elegida y por qué. Luego se presentan las versiones paralelas de todos los algoritmos que a lo largo de la tesis se usan. En este apartado no se presenta ningún estudio comparativo entre versiones, se dedica al desarrollo y explicación de los mismos, así como a la justificación de éstos aún cuando puedan existir versiones distribuidas de los métodos básicos.

Estudios experimentales: Como su nombre indica, en este apartado se han englobado los estudios experimentales de las prestaciones de los algoritmos desarrollados, tanto desde el punto de vista de la calidad de recuperación de información, como de los costes computacionales, secuenciales y paralelos. También se presenta un subapartado explicativo de las colecciones de prueba utilizadas para determinar la calidad de los distintos métodos y se finaliza con otro subapartado a modo de resumen de las conclusiones que se pueden extraer de las distintas comparaciones experimentales entre los diferentes métodos.

En el apartado de conclusiones se han incluido las conclusiones que se pueden extraer del trabajo presentado en esta tesis, así como las líneas de trabajos futuros que se abren a partir de ella.

1.3. Estado del arte

Aunque el estado del arte se desarrolla a lo largo de toda la tesis incluyéndolo dentro del desarrollo de cada apartado, se ha querido presentar también un punto

general del mismo. Existen varias revisiones del estado del arte ofreciendo un aporte más extenso y detallado de los distintos métodos y enfoques en la recuperación de información, entre los cuales podemos destacar [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

1.3.1. Los sistemas de recuperación de información

Desde sus origienes el ser humano ha sabido de la necesidad de organizar y extraer información. Pero en cada momento de su historia lo ha realizado de una forma u otra, en función de la información y en función de los medios disponibles. La disponibilidad de la información mejora cuando está bien estructurada, ordenada y no es muy extensa. Hace decadas era una situación común, al encontrarse la información en bibliotecas y archivos. Pero al crecer el volumen de información considerablemente, fue necesario construir estructuras de datos especializadas para asegurar un acceso rápido a la información almacenada. Los índices son una antigua y popular estructura de datos, que de una forma u otra, es el corazón de todos los sistemas de recuperación de información modernos.

Los sistemas de organización de las bibliotecas, basados fundamentalmente en índices, se han ido refinando durante mucho tiempo, de hecho se puede considerar que éstas fueron de las primeras instituciones en adoptar sistemas de recuperación de información, aunque éstos eran muy diferentes a los actuales [11, 12], incluso en 1995 se trabajo en su paralelización [13].

Se pueden ver tres generaciones en el desarrollo de los sistemas de recuperación de información [11]:

Primera generación: Automatización de tecnologías previas, por ejemplo catálogos de tarjetas. Permitían búsquedas básicas basadas en el título y autor del documento.

Segunda generación: Inclusión de nuevas formas de búsqueda: por títulos de capítulo, palabras claves, etc. Tambien se facilita la realización de consultas más complejas.

Tercera generación: Mejora de interfaces con el usuario, de modelos, de visualización de la información, de la clasificación y categorización de los documentos, etcétera. Esta se puede considerar la generación actual.

Con la aparición de los sistemas de información digitales, sobre todo de la *World Wide Web*, a principios de los noventa, la cantidad de información disponible para ser compartida sufrió un cambio sin precedente alguno. Lo cual conllevó un nuevo problema, la búsqueda de dicha información se convirtió en difícil y tediosa, ya que ésta es estructuralmente de baja calidad. De hecho aunque pueda parecer que los actuales motores de búsqueda de la *Web* utilizan métodos de indexación similares a los empleados en las bibliotecas de hace siglos, es una mera apariencia. Existen trabajos enfocados directamente a la *Web* como por ejemplo [14, 15]. Son fundamentalmente tres los cambios debidos al avance tecnológico de la informática moderna y la explosión de la *Web* [11]:

1. De forma muy barata se puede acceder a la información de varias fuentes.
2. A la información se puede acceder de manera muy rápida incluso a pesar de las distancias físicas, gracias a la comunicación digital.
3. La libertad de la gente a incluir la información que considera útil a disposición del resto de personas.

Este entorno conlleva la necesidad de una herramienta que facilite que se pueda extraer la información que sea de interés. La investigación en este campo ha seguido distintas direcciones en función del tipo de información buscada o de si se pretende la organización de la información.

Minería de datos (*Data Mining*): Busca datos concretos entre grandes cantidades de información [3].

Navegación (*Browsing*): Realiza una búsqueda interactiva a través del sistema de información cuando la información buscada es amplia o no está claramente definida [4, 11].

Clasificación: Agrupa de forma supervisada o no supervisada la información en conjuntos relacionados directamente [3, 4].

Filtrado (*Filtering*): Filtra según unas consultas predefinidas la información que continuamente llega [4, 11].

Recuperación de información (*Information Retrieval*): Obtiene de un sistema de información más o menos estático la información relevante a una consulta puntual dada [4, 11].

Los primeros aportes serios de paralelización surgen en 1994, cuando se intenta introducir la paralelización de sistemas de recuperación de información en las bases de datos [16] y en 1995, cuando se trabaja en la paralelización de índices y de algunos métodos de *clustering* (*Fuzzy Covarianze* y *Affinity Decomposition*) sobre distintas arquitecturas (maquinas vectoriales, computadores de memoria distribuida y memoria compartida) [13].

El uso de la recuperación de información se ha llevado a varios campos distinto del puramente textual. Por ejemplo, en [17] se une el reconocimiento de voz y la recuperación de información en un único sistema, el cual recibe las consultas habladas y devuelve los documentos escritos relevantes a dicha consulta. Y en [18, 19, 20] se ha intentado llevar a los en sistemas de imágenes.

1.3.2. Los paradigmas de la recuperación de información

Durante mucho tiempo la información se ha organizado manualmente como jerarquías categorizadas [11]. Forma que hoy en día todavía se utiliza en muchos lugares, sobre todo en bibliotecas. Actualmente los ordenadores han posibilitado la construcción de grandes índices de forma automática. Y aunque los sistemas de indexación

manual, a pesar de sus problemas, trabajan muy bien, la proliferación de gran cantidad de información ha provocado la necesidad de los sistemas automatizados [12].

Un claro ejemplo de que la organización manual es, también hoy en día, una posible solución a la organización y estructuración de la información se ve al adoptarse para organizar una parte de Internet, concretamente la *Web* www.yahoo.com [21]. De todas formas este proceso sólo es viable cuando se dispone de grandes recursos.

A medida que ha ido pasando el tiempo y el problema de la recuperación de información se ha ido conociendo y estudiando más profundamente, han ido apareciendo otros paradigmas de modelado alternativos a los clásicos (lógico, vectorial y probabilístico). Dentro del paradigma lógico destacan los modelos de lógica difusa (*Fuzzy*) y el modelo lógico extendido. El paradigma algebraico incluye el modelo vectorial generalizado, la indexación semántica latente (*Latent Semantic Index*ing) y las redes neuronales. Las alternativas del modelo probabilístico son las redes de inferencia (*Inference Network*) y las redes de convicción (*Belief Network*). Los métodos de *clustering* se podrían englobar en cualquiera de los paradigmas en función del modelo en el que se basen, aunque habitualmente trabajan dentro del modelo vectorial.

Durante la historia de los sistemas de recuperación de información han habido multitud de intentos de unir las características de distintos paradigmas. Por ejemplo: en [22] se intenta unir el LSI y el probabilistico, en [23, 24] se intenta relacionar la matriz de vectores conceptos obtenida a partir del *Spherikal K-Means* con la matriz de bajo rango obtenida con el modelo LSI utilizando la SVD, en [25] se quiere unir el modelo LSI con la formulación de consultas boleanas. E incluso de unir varios métodos indistintamente cuales sean éstos, al centrarse en como combinar los resultados de varios métodos de recuperación para unificar y dar sólo una relación de documentos relevantes, aún cuando cada método puede dar lugar a un conjunto distinto [26].

Diferentes trabajos sobre el modelo LSI han ido apareciendo, destacan [27, 28, 29], más tarde se afrontan problemas específicos y surgen distintos tipos de optimizaciones. En [30] se presenta un estudio para optimizar la SVD mediante descomposiciones semidiscretas, el cual a su vez se basó en un anterior trabajo [31]. En [32] se plantea la utilización de alternativas a la SVD para mejorar computacionalmente las descomposiciones de bajo rango. En [33] se presenta una nueva implementación del modelo LSI mejorando implementaciones anteriores. En [14] se estudian diferentes alternativas basadas en la LSI enfocadas a un entorno Web. En [34] se afronta el problema de determinar el rango buscado en la LSI, concretamente se basan en cálculos de la SVD de muestras de la matriz de pesos.

1.3.3. El *clustering* en la recuperación de información

Las técnicas de *clustering* trabajan directamente con la colección de documentos y no con el texto de los documentos [11]. Esta es la razón por la que el modelo de *clustering* no pertenece a ninguno de los paradigmas de recuperación de información (lógico, vectorial o probabilístico). Incluso se pueden encontrar trabajos basados en modelos de grafos que presentan técnicas de *clustering* [35, 36, 37].

Inicialmente el *clustering* fue investigado para mejorar la precisión y la cobertura (*recall*) de los sistemas de recuperación de información y como un camino eficiente

de encontrar los vecinos más próximos de un documento. Después, fue propuesto para usarse en la navegación por las colecciones de documentos, o en la organización de los resultados de una consulta, o en la generación automática de jerarquías de documentos, o en el análisis de conceptos latentes en conjuntos de documentos desestructurados, etc. [23, 38].

En la literatura existe una vasta colección de algoritmos de *clustering* disponibles, lo cual puede a priori confundir a la hora de seleccionar cual es el más conveniente para un problema dado, pero no hay que olvidar que no existe una técnica de *clustering* universal que se pueda aplicar a todos los conjuntos de datos [3, 39]. No todas las técnicas de *clustering* pueden encontrar todos los *clusters*, y aún pudiéndolos encontrar no todos lo hacen con la misma facilidad. Esto es debido a las asunciones implícitas que cada algoritmo hace, tales como la forma de los *clusters* buscados, la configuración de múltiples *clusters* o los criterios de agrupación entre *clusters* [3, 39]. Toda esta multitud de métodos de *clustering* disponibles en la literatura comúnmente se divide en dos tipos: [3, 4, 5, 6, 40]:

Particional: Construye particiones, donde cada *cluster* optimiza un criterio de *clustering*. Se caracterizan por su alta complejidad, algunos incluso enumeran exhaustivamente todas las posibles particiones intentando encontrar el óptimo global. Normalmente se empieza con una solución inicial aleatoria y luego se refina.

Jerarquico: Crea una descomposición jerárquica. Se subdividen a su vez en aglomerativos y divisivos. Los aglomerativos parten siendo cada documento un *cluster* independiente y sucesivamente los mezclan de acuerdo a una medida de distancia. Los divisivos parten de un único *cluster* formado por toda la colección y sucesivamente lo divide en grupos más pequeños según algún criterio hasta que cada documento forma un único *cluster*.

Un método jerárquico se puede utilizar como un método particional y el proceso inverso también es posible, de una técnica particional se pueden obtener resultados jerárquicos [38, 41]. A parte de estas dos principales categorías de algoritmos de clustering han surgido otros métodos, los cuales suelen estar enfocados a problemas o conjuntos de datos específicos [3, 5, 6]:

Clustering basado en densidades: Los elementos vecinos se agrupan en un mismo conjunto basándose en funciones objetivo específicas de densidad (número de objetos en un vecindario particular).

Clustering basado en grid: Están pensados principalmente para datos espaciales (por ejemplo, los que modelan estructuras geométricas). El procedimiento que siguen es dividir el espacio en un número finito de celdas y entonces trabajar con dichas celdas y su contenido.

Clustering basado en modelos: Se utilizan para encontrar buenas aproximaciones de los parámetros del modelo que mejor define los datos. Están bastante cerca de los algoritmos basados en densidades aunque no tienen por que usar el mismo concepto de densidad.

Clustering de información categórica: Están especialmente desarrollados para datos categóricos donde la medida de distancia Euclídea u otras numéricas no pueden ser aplicadas.

En [42] se presenta un estudio comparativo de varios métodos de *clustering* para analizar el tráfico de una red, entre ellos se encuentra el DBSCAN y el *K-Means*. Los resultados de este trabajo presentan al DBSCAN como el mejor de los métodos comparados. Este estudio es un ejemplo del buen comportamiento de los métodos de *clustering* presentados en esta tesis en otros campos distintos a los sistemas de recuperación de información.

1.3.4. La familia del *K-Means*

El algoritmo *K-Means* es la técnica iterativa particional más ampliamente usada [43, 44], y el algoritmo *K-Means* típico ha sido varias veces presentado [3, 6, 38, 45].

A lo largo del tiempo se han presentado diversas mejoras y enfoques del *K-Means* y sus principales variantes. Una mejora típica es intentar alcanzar el mínimo global, en lugar de uno local intentando para ello mejorar la elección de los centros iniciales [3]. Otra mejora importante es permitir separar y juntar *clusters* en función de dos umbrales, separando cuando la varianza del *cluster* supera el umbral determinado y juntando cuando los centros de dos *clusters* superan el umbral de cercanía [3]. Otra posible mejora es actualizar los centros de los *clusters* incrementalmente, es decir a medida que los documentos se asignan a los *clusters* [38]. También se ha intentado mejorar el método de inicialización, tanto basándose en modos de distribución probabilísticos [46], como en la densidad de puntos de las celdas en las que se distribuye la región de puntos con los que se trabaja [47] (este trabajo se centra en una baja dimensionalidad, concretamente en dos dimensiones). En [48] se intenta acelerar el método *K-Means* mediante razonamiento geométrico usando la estructura *kd-tree* (también utilizada anteriormente en [49] para también intentar mejorar el *K-Means*), las pruebas realizadas llegan sólo hasta datos de cinco dimensiones. En [43, 44] se aporta una técnica alternativa a las clásicas de elejir el elemento a dividir en los métodos basados en divisiones como el *Bisecting K-Mean*. En [21] se trabaja en una implementación eficiente de un sistema completo de *clustering* basado en el *Spherical K-Means*. En [50] se extiende el *K-Means* para que pueda trabajar con distintas características (además de la habitual, la aparición de términos en el documento, se puede utilizar también, por ejemplo, los enlaces que aparecen en las páginas *Web*). Este aspecto lo afrontan usando tuplas de características en la modelización.

El algoritmo *K-Means* has sido paralelizado muchas veces (por ejemplo en [51, 52, 53, 54]) partiendo en la mayoría de los casos de un conjunto de datos centralizado o repetido. Es decir, o los datos residen en un servidor central y entonces se distribuyen entre los diferentes elementos de proceso, o cada elemento de proceso presenta una copia de todos los datos. La filosofía seguida en esta tesis parte de que la colección de documentos se encuentra ya distribuida entre los distintos elementos de proceso.

Estudiando un poco el algoritmo *K-Means* en seguida se encuentra una filosofía de paralelización básica y sencilla, operaciones de suma y producto vectoriales en

paralelo y una operación de reducción global, la cual también sirve como método de sincronización. Ésta es la línea que han seguido en general las versiones paralelas que se encuentran en la literatura (ver por ejemplo en [51, 52, 53]) y es también la que se seguirá en esta tesis. En [55, 56] se presenta una generalización de cómo paralelizar los métodos basados en centroides, tales como el *K-Means*.

1.3.5. La familia del DBSCAN

Tras su aparición en 1996 comparándose con el CLARANS, el DBSCAN [40] ha dado lugar a diferentes variantes y mejoras destacando las siguientes:

GDBSCAN: Versión generalizada del DBSCAN que permite trabajar con datos numéricos y con atributos categóricos [57].

DBCLASD: Variante del DBSCAN que permite trabajar sin ningún parámetro de entrada, al asumir que los objetos dentro de un *cluster* son distribuidos uniformemente [58, 59]. Estudios muestran que el DBSCAN es más rápido (tendiendo al doble) que el DBCLASD [59].

DBSCAN Incremental: Se ha desarrollado una versión incremental eficiente [60].

DBSCAN Paralelo: Existe una versión paralela del DBSCAN la cual empieza con el conjunto de datos residente en un servidor central y entonces distribuye la información a los diferentes clientes [61]. También existe el DBDC (*Density Based Distributed Clustering*) una versión distribuida, que parte de la colección ya distribuida, entonces hace un *clustering* local y envía información del modelo local a un servidor central, donde se genera un modelo global a partir de los locales, el cual es distribuido para actualizar los modelos locales [62, 63, 64].

OPTICS: Extensión del DBSCAN que relaja el requerimiento estricto de los parámetros de entrada, computando una descomposición jerárquica de *clusters* basada en un rango de configuración de los parámetros [65, 66]. El DBSCAN y el OPTICS son similares en estructura y presentan la misma complejidad computacional [6]. Este método aunque tiene el proposito del análisis de *clusters* no construye explícitamente los *clusters*, en su lugar crea un orden que representa la estructura de *clustering* basada en densidades [65]

Más recientemente han aparecido bastantes trabajos enfocados al DBSCAN, a su mejora, a su extensión a otros entornos y formas de aplicación, a su aprovechamiento para mejorar otras herramientas, etc. A modo de ejemplo citaremos los siguientes. En [67, 68, 69] trabajan con subespacios de densidades basados en métodos de densidades como el DBSCAN. En [70] se presenta una ampliación y mejora del DBSCAN, basado en el trabajo con muestras, igualmente que en [71] que se pretende mejorar el tiempo de respuesta del DBSCAN basándose en cálculos parciales sobre muestras. Y buscando también la aceleración en [72, 73] reducen las comprobaciones sobre el vecindario de los elementos, y en [73] se presenta una variante que basándose en ciertas ordenaciones lo acelera. En [74] se trabaja en un método basado en el modelo de densidades para

objetos representados de varias formas, y en [75] se presenta un nuevo método basado en el DBSCAN para descubrir *clusters* de acuerdo a valores no espaciales, espaciales y temporales de los objetos. En [76] se intentan hacer métodos híbridos del modelo basado en densidades junto con *grid*, en [77] con un método de *clustering* de líderes, y en [78] se conjugan con los k-vecinos más próximos.

También es interesante observar como en la literatura cada vez aparecen más métodos para diferentes campos que se basan en el DBSCAN principalmente [67, 68, 74, 76, 79, 80, 81, 82, 83, 84] (y en ciertos casos de alguna de sus variantes, sobre todo del OPTICS [74, 75, 85]).

1.4. Justificación de objetivos

Una vez expuestos los objetivos, la estructura de la tesis y el estado del arte, se van a relacionar los tres apartados, concretando los objetivos en relación a las áreas de investigación dónde se quieren mejorar aspectos, a qué contribuciones se quieren hacer y a cómo se abordan éstos a lo largo de la tesis.

1. Mejorar dos de los principales y más importantes métodos de la literatura, *K-Means* y DBSCAN. Los métodos de *clustering* existentes suelen estar enfocados y funcionar en entornos muy específicos, dependiendo mucho del conocimiento de dicho entorno para seleccionar correctamente los parámetros del método. Por ello, se ha querido trabajar con dos de los métodos más generalistas y aprovechar sus características y ventajas y mejorar aquellos puntos más débiles. Para ello, al hablar de los métodos de *clustering* (ver apartado 3 en la página 71) se han estudiado las cualidades de éstos métodos y como en la literatura se han afrontado posibles mejoras. Luego se han desarrollado variantes de los mismos para mejorarlos.

2. Desarrollar un método menos sensible que el *K-Means* a la inicialización de sus parámetros, el *α-Bisecting Spherical K-Means*. Aunque el *K-Means* es uno de los métodos más importantes de la literatura, debido principalmente a su simplicidad, velocidad y generalidad, tiene un problema muy importante, depende mucho de la elección de sus parámetros. Por ello, se ha desarrollado el método *α-Bisecting Spherical K-Means* que aprovecha las características de otros métodos también conocidos como el *Bisection* y que gracias a un enfoque algo diferente minimiza la sensibilidad del algoritmo al seleccionar los parámetros de entrada. Todo esto se trabaja fundamentalmente en el apartado 3.3 en la página 82.

3. Eliminar la elección aleatoria de los parámetros de inicialización cuando no se tiene información suficiente sobre el sistema de información, desarrollando el método VDBSCAN a partir del método DBSCAN. Nuevamente el principal problema de los métodos de *clustering* generales que obtienen buenas prestaciones es la elección correcta de sus parámetros de entrada cuando se desconoce a priori información sobre la colección (o conjunto de datos) sobre los que se trabaja. Y esto ocurre también con el DBSCAN. Por ello, se ha desarrollado un

método alternativo el VDBSCAN basado en el DBSCAN que mantiene todas las cualidades del DBSCAN pero que mejora en tiempos de respuesta y que prácticamente no necesita ningún parámetro de entrada. Todo esto se trabaja fundamentalmente en los apartados 3.4 en la página 85 y 3.5 en la página 90.

4. Desarrollar métodos capaces de trabajar en entornos distribuidos. La paralelización y los sistemas distribuidos son áreas de investigación muy importantes a la hora de mejorar tiempos de respuesta o de posibilitar abordar problemas que de otra forma serían inviables. Pero también es una área muy importante cuando el problema planteado es de naturaleza distribuida, situación bastante natural de los sistemas de recuperación de inforamción. Por ello, todos los métodos desarrollados lo han sido con la capacidad de trabajar en sistemas paralelos sin afectar a la propia naturaleza de la distribución de los datos y mejorando en la medida de lo posible las prestaciones temporales y computacionales. Este aspecto se ha trabajado fundamentalmente en el apartado 4 en la página 105.

Capítulo 2

Los sistemas de recuperación de información

2.1. Aspectos generales

2.1.1. Qué vamos a entender por sistema de recuperación de información

El concepto de sistema de recuperación de información es muy amplio, por ello vamos a centrar lo que se va a entender a lo largo de esta tesis por sistema de recuperación de información, ya que éstos se pueden caracterizar por multitud de factores:

- El formato de la información contenida: textual, sonora, multimedia, mixta, etc.
- El medio en el que se almacena la información: bases de datos, ficheros de texto, ficheros de imágenes, libros, mixtos, etc.
- La vigencia de la información: estática, se añade información continuamente, se actualiza la información continuamente, etc.
- El tipo de información que se desea extraer: información concreta y puntual, información general o documental sobre un tema, etc.
- Y muchos otros factores.

En concreto en esta tesis se va a trabajar con sistemas formados por documentos textuales, sin una estructura determinada, almacenados digitalmente. Por tanto, se puede ver el sistema de información como una colección de documentos digitales, pudiendo éstos representar a libros, artículos, patentes, resúmenes, etc. Además, se presupone que dicha colección no varía sustancialmente en su contenido. Pueden añadirse documentos de forma esporádica y no suelen eliminarse ninguno, consecuentemente se puede considerar estática. La información que se pretende obtener de dicho sistema

es la relación de documentos relevantes incluidos en la colección. O en otras palabras, una consulta (la información buscada) dará lugar a una relación de documentos relevantes (la parte relevante de la información disponible) [31].

En estas condiciones el sistema de recuperación de información no proporciona una solución a la búsqueda de información sobre un tema concreto, sino que es preciso "interpretar" el contenido de los documentos relevantes obtenidos. En los sistemas de información el concepto quizá más importante es el de relevancia ya que el principal objetivo del mismo es proporcionar aquellos documentos que se consideran relevantes sin recuperar los considerados irrelevantes (no relevantes) a partir de una consulta [11]. La medida de relevancia se estudia principalmente teniendo en cuenta la información sintáctica o léxica que presenta el texto, a veces también se tiene en cuenta la información semántica e incluso estructural del documento. Al ser tan importante la relevancia también ocasionará los principales problemas de los sistemas de recuperación de información.

2.1.2. ¿Para qué un sistema de recuperación de información?

Para entender la utilidad, o incluso se podría decir la necesidad, de los sistemas de recuperación de información hay que tener claro que la información es útil si cuando se necesita está disponible y se puede hacer uso de ella. La disponibilidad de la información está directamente relacionada con el esfuerzo necesario para obtenerla. Es bastante importante para mejorar la disponibilidad que la información esté bien estructurada y ordenada, y preferiblemente no ser muy extensa. Normalmente esta situación no es la más común, en cambio, hace años sí lo era, ya que la información principalmente se encontraba en bibliotecas o archivos bien estructurados y ordenados, siendo además de tamaños aceptablemente pequeños. Dicha ordenación y estructuración se debe fundamentalmente a un trabajo manual previo, sólo posible al ser la cantidad de información manejable.

Pero, cada vez más, se tiende a que la cantidad de información ofrecida crezca de forma desmesurada, que, además, esté desestructurada, al menos a nivel global, y que tampoco presente un orden claro, cuando lo tiene. A todo ello hay que añadir que no es viable la estructuración u ordenación manual de la información, debido a las dimensiones de la misma. Esto se ve claramente al tratar con la información ofrecida por la *World Wide Web*, o la multitud de *intranets* existentes, o cualquier otro entorno de trabajo con documentación digital: revistas científicas, patentes, etc.

Dentro de este contexto se hace precisa una herramienta, en su amplio sentido, que ayude a extraer la información que en un momento dado sea de interés. Dependiendo del tipo de información que se quiera obtener o si lo que se pretende es organizar de forma más global la información se ha investigado en distintas direcciones:

Minería de datos (*Data Mining*): Pretende buscar datos específicos entre grandes cantidades de información [3].

Navegación (*Browsing*): Ante una necesidad de información amplia o no claramente definida, no concreta, se pretende realizar una búsqueda interactiva por el sistema de información [4, 11].

Clasificación: Pretende agrupar la información en conjuntos directamente relacionados. Puede ser de forma supervisada donde se conocen los patrones que definen cada grupo, o no supervisada donde no se conocen dichos patrones. [3, 4].

Filtrado (*Filtering*): Pretende seleccionar la información que continuamente está llegando en función de ciertas consultas predefinidas o al menos relativamente estáticas [4, 11].

Recuperación de información (*Information Retrieval*): Pretende obtener de un sistema de información más o menos estático la información relevante a una consulta puntual dada [4, 11].

En conclusión, los sistemas de recuperación de información son la herramienta que cuando la cantidad de información donde buscar es enorme, permite que la obtención de información relevante tenga buenas prestaciones, tanto en tiempo de respuesta como en calidad de respuesta. No hay que olvidar que grandes cantidades de información no son útiles a menos que la información se pueda buscar de manera eficiente y efectiva [31].

2.1.3. Evolución de los sistemas de recuperación de información

Desde siempre el ser humano ha sido consciente de la necesidad de organizar y extraer información. Pero en cada momento de su historia lo ha realizado de una forma u otra, en función de la información y en función de los medios disponibles.

En el momento que el volumen de información creció más allá de unos pocos libros, fue necesario construir estructuras de datos especializadas para asegurar un acceso rápido a la información almacenada. Una antigua y popular estructura de datos es el índice. El cual, de una forma u otra, es el corazón de todos los sistemas de recuperación de información modernos [11].

Durante décadas, incluso centurias, de experiencia se han refinado los sistemas de organización en las bibliotecas, donde todo está catalogado en base a alguna valoración individual o de grupo, y todo ello completado con las apropiadas referencias en catálogos [12]. Las bibliotecas fueron de las primeras instituciones en adoptar sistemas de recuperación de información. Aunque lógicamente éstos eran muy diferentes a los actuales. Se pueden ver tres generaciones en el desarrollo de los sistemas de recuperación de información [11]:

Primera generación: Los sistemas consistían en automatizar tecnologías previas, tales como los catálogos de tarjetas, y básicamente permitían búsquedas basadas en el nombre del autor y el título del documento.

Segunda generación: Los sistemas incrementaron su funcionalidad añadiendo la posibilidad de buscar por títulos de capítulo, por palabras clave, y añadiendo también algunas facilidades de consulta más complejas.

Tercera generación: Se puede considerar la actual y se está enfocando hacia la mejora de interfaces con el usuario, características de hipertexto, modelos, visualización de la información, clasificación y categorización de documentos, etc.

Con la aparición, a principios de los noventa, de la *World Wide Web* hubo un cambio sin precedente alguno en el volumen de la información disponible para ser compartida, ya que guarda el conocimiento universal y la cultura humana, permitiendo compartir ideas a una escala grandiosa. De todas formas este avance a conllevado un nuevo problema. La tarea de búsqueda de información en la *Web* se ha convertido frecuentemente en difícil y tediosa, ya que no existe un modelo de datos subyacente bien definido, dando lugar a información estructuralmente de baja calidad.

Aunque en un principio pueda parecer que los actuales motores de búsqueda de la *Web* utilizan métodos de indexación similares a los empleados en las bibliotecas de hace siglos, es una mera apariencia ya que existen fundamentalmente tres cambios debidos al avance tecnológico de la informática moderna y la explosión de la *Web* [11]:

1. El acceso a varias fuentes de información se ha vuelto muy barato, permitiendo alcanzar mayor audiencia.

2. Los tipos de comunicación digital con su avance tecnológico permiten un mayor acceso a las redes de conexión, posibilitando que las fuentes de información estén disponibles con accesos rápidos incluso a pesar de las distancias físicas.

3. Ha crecido la libertad existente en la colocación de la información que la gente considera útil a disposición del resto de personas, popularizando así la *Web*.

La mayoría de los sistemas de recuperación de información comerciales utilizan el modelo lógico o booleano extendido [25]. Pero estos sistemas no resuelven las necesidades de información reales. Todavía se debe avanzar bastante en varios puntos. A pesar de las mejoras actuales, todavía es difícil, e incluso en ocasiones imposible, recuperar información relevante ante ciertas necesidades de información. Se han de encontrar técnicas que permitan una recuperación de información de alta calidad. Además, en muchas ocasiones se tarda mucho en conseguir la información relevante. Se precisan técnicas con tiempos de respuesta cortos. Y también, hay que tener en cuenta la separación semántica entre lo que se quiere buscar y lo que se busca (por problemas de transformación de una idea en palabras, etc.). La solución pasa por la interacción con el usuario para ofrecer algún tipo de realimentación en la estrategia de búsqueda.

2.1.4. Organización manual o automática de la información

Existen dos formas fundamentales de organizar la información, la realizada manualmente por una o varias personas, y la realizada de forma automática mediante una máquina. También se podrían estudiar situaciones mixtas, por ejemplo una organización automática supervisada y perfeccionada por humanos, pero no son habituales.

Las características principales de la indexación u organización manual se deben a aspectos propios del ser humano, tanto para bien como para mal. La principal ventaja radica en que un revisor humano puede establecer relaciones entre conceptos

aparentemente diferentes pero que más tarde pueden ser útiles para recuperar información relevante [12]. En contrapartida dicha ventaja se puede convertir en algo negativo, ya que la relación establecida entre conceptos puede ser demasiado subjetiva, influye mucho los conocimientos, la experiencia, la opinión y la personalidad del revisor. La decisión sobre las palabras clave importantes y los conceptos puede basarse en atributos como la edad, la formación cultural, la educación, la lengua, e incluso las ideas políticas del evaluador. De hecho existen estudios que han mostrado que hay un 20 % de disparidad de media en los términos elegidos como apropiados por dos indexadores profesionales independientes para describir un documento dado [29]. De todas formas, la principal desventaja de la organización manual reside en que precisa de mucho tiempo, además de ser cara. Tampoco hay que olvidar que una indexación manual puede ser muy difícil de reproducir o reconstruir [12].

Análogamente, la caracterización de la organización automática también está directamente relacionada a las características propias de los ordenadores. La principal ventaja es la velocidad y el volumen de información que puede procesar. Pero en contrapartida, le es muy difícil establecer relaciones entre conceptos, e incluso entre términos (por ejemplo, sinónimos). Además, el desarrollo de buenos algoritmos de clasificación, sobre todo los de carácter general, es también una tarea complicada.

A las diferencias entre la organización manual y la automática debidas a sus propias características, también existe una diferencia significativa respecto a la visión del problema. La indexación automática provee una visión del problema de recuperación de información más relacionado con el sistema en sí mismo que con las necesidades del usuario [11].

Durante mucho tiempo la información se ha organizado manualmente como jerarquías categorizadas [11]. Forma que hoy en día todavía se utiliza en muchos lugares, sobre todo bibliotecas. Actualmente los ordenadores han posibilitado la construcción de grandes índices de forma automática. Y aunque los sistemas de indexación manual, a pesar de sus problemas, trabajan muy bien, la proliferación de cantidad de información ha provocado la necesidad de los sistemas automatizados [12].

Un claro ejemplo de que la organización manual es, también hoy en día, una posible solución a la organización y estructuración de la información es que ha sido adoptada para organizar una parte de Internet, concretamente la *Web* www.yahoo.com [21]. De todas formas este proceso sólo es viable cuando se dispone de grandes recursos.

2.1.5. Fases de un sistema de recuperación de información

El desarrollo de un sistema de recuperación de información comprende básicamente cuatro fases distintas. Las tres primeras corresponderían al desarrollo propiamente dicho del sistema y serían: el preprocesamiento, la modelización y la evaluación. En la figura 2.1 en la página siguiente se muestran las dos primeras fases, que corresponderían con la creación propiamente dicha del sistema de recuperación de información. La cuarta fase sería la de utilización del sistema para la obtención de información relevante a partir de consultas.

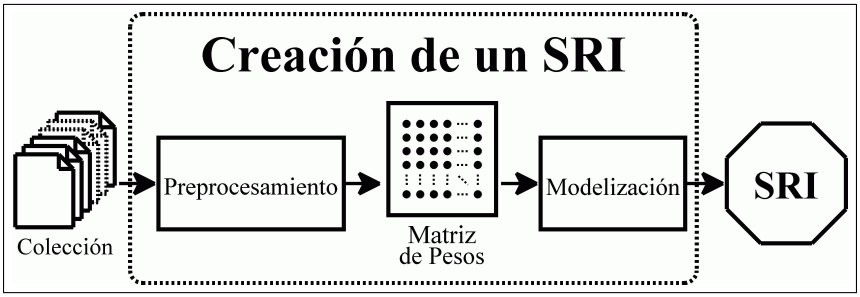

Figura 2.1: Creación de un sistema de recuperación de información.

Preprocesamiento

El preprocesamiento es la primera fase en la construcción de un sistema de recuperación de información. Básicamente durante esta etapa se hacen todas las acciones necesarias para transformar los documentos de la colección en una estructura de datos con la información relevante de los documentos que será la base para la modelización.

Fundamentalmente en el preprocesamiento tienen lugar cinco transformaciones diferentes [11], no siempre se tienen que dar todas:

1. Análisis léxico del texto con el objetivo de identificar los términos que forman el documento. Puede haber un tratamiento especial para los dígitos, las palabras con guiones, los signos de puntuación, etc.

2. Eliminación de las palabras o términos carentes de significado, conocidas en la literatura como *stopwords* con el objetivo de eliminar palabras con muy poco valor discriminante para los propósitos de recuperación.

3. Obtención de la raíz de las palabras restantes, conocido en la literatura como *stemming*, con el objetivo de eliminar afijos (prefijos y sufijos) y permitir la recuperación de documentos que contienen variaciones sintácticas de los términos de la consulta. También se puede trabajar con la extracción de la raíz semántica (lematización) [86, 87, 88], aunque esta línea está mucho menos trabajada.

4. Selección de los términos índice, que pueden ser: palabras, raíces de palabra, números, grupos de palabras, fechas, etc. Éstos términos serán usados como elementos de indexación. La decisión sobre qué elementos serán usados como términos índices puede estar relacionada con la naturaleza sintáctica de la palabra (un sustantivo frecuentemente presenta más significado que un adjetivo o un adverbio, por ejemplo). La decisión también puede venir dada por el número de veces que aparece el término en un documento en relación a la colección, aquí es donde aparece la utilización de heurísticas (una de las más básicas es eliminar los términos que aparecen en todos los documentos, ya que no ofrecen información discriminante).

5. Construcción de estructuras de categorización de términos tales como los tesauros, o la extracción de la estructura directamente representada en el texto, para permitir la expansión de la consulta original con los términos relacionados (un procedimiento útil normalmente).

Las distintas transformaciones descritas se han presentado siguiendo una lógica temporal de aplicación, pero también se podrían analizar en relación al aspecto que trabajan:

- Análisis a nivel de palabra: análisis léxico, tratamiento de los signos de puntuación, eliminación de las *stopwords*, aplicación de métodos de *stemming*, etc.
- Análisis a nivel de documento o contexto de la palabra: conjunción de palabras (palabras juntas con significado propio), formato de números y fechas, lematización, etc.
- Análisis a nivel de colección: reducciones heurísticas, tesauros, etc.

Tras el preprocesamiento, en general, se dispone de una lista de términos por cada documento de la colección. A partir de esta estructura de datos básica se procederá a la modelización del sistema.

Modelización

La fase de modelización es la que se encarga de definir la estructura y comportamiento del sistema de recuperación de información, tanto a nivel de colección, como de consulta, sin olvidar la presentación de resultados. Existen tres modelos clásicos de recuperación de información: el lógico o booleano (*boolean*), el vectorial y el probabilístico [11, 31]. El modelo lógico se fundamenta en la teoría de conjuntos, y representa a los documentos y consultas mediante un conjunto de términos índice. El modelo vectorial se fundamenta en la teoría algebraica, y define los documentos y consultas como vectores en un espacio m-dimensional. El modelo probabilístico se fundamenta en la teoría de probabilidades y modela los documentos y consultas como relaciones de probabilidades.

Dependiendo del modelo elegido el marco de trabajo se centrará en una teoría matemática u otra. El paradigma más utilizado hoy en día en los sistemas de recuperación de información comerciales es el modelo lógico [25]. Dicho modelo tiene como estructura de datos subyacente los índices invertidos. Por su lado, los demás modelos cada vez están siendo más utilizados, ya que aunque algo más complejos también ofrecen nuevas características que mejoran las prestaciones del modelo lógico (facilidad para generar *rankings*, facilidad de agrupación automática -*clustering*-, etc.).

Esta tesis se centra en el modelo vectorial, por lo que se hablará en relación al mismo, aunque la mayoría de aspectos y características comentadas son extrapolables a los demás paradigmas con algunas o ninguna modificación.

Dentro del paradigma vectorial existen diferentes modelizaciones y enfoques, pero todos tienen en común una primera fase. El primer paso tras el preprocesamiento es construir una estructura de datos vectorial que será la base de cualquiera de las

modelizaciones. Concretamente se genera una matriz de pesos que representará la colección. Esta matriz tendrá tantas filas como términos caractericen a la colección y una columna por documento presente. Cada elemento tendrá un peso, un valor entre 0 y 1 que corresponderá a lo importante que es cada término en cada documento. Existen multitud de fórmulas, cada una potenciando una características, para determinar dichos pesos, aunque todas ellas coinciden en que un valor de cero significa ninguna importancia. Lógicamente el número de términos que caracterizan la colección es muy superior al número de términos que identifican un documento en particular. Esta situación conlleva que la matriz de pesos presente una estructura dispersa. Aspecto muy importante para las prestaciones espaciales y temporales del sistema de recuperación.

Las consultas, en el modelo vectorial, se representan como si fuesen un documento más de la colección, pero con algunas particularidades. Es decir, una consulta dada se modelizará mediante un vector con tantos elementos como términos caractericen la colección, y cada uno de éstos presentará un peso. Los resultados de la consulta se presentan como una lista de documentos ordenados por relevancia (*ranking*).

Una vez la colección, la consulta y los resultados se encuentran representados como se ha descrito, se puede decir que ya se dispone del modelo vectorial más básico. Pero a partir de éste se pueden desarrollar nuevos modelos más sofisticados. Por ejemplo si se desarrolla la modelización de la colección y la consulta pueden aparecer: modelos basados en métodos de *clustering* (por ejemplo, familia del *K-Means*), modelos de indexación semántica latente (LSI -*Latent Semantic Indexing*-), etc.; o también pueden aparecer variantes si se modeliza teniendo en cuenta las estructuras de los textos (se diferencian los títulos del resto del texto, por ejemplo). Si lo que se pretende es mejorar los *rankings* entonces aparecen métodos como priorizar una respuesta u otra en función de la estructura de los documentos (no se valora igual que los términos buscados aparezcan en el título que en el cuerpo del texto, por ejemplo), o se utilizan métodos de retroalimentación, o de agrupación semántica, etc.

Evaluación

La fase de evaluación es la encargada de determinar la calidad del sistema de recuperación de información. Al menos lo hará en la fase de desarrollo y de una forma más o menos teórica, ya que lo bueno o malo que sea el sistema lo decidirán realmente los usuarios durante la fase de utilización. La figura 2.2 en la página siguiente muestra el proceso de evaluación de un sistema de recuperación de información. Esta etapa es una de las más complejas, si no la que más, ya que en la decisión sobre la calidad del sistema intervienen multitud de aspectos, muchos de ellos subjetivos y dependientes del usuario final, que realmente son multitud de usuarios.

Pero la evaluación no es sólo importante ante nuevos métodos, algoritmos o modelos desarrollados, es también decisiva cuando se presenta una modificación de alguno ya existente, ya que aparece el problema crucial de decidir si, y en que magnitud, lo nuevo mejora las prestaciones de lo ya existente.

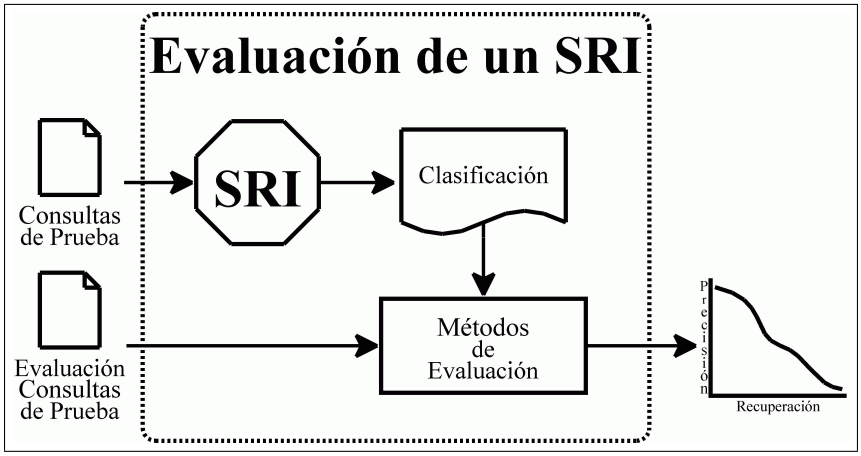

Figura 2.2: Evaluación de un sistema de recuperación de información.

Para poder comparar métodos o modelos y poder determinar las prestaciones que ofrecen, se han desarrollado algunos estándares de evaluación [12]. Las dos definiciones estándares más usadas, directa o indirectamente, son la precisión (*precision*) y la cobertura (*recall*) [11, 12]. La precisión sería la relación entre el número de documentos relevantes recuperados y el número total de documentos recuperados. Mientras que la cobertura (*recall*) sería la relación entre los documentos recuperados como relevantes y los realmente relevantes. Normalmente ambos aspectos están directamente relacionados en todas las medidas de evaluación, ya que el objetivo de un buen sistema de recuperación de información es maximizar ambos aspectos, obtener todos los documentos relevantes y sólo los documentos relevantes. De todas formas no hay que olvidar que la decisión de si un documento es relevante dependerá totalmente del sistema, dos sistemas con la misma colección y con la misma consulta pueden no decidir que los documentos relevantes sean los mismos (o en la misma relevancia), y de hecho es una situación normal. Además, independientemente de la decisión del sistema, con la misma colección y la misma consulta, para un usuario un documento puede ser relevante y para otro no.

Cuando se habla de evaluación, de alguna forma se está hablando de la precisión y cobertura del sistema. Y para determinar ambos aspectos es preciso tener información sobre qué documentos son realmente relevantes para una consulta dada. Lógicamente esto sólo se puede conocer disponiendo de una relación de consultas de prueba con sus respectivos documentos relevantes. Éste suele ser el método más común para evaluar los sistemas de recuperación de información desarrollados. De todas formas, también existen otros métodos más teóricos, aunque éstos suelen utilizarse más en los métodos de *clustering*, ya que permiten determinar la calidad de los *clusters* en función únicamente de la relación entre los elementos de un *cluster* con los elementos de los demás *clusters*.

Hasta ahora se ha hablado de evaluar la calidad del sistema de recuperación de información desde el punto de vista de la calidad de la respuesta. Pero no hay que

olvidar la evaluación del sistema desde el punto de vista computacional. Hay que tener en cuenta el espacio de almacenamiento que precisa y el tiempo de respuesta de las consultas. No sería muy útil, salvo casos concretos, un sistema cuya respuesta fuese casi perfecta pero que tardará muchísimo tiempo en dar dicha respuesta.

Luego también se pueden considerar otros aspectos más secundarios para evaluar la calidad general de un sistema de recuperación de información, tales como el interfaz con el usuario (la sencillez para hacer las consultas o la claridad en las respuestas serían factores a tener muy en cuenta), la disponibilidad de los documentos seleccionados como relevantes (todo el documento o sólo la referencia), etc.

Utilización

La utilización del sistema de recuperación de información es la fase final y el objetivo en sí mismo de las fases de desarrollo. El funcionamiento de esta fase es muy sencillo, los usuarios realizan consultas y el sistema devuelve los documentos que determina como relevantes; este proceso se presenta en la figura 2.3. A partir de ahí dependiendo del sistema de recuperación concreto se puede complicar más o menos las posibilidades. Por ejemplo, se podría dar la opción de refinar consultas (se trabajaría únicamente con los documentos respuesta de otra consulta), o se podría disponer de un detector de errores ortográficos que sugiriera la consulta corregida, etcétera.

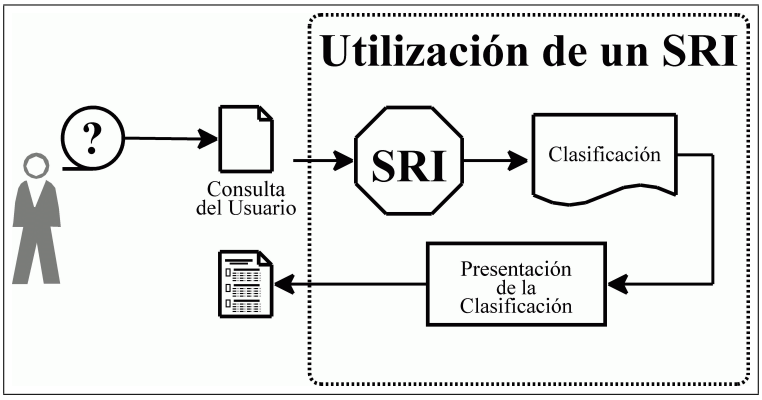

Figura 2.3: Utilización de un sistema de recuperación de información.

La fase de utilización también se considera como la evaluación final del sistema de recuperación de información, ya que realmente la calidad del sistema se valorará en función de la satisfacción de los usuarios ante los resultados de sus consultas [12]. De todas formas este tipo de evaluación no es del todo fidedigna, ya que puede convertirse en demasiado simplista al no quedar claro cómo medir la satisfacción del usuario ¿de forma binaria, sí o no? ¿o relativamente, mediante escalas? Sin contar que al considerarse la satisfacción del usuario únicamente en función de los resultados del sistema, no se tiene en cuenta que éstos dependen también directamente de otros factores independientes del sistema. El más importante sería la calidad de la consulta realizada por

el usuario. Por muy bueno que sea el sistema de recuperación de información si la consulta está mal realizada es dificilísimo obtener resultados que satisfagan. Qué pasaría si se quiere buscar información sobre "casas" y resulta que al escribir la consulta se comente una equivocación y se escribe "cosas". El sistema aunque tuviera detector de faltas de ortografía no se daría cuenta del error. Lógicamente los documentos que el sistema de recuperación de información devolvería como relevantes no satisfarían al usuario ¿y eso significa que el sistema es malo?

2.1.6. Sistemas parecidos y relacionados

La recuperación de información en un amplio sentido se puede enfocar desde distintos puntos de vista, ya que no siempre interesa el mismo tipo de información o tratarla de la misma forma. En ocasiones lo que importa es obtener una serie de documentos relevantes a una consulta efectuada, pero en otros muchos casos la intención puede ser otra: interpretar la información para obtener datos específicos (Minería de datos), buscar información interactivamente (Navegación), agrupar la información relacionada (Clasificación), o seleccionar parte de la información que continuamente llega (Filtrado). Todos estos sistemas son parecidos y están relacionados, pero cada uno de ellos tienen sus características particulares que diferencian los unos de los otros.

Minería de datos (*Data Mining*)

La minería de datos se podría definir como la búsqueda de datos específicos y útiles entre grandes cantidades de datos [3]. La aparición de la minería de datos se ha debido fundamentalmente a la necesidad de desarrollar métodos que permitiesen extraer la información subyacente de grandes cantidades de datos de todo tipo, cuyo volumen ha crecido enormemente en los últimos años.

Aunque la minería de datos se aplica fundamentalmente a bases de datos relacionales y transaccionales, donde los datos están bien definidos y establecidos, no es éste su único campo de acción. La investigación en minería de datos también trabaja con datos desestructurados como la *World Wide Web* [3].

La minería de datos se basa en una actividad exploratoria de datos, por ello los métodos de *clustering* son buenos en este campo. De hecho normalmente un paso inicial importante aplicado en cantidad de procesos de minería de datos es algún método de *clustering* [3]. Estas primeras etapas se utilizan para la segmentación de las bases de datos en grupos homogéneos, permitiendo así identificar características propias de cada subgrupo e incluso poder visualizar la gran cantidad de datos de forma compacta.

La principal diferencia entre la minería de datos y la recuperación de información es el tipo de información que se pretende obtener. Mientras que la recuperación de información busca un conjunto de documentos de una colección que pueden aportar información relevante sobre el tema de la consulta realizada, la minería de datos pretende conseguir datos que den una respuesta concreta a la consulta. Esto se traduce en que pequeños errores en minería de datos pueden ocasionar la invalidez de los

resultados, puesto que, ésta debe satisfacer unas condiciones claramente definidas y suele trabajar con datos bien estructurados [11]. Por el contrario, en los sistemas de recuperación de información la existencia de pequeñas imprecisiones o errores, que se dan normalmente, suelen pasar desapercibidos o simplemente se desprecian, ya que estos sistemas trabajan con textos en lenguaje natural habitualmente mal estructurados e incluso ambiguos semánticamente [11].

Navegación (Browsing)

La navegación cobra sentido cuando el usuario no tiene claro lo que busca, o simplemente no busca, sólo quiere información general, sobre nada en concreto. En otras palabras, la navegación, aunque es un proceso de recuperación de información, no tiene su principal objetivo claramente definido desde un principio, pudiendo éste cambiar varias veces y de forma sustancial durante la navegación [4, 11]. Cuando el usuario tiene un interés pobremente definido o inherentemente amplio, lo que le interesa es una búsqueda interactiva sobre la colección de documentos, que le permita cambiar entre documentos de temas relacionados.

Se pueden distinguir tres tipos fundamentales de navegación: Navegación plana (*flat browsing*), navegación guiada estructuralmente y navegación por hipertexto [11].

Navegación plana: Es aquella navegación que se produce cuando el espacio a explorar tiene una estructura organizativa plana. En este caso, el usuario de un vistazo busca la información dentro de los documentos visitados.

Navegación guiada estructuralmente: Esta técnica trata de facilitar la tarea de navegación a través de la colección, para lo cual organiza los documentos mediante directorios. Los directorios son jerarquías de clases que agrupan documentos que cubren temas relacionados. Este tipo de jerarquías de clases han sido ampliamente utilizadas durante muchísimo tiempo para la clasificación de documentos.

Navegación por hipertexto: Pretende organizar el contenido de los documentos, su estructura, no de una forma secuencial, si no por hiperenlaces. De esta forma se consigue localizar la información más fácilmente, al igual que facilita omitir parte de la información y moverse entre diferentes partes. Su mayor desventaja es que no permite obtener una versión completa y secuencial del texto.

En conclusión la utilidad de la navegación aparece cuando se dan situaciones en las que el usuario no es capaz de formular una consulta con precisión que exprese su necesidad de información y que permita por tanto obtener un conjunto de documentos relevantes. Esta situación se puede dar fácilmente cuando el usuario no está familiarizado con el vocabulario propio del tema que le interesa, o cuando el usuario no busca nada específico, sino únicamente información genérica [89]. De hecho existen métodos que se preocupan fundamentalmente por el proceso iterativo de búsqueda, como por ejemplo [89, 90].

Clasificación

Se puede entender por clasificación el procedimiento de agrupar los documentos de una colección dentro de una serie de conjuntos cuyas características diferenciadoras ya están definidas y establecidas [4, 5, 91]. La clasificación y los métodos de *clustering* son muy parecidos, pero hay que tener muy claras las diferencias, ya que son procedimientos distintos. Algunos autores entienden la clasificación anteriormente definida como clasificación supervisada y el *clustering* como clasificación no supervisada [3]. La diferencia fundamental entre clasificación y *clustering* es que la primera presenta una serie de características propias de cada grupo que son obtenidas independientemente de su contenido, pudiendo, además, incluir información sustraída de su contenido. En cambio en el *clustering* la información disponible para identificar un grupo viene dada únicamente por el contenido del grupo [4]. Otra diferencia importante, que no siempre se tiene por que dar, es que los métodos de *clustering* trabajan muchas veces sin conocimiento previo del número de agrupaciones que deben generar y menos todavía de información referente a ellas. Son métodos iterativos que van refinando una solución inicial. Es en las etapas del proceso y en función del contenido de cada grupo cuando se extrae información característica de la agrupación.

Al igual que el resto de métodos de obtención de información, la clasificación crece en importancia a medida que crece el volumen de documentos disponibles, que es lo que sucede con Internet, bibliotecas digitales, intranets, etc. Cuando se está trabajando con tal cantidad de información no estructurada es casi imprescindible la categorización de la misma en diferentes grupos preespecificados, ya que la información desorganizada dificulta enormemente encontrar la información buscada.

Los varios algoritmos de categorización de documentos que han sido desarrollados durante estos años se encuentran dentro de dos categorías generales [91]:

1. Algoritmos tradicionales de aprendizaje que han sido usados directamente o tras una adaptación para su uso con documentos. Como ejemplos están los árboles de decisión, conjuntos de reglas, clasificadores basados en instancias, clasificadores probabilísticos, máquinas con soporte vectorial (*support vector machines*), etc.

2. Algoritmos de categorización especializados que han sido desarrollados dentro de la comunidad de la recuperación de información. Ejemplos de tales algoritmos incluyen la retroalimentación, clasificadores lineares, clasificadores de conjuntos de instancias generalizados, etc.

Los algoritmos basados en el concepto de similitud generalmente se han presentado con buenas prestaciones en la categorización o clasificación de documentos [91]. Algoritmos como k-vecinos más próximo (*k-nearest neighbor*), conjunto de instancias generalizado, o los basados en centroides son un claro ejemplo de algoritmos basados en el concepto de similitud. Este tipo de métodos determinan a que grupo pertenece un nuevo documentos midiendo su similitud entre ciertos documentos de prueba que caracterizan a un grupo o entre los elementos de ese grupo o una mezcla de ambas acciones.

Filtrado (Filtering)

La tarea de filtrado consiste en seleccionar o no los documentos que fluyen hacía el sistema, la decisión se toma en función de una consulta que permanece relativamente estática [4, 11, 28, 92]. El filtrado sería como el proceso contrario al de recuperación de información convencional. En este último son los documentos los que permanecen en el sistema prácticamente sin cambios y son las consultas las que llegan al sistema, siendo éstas muy variadas, mientras que en los sistemas de filtrado son los documentos los que llegan al sistema y las consultas las que permanecen estáticas.

Las consultas de los sistemas de filtrado se conocen también como perfil de usuario, ya que suelen corresponder con el perfil de los documentos que interesan a un usuario en concreto. Se supone que dicho perfil no varía significativamente, al menos a medio plazo, aunque sí se asume que pueden haber pequeñas modificaciones que lo perfeccionen definiendo mejor aquello que el usuario quiere. Estas pequeñas correcciones del perfil de filtrado se realizan a través de la retroalimentación suministrada a partir de la relevancia real de los documentos recuperados por el sistema de filtrado.

Normalmente para el filtrado de documentos sólo se trabaja con la información del perfil y el nuevo documento a filtrar, la decisión de relevancia es instantánea, no se espera a obtener más información sobre los siguientes documentos que llegan.

En el filtrado, el aspecto más crucial es la construcción de un perfil de usuario que verdaderamente refleje las preferencias de éste. Existen básicamente dos aproximaciones diferentes para la construcción del perfil de usuario: una aproximación simplista y otra más elaborada o sofisticada [11].

La aproximación simplista de construcción de perfiles de usuario consiste en describir éste a través de un conjunto de palabras clave requiriendo al usuario que las suministre. La simplicidad radica en que se le exige al usuario que haga el trabajo. Esta forma de actuar falla en que si el usuario no está familiarizado con el contexto de los documentos le puede ser muy difícil determinar que palabras clave describen sus preferencias en dicho contexto. Además, si el usuario intenta familiarizarse con el vocabulario de los documentos, el proceso se convierte fácilmente en largo y tedioso. Y por otro lado, requerir al usuario que describa de forma precisa su perfil puede ser algo impracticable.

Por todo ello, aparece una alternativa más elaborara, que consiste en recolectar información sobre las preferencias del usuario y usarla para construir el perfil de forma dinámica. Se parte de un perfil inicial de las preferencias del usuario, éste se puede realizar a partir de un pequeño conjunto de palabras claves suministradas por el usuario. Sería algo parecido a la aproximación simplista, pero sin exigir una calidad mínima. Luego, a medida que los nuevos documentos llegan al sistema se utiliza dicho perfil inicial para seleccionar los documentos que potencialmente interesarían al usuario. Entonces se muestran al usuario, el cual inicia un proceso de retroalimentación que consiste en indicar los documentos que son relevantes y los que no lo son. A partir de dicha información el sistema ajustará automáticamente el perfil del usuario para que refleje las nuevas preferencias descritas. Aunque en un principio parezca que esta forma de actuar conlleva a un cambio continuo del perfil de usuario, lo normal, al menos desde un punto de vista optimista, es que el perfil se estabilice después de un

tiempo, consiguiendo que no sufra ningún cambio drástico (lógicamente se presupone que los intereses del usuario no se alteran repentinamente).

2.1.7. Problemática de los sistemas de recuperación de información

El objetivo de los sistemas de recuperación de información es seleccionar los documentos de una colección que están estrechamente relacionados, debido a la información que incluyen, respecto a una consulta realizada. En general, la gente espera mucho de los sistemas de recuperación de información [12], pero normalmente los resultados no son tan buenos. Esta diferencia entre lo esperado y lo obtenido se debe fundamentalmente a la problemática intrínseca de los sistemas de recuperación de información. Estos problemas se pueden agrupar básicamente en dos tipos: los influenciados por la calidad de la consulta y los influenciados por la organización y representación de la información incluida en los documentos. Luego también hay ciertos problemas que afectan tanto a la caracterización de la consulta como de los documentos.

Siempre va a ver una diferencia, mayor o menor, entre la necesidad de información de un usuario y la expresión o representación de la misma en una consulta. Normalmente las consultas presentadas suelen ser vagas y poco precisas, mientras que en cambio se espera obtener respuestas concisas y bien organizadas [12]. El usuario espera conseguir información sobre lo que quiere buscar y lo tiene muy claro en su cabeza, el problema viene cuando lo tiene que expresar en una consulta. Se espera del sistema de recuperación de información que si una palabra puede tener más de un sentido éste sepa exactamente a cual se refiere el usuario, que si se ha cometido una falta de ortografía la detecte y utilice la palabra corregida, que si se ha consultado sobre una palabra también debe tener en cuenta sus sinónimos, que sepa el nivel de importancia de las distintas partes de la consulta y de la importancia relativa de las palabras en un documento, y un largo etcétera que lógicamente el sistema de recuperación de información no puede tener en cuenta.

Por otro lado, se pretende seleccionar ciertos documentos de una colección que incluyan información relevante sobre una consulta. Claro está que para poder medir la relación entre la información que pide una consulta y la información incluida en un documento, es necesario primero caracterizar ésta tanto en la consulta como en el documento. Dicha caracterización se suele hacer mediante la selección de ciertos términos identificativos. Aunque los métodos aplicados variarán mucho, pasando de los más simples a los más sofisticados, todos ellos en mayor o menor medida se basan en un conjunto de términos que identifican la información de la colección (esto es así fundamentalmente por que la unidad de información más pequeña de los documentos textuales es la palabra). Aquí es donde aparece otra fuente de problemas de los sistemas de recuperación de información.

La representación y organización de la información debería ofrecer al usuario un fácil acceso a aquella información en la que el usuario está interesado. Pero, desafortunadamente, caracterizar la información que el usuario necesita no es un problema trivial [11]. En este campo, aunque también están los problemas derivados de la cantidad de información que ofrece cada término, el problema fundamental viene dado por

la polisemia y la sinonimia, teniendo en cuenta que se pueden dar también a expresiones y no sólo a palabras sueltas. Además, hay que entenderla en un sentido relajado y no estricto. Ya que aunque por definición dos palabras no sean lo mismo, para los usuarios la diferencia de significado es despreciable en relación a la información que puede ser recuperada. Y de forma análoga se encuentran las palabras relacionadas, que son conjuntos de palabras que por separado tienen un sentido, pero que si se toman como un grupo tienen otro distinto (por ejemplo, en un contexto "casa blanca" puede tomarse como palabras independientes y referirse a una casa que es de color blanco y en otro tomarse como un conjunto y referirse a la residencia oficial del presidente de los Estados Unidos de America).

Estos problemas de caracterizar la información, añadidos a que no todas las palabras de un documento ofrecen la misma información y a que pocos términos hacen perder información y muchos hacen que se pierda eficiencia en la recuperación, hacen que determinar los términos que caracterizan a un documento sea tanto una ciencia como un arte.

Además de los problemas intrínsecos descritos, en los sistemas de recuperación de información también aparecen otros problemas también muy importantes:

- La enormidad de la información a manejar, que hace fundamental encontrar métodos con muy buenas prestaciones de almacenamiento y computación. Los sistemas de recuperación de información para ser útiles deben dar una respuesta de calidad, pero sin olvidar que la deben de dar en un tiempo de respuesta razonable.

- La heterogeneidad de los documentos, que dificulta la caracterización de los mismos de forma que se puedan comparar unos con otros para evaluar la información que ofrecen. Por ejemplo, en todos los documentos la información que ofrece un término tiene que tener en cuenta no sólo las veces que aparece en el texto, sino también la longitud del documento.

- La influencia de las distintas partes de un documento, que puede obligar a ponderar la importancia de un término en función de su localización en el texto. A lo mejor no tiene la misma importancia un término que aparece en el título de un artículo que si sólo aparece en el texto.

- La interfaz de usuario, que puede provocar que un sistema se utilice o se deje de utilizar independientemente de la calidad del sistema en la recuperación de información. Si hacer la consulta es fácil para el usuario y los resultados están claros de entender a lo mejor ese sistema se utilizará más que otro, que ofrezca una mayor calidad en el resto de prestaciones.

Enormidad de la información a manejar: millones de documentos

Uno de los aspectos más destacables de los sistemas de recuperación de información actuales es el volumen de información con el que tienen que trabajar. Las colecciones de documentos están creciendo en poco tiempo de forma exponencial [14, 24, 33], este crecimiento es debido fundamentalmente al surgimiento de la *World Wide Web*

y en general con la disponibilidad de documentación en formato digital (bibliotecas digitales, fuentes de noticias, *intranets*, artículos científicos, etc.) [21, 23, 24, 29, 33].

Lógicamente este aumento en el número de documentos a manejar ha conllevado el desarrollo y mejora de nuevas técnicas para poder automatizar en la medida de lo posible los sistemas de recuperación de información. Pero en ocasiones no es suficiente con sistemas automáticos y se precisan métodos basados en computación de altas prestaciones, donde destaca la computación paralela o distribuida. Esta situación se da fundamentalmente en dos casos: cuando el volumen de información a manejar es tan grande que una única máquina no es capaz de manejarlo en un tiempo razonablemente aceptable; y cuando los documentos ya se encuentran distribuidos de forma natural y no es viable o no tiene sentido centralizarlos.

Sinonimia y polisemia

Se podría decir que la sinonimia y la polisemia son los dos problemas más comunes y más complejos de los sistemas de recuperación de información [12, 29]. Se entiende por sinonimia el uso de sinónimos, es decir de palabras diferentes que tienen el mismo significado (múltiples palabras tienen un único significado); y se entiende por polisemia el uso de una palabra que varía su significado dependiendo del contexto en el que se encuentre (un único término tiene múltiples significados).

Tradicionalmente, los sistemas de recuperación de información trabajan con emparejamiento literal entre los términos de los documentos y los de la consulta. Pero esta forma de actuar debido a los problemas de sinonimia y polisemia puede dar lugar a bajas prestaciones en la recuperación de documentos relevantes [24]. Cuando un término tiene varios significados y se realiza un emparejamiento puramente léxico el sistema recuperará ciertos documentos irrelevantes al no diferenciar entre las diferentes semánticas del término y tratarlas todas por igual. Por otro lado, cuando en un documento aparezca un sinónimo del término de la consulta, ese documento no será recuperado como relevante, por lo tanto se perderán muchos documentos realmente relevantes porque el sistema sólo determina la relevancia por coincidencia léxica.

El análisis de este tipo de problemática permite deducir que es más importante la relación entre los conceptos tratados en los documentos y los descritos en la consulta, que la relación léxica directa entre los términos que aparecen en los documentos y los presentes en la consulta. Al trabajar con conceptos y no con términos se podrían obtener mejores prestaciones, ya que se podrían recuperar documentos que aún no teniendo ningún término en común con la consulta describan los mismo conceptos, y también se podría prevenir la recuperación de documentos que aún teniendo términos comunes con la consulta éstos no tuvieran la misma semántica.

La indexación semántica latente (LSI -*Latent Semantic Indexing*-) es una modelización utilizada en los sistemas de recuperación de información que intenta superar los problemas de emparejamiento léxico, es decir, los problemas de sinonimia y polisemia, usando índices conceptuales en lugar de término individuales [25]. La LSI asume que existen estructuras semánticas latentes o subyacentes a los términos usados en los documentos, la cual se encuentra parcialmente oculta por la variabilidad de uso de los términos.

Aunque la lógica puede hacer pensar que la utilización de conceptos para obtener la relación de relevancia entre la consulta y los documentos de la colección tendría que ofrecer mejores prestaciones, en muchas ocasiones los usuarios realizan consultas presuponiendo el emparejamiento léxico y haciendo búsquedas literales a propósito. Este comportamiento sucede sobre todo cuando el usuario consciente o inconscientemente valora más la precisión de los resultados, aunque no obtenga todos los documentos relevantes, que la obtención de toda la información relevante disponible. En otras palabras, cuando una consulta devuelve muchos documentos relevantes y el usuario prefiere concretar la información, éste suele basarse en que el sistema de recuperación de información hace emparejamiento léxico para hacer la consulta más precisa.

Palabras relacionadas

Una situación análoga a la sinonimia y a la polisemia son los problemas que ocasionan las palabras relacionadas. Dentro del concepto de palabras relacionadas podemos definir varios tipos:

1. Aquellas palabras que solas tienen un significado pero que unidas o relacionadas pueden tener otro significado distinto: Por ejemplo, la expresión "casa blanca", según el contexto puede significar varias cosas; si se toma como palabras independientes hablaría de una casa de color blanco, pero si se toman juntas entonces cambia su significado y puede referirse a la casa presidencial de Estados Unidos.

2. Aquellas palabras que definen un mismo concepto pero desde un punto de vista morfosintáctico distinto: Por ejemplo las diferentes formas verbales, el plural frente al singular, los adjetivos frentes a los nombres, etc.

3. Aquellas palabras que son antónimas: Por ejemplo, se puede buscar información sobre las subidas en bolsa de una empresa, pero la información puede estar expresada en relación a las bajadas en bolsa.

4. Aquellas palabras que se centran en aspectos complementarios a un concepto: Por ejemplo, los términos como enseñar y aprender, que están estrechamente relacionados y se complementan.

Heterogeneidad de los documentos

El concepto de documento se refiere a una unidad simple de información, formada por texto, generalmente en formato digital, aunque también podría incluir otros tipos de contenido no textuales. Un documento se definiría como una unidad lógica completa, la cual puede presentarse de multitud de formas: artículos de investigación, libros, manuales, noticias, juicios de opinión, descripciones, y un largo etcétera. Además, la representación física del documento también varía, pudiendo ser desde un fichero, a un correo electrónico, pasando por una página de la *World Wide Web* o cualquier otra forma.

A partir del amplio concepto de documento se puede decir que todo documento tiene una sintaxis, una estructura, una semántica y un diseño (la forma en que se

presenta), aspectos todos ellos que pueden cambiar significativamente de un documento a otro. El primer problema que plantea esto es como unificar los criterios para que a partir de documentos con diferente estructura se puedan hacer comparaciones que permitan determinar la relevancia de los documentos de forma equiparable. Para conseguirlo, se suelen tomar decisiones más o menos arbitrarias que aún conllevando pérdida de parte de la información que ofrecen algunos documentos permiten homogeneizar aceptáblemente todos ellos. Un ejemplo de ello podría ser, no tener en cuenta la estructura de los documentos y tomarlos como un único texto largo y continuo (eliminar capítulos de libros, etc). También se suele normalizar la longitud de los documentos para que no influya ésta en la evaluación de relevancia (en un texto largo un término buscado probablemente aparecerá más veces que en un texto corto, sin significar ello que necesariamente sea más relevante el texto largo).

Cuando se está trabajando con una colección formada por documentos que mantienen todos ellos al menos una mínima coherencia en su estructura, sintaxis y forma, es posible mejorar los sistemas de recuperación de información aprovechando la información incluida en la estructura de los documentos. Si todos los documentos están formados por varios apartados (por ejemplo, título, resumen y texto), puede ser útil diferenciar dichos apartados para ponderar la relevancia de los documentos. Puede ser interesante potenciar aquellos documentos donde los términos buscados se encuentran en unos apartados concretos (por ejemplo en el título o en el resumen), sobre aquellos en los que sólo aparecen en otro apartado menos importante (por ejemplo si los términos únicamente se encuentran en el texto).

Lógicamente uno de los mayores problemas respecto a la heterogeneidad de los documentos viene dado por las diferencias idiomáticas de los documentos. Si en una misma colección se encuentran documentos escritos en varias lenguas diferentes y se pretende que el sistema ante una consulta devuelva los documentos relevantes indistintamente del idioma de los documentos, entonces a los problemas normales de los sistemas de recuperación de información, que no son pocos, hay que añadirles los de traducción automática, además de los problemas propios de cada uno de los idiomas con los que se trabaje.

Además, de las diferencias propias de la estructura de los documentos, éstas pueden provocar indirectamente problemas al establecer qué características de los documentos se utilizarán para representar los documentos. Por ejemplo, en documentos de hipertexto, además de los términos propios del texto se podrían utilizar los enlaces entrantes y los salientes, los cuales no aparecerían en documentos textuales estándares. Los documentos de una colección pueden tener múltiples y heterogéneas características como situación natural y común [50].

¿Tienen la misma importancia todas las partes de un documento?

Es bastante lógico pensar que no todas las partes de un documento tienen la misma importancia a la hora de determinar la relevancia del documento en relación a los términos de la consulta. La principal causa es que el contenido de las distintas partes del documento se escriben de forma diferente en función de su objetivo: resumir la información, sintetizarla, detallarla, etc.

Por ejemplo, no se escribe igual el título de un documento, que un resumen del mismo, o que su texto completo. En el título se utilizan muy pocas palabras pero con mucha significación del tema principal del documento. En el resumen se utilizan también pocas palabras, pero muchas más que en el título, y se nombra sólo el tema principal del documento y posiblemente algún tema secundario también importante. Los resúmenes son compactos y densos en información [93]. Lo cual implica que se utilicen términos específicos y muy diferenciadores. Estos términos actúan como representantes de múltiples ocurrencias en el texto original [93]. Por su parte en el texto completo, la utilización de palabras crece enormemente y se tratan todos los temas del documento. Dependiendo de la zona se puede hablar del tema principal o ni siquiera mencionarlo y sólo desarrollar algún tema complementario. El texto completo tratará de muchos subtemas que no han sido mencionados en el resumen, si existiese [93]. Consecuentemente es importante conocer el patrón de distribución de los términos en el documento, ya que de esta forma es más fácil interpretar como influyen los términos para determinar la relevancia del documento [93].

De todo ello se deduce que, en principio, si la relación entre la consulta y el documento se establece en el título del documento entonces la relevancia de dicho documento seguramente será mucho mayor que si se establece en una parte específica del texto completo.

A pesar de que está clara la influencia de la estructura no se suele tener en cuenta en los sistemas de recuperación de información habituales (o mínimamente, el título y poco más). Esto es debido principalmente a la complejidad de extraer la estructura de los documentos, a que las colecciones suelen presentar documentos con estructuras muy heterogéneas difíciles de unificar, y a que es difícil evaluar y medir la diferencia de influencia de las diferentes partes del documento. Por otro lado hay una clara influencia, en este sentido, de la evolución de los sistemas de recuperación de información. Tradicionalmente la mayoría de los sistemas trabajaban únicamente con documentos cortos (resúmenes) [93], pero con el aumento de las capacidades computacionales y de almacenamiento se ha evolucionado hasta trabajar con documentos largos (documentos completos, no sólo resúmenes).

Interfaz con los usuarios

Los usuarios interactúan con el sistema de recuperación de información básicamente de dos formas: formulando las consultas y revisando los resultados; también podría haber una tercera forma si existiera la opción de refinar los resultados de la consulta [4]. Consecuentemente, las interfaces de usuario básicamente cumplen dos funciones distintas. Por un lado deben facilitar que el usuario realice la consulta y por otro lado debe presentar de forma útil los resultados de la búsqueda. Pero tampoco hay que olvidar que la interfaz de usuario debe mantener a éste informado durante el transcurso de la búsqueda. Es decir, la interfaz de usuario debería ayudar a formular las consultas, a seleccionar entre las fuentes de información disponibles, a entender los resultados de la búsqueda, y a informar del progreso de la búsqueda [11].

Normalmente los usuarios no tienen claro cómo deben actuar para obtener la información que precisan, aunque sea simplemente por que la búsqueda de información

sea un proceso impreciso. Por lo tanto la interfaz de usuario debe guiar al usuario para que éste pueda obtener la información deseada. Una interfaz que sea una pantalla vacía o un formulario de entrada en blanco no ayuda mucho al usuario a decidir por dónde empezar el proceso de búsqueda, la interfaz debe proveer un buen camino para empezar [11].

Una vez el usuario ha iniciado el proceso de búsqueda es necesario que la interfaz ofrezca algún tipo de información que indique como está evolucionando el proceso. Es decir, mientras la búsqueda está siendo conducida el usuario necesita saber lo que el sistema está haciendo antes de que el resultado llegue, y además, aunque parezca una ironía es importante que cuando el sistema devuelva que no existen documentos relevantes, esta respuesta no sea demasiado rápida, pero si existen documentos relevantes éstos han de ser devueltos lo más pronto posible [12].

Cuando un sistema de recuperación de información devuelve una serie de documentos relevantes ante la consulta de un usuario, la interfaz debería proveer más información que únicamente el conjunto de documentos. Aunque la mayoría de los sistemas de recuperación de información devuelven únicamente una lista o un *ranking* de documentos [4], es importante el contexto que ha provocado que un documento sea relevante, ya que esto hace más comprensible al usuario el porque ese documento le va a aportar más o menos información. El contexto se puede establecer entre los documentos devueltos con los términos de la consulta y también todos los documentos recuperados entre sí. Para cumplir este objetivo existen distintos tipos de información que se puede presentar [4, 11], por ejemplo: título del documento, longitud, porcentaje de similitud, resumen del documento, formato del documento, fragmento del texto con los términos de la consulta, enlace al documento completo, presentación gráfica de la distribución de los términos de la consulta en el documento (el paradigma de presentación *TileBars* permite ver simultáneamente la longitud relativa de los documentos recuperados, la frecuencia relativa de los términos de la consulta, y su distribución con respecto al documento y respecto a los demás documentos [93]), etc. El patrón de distribución de los términos de una consulta en el texto completo de un documento es un aspecto muy importante para valorar la relevancia de un documento [93]. Es importante saber, sobre todo en los documentos largos, tanto como de frecuente es un término, cómo cual es su distribución. No es lo mismo que el término se comente a lo largo de todo el texto (es un tema general del documento, principal o no) a que sólo se comente en una parte muy concreta del texto (es un tema de apoyo o parcial del documento). Existen trabajos que se centran en la la presentación visual de los resultados del sistema de recuperación de información [85, 93].

Una característica muy importante para las interfaces de usuario es la posibilidad de disponer de interfaces alternativas para novatos y para expertos. No valoran de igual forma una interfaz un usuario que otro, dependiendo totalmente de su perfil. Mientras que el novato agradecerá más la simplicidad de manejo y claridad en la información, el experto le dará más importancia a la potencia del sistema [12]. Los sistemas simples son más fáciles de aprender, a expensas incluso de menor flexibilidad e incluso menor eficiencia de uso. Las interfaces potentes permiten al usuario estar bien informado haciendo más cosas y teniendo más control sobre las operaciones, pero en cambio suponen mayor tiempo de aprendizaje. El buscador experto prefiere

entender cómo la búsqueda opera en orden a estrecharla o ensancharla y asegurar su minuciosidad, mientras el novato meramente quiere una respuesta [12].

Aunque se suela infravalorar la importancia de la interfaz de usuario no hay que olvidar que el usuario juzgará las prestaciones del sistema de recuperación de información tanto por la calidad del resultado final de la búsqueda como por cuanto le ha costado conseguirlo. Por lo tanto, la importancia de la comunicación entre el usuarios y el motor del sistema de recuperación de información, es decir, la interfaz de usuario, no se puede subestimar [12].

2.2. Preprocesamiento

La fase del preprocesamiento es la inicial en la creación de un sistema de recuperación de información. Durante esta etapa se transforman los documentos de la colección en una estructura de datos que sintetiza la información relevante de los documentos. Las cinco transformaciones fundamentales que suelen tener lugar durante el preprocesamiento de la colección son [11]:

- Análisis léxico para identificar los términos.
- Eliminación de las palabras carentes de significado (*stopwords*).
- Unificación de términos según su raíz sintáctica (*stemming*) o semántica (lematización).
- Reducción de los términos identificativos de la colección (unificación de formatos, reducciones heurísticas, etc.).
- Utilización de tesauros.

Estas transformaciones básicas son descritas con mayor detalle en los siguientes subapartados, haciendo especial hincapié en los aspectos generales de cualquier modelización y en los más particulares del paradigma vectorial.

2.2.1. Selección de los términos que identificarán la colección

Los documentos de las colecciones se suelen representar mediante un conjunto de términos índice o palabras clave, adoptando la idea que la semántica del documento y la necesidad de información del usuario puede expresarse mediante términos índice. Esta asunción es una considerable amplificación del problema porque muchos de los significados en un documento o en la consulta de usuario se pierden cuando se reemplaza su texto con un conjunto de palabras clave [11]. La selección de los términos índice o palabras clave puede realizarse extrayéndolas directa y automáticamente del propio texto del documento o puede ser un especialista quien las determine [2, 11]. Lógicamente cuando la selección la realiza un experto humano sus decisiones estarán basadas en su edad, su bagaje cultural, su educación, el idioma e incluso sus tendencias políticas. De hecho existen experimentos que han demostrado que hay en media

un 20 % de disparidad en los términos elegidos como apropiados para describir un documento dado por dos indexadores profesionales diferentes [29]. A pesar de ello hay ciertas comparativas que muestran que la indexación automática simple y los métodos manuales ofrecen prestaciones similares [2].

En función de qué palabras clave se seleccionen para describir un documento se tienen distintas vistas lógicas de dicho documento [11]. Si los términos índice elegidos fuesen el texto completo del documento, entonces se estaría frente a la vista lógica más completa. Esta vista implica mayores costes computaciones, tanto temporales como espaciales. En el otro extremo estaría la vista lógica más concisa del documento, que se obtendría por la selección de un pequeño conjunto de palabras clave elegidas por un especialista humano. Pero esta vista puede conllevar recuperaciones de pobre calidad. Como es habitual, en un estado intermedio suele estar la mejor solución. Se pueden adoptar multitud de vistas lógicas intermedias, las cuales pueden contemplar gran número de términos índice pero al mismo tiempo eliminar muchas palabras clave que aportan muy poca información, también pueden reconocer la estructura propia interna del documento (título, capítulo, secciones, apéndices, etc.), e incluso estructurar y relacionar los términos índice del texto para obtener un valor añadido a través de nueva información.

Aunque los ordenadores modernos hacen posible representar los documentos por el conjunto completo de palabras que forman su texto, vista lógica completa, también es verdad que ciertas alternativas pueden mejorar las prestaciones de la vista lógica completa pasando a una vista intermedia. Por ejemplo, al reducir el número de palabras clave las prestaciones computacionales mejoran, pero lógicamente esta reducción debe hacerse minimizando la pérdida de prestaciones de recuperación. Una buena opción para esta situación es la eliminación de las palabras carentes de significado (*stopwords*), o también está la opción de agrupar todos los términos índice con la misma raíz gramatical en una única palabra clave (*stemming*).

Una vez el concepto abstracto de selección de términos índice está descrito, se puede proceder a describir el proceso que se sigue para pasar de un documento a las palabras claves seleccionadas. El análisis léxico es el proceso de convertir un flujo de caracteres (el texto del documento) en un flujo de palabras (las palabras candidatas a ser adoptadas como términos índice) [11]. Es decir, a través del análisis léxico del documento se extraen una lista inicial de palabras claves, luego está lista se irá refinando por diversos métodos hasta llegar a la lista final de términos índice. Aunque a simple vista el análisis léxico parece consistir en un sencillo proceso de identificar las palabras del texto, las cuales estarían separadas por espacios en blanco, el procedimiento va mucho más allá, teniendo que normalizar dichas palabras tomando decisiones clave sobre que será un término, como se diferencian dos, y un largo etcétera [11, 12].

Hay que decidir qué se hace con los dígitos. Aunque los números no suelen ser unos buenos términos índice al ser muy vagos sin un contexto, en ciertas ocasiones hay que tenerlos en cuenta. Por ejemplo, cuando aparecen mezclados con palabras, formando fechas, o aunque estén sueltos identifican una información importante (número de tarjeta de crédito, cuentas bancarias, etc.). Tampoco hay que olvidar que sería importante la normalización de fechas y números para unificar formatos entre distintos documentos.

También es importante tener en cuenta cómo se van a tratar los guiones, éstos pueden ser parte integral de una palabra, pueden unir palabras formando un único término, e incluso pueden separar una misma palabra al final de línea. Habría que identificar cada caso y decidir como tratarlo.

Aunque en principio los signos de puntuación no deberían tener ningún trato especial y eliminarlos enteramente del proceso de análisis léxico, realmente hay que plantearse que algunos signos de puntuación son parte integral de ciertas palabras (por ejemplo en las siglas, "A.C."). De todas formas normalmente su eliminación no suele tener un gran impacto en las prestaciones de recuperación ya que el riesgo de mal interpretaciones es mínimo [11].

La decisión más fácil versa sobre la diferenciación de mayúsculas y minúsculas a la hora de distinguir dos términos, ya que salvo casos muy concretos (por ejemplo, cuando se describen detalles sobre comandos Unix) no se suele hacer distinción alguna.

Tampoco hay ninguna razón para limitarse a la utilización de palabras simples como términos índice, también se pueden utilizar frases [31]. Las frases conllevan mayor contenido semántico, pero también conllevan mayor especificidad semántica, lo cual puede reducir las prestaciones de recuperación [2]. De todas formas, la utilización de frases permite capturar el orden de éstas y según ciertos estudios mejoran la precisión de los sistemas de recuperación de información [50].

Después de obtener una lista de palabras o frases clave tras haber tomada diferentes decisiones sobre todos los aspectos comentados en el análisis léxico, dicha lista se simplifica eliminando todas aquellas palabras carentes de significado o con un valor prácticamente nulo, las *stopwords*. Y luego todavía se reduce más esta lista unificando distintos términos en una única raíz gramatical, el proceso de *stemming*.

2.2.2. Eliminación de palabras carentes de información: *Stopwords*

Hay ciertas palabras que independientemente del documento en el que estén no aportan información discriminante, o ésta es tan mínima que no compensa tenerlas en cuenta. Este tipo de palabras son conocidas como *stopwords* y no son tenidas en cuenta como términos clave. Generalmente las palabras que son excesivamente frecuentes entre los documentos de la colección corresponden con las *stopwords*. De hecho, una palabra que aparece en el 80 % de los documentos en la colección es inútil para propósitos de recuperación [11]. Claros candidatos de palabras que deberían estar en las *stoplists* (listas de palabras carentes de significado) son los artículos, las preposiciones y las conjunciones.

La eliminación de las *stopwords* tiene un claro e importante beneficio en el proceso de recuperación de información. Reduce considerablemente la lista de los términos índice de la colección. Es típico obtener una reducción en la talla de la estructura de indexación, de un 40 % o más, solamente con la eliminación de las stopwords [11]. Rebajar el número de palabras clave conlleva por tanto dos ventajas importantes en la mejora de las prestaciones computacionales, por un lado reduce el espacio de almacenamiento necesario para guardar la estructura de datos que define la colección, y por otro lado reduce el tiempo necesario para construir y manejar dicha estructura.

Y, además, minimizando las repercusiones en las prestaciones de recuperación de información. Todo ello hace que en ocasiones se plantee la posibilidad de extender la lista de *stopwords* para incluir otras palabras que también aportan poca información, por ejemplo, algunos verbos, adverbios y adjetivos.

A pesar de estas importantes ventajas, no hay que olvidar que la eliminación de las *stopwords* puede reducir la cobertura (*recall*) del sistema de recuperación de información, al eliminar términos clave en la evaluación de la relevancia de un documento [4], aunque esta reducción suele ser insignificante a efectos de la calidad de la recuperación (prácticamente no se pierden documentos importantes). De todas formas esta es una razón, junto a otras varias, que da lugar a la adopción de la indexación completa de los documentos utilizada por algunos motores de búsqueda de la Web [11]. También hay que tener en cuenta que cuando se está trabajando con una estructura de fichero invertido comprimido la mejora espacial obtenida eliminando las *stopwords* puede ser cuestionable, ya que las palabras omitidas típicamente requieren pocos bits por puntero haciendo que la mejora total en almacenamiento no sea destacable [12].

2.2.3. Trabajar únicamente con la raíz de las palabras: *Stemming*

El *stemming* es el proceso por el cual las palabras, los términos índices de un documento, se sustituyen por su raíz morfológica. Es decir, el *stemming* consiste en la eliminación de los afijos (prefijos y sufijos) de la palabra [11, 12]. La idea subyacente es la sustitución de las variantes morfológicas de una palabra por el concepto que define dicha palabra, representado éste por la raíz de la misma, consiguiendo así mejorar las prestaciones en la recuperación de información. Su utilidad se ve claramente al analizar el proceso de consulta. El usuario especifica una palabra clave en la consulta, pero lo más seguro es que en el documento relevante para el usuario aparezca una variante de dicha palabra clave. Las variaciones sintácticas más comunes que impiden el correcto emparejamiento entre los términos índice de una consulta y las palabras clave de un documento suelen ser: plurales, gerundios, participios, formas verbales, etc. El *stemming* en definitiva busca la eliminación de ruido, pero no hay que olvidar que también puede eliminar términos cruciales para juzgar la relevancia de un documento [4], aunque esta potencial pérdida suele ser mínima y se da en casos muy específicos, además suele ser totalmente compensada por los beneficios que aporta.

La utilización de los métodos de *stemming* tiene una tradición relativamente larga en el proceso de selección de los términos índice [12]. El *stemming* no es una tarea para ser tomada a la ligera, ya que puede ser una empresa tediosa que tiene diversas variantes y enfoques. Afortunadamente, varios métodos de *stemming* automáticos han sido desarrollados [12], destacando el Porter [94] como el más popular debido a su simplicidad y elegancia [11]. A pesar de ser muy simple, el método Porter produce resultados comparables con otros algoritmos más sofisticados [11].

El uso del *stemming* tiene el efecto secundario, en este caso bueno, de reducir la talla de la estructura de indexación porque reduce el número de distintas palabras clave utilizadas [11]. Concretamente, en el modelo vectorial, el *stemming* reduce el número de filas en la matriz de términos por documentos [12]. Esta propiedad no hay

que infravalorarla ya que es una ventaja añadida muy importante, sobre todo debido al volumen de información que se suele manejar.

Por lo tanto, el *stemming* favorece la mejora de prestaciones a todos los niveles buscados. Mejora las prestaciones en la calidad de la recuperación de información, y mejora las prestaciones computacionales, tanto espaciales (estructuras de datos más pequeñas) como temporales (menos datos a manejar y consultar).

2.2.4. Utilización de información sintáctica y/o semántica: Lematización

Análogamente al *stemming* aparece la lematización, la cual obtiene los términos de la colección a partir del lexema de las palabras de la colección (lematización) [95] o a partir de los sentidos de las palabras de la colección (lematización semántica). A la utilización de técnicas sintácticas y/o semánticas para mejorar las prestaciones en recuperación de información se le conoce como *word sense desambiguation* [4].

La lematización obtiene la raíz de las palabras, pero a diferencia del *stemming* que simplemente elimina afijos, se basa en la raíz léxica: los verbos conjugados pasan a la forma del infinitivo, los sustantivos a la forma singular masculina, etc. La lematización semántica, que es una extensión de la lematización al necesitar de la raíz léxica de las palabras, se basa en el significado de éstas, teniendo en cuenta su contexto. Para trabajar con los significados en relación al contexto se precisa un recurso externo, una ontología (relación conceptual exhaustiva y rigurosa) siendo una de las más populares y utilizadas la de WordNet [96]. Un estudio interesante sobre el uso de las ontologías en la recuperación de información fue [97] y también destaca [98] que muestran cómo se pueden obtener mejoras con el uso de la lematización en la recuperación de información. En [86] se hizo una comparación entre la influencia de la lematización semántica y del *stemming* en las prestaciones de recuperación. Se obtuvieron mejores prestaciones al utilizar en el preprocesamiento técnicas de *stemming* en contraposición a las de lematización, aunque el comportamiento de ambas fue parecido.

2.2.5. Reducciones heurísticas de términos poco o demasiado utilizados

En un intento por reducir más el número de palabras clave o términos índice de una colección se utilizan criterios heurísticos para eliminar aquellas palabras que aportan muy poca información discriminante entre documentos de la colección [12, 23, 33]. Concretamente, las palabras que se plantea eliminar son aquellas que aparecen de manera muy infrecuente en la colección, generalmente en un único documento de la colección (o en dos documentos), y también aquellas que aparecen en prácticamente todos los documentos de la colección (palabras candidatas a incluir en la lista de *stopwords*). Es decir, se eliminan aquellos términos de muy poca frecuencia o de muy alta frecuencia.

Con las reducciones heurísticas se pretende mejorar las prestaciones computacionales del sistema de recuperación de información, al tener que manejar menos información y al tener que almacenar también menos información. Lógicamente esta

mejora puede conllevar un empeoramiento de las prestaciones de calidad de la recuperación de información. Pero se prevé que ésta sea mínima, al eliminar los términos que prácticamente no aportan información. Ahí es donde aparece la heurística.

2.2.6. Tesauros

La palabra tesauro tiene su origen en el Griego y en el Latín y es usada como una referencia a un tesoro de palabras. De forma simple un tesauro consiste en una lista precompilada de palabras a las que se asocia un conjunto de palabras relacionadas generalmente derivadas de una relación de sinonimia. De forma más general, también involucra alguna normalización del vocabulario e incluye una estructura más compleja.

Básicamente, los principales propósitos de un tesauro son [11]:

- Proveer un vocabulario estándar (o sistema de referencia) para la indexación y la búsqueda.
- Asistir al usuario en la elección de términos para crear consultas apropiadas.
- Proveer jerarquías clasificadas que permitan ensanchar y ampliar la petición de consulta actual de acuerdo con las necesidades del usuario.

Una de las principales motivaciones para construir un tesauro está basada en la idea de usar un vocabulario controlado para la indexación y la búsqueda, lo cual conlleva importantes ventajas como la reducción de ruido, la recuperación basada en conceptos, etc. [11]

Los tesauros favorecen la reformulación automática de consultas [11, 31]. Es sabido que en muchas ocasiones la calidad de los resultados del sistema de recuperación de información dependen directamente de la calidad de la consulta. Generalmente si no se tiene claro cómo construir una consulta de calidad y además se desconoce cómo es la colección en la que se busca, lo más normal es que la consulta realizada no permita recuperar los documentos relevantes. Es entonces, cuando es interesante reformular o expandir la consulta para mejorar sus prestaciones. Los tesauros se pueden utilizar para este fin sustituyendo los términos que aparecen en la consulta por aquellos que el tesauro determina que están directamente relacionados.

2.3. Modelización

2.3.1. Clasificación de los modelos

El propósito principal de los sistemas de recuperación de información es ayudar a los usuarios a acceder eficientemente a grandes colecciones para así satisfacer su necesidad de información. Con la finalidad de conseguir tal objetivo existen tres modelos clásicos de recuperación de información: el lógico (*boolean*), el vectorial y el probabilístico [11, 31, 93, 99]. El paradigma lógico modela los documentos y las consultas mediante conjuntos de términos índice y se basa en la teoría de conjuntos. En cambio, el paradigma vectorial modeliza los documentos y las consultas como vectores

en un espacio m-dimensional (m sería el número de términos índice) y se basa en la teoría algebraica. Por su parte, el paradigma probabilístico modeliza los documentos y las consultas como probabilidades y se basa en la teoría probabilística.

A medida que ha ido pasando el tiempo y el problema de la recuperación de información se ha ido conociendo y estudiando más profundamente, han ido apareciendo otros paradigmas de modelado alternativos a los clásicos. En la figura 2.4 se presenta una taxonomía sobre los modelos de recuperación de información [11]. Dentro del paradigma lógico destacan los modelos de lógica difusa (*Fuzzy*) y el modelo lógico extendido. El paradigma algebraico incluye el modelo vectorial generalizado, la indexación semántica latente (*Latent Semantic Indexing*) y las redes neuronales. Las alternativas del modelo probabilístico son las redes de inferencia (*Inference Network*) y las redes de convicción (*Belief Network*). Los métodos de clustering se podrían englobar en cualquiera de los paradigmas en función del modelo en el que se basen, aunque habitualmente trabajan dentro del modelo vectorial.

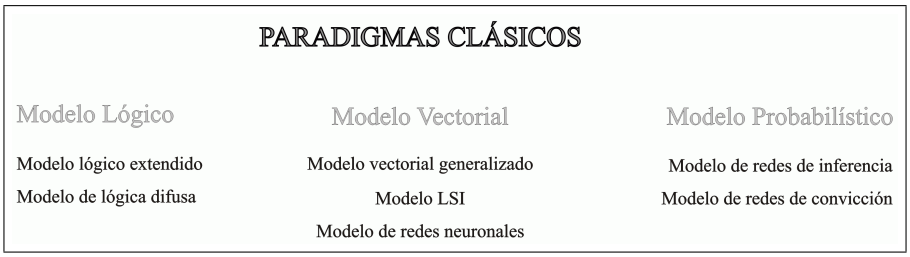

Figura 2.4: Taxonomía de los modelos de recuperación de información.

El modelo lógico está basado en la teoría de conjuntos y en el álgebra booleana. Las consultas, dentro de este modelo, están definidas en términos de disyunciones, conjunciones y negaciones entre conjuntos de documentos que contienen ciertos términos índices. Los documentos recuperados son aquellos cuyo contenido satisface el enunciado lógico que es la consulta.

Las principales ventajas del modelo lógico son su claro formalismo de base y su simplicidad. Al ser el concepto de conjunto bastante intuitivo y al ser las consultas específicas (las expresiones lógicas tienen un significado preciso), el modelo lógico es fácil de comprender por un usuario común. Estas características destacadas provocaron el auge del modelo lógico, siendo éste adoptado por los primeros sistemas comerciales.

De todas formas el modelo lógico presenta algunas desventajas importantes. En la forma más básica del modelo lógico no se obtiene una clasificación ordenada (*ranking*). Esto es debido a que su estrategia de recuperación está basada en un criterio de decisión binario o lógico (un documento es relevante o no) sin ninguna noción de gradiente. Al no haber una noción de correspondencia parcial para los términos índice de las consultas, las prestaciones en la recuperación se ven resentidas.

Aunque las expresiones lógicas tienen un significado preciso, normalmente es difícil traducir una necesidad de información en expresión lógica. De hecho, la mayoría de los usuarios encuentran complicado definir las consultas como expresiones lógicas [11].

Existen dos alternativas al paradigma lógico, el llamado modelo de conjuntos difusos y el modelo booleano extendido [11, 99]. La primera se basa en el hecho que representar los documentos y las consultas mediante conjuntos de palabras clave conlleva descripciones sólo parcialmente relacionadas con el contenido semántico real de dichos documentos y consultas. Por lo tanto, la relación de un documento con los términos de la consulta es aproximada o vaga. Para solventar esta situación aparece la teoría de conjuntos difusos, donde cada documento presenta un grado de pertenencia al conjunto difuso que define cada término de la consulta. Por su parte, el modelo booleano extendido, está basado en una crítica al axioma básico de la lógica booleana que dice que un documento que contiene sólo parte de los términos de la consulta es tan irrelevante como aquel que no contiene ninguno [11]. Dado que este criterio de decisión binario en la mayoría de los casos es ilógico desde el punto de vista del sentido común, el modelo booleano extendido se lo salta tratando de forma diferente los documentos en función de los términos de la consulta que son relevantes en el documento.

El modelo vectorial intenta superar la restricción del uso de pesos binarios proponiendo la posibilidad de una correspondencia parcial. Para ello utiliza pesos no binarios en los términos de los documentos y de las consultas, los cuales se utilizan para determinar el grado de similitud entre consultas y documentos, permitiendo ordenar los documentos recuperados por similitud.

En el paradigma vectorial tanto los documentos como las consultas se modelizan mediante vectores m-dimensionales que contienen el peso de cada término. La relación de similitud puede ser cuantificada de multitud de formas, aunque una de las más comunes suele ser calculando el coseno del ángulo que forman el vector representante de una consulta con el de un documento. Actuando de esta forma el modelo vectorial puede clasificar los documentos en relación al grado de similitud respecto a la consulta y así recuperar documentos que presentan una relación sólo parcial con la consulta.

Destacan tres modelos como alternativas al paradigma algebraico clásico [11]: el modelo vectorial generalizado, en el que se asume que los vectores índice son linealmente independientes pero no ortogonales; el modelo de indexación semántica latente (LSI -*Latent Semantic Indexing*-); y el modelo de redes neuronales.

Aunque el modelo de *clustering* no está únicamente relacionado con el paradigma vectorial, muchos de los métodos de *clustering* (y todos los desarrollados en el marco de esta tesis) trabajan partiendo de una representación vectorial y utilizando técnicas de cálculo de distancias iguales o similares a las utilizadas en el modelo vectorial para calcular similitudes.

Los métodos de *clustering*, de forma simple, tratan de separar una colección de objetos (documentos) en dos conjuntos distintos (normalmente disjuntos) basándose en una vaga descripción de un conjunto: el primero de los conjuntos estará formado por todos los objetos relacionados con dicha descripción vaga y el otro por todos los objetos que no están relacionados. Hay que entender por descripción vaga aquella información qué no permite discernir con precisión que objetos pertenecen o no al conjunto descrito.

Cuando se está trabajando con modelos de *clustering* aparecen principalmente dos problemas: la necesidad de determinar cuales son las características (términos) que

mejor describen los objetos (documentos) de un conjunto y la necesidad de determinar cuales son las características (términos) que mejor distinguen los objetos (documentos) de un conjunto de los restantes objetos (documentos) de la colección. Las primeras características servirán para cuantificar la similitud intracluster (se utiliza la frecuencia de los términos dentro de los documentos) y las segundas para valorar la similitud intercluster (se hace uso de la inversa de la frecuencia de los términos entre los documentos de la colección).

El modelo probabilístico intenta estimar la probabilidad de que un documento de la colección sea relevante para una consulta dada. Para ello asume que dicha probabilidad depende sólo de la representación del documento y de la consulta. Este modelo tiene el problema de no establecer explícitamente cómo calcular las probabilidades de relevancia, ni siquiera determina cual sería el espacio muestral utilizado para definir tales probabilidades [11].

El procedimiento simplificado que realiza el paradigma probabilístico es asignar a cada documento un ratio como medida de similitud, el cual es la relación entre la probabilidad que dicho documento sea relevante para una consulta dada y el de que no lo sea. Luego toma la probabilidad de relevancia como el rango que minimiza la probabilidad de un juicio erróneo [11].

Las alternativas al paradigma probabilístico están basadas en las redes bayesianas, son el modelo de redes de inferencia (*inference networks*) y una generalización de éste, el modelo de redes de convicción (*belief networks*) [11].

2.3.2. Modelo vectorial básico: Matriz de pesos

El origen del modelo vectorial se basa en un intento de eliminar algunos de los mayores problemas asociados a las técnicas de emparejamiento léxico y exacto, concretamente la polisemia (varios significados expresados por un mismo término) y sinonimia (el mismo significado expresado por varios término) [33]. El modelo vectorial al trabajar en el espacio de los términos y documentos y al calcular similitudes en dicho espacio con las consultas, permite clasificar el resultado de una consulta en relación a la medida de similitud usada. Tales clasificaciones no son posibles con el emparejamiento léxico. El modelo vectorial, también gracias a trabajar en espacios vectoriales, facilita la implementación de métodos de retroalimentación, los cuales permiten ir mejorando la consulta realizada a partir de la información devuelta por la propia consulta.

Partiendo de la premisa de que el significado de un documento puede ser derivado de los términos que lo forman, el modelo vectorial crea una estructura matricial de la colección de documentos. Concretamente, los documentos son representados como vectores de términos, siendo cada uno de los elementos de dicho vector un peso que denota la importancia del término correspondiente en el documento. El conjunto de los vectores documento forman la matriz que representa la colección. Las consultas se definen análogamente a los documentos, como vectores de términos. Y para determinar los documentos relacionados con una consulta se mide el grado de similitud (por ejemplo mediante la distancia euclídea) entre el vector consulta y los vectores documentos.

En el modelo vectorial más básico, la representación de los documentos solamente se basa en la frecuencia de términos, la cual no modeliza adecuadamente el contenido semántico dando lugar a ciertos errores. Estas incertidumbres han sido tratadas mejorando la representación vectorial de los documentos (y por ende de las consultas). Fundamentalmente se han desarrollado dos soluciones. La primera se basa en el pesado de los términos, y la segunda utiliza aproximaciones de bajo rango de la matriz original [12]. Esta última solución, que suele incluir también la primera, se basa en la premisa de que con la reducción del rango de la matriz se reduce también el ruido de esta (es decir, los problemas de sinonimia y polisemia). El modelo vectorial de indexación semántica latente (LSI -*Latent Semantic Indexing*-) es el que trabaja la solución de las aproximaciones de bajo rango, utilizando algoritmos tales como la factorización QR y la descomposición en valores singulares (SVD -*Singular Value Decomposition*-). Desafortunadamente, tanto en el modelo vectorial como el LSI, no es posible distinguir las consultas que usan conectivas Y (*and*) de las que usan conectivas O (*or*) [25].

El modelo vectorial, excepto su variante LSI, trata cada término independientemente imitando los índices invertidos, sin embargo es más flexible que éstos últimos, ya que cada término puede ser pesado individualmente [33]. Además, al utilizar medidas de similitud para comparar las consultas con los documentos permiten enfatizar o desenfatizar ciertas características de los documentos o de la colección (por ejemplo, la longitud de los documentos).

El éxito o fracaso del método vectorial se basa principalmente en la elección adecuada del esquema de pesado de los términos. Existen muchas investigaciones respecto a las técnicas de pesado pero no parece haber consenso respecto al mejor método [12, 31], lo cual se debe a que dependiendo de la naturaleza del vocabulario de la colección un tipo de pesado u otro será mejor. En [12, 31] se incluye una comparación computacional de varias de las técnicas sugeridas en la literatura.

La matriz de pesos

La relación existente entre los términos clave y los documentos de la colección se establece mediante una matriz dispersa de términos por documentos [2, 11, 12, 14, 33, 100]. Cada documento se representa mediante un vector m-dimensional, siendo m el número total de términos clave de la colección. Por lo tanto, si existen n documentos en la colección, ésta será expresada por una matriz A con m filas y n columnas. Normalmente cuando una colección es pequeña (tiene pocos documentos) el número de términos en esa colección es mucho mayor que el número de documentos y más, cuando la colección versa sobre temas heterogéneos. Ahora bien, la tendencia natural es a que las colecciones sean cada vez más grades (el ejemplo más claro es la Web), pero lógicamente aunque el vocabulario que los forma (los términos) también vaya creciendo está limitado al vocabulario del idioma usado, además de que luego se suele reducir el número de términos durante el preprocesamiento. Por tanto, la situación con colecciones normales es que existen muchos más documentos que términos.

Cada elemento a_{ij} de la matriz $A_{m,n}$ ofrece información sobre si el término i aparece o no en el documento j. Dicha información puede referirse únicamente a la aparición del término en el documento u ofrecer mayor información referente a

la importancia de ese término en ese documento. Para cumplir este objetivo cada elemento toma un valor, un peso, entre 0 y 1. La ausencia del término en el documento es indicada mediante un 0, y un valor positivo define el peso de dicho término en el documento. Esta forma de representación provoca que la matriz presente una enorme cantidad de elementos iguales a cero. Ya que el número de términos en un documento es un subconjunto muy pequeño de todos los términos de todos los documentos de la colección. Ello es la causa de la dispersión característica de estas matrices.

Normalmente se asume que los pesos son mutuamente independientes [11]. En otras palabras conocer el peso de un término asociado a un documento no da ninguna información sobre el peso de otro término asociado a ese mismo documento. Esta simplificación, ya que los términos de un documento están correlacionados, facilita la computación de los pesos y acelera la creación de clasificaciones (*rankings*). Además, ninguna de las soluciones propuestas en el pasado ha demostrado claramente que tener en cuenta la correlación de términos sea ventajosa para mejorar las clasificaciones, al menos en condiciones generales [11].

Esquemas de pesado

Los esquemas de pesado son un método común y eficaz para mejorar las prestaciones de recuperación en el modelo vectorial, el cual consiste en sustituir la frecuencia de aparición de un término en un documento por una función que otorgue un peso a cada elemento de la matriz de términos por documentos [28]. Esta transformación presenta tres componentes distintos, un peso local que presenta la importancia relativa de un término en un documento, un peso global que presenta la importancia de un término en relación a todos los documentos de la colección, y un factor de normalización que minimiza la influencia de la longitud de los documentos [12, 21, 23, 31].

Cada elemento a_{ij} de la matriz $A_{m,n}$ de términos por documentos presentará la siguiente forma: $a_{ij} = l_{ij} g_i d_j$ donde l_{ij} corresponde con el peso local del i-ésimo término en el j-ésimo documento, g_i corresponde con el peso global del i-ésimo término y d_j corresponde al factor de normalización del j-ésimo documento [12, 21, 23, 31].

Los esquemas de pesado se especifican mediante una combinación de seis letras [12, 31]. Las tres primeras indican la componente local, global y de normalización para la matriz de documentos, y las tres segundas determinan la componente local, global y de normalización utilizadas en la consulta. Cuando se utiliza un arterisco en lugar del símbolo de una fórmula se está denotando que puede ser cualquier fórmula.

La mayoría de los esquemas de pesado más utilizados se basan en las dos funciones siguientes:

$$\chi(r) = \begin{cases} 1 & \text{si} \quad r > 0 \\ 0 & \text{si} \quad r = 0 \end{cases} \qquad\qquad p_{ij} = \frac{f_{ij}}{\sum_j f_{ij}}$$

En la tabla 2.1 en la página siguiente se muestran los esquemas de pesado local más comunes [12, 31], que como se puede ver sólo dependen de las frecuencias intradocumento (dentro del documento) y no de las frecuencias interdocumento (entre documentos).

Símbolo	Nombre	Fórmula
b	Binario	$\chi(f_{ij})$
t	Frecuencia	f_{ij}
n	Frecuencia Normalizada Aumentada	$\frac{\chi(f_{ij})+\frac{f_{ij}}{\max_k f_{kj}}}{2}$
l	Logaritmo	$\log(f_{ij}+1)$
a	Logaritmo Alternativo	$\chi(f_{ij})(\log(f_{ij})+1)$

Tabla 2.1: Fórmulas de pesado local.

Tanto la fórmula binaria (b) como la de frecuencia de términos (t) son simples, claras y obvias. En la binaria se da igual importancia a todos los términos independientemente del número de veces que aparecen. Por su parte la fórmula de frecuencia otorga relevancia a los términos proporcionalmente al número de veces que se presentan. El primer caso puede ser útil si el número de apariciones no se considera importante, aunque normalmente es su principal desventaja. En el segundo caso usualmente se da demasiada importancia a los términos que más aparecen. La frecuencia normalizada aumentada intenta mejorarlo, para ello le da importancia a todos los términos que aparecen y además añade cierta importancia a aquellos términos que más se repiten. Por su parte las fórmulas con logaritmos (l,a) también se utilizan para desenfatizar el efecto de la frecuencia.

La elección de la componente local del esquema de pesado depende de las características de la colección, principalmente de su vocabulario. Por ejemplo, ante vocabularios técnicos o científicos los esquemas que mejor suelen funcionar son los basados en la frecuencia de términos normalizada (n**), en cambio, cuando el vocabulario es variado o generalizado, normalmente utilizando únicamente la frecuencia del término suele ser suficiente (t**) [12].

En la tabla 2.2 en la página siguiente se muestran los esquemas de pesado global más comunes [12, 31], los cuales son utilizados para enfatizar los términos que son discriminantes basándose en la dispersión de un término a través de los documentos. De hecho el pesado global teóricamente hace innecesaria la eliminación de las *stopwords*, al presentar éstas un peso global muy pequeño [31]. De todas formas, en la práctica, sí que se eliminan en la fase de preprocesamiento, puesto que de esta forma hay que trabajar con un volumen menor de términos, con las consiguientes ventajas que ello supone.

La fórmula de la entropía (e) asigna pesos entre cero y uno, siendo cero cuando el término aparece con la misma frecuencia en todos los documentos y uno si sólo aparece en un único documento. Como su propio nombre indica, la fórmula de frecuencia de documento inversa (f), hace decrecer más el peso de un término en cuantos más documentos aparece éste, si el término aparece en todos los documentos el peso global es cero. La fórmula GfIdf (g) calcula un ratio entre el número de veces total que el término aparece en la colección y el número de documentos en los que aparece.

El pesado inverso probabilístico (p) también se basa en la frecuencia de documentos inversa, concretamente asigna valores entre menos infinito (realmente elimina el término) cuando el término aparece en todos los documentos y $\log(n-1)$ cuando el término aparece en un sólo documento.

Símbolo	**Nombre**	**Fórmula**
x	Ninguna	1
e	Entropía	$1+\sum_{j=1}^{n}\left(\frac{p_{ij}\log p_{ij}}{\log n}\right)$
f	Frecuencia de documento inversa (IDF -*Inverse Document Frecuency*-)	$\log\left(\frac{n}{\sum_{k=1}^{n}\chi(f_{ij})}\right)$
g	GfIdf	$\frac{\sum_j f_{ij}}{\sum_j \chi(f_{ij})}$
n	Normal	$\frac{1}{\sqrt{\sum_j f_{ij}^2}}$
p	Inversa probabilística	$\log\left(\frac{n-\sum_{k=1}^{n}\chi(f_{ij})}{\sum_{k=1}^{n}\chi(f_{ij})}\right)$

Tabla 2.2: Fórmulas de pesado global.

A la hora de elegir el factor de pesado global hay que tener en cuenta como es de habitual que la colección cambie, ya que tener que reajustar el peso global cada vez que cambie el vocabulario afectará a todas las filas de la matriz de pesos. Por lo tanto, para evitar en la medida de lo posible tales actualizaciones se puede simplificar el esquema de pesado descuidando el factor global (*x*) cuando el vocabulario de la colección cambia continuamente. Pero cuando el vocabulario es estático (o es asumible) entonces la elección más común suele ser la frecuencia de documento inversa (*f*).

Símbolo	**Nombre**	**Fórmula**
x	Ninguna	1
n,c	Normal o Coseno	$\frac{1}{\sqrt{\sum_i (g_i l_{ij})^2}}$

Tabla 2.3: Fórmulas de normalización.

Una vez el peso local y el global están establecidos comúnmente se procede a normalizar, concretamente las columnas de la matriz de términos por documentos. Si esta normalización no se realiza puede conllevar que los documentos cortos no

sean reconocidos cómo relevantes aún siéndolo. En la tabla 2.3 en la página anterior se muestran el esquema de normalización más utilizado [12, 31], basado en la 2-norma, aunque muchas otras normalizaciones son posibles [31]. La normalización de la consulta no da lugar a ninguna diferencia en la clasificación final de los documentos recuperados, por lo cual no se realizará nunca.

Intuitivamente, con la normalización se busca retener sólo la información sobre la dirección de los vectores documentos, es decir con información únicamente sobre el tema del documento, ya que documentos con diferentes longitudes pero que versan sobre el mismo tema darán lugar a vectores similares.

Aunque existen muchos esquemas de pesado, sólo hay que contar el número de posibles combinaciones, no todos los esquemas son interesantes. Uno de los más populares y efectivos es el (tfn) que se muestra en la figura 2.5, donde el peso local utiliza la frecuencia de los términos, el peso global es la frecuencia de documento inversa y la normalización utilizada es la normal.

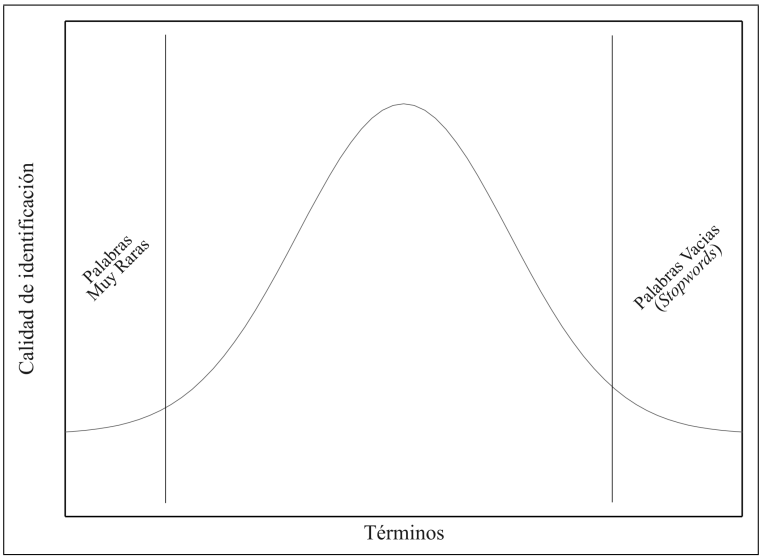

Figura 2.5: Comportamiento del esquema de pesado tfn.

La norma general de los esquemas de pesado de términos, al menos en el modelo vectorial, es la asignación de un peso cero a aquellos términos que están ausentes. Esta forma de actuar conlleva a que no se tengan en cuenta dichos términos para calcular similitudes. Mediante un nuevo esquema de pesado, el BTWS (*Balanced Term-Weighting Scheme*), se hace que los términos ausentes también contribuyan en el cálculo de similitud, dándoles un peso negativo [101]. El principal problema del esquema BTWS es su coste computacional, al utilizar todos los términos en la comparación con la consulta. Para suavizar este problema se utilizan métodos de compresión de dimensión que reducen dicho coste drásticamente, pero todavía sin llegar a un nivel tan pequeño como el del coseno, que sólo considera los positivos [101].

Utilización del modelo básico

La utilización del modelo vectorial básico consiste en transformar la consulta del usuario en un vector de pesos, según el esquema elegido, y entonces buscar que vectores documento de la matriz de términos por documentos son más parecidos al vector consulta. La medida de similitud puede ser cualquiera, aunque la más común y más utilizada suele ser el coseno del ángulo que forman el vector consulta y los vectores documento.

Sea $A_{m,n}$ la matriz de términos por documentos con m términos y n documentos, sea a_j el vector documento j-ésimo, y sea q el vector consulta. Entonces la expresión del coseno sería:

$$\cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^{m} a_{ij} q_i}{\sqrt{\sum_{i=1}^{m} a_{ij}^2}\sqrt{\sum_{i=1}^{m} q_i^2}} \tag{2.1}$$

Hay que tener en cuenta que las normas de los documentos pueden estar precalculadas y que la norma de la consulta no afecta a la clasificación final, con lo cual el cálculo se puede acelerar. Además, si el esquema de pesado incluye normalización, entonces ni siquiera habría que preocuparse de las normas ya que éstas valdrían uno ($\|a_j\|_2 = 1$; $\|q\|_2 = 1$). Esta situación es muy habitual, con lo cual normalmente la medida de similitud utilizada es el producto escalar entre el vector documento y el vector consulta:

$$\cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{a_j^T q}{1} = a_j^T q \tag{2.2}$$

Al igual que la matriz de pesos (matriz de términos por documentos), el vector consulta es de naturaleza dispersa, lo cual implica que todos los cálculos en el modelo vectorial se pueden optimizar recurriendo a técnicas que aprovechen dicha estructura dispersa.

Una vez calculado el grado de similitud entre el vector consulta y todos los vectores documento, se ordenan los documentos de mayor a menor similitud con la consulta. Si hay algún criterio de corte, todos los documentos que no llegan al umbral de similitud establecido no se presentan al usuario.

2.3.3. Modelo de indexación semántica latente (LSI): Mediante la SVD

En el modelo vectorial la elección de los documentos relevantes a una consulta se realiza basándose en emparejamientos literales de términos (o de transformaciones léxicas). Pero existen diversas formas de expresar un concepto (sinonimia) y varios conceptos se pueden expresar de la misma forma (polisemia). Al contemplar únicamente la relación léxica se pueden perder otras formas de expresar el mismo concepto

y recuperar conceptos que se expresan igual. El modelo de Indexación Semántica Latente (LSI -*Latent Semantic Indexing*-) [27], una variación del modelo vectorial, trata de superar esta situación pasando a una recuperación de información conceptual, basándose en la asunción de que existe una estructura latente en los términos utilizados, la cual se encuentra oculta por la variabilidad en la elección de términos [28, 33]. Ya que en la LSI las búsquedas están fundamentadas en los conceptos y no en los términos se pueden obtener documentos que no comparten ningún término con la consulta (sinonimia), e incluso no recuperar documentos que sí que comparten términos con la consulta (polisemia).

Las colecciones de documentos grandes dan lugar, en prácticamente todos los casos, a matrices de términos por documentos que no son de rango completo, es decir no son todas las columnas linealmente independientes. En este hecho se fundamenta la LSI para llegar a la indexación conceptual (búsqueda por conceptos), ya que aproxima la matriz de pesos de términos por documentos en una nueva matriz de rango menor [12]. La idea principal que reside en la reducción de dimensionalidad, de rango, es proyectar cada documento dentro de un espacio dimensional más pequeño, el cual de forma explícita tendrá en cuenta la relación y dependencia entre términos. El rango k de la matriz aproximada, será mucho menor que el rango de la matriz de pesos mín (m, n), es decir $k \ll$ mín (m, n). La reducción del rango de la matriz de términos por documentos es, además, un medio de eliminar información extraña, ruido, de la colección [28, 33]. Ahora bien, la elección del número de dimensiones (k), es decir, del rango, es un problema importante, ya que cuanto menor sea k más ruido se puede eliminar, pero reducir demasiado el rango puede dar lugar a perder información importante, y por otro lado hay que tener cuidado de no reconstruir la matriz original de pesos perdiéndose las ventajas de la reducción de rango. Realmente la determinación del rango óptimo con el cual aproximar la matriz de términos por documentos es una cuestión abierta, aunque está bastante claro que es muy dependiente de la colección concreta con la que se esté trabajando [12]. El hecho que la LSI trabaje bien con un relativo número pequeño de dimensiones respecto al número de términos de la colección, muestra que realmente se está capturando parte importante de la estructura subyacente.

La LSI puede conseguir mejores prestaciones de recuperación de información que otras técnicas vectoriales [2, 33, 34], sin embargo no es bien entendida la razón de las buenas prestaciones de la LSI, la búsqueda de dicha razón es un área activa de investigación [24]. En cambio la complejidad del modelo LSI suele conllevar que sus prestaciones computacionales, de ejecución y respuesta, estén muy por debajo de los otros modelos vectoriales. Aunque mediante diferentes optimizaciones de implementación las prestaciones de la LSI podrían acercarse mucho a las demás técnicas vectoriales. Otras ventajas importantes de este modelo es su automatismo y su facilidad de uso y retroalimentación.

Existen multitud de técnicas para la reducción dimensional de una matriz, entre ellas destacan la factorización QR, el análisis de componentes principales (PCA -*Principal Component Analysis*-) y la descomposición en valores singulares (SVD -*Singular Value Decomposition*-). Siendo esta última la más utilizada debido principalmente a que permite la aproximación de bajo rango de la matriz original con el

mínimo error [12, 29]. Además, mientras la aproximación QR únicamente suministra una base de rango reducido para el espacio de las columnas de la matriz de términos por documentos, sin aportar información sobre las filas, la SVD, aunque con mayor coste computacional, ofrece una aproximación de bajo rango de ambos espacios, el de columnas y el de filas.

Base matemática

La factorización QR transforma la matriz de términos por documentos $A_{m,n}$ en dos matrices de la siguiente forma: $A = QR$, donde la matriz $Q \in \Re^{m \times m}$ es ortogonal y la matriz $R \in \Re^{m \times n}$ es triangular superior [12, 29, 102]. La factorización QR existe para cualquier matriz y hay diversos métodos para calcularla. La factorización QR también se puede representar como sigue:

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1 + Q_2 0 = Q_1 R_1 \tag{2.3}$$

En la expresión 2.3, Q_1 es la porción de la matriz Q asociada a R_1, Q_2 es el resto de la matriz Q y R_1 representa las filas de R distintas de cero. Claramente, las columnas de Q_2 no contribuyen para producir la matriz A, lo cual implica que el rango de las matrices A, R y Q_1 es el mismo. Como las columnas de la matriz A son todas combinación lineal de las columnas de Q, un subconjunto de k columnas de la matriz Q, concretamente las columnas de Q_1, forman una base para el espacio de columnas de la matriz A, donde $k = \text{rango}\,(A)$ [12, 29]. Cuando se utiliza la factorización QR en el modelo LSI, el cálculo de coseno del ángulo que forman los vectores documento y los vectores consulta sufre una lógica modificación [12, 29], presentada en la expresión 2.4:

$$\left.\begin{array}{l} A = [a_j]\,\forall j \in [1,n] \\ R_1 = [r_j]\,\forall j \in [1,n] \end{array}\right\} \cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{(Q_1 r_j)^T q}{\|Q_1 r_j\|_2 \|q\|_2} = \frac{r_j^T \left(Q_1^T q\right)}{\|r_j\|_2 \|q\|_2} \tag{2.4}$$

Otra de las técnicas utilizadas para la reducción de dimensionalidad es el análisis de componentes principales (PCA -*Principal Component Analysis*-), el cual consiste en utilizar los k principales vectores propios de la matriz de covarianza de la matriz de términos por documentos [24]. La principal desventaja de la PCA son los altos requerimientos computacionales y espaciales que precisa.

Por último, otra técnica es la descomposición en valores singulares, que existe para cualquier matriz A y se define como $A = USV^T$, donde $U \in \Re^{m \times m}$ es la matriz ortogonal cuyas columnas se corresponden con los vectores singulares izquierdos de A, $V \in \Re^{n \times n}$ es la matriz ortogonal cuyas columnas se corresponden con los vectores singulares derechos de A, y $S \in \Re^{m \times n}$ es la matriz diagonal cuyos elementos definen los valores singulares $s_1 \geq s_2 \geq \ldots \geq s_{\text{mín}(m,n)}$ de A ordenados a lo largo de su diagonal [12, 14, 23, 25, 29, 33, 102]. Existen diversos métodos para computar la SVD tanto de matrices densas como dispersas y la mayoría de ellos están muy bien documentados [12, 14, 102]: Householder, rotaciones de Givens, método de la potencia, iteración del subespacio, Lanczos, Arnoldi, etc. Los métodos que trabajan con matrices dispersas (Arnoldi, Lanczos, iteración del subespacio, etc.) sólo referencian a ésta a

través de operaciones de multiplicación de matriz por vector, y por lo tanto pueden ser implementadas aprovechando los formatos de almacenamiento dispersos [12, 29].

Los primeros k valores singulares (los primeros k elementos diagonales de S), junto con las k primeras columnas de U y V, se utilizan para obtener la aproximación de rango k de la matriz de términos por documentos, es decir, en realidad se hace uso de la SVD truncada: $A_k = U_k S_k V_k^T$ [25]. Los cambios relativos en los valores singulares al pasar del mayor al menor puede dar información para determinar que dimensión del subespacio reducido es la apropiada para modelizar el espacio de términos por documentos [34].

En el cálculo de la SVD se aprovecha la característica dispersa de la matriz de términos por documentos, pero no se hace ningún esfuerzo por preservar dicha dispersión en la aproximación de bajo rango. De hecho las matrices de vectores singulares (U_k y V_k) son normalmente densas, lo que conlleva grandes requerimientos de almacenamiento, superiores a los necesarios para la matriz de términos por documentos dispersa. Y, aunque la matriz de bajo rango no se reconstruye (se trabaja con las matrices U_k, S_k y V_k) los costes computacionales también aumentan, eso sí menos significativamente que los costes de almacenamiento. Como posible solución a este decremento de prestaciones aparece una variante de la SVD, la SDD (*Semi-Discrete Decomposition*) que utiliza únicamente tres valores ($\{-1, 0, 1\}$ representados por dos bits cada uno) para definir los elementos de U_k y V_k, reduciendo así los requerimientos de almacenamiento, aunque los computacionales aumentan al necesitar resolver una secuencia de problemas de programación entera para producir la descomposición [12, 30].

A la hora de calcular la similitud de las consultas con los documentos de la matriz de términos por documentos reducida mediante el coseno del ángulo que forman los respectivos vectores, es necesario sustituir la matriz de pesos por su correspondiente descomposición en valores singulares. Si se define e_j como el j-ésimo vector canónico de dimensión n (es decir, es la j-ésima columna de la matriz identidad de dimensión n, $I_{n,n}$), entonces el vector $A_k e_j$ es simplemente la j-ésima columna de la matriz A_k (es decir, el vector columna j-ésimo). Teniendo en cuenta todo esto el cálculo del coseno del ángulo sería [12, 33]:

$$\cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{(A_k e_j)^T q}{\|A_k e_j\|_2 \|q\|_2} = \frac{\left(U_k S_k V_k^T e_j\right)^T q}{\|U_k S_k V_k^T e_j\|_2 \|q\|_2} = \frac{e_j^T V_k S_k \left(U_k^T q\right)}{\|S_k V_k^T e_j\|_2 \|q\|_2} \tag{2.5}$$

Utilización del modelo LSI

El proceso que se sigue en el modelo LSI es análogo al utilizado en el modelo vectorial básico.

1. A partir de la colección de documentos se genera una matriz de pesos de términos por documentos, de igual forma que en el modelo vectorial básico. La elección del esquema de pesado y otros aspectos del preprocesamiento, se elegirán en relación a la colección, independientemente de que se vaya a usar el modelo LSI.

2. El siguiente paso es hacer la descomposición truncada en valores singulares de la matriz de pesos. El bajo rango deseado y por lo tanto el número de valores singulares a calcular se determinará en función de la información disponible y por el método que se considere oportuno.

3. Las consultas se evaluarán proyectando las mismas dentro del subespacio de relaciones conceptuales construido. Para ello se utiliza el coseno del ángulo que forman el vector consulta y el vector documento, ambos proyectados, según se muestra en la ecuación 2.5 en la página anterior. Todos estos cálculos no precisan la reconstrucción de la matriz de bajo rango, ya que se puede hacer uso directamente de las matrices U_k, S_k y V_k.

2.3.4. Modelo de clustering

La idea fundamental subyacente a los modelos de clustering es particionar la colección de documentos en diferentes grupos (los *clusters*), donde los miembros de una agrupación son documentos más similares entre sí que a los pertenecientes a las demás agrupaciones [5, 6, 103]. La hipótesis de base es que los documentos relevantes tienden a estar más cerca los unos de los otros que los documentos no relacionados [92].

No hay que confundir la clasificación con el *clustering*, mientras el primero asigna elementos a conjuntos de categorías predefinidos, el *clustering* no dispone de clases predefinidas o información sobre que relación debería ser validada entre los elementos, es un proceso no supervisado.

El *clustering* puede encontrarse en muy diferentes contextos a parte de la recuperación de información: generación de hipótesis, prueba de hipótesis, reducción de información, predicción basada en grupos, negocios, análisis de datos espaciales, minería en la web, etc.; aunque también puede aparecer referenciado de distintas formas tales como aprendizaje no supervisado en reconocimiento de patrones, taxonomías numéricas en biología o ecología, tipologías en ciencias sociales, o particiones en teoría de grafos [5].

El *clustering* en la recuperación de información

La recuperación de información se puede modelizar como un problema de *clustering* haciendo la siguiente analogía. La colección de documentos es el conjunto de objetos C y la consulta del usuario es una especificación vaga del conjunto de objetos A, entonces el problema de encontrar los documentos relevantes a la consulta se traduce en el problema de determinar que objetos del conjunto C pertenecen al conjunto A y cuales no [11]. En este proceso hay que hacer especial hincapié en dos hechos, el primero sería determinar cuales son las características que mejor describen a los objetos del conjunto A para establecer la similitud intracluster, y segundo cuales son las que mejor los distinguen del resto de objetos de C para cuantificar la diferencia intercluster [11]. Uno de los grandes problemas de las técnicas de *clustering* es determinar el número de *clusters* (de agrupaciones) que el conjunto de datos presenta, de hecho debido a su particular dificultad los algoritmos normalmente lo ignoran [103].

El procedimiento de clustering debería idealmente ser eficiente y entero. La eficiencia se suele medir en base al tiempo requerido para realizar el *clustering* y la entereza teórica se valora bajo los siguientes criterios [2]:

- El particionado no debe variar drásticamente con la inserción de documentos.
- Pequeños errores de descripción de los documentos deben ocasionar pequeños cambios de particionado.
- El método debe ser independiente de la ordenación inicial de los documentos.

Las técnicas de *clustering* trabajan directamente con la colección de documentos y no con el texto de los documentos [11]. Esta es la razón por la que el modelo de *clustering* no pertenece a ninguno de los paradigmas de recuperación de información (lógico, vectorial o probabilístico), ya que el tratamiento del texto puede estar basado en cualquiera de dichos paradigmas, y luego utilizar el modelo de *clustering*. No hay que olvidar que el *clustering* pretende trabajar con la estructura intrínseca de la colección [4].

Inicialmente el *clustering* fue investigado para mejorar la precisión y la cobertura (*recall*) de los sistemas de recuperación de información y como un camino eficiente de encontrar los vecinos más próximos de un documento. Después, fue propuesto para usarse en la navegación por las colecciones de documentos, o en la organización de los resultados de una consulta, o en la generación automática de jerarquías de documentos, o en el análisis de conceptos latentes en conjuntos de documentos desestructurados, etc. [23, 38]

En la literatura existe una vasta colección de algoritmos de *clustering* disponibles, lo cual puede a priori confundir a la hora de seleccionar cual es el más conveniente para un problema dado, pero no hay que olvidar que no hay una técnica de *clustering* universal que se pueda aplicar a todos los conjuntos de datos [3, 39]. No todas las técnicas de *clustering* pueden encontrar todos los *clusters*, y aún pudiéndolos encontrar no todos lo hacen con la misma facilidad. Esto es debido a las asunciones implícitas que cada algoritmo hace, tales como la forma de los *clusters* buscados, la configuración de múltiples *clusters* o los criterios de agrupación entre *clusters* [3, 39]. Toda esta multitud de métodos de *clustering* disponibles en la literatura puede ser clasificada fundamentalmente en los siguientes tipos: *clustering* particional, *clustering* jerárquico, *clustering* basado en densidades y *clustering* basado en *grid* [3, 5, 6, 40].

Utilización del modelo *clustering*

Al igual que en el modelo LSI o en el vectorial básico se parte de una matriz de pesos de términos por documentos generada a partir de la colección de documentos. A continuación se utiliza el método de *clustering* elegido y se obtienen los distintos *clusters* de la colección, en función de la técnica elegida pueden estar estructurados en una jerarquía. La consulta del usuario se compara inicialmente con los representantes de cada *cluster*, generalmente el centroide, mediante una función de cercanía o similitud, generalmente el coseno. El *cluster* más relacionado con la consulta se elegirá y sólo

los documentos incluidos en dicho *cluster* serán devueltos como relevantes, ordenados por su grado de proximidad con la consulta.

Los métodos de *clustering* se basan en que los documentos similares a uno que es relevante a una consulta, también tenderán a ser relevantes a la misma consulta, consiguiendo con ello mejorar la cobertura (*recall*) del sistema de recuperación de información gracias a la determinación previa de los *clusters* [89].

2.4. Tratamiento de la consulta

La consulta del usuario debe ser introducida al sistema de recuperación de información de tal forma que éste sea capaz de interpretarla y resolverla, este proceso se realiza en tres fases. La primera conlleva la formulación de la consulta por parte del usuario intentando representar su necesidad de información, esta formulación dependerá mucho de la experiencia y conocimientos del usuario, así como de la forma que el propio sistema de recuperación de información permita introducir la consulta.

En una segunda fase, el sistema de recuperación de información traducirá la consulta realizada por el usuario al modelo con el que trabaja. Convertirá las palabras, frases y/o operadores utilizados por el usuario en los términos y relaciones que utiliza para representar la colección de documentos. Por ejemplo, se eliminarán de la consulta cualquier término que no se haya utilizado para caracterizar la colección de documentos (ello incluye las *stopwords*), se utilizarán métodos de *stemming* para generar los términos de la consulta si se utilizaron para obtener los términos de la colección. Se podría dar el caso, también, de expandir la consulta añadiendo términos sinónimos o relacionados (mediante tesauros) a aquellos que aparecen en la consulta.

Por último, en la tercera fase, el sistema de recuperación de información utilizará la consulta traducida para obtener los documentos de la colección que son relevantes para dicha consulta, presentándolos al usuario. Puede existir una cuarta fase optativa que trataría de mejorar la consulta mediante retroalimentación de la misma.

2.4.1. Tipos de consultas

Existen diversos tipos de consultas: booleanas, vectoriales, en lenguaje natural, etc.; y cada tipo, además, se puede sofisticar todavía más, fundamentalmente permitiendo utilizar información de dos tipos: de proximidad entre términos y estructural del documento [11, 12].

El tipo de consulta que el usuario podrá formular depende directamente de que tipo o tipos de consulta acepte el sistema de recuperación de información que se va a utilizar, y ello dependerá concretamente de la modelización subyacente al sistema.

La preferencia por parte del usuario del tipo de consulta dependerá en parte de su formación y conocimientos, mientras que un usuario más novato o más inexperto probablemente estará más cómodo con una consulta en lenguaje natural un profesional de la información seguramente prefiera un tipo de búsqueda booleana con parámetros de proximidad entre términos [4, 12].

Una consulta booleana está formada por términos que recuperan documentos y por operadores booleanos que operan dichos conjuntos de documentos recuperados, devolviendo al final un conjunto concreto de documentos [11]. Los operadores booleanos más habituales son: la disyunción lógica (O), la conjunción lógica (Y) y la negación (No). Este tipo de consulta es el más antiguo y todavía es muy utilizado, aunque la mayoría de los usuarios no están bien entrenados en operadores booleanos [11, 12].

En las consultas de tipo vectorial se introducen una serie de términos que determinan los documentos relevantes, pero no se utilizan operadores como en las consulta booleanas, de hecho los operadores booleanos generalmente son reconocidos como *stopwords* y por tanto ignorados [11, 12]. La consulta vectorial, compuestas por términos o palabras clave, son intuitivas, fáciles de expresar y permiten clasificaciones fáciles [11]. En muchas ocasiones este tipo de consultas están formadas por una única palabra, aunque la situación más normal, al menos en Internet, es suministrar unas pocas palabras sobre todo dos o tres. Si la consulta vectorial se comportase como una consulta booleana con operadores disyuntivos (O) dos o tres términos podría ser un buen número, pero al estar basado en el modelo vectorial más términos pueden dar lugar a mejores prestaciones, al definir y sintetizar mejor la consulta [12].

Las consultas en lenguaje natural son formuladas como una oración, es entonces cuando los sistemas de recuperación de información deben extraer los términos que se utilizarán para realizar la búsqueda real; con este tipo de actuación se suele perder el contexto de las palabras perdiendo con ello información discriminante [12].

Algunos sistemas de recuperación de información permiten tener en cuenta en las búsquedas la proximidad de los términos buscados, es decir, tienen la habilidad de buscar términos en un contexto dado. Esta característica se fundamenta en que los términos que aparecen cerca de otro pueden dar mayor probabilidad de relevancia que si aparecen lejos [11]. La proximidad puede ser de dos tipos: los términos están contiguos formando una frase o están cerca pudiendo haber cierta distancia entre ellos. La distancia se puede medir de varias formas, destaca la medición por caracteres o términos. Para poder implementar este tipo de búsquedas el sistema de recuperación debe guardar, además de que términos aparecen en cada documento, también información sobre la posición que ocupan cada uno de ellos. Esto conlleva mayores capacidades y requerimientos de computación y almacenamiento.

Si el sistema de recuperación de información dispone de información sobre la estructura de los documentos de la colección, entonces se puede aprovechar ésta para realizar las consultas incluyendo la estructura de los documentos. Es decir, permiten al usuario realizar consultas basándose en el contenido y en la estructura del documento, lo cual puede dar lugar a consultas muy potentes y expresivas [11].

2.4.2. Retroalimentación de la consulta

La retroalimentación es un método iterativo de mejora de la consulta basándose en los documentos relevantes de dicha consulta [4, 11, 12, 28, 33, 104]. La idea subyacente es que los términos incluidos en los documentos relevantes a una consulta tienen similitudes entre ellos y que los términos que aparecen en los documentos no relevantes son diferentes a los de los documentos relevantes.

Por tanto, el procedimiento a seguir consistirá en reformular la consulta de tal forma que se acerque más a los términos de los documentos relevantes (se da más peso a estos términos) y si es posible se aleje más de los términos de los documentos no relevantes (se reduce el peso de estos términos). Con esta forma de actuar se pueden mejorar las prestaciones de recuperación en términos de precisión y cobertura (*recall*). Existen experimentos que han mostrado que al reemplazar la consulta con una combinación de unos pocos documentos relevantes la precisión mejora para algunas colecciones [12, 29].

Todo el rato se está hablando de aprovechar la información dada por documentos relevantes, pero lógicamente éstos son el objetivo de la consulta y por lo tanto no se dispone de ellos (una vez alcanzados no tendría sentido seguir con la búsqueda). Entonces lo que hacen los sistemas de recuperación de información es utilizar documentos que a priori se suponen relevantes, o que al menos lo son parcialmente. Hay dos métodos básicos de llevar a cabo la retroalimentación:

Automático: El sistema evalúa la consulta y obtiene una clasificación de documentos ordenados por relevancia. Entonces selecciona los primeros de la clasificación y los utiliza para realizar la retroalimentación. Tiene la ventaja de que no precisa la participación del usuario (se hace transparentemente a él) y que es rápido. En contrapartida, la eficacia del método dependerá totalmente de las prestaciones iniciales del sistema. Si los primeros documentos de la clasificación no son realmente relevantes en las siguientes iteraciones las prestaciones empeorarán, por ello es importante priorizar la precisión a la cobertura (*recall*).

Manual: El usuario evalúa unos pocos documentos de la clasificación ofrecida por el sistema de recuperación y determina si son o no son relevantes. En el paso siguiente el sistema utiliza dicha selección para realizar la retroalimentación. Tiene la ventaja que los documentos que pretenden mejorar la consulta son realmente relevantes o no para el usuario, y por tanto la consulta se acercará o alejará de ellos de forma correcta. El principal inconveniente es que el proceso es lento, ya que se deben analizar algunos documentos. De hecho será tanto más lento cuantos más documentos se evalúen.

Basándose en que sea D_r el conjunto de documentos relevantes identificados por el usuario entre los documentos recuperados, sea D_n el conjunto de documentos no relevantes entre los documentos recuperados, sea C_r el conjunto de documentos relevantes entre todos los documentos de la colección; basándose también en que $|D_r|$, $|D_n|$ y $|C_r|$ sean el número de documentos de los conjuntos D_r, D_n y C_r respectivamente; y en que α, β y γ sean constantes de ajuste; se puede definir de forma más matemática la retroalimentación óptima ideal por la siguiente fórmula (siendo n el número de documentos total de la colección) [11].

$$q_{\text{opt}} = \frac{1}{|C_r|} \sum_{\forall d \in C_r} d - \frac{1}{n - |C_r|} \sum_{\forall d \notin C_r} d \tag{2.6}$$

La fórmula 2.6 define una situación ideal e irreal, ya que el conjunto completo C_r de documentos relevantes para una consulta q no se conoce, de hecho ese es el

conjunto de documentos buscado. Entonces lo que se hace es reformular la consulta en base a los conjuntos D_r y D_n que sí se pueden conocer. Aunque existen multitud de maneras de hacerlo es famosa la formulación de Rochio [11]:

$$q_m = \alpha q + \frac{\beta}{|D_r|} \sum_{\forall d \in D_r} d - \frac{\gamma}{|D_n|} \sum_{\forall d \in D_n} d \tag{2.7}$$

En la fórmula de Rochio (expresión 2.7) destaca que los términos de la consulta original son utilizados y que usualmente a la información contenida en los documentos relevantes se le da más importancia que a la suministrada por los documentos no relevantes (por tanto la constante γ será más pequeña que la constante β). Una alternativa, también ampliamente utilizada, es utilizar sólo una estrategia de retroalimentación positiva estableciendo γ a cero.

La principal ventaja de la técnica de retroalimentación de Rochio (expresión 2.7) es su simplicidad y sus buenos resultados. La simplicidad se debe a que los términos modificados se obtienen directamente de los documentos recuperados. Y los buenos resultados se observan experimentalmente, los cuales se deben a que la consulta modificada refleja una parte de la semántica pensada, de la necesidad de información real del usuario.

Otra forma de mejorar la consulta consistiría en expandirla, añadiendo o sustituyendo cada término de la consulta original con un sinónimo o un término relacionado mediante, por ejemplo, un tesauro [4]. Lógicamente, esta forma de actuar no excluye al método citado anteriormente, siendo ambos complementarios.

2.5. Métodos de evaluación

Cuando se crea un nuevo sistema de recuperación de información (básicamente un nuevo algoritmo) o se modifica uno ya existente hay que estudiar si éste aporta mejores prestaciones, y en que magnitud lo hace. Este problema es muy sutil y resbaladizo, y normalmente se embellece o desenfatiza [43]. A la hora de valorar la calidad de un sistema de recuperación o algoritmo se plantea la difícil y delicada tarea de balancear tres aspectos fundamentales: velocidad, precisión y cobertura [14].

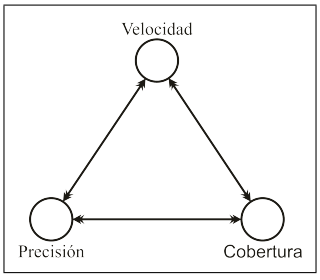

Figura 2.6: Relación de equilibrio de prestaciones.

Como se puede observar en la figura 2.6 en la página anterior [14] llegar a mejorar los tres factores es imposible, siempre hay alguno que empeora al mejorar los otros dos, por lo cual es necesario llegar a un compromiso entre ellos.

También se podría hablar de un cuarto aspecto importante para medir la calidad de un sistema, los requerimientos de espacio de almacenamiento, valorándose más cuanto menos espacio sea necesario. De hecho, en general, para cualquier algoritmo las prestaciones de respuesta se miden en función del tiempo de respuesta (velocidad) y de la complejidad espacial (espacio de almacenamiento requerido). Lo que se particulariza es la medida de prestaciones referentes a la calidad de los resultados, ya que dependiendo de los objetivos del algoritmo dichas prestaciones se miden de una forma u otra. En la recuperación de información se han desarrollado algunos estándares de evaluación que permiten discutir y comparar la calidad de los resultados de diferentes búsquedas en distintos sistemas de recuperación de información [12].

La evaluación de la calidad de las respuestas de un sistema de recuperación de información se suele basar en la medición directa o indirectamente de la precisión (los documentos devueltos son sólo los relevantes) y la cobertura (todos los documentos relevantes son devueltos) de dicho sistema en relación a una colección de prueba (conjunto de documentos y consultas de las cuales se conocen los documentos relevantes).

Cambia la forma de evaluar el resultado de una consulta del de una sesión interactiva (o una combinación de ambas). Durante una sesión interactiva se debería evaluar las características del diseño de la interfaz, la ayuda del sistema, la duración de la sesión y el esfuerzo del usuario, y aunque pueden ser aspectos críticos, pero en la evaluación de los resultados de una consulta ninguno de estos aspectos es importante en comparación con la calidad del conjunto respuesta devuelto [11].

Existen diferentes medidas de evaluación de calidad y dependiendo de las utilizadas pueden variar los resultados al evaluar distintos sistemas de recuperación; sin embargo, si un algoritmos funciona mejor que otro en varias de esas posibles medidas de evaluación entonces se puede decir que es mejor, al menos para la situación que se está evaluando [38].

2.5.1. Evaluación específica de clusters

Los métodos de *clustering* tienen algunas características específicas respecto a otros métodos de recuperación de información, por ello mismo también tienen ciertas características propias a la hora de ser evaluados. De hecho, casi lo más importante es determinar si los *clusters* generados contribuyen a una recuperación de información efectiva [4], en otras palabras, si son útiles.

Cualquier proceso de *clustering* puede ser realizado [43]:

- Por expertos humanos: Se asume que estas particiones son "correctas", aunque el problema es que los expertos humanos únicamente pueden procesar pequeñas cantidades, y que su evaluación es subjetiva.

- Por métodos automáticos: Tienen la ventaja de ser capaces de procesar ingentes cantidades de información, pero su evaluación no es sencilla.

La evaluación de los resultados obtenidos por un algoritmo de *clustering* se puede enfocar desde dos puntos de vista distintos [4, 5, 7, 8, 38, 43, 103]:

- Evaluación de la calidad externa: Se asume que es buena si da lugar a la misma partición que un experto humano. La entropía o la medida F son ampliamente usadas para evaluar la calidad externa.
- Evaluación de la calidad interna: Se asume que no hay información externa (incluida una partición por expertos humanos), por tanto, sólo el algoritmo es evaluado, no el proceso entero de *clustering*. La similitud global es muy utilizada para evaluar la calidad interna.

Los méritos y peligros de ambas evaluaciones se puede resumir en [43]:

- Maximizar la calidad externa es el objetivo final en cualquier aplicación práctica de *clustering*. El principal límite de la evaluación de la calidad externa es que es subjetiva, dado que es fuertemente dependiente de un proceso de *clustering* dirigido por humanos.
- Usar la calidad interna es el mejor camino de medir las prestaciones de un algoritmo de *clustering*. Obviamente alta calidad interna no garantiza buenos resultados del proceso global de *clustering* en una aplicación específica, dado que tales resultados también dependen de otros pasos críticos como la selección de características y el preprocesamiento.

2.5.2. Medidas de evaluación estándar

La precisión (*precision*) y la cobertura (*recall*) son dos de las definiciones estándares más usadas en la evaluación de sistemas de recuperación de información. Cómo están directamente relacionadas se suelen estudiar juntas, y prácticamente todas las medidas de evaluación existentes se basan directa o indirectamente en ellas.

Siendo D_r el número de documentos relevantes recuperados y D_t el número total de documentos recuperados; la precisión P se define como la fracción de los documentos recuperados que son relevantes:

$$P = \frac{D_r}{D_t} \tag{2.8}$$

Siendo, de nuevo, D_r el número de documentos relevantes recuperados y N_r el número total de documentos relevantes en la colección; la cobertura (*recall*) R se define como la fracción de los documentos relevantes que son recuperados:

$$R = \frac{D_r}{N_r} \tag{2.9}$$

En la figura 2.7 se muestra gráficamente el resultado de una consulta, en dicha imagen se puede observar claramente los conjuntos de documentos que determinan la precisión (expresión 2.8 en la página anterior) y la cobertura (expresión 2.9 en la página anterior).

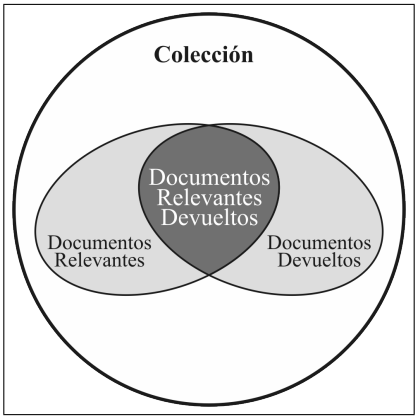

Figura 2.7: Representación del resultado de una consulta.

Es importante no olvidar que la relevancia de un documento es una cuestión subjetiva, de hecho dos sistemas de recuperación ante la misma consulta pueden dar distinta relevancia al mismo documento. También hay que tener en cuenta que el número total de documentos relevantes nunca es conocido, aunque ello no reduce la utilidad de la cobertura.

La relación entre precisión y cobertura es clara, un sistema de recuperación de información busca la mayor precisión y la mayor cobertura, pero ambas características se afectan mutuamente obligando a llegar a un compromiso entre ellas. Si se desea mucha precisión se corre el riesgo de perder documentos relevantes. Pero si se obtienen muchos documentos respuesta para garantizar la obtención de todos los relevantes, entonces existirá mucho ruido, muchos documentos que realmente no son relevantes.

La precisión y la cobertura asumen que todos los documentos del conjunto respuesta han sido examinados. Sin embargo, el usuario examina la clasificación desde el documento más relevante, con lo cual, la precisión y la cobertura irán variando a medida que el usuario revise los documentos [11]. Sea D_i el número de documentos relevantes hasta la posición i, inclusive, de la clasificación de documentos devuelta. Entonces la cobertura en el documento i-ésimo de la clasificación, R_i se define como la proporción de documentos relevantes hasta ese momento ($R_i = \frac{D_i}{N_r}$). Y la precisión en el documento i-ésimo, P_i se define como la proporción de documentos hasta la posición i, inclusive, que son relevantes para la consulta dada ($P_i = \frac{D_i}{i}$).

Un gráfico de precisión versus cobertura representa p puntos de precisión generados sobre la cobertura de forma uniformemente espaciada tomando valores entre 0 y 1, formando siempre una curva no incremental [31]. La curva de precisión versus cobertura normalmente se basa en once niveles (11 puntos) de cobertura estándar

(0 %, 10 %, 20 %,..., 100 %), los cuales son obtenidos por interpolación [11]. En la figura 2.8 se muestran tres curvas típicas de precisión versus cobertura para los once niveles estándares.

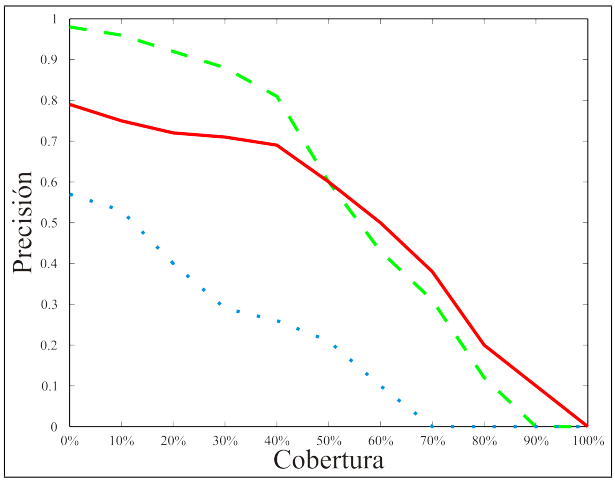

Figura 2.8: Curvas típicas de precisión versus cobertura.

El procedimiento de interpolación, imprescindible en casos tales como cuando una consulta presenta menos de diez documentos relevantes, se basa en elegir la precisión máxima entre un nivel y el siguiente. Sea r_j la referencia a los j-ésimos niveles de cobertura estándar, teniendo en cuenta que $j \in \{0, 1, 2, ..., 10\}$. Entonces la precisión interpolada en el nivel de cobertura estándar j-ésimo, el cual es la máxima precisión conocida en cualquier nivel de cobertura entre el nivel j-ésimo y j-ésimo más uno, es:

$$P(r_j) = \max_{r_j \leq r \leq r_j+1} P(r) \tag{2.10}$$

De esta forma se consigue la curva de precisión versus cobertura para una consulta simple, pero lo normal es evaluar mediante varias consultas distintas. En estos casos lo que se genera es la media de todas las curvas de precisión versus cobertura. Siendo N_q el número de consultas y $P_i(r)$ la precisión al nivel de cobertura r para la consulta i-ésima, la precisión media en el nivel r será:

$$\overline{P}(r) = \sum_{i=1}^{Nq} \frac{P_i(r)}{N_q} \tag{2.11}$$

Una opción alternativa o adicional a la curva media de precisión versus cobertura es la precisión media en un valor de corte de documentos establecido (por ejemplo al encontrar 5, 10, 20, 50 ó 100 documentos relevantes). En general recibe el nombre de *r-precision* definiéndose como la precisión después de r documentos y siendo la *r-precision* para múltiples consultas la media de las *r-precision* para todas ellas aunque dicha media puede ser bastante imprecisa [11, 31].

La utilidad de las curvas de precisión versus cobertura radica en que permiten evaluar cuantitativamente tanto la calidad del conjunto de respuestas como la calidad de la cobertura del sistema de recuperación de información, permitiendo así comparar diferentes sistemas. Además, son intuitivas y se pueden combinar en una única curva simple. De todas formas, a veces, también es interesante utilizar otras medidas que combinen la precisión y la cobertura en una única medida simple, ya que éstas están relacionadas y capturan aspectos diferentes a tener en cuenta.

2.5.3. Otras medidas de evaluación menos comunes

Aunque la precisión y la cobertura son medidas estándares y muy populares, no siempre son las medidas más apropiadas para evaluar las prestaciones de recuperación [11], por ello durante años han ido apareciendo y siendo propuestas numerosas medidas alternativas, entre las que destacan las siguientes:

Media harmónica: Sirve para medir la calidad de un cluster y su definición combina la precisión y la cobertura estándar del j-ésimo documento de la clasificación, tomando el valor uno con la precisión y cobertura perfectas y el valor cero con precisión y cobertura nulas [11, 92]:

$$F_j = \frac{2}{\frac{1}{p_j} + \frac{1}{r_j}} \tag{2.12}$$

Medida *E* (*E measure*): También combina la precisión y la cobertura para el j-ésimo documento de la clasificación, pero permitiendo especificar la importancia relativa entre ellas mediante el parámetro b ($b = 1$ equivale al complemento de la media harmónica, $b > 1$ potencia la precisión y $b < 1$ potencia la cobertura [11]:

$$E_j = 1 - \frac{1 + b^2}{\frac{b^2}{r_j} + \frac{1}{p_j}} \tag{2.13}$$

Entropía: Definiendo p_{ij} como la probabilidad de que un miembro del *cluster* j pertenezca al *cluster* i, la entropía de cada cluster j sería E_j y la entropía total E_T es la suma de las entropías de cada *cluster* pesadas por sus tallas, siendo n_j la dimensión del *cluster* j, m el número de *clusters* y n el número total de datos [92]:

$$E_j = -\sum_i p_{ij} \log p_{ij} \qquad E_T = \sum_{j=1}^{m} \frac{n_j E_j}{n} \tag{2.14}$$

Histogramas de precisión: La medida de evaluación *r-precision* cuando se aplica a varias consultas puede ser utilizada para comparar la calidad de recuperación de dos sistemas distintos. Siendo $RP_A\,(i)$ y $RP_B\,(i)$ los valores de la *r-precision* de los sistemas de recuperación A y B, respectivamente, para la consulta i-ésima;

se define $RP(i) = RP_A(i) - RP_B(i)$ [11]. Un valor de $RP(i)$ igual a cero significa que ambos sistemas presentan prestaciones equivalentes para la consulta i-ésima, un valor positivo indicaría que el sistema A alcanza mejores prestaciones, y un valor negativo destacaría al sistema de recuperación B como el de mejores prestaciones. Los análisis mediante diferencias entre medidas *r-precision* se suelen representar mediante histogramas de precisión que representan las diferencias de prestaciones entre los dos sistemas para todas las consultas estudiadas. En la figura 2.9 se presenta un histograma de precisiones típico donde el sistema de recuperación A presenta en general mejores prestaciones que el sistema de recuperación de información B.

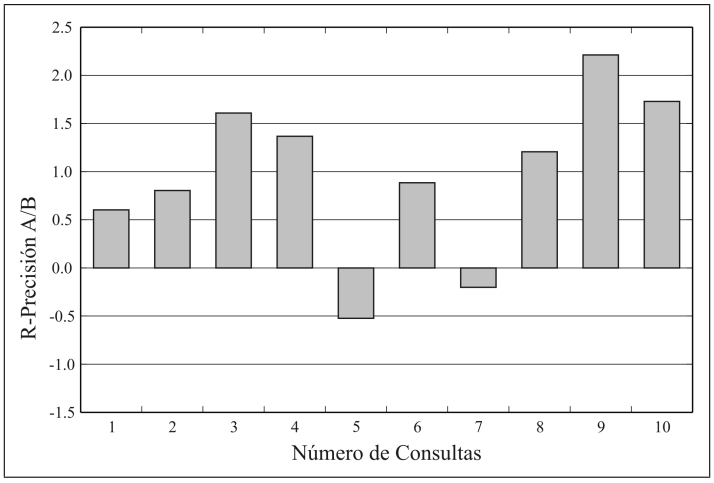

Figura 2.9: Histograma de precisión típico.

Ratios *coverage* y *novelty*: La precisión y la cobertura se basan en la asunción de que los documentos relevantes para una consulta dada son los mismos independientemente de los usuarios, pero usuarios diferentes seguramente tendrán interpretaciones distintas de cuales documentos son relevante y cuales no. Con el objetivo de tener este aspecto en cuenta surgieron dos ratios, el *coverage* y el *novelty*, que miden la cobertura relativa y el esfuerzo de cobertura [11].
Dada una colección y una consulta, se define R como el conjunto de documentos relevantes, A como el conjunto de documentos respuesta, U como el subconjunto de R que es conocido por el usuario, R_K como la intersección del conjunto A y U que serán los documentos relevantes recuperados y conocidos por el usuario, y R_U como el conjunto de documentos relevantes recuperados desconocidos previamente por el usuario. Sea $|U|$, $|R_K|$ y $|R_U|$ el número de documentos de los conjuntos U, R_K y R_U respectivamente. Basándose en todo ello se define [11]:

- *Coverage*: Fracción de documentos conocidos por el usuario como relevantes que han sido recuperados: $coverage = \frac{|R_k|}{|U|}$.

- Novelty: Fracción de documentos relevantes recuperados desconocidos por el usuario: $novelty = \frac{|R_u|}{|R_u|+|R_k|}$.
- Cobertura relativa: Relación entre el número de documentos relevantes encontrados por el sistema y el número de documentos relevantes que el usuario esperaba encontrar.
- Esfuerzo de cobertura: Relación entre el número de documentos relevantes que el usuario espera encontrar y el número de documentos examinados intentando encontrar dichos documentos relevantes esperados.

2.5.4. Colecciones de prueba para evaluación

Las colecciones de prueba consisten en colecciones de documentos, las cuales llevan asociadas información añadida que permite hacer evaluaciones. Concretamente, suelen presentar un conjunto de consultas y los documentos relevantes a las mismas (decididos por especialistas). Estas colecciones permiten estudiar el comportamiento de diversos sistemas de recuperación de información, ya que facilitan enormemente la tarea de calcular medidas de calidad de recuperación.

Aunque históricamente casi todas las colecciones de prueba ampliamente aceptadas han sido de tamaño relativamente pequeño, cada vez existen más colecciones de prueba de dimensiones mayores que caracterizan mejor a las grandes colecciones reales.

Dentro de este campo destaca la conferencia anual TREC (*Text REtrieval Conference*) [105], nacida a principios de los noventa, la cual se dedica a la experimentación con colecciones de pruebas enormes, llegando incluso a un millón de documentos. En cada conferencia TREC se designan un conjunto de experimentos de referencia y los investigadores participantes los usan para comparar sus sistemas de recuperación de información.

2.5.5. Colecciones de prueba utilizadas

En esta tesis se ha utilizado como principal método de evaluación las colecciones de prueba, concretamente se han empleado para tal fin tres colecciones de prueba distintas. Las tres están formadas por una colección de documentos, un conjunto de consultas y los documentos relevantes a las mismas.

Las tres colecciones versan sobre temáticas totalmente diferentes, una sobre noticias principalmente políticas, otra sobre un tema muy concreto, la fibrosis cística, y la última sobre resúmenes del departamento americano de energía. El tamaño de dichas colecciones también es variado, va desde una pequeña colección de 425 documentos, pasando por otra que aproximadamente los triplica, hasta una colección muy grande con más de 226000 documentos.

Colección: *Times Magazine* 1963

La colección *"Times Magazine 1963"* contiene artículos del '*Times Magazine*' de 1963, fue obtenida de ftp://ftp.cs.cornell.edu/pub/smart/time/. El idioma de la colección es el inglés. La colección incluye 425 documentos, 83 consultas de prueba con las

consiguientes referencias a los documentos relevantes a las mismas, y una lista de *stopwords*. El tema principal de la colección son las noticias mundiales, especialmente las referidas a política internacional. Incluso parecen centradas en la guerra fría.

La estructura de los ficheros es la siguiente. Todos los documentos se encuentran en un único fichero, separados uno de otro por una línea del tipo "-TEXT-043 01/18/6326". La palabra identificativa "TEXT" entre guiones, un número diferente para cada texto y tres números más separados por "/" los cuales se pueden repetir. El fichero de *stopwords* está formado por un término o símbolo en cada línea. Existe otro fichero que incluye todas las consultas una detrás de la otra, cada una empieza con una línea que incluye el texto "-TOPIC-". El fichero con la relación de documentos relevantes a cada consulta está formado por una columna para cada una de ellas, en el mismo orden que aparecen en el fichero de consultas. Cada columna está formada por los números que identifican al documento relevante correspondiendo el uno al primero que aparece en el fichero de documentos, dos al segundo y así sucesivamente, el cero no indica nada sólo está para rellenar.

Colección: *Cystic Fibrosis Database*

La colección *"Cystic Fibrosis Database"* [11, 106] está formada por 1239 documentos escritos en inglés y publicados desde 1974 a 1979 sobre aspectos de fibrosis cística, e incluye 100 consultas con sus respectivos documentos relevantes.

Existe un fichero por cada uno de los años contemplados, que incluye los documentos referentes a ese año. Todos los ficheros presentan el mismo formato, once campos, donde destacan el campo "RN" que identifica cada documento con un número entre 1 y 1239, y el campo con el resumen del texto original que puede estar identificado con la etiqueta "AB" o con la "EX".

El fichero con las consultas y sus documentos relevantes incluye cuatro campos por consulta. El primero "QN" es el número identificativo de la consulta. El segundo "QU" es el texto de la consulta. El tercero "NR" indica el número de documentos relevantes. Y el cuarto y último "RD" da información sobre los documentos relevantes para esa consulta desde cuatro evaluaciones distintas y tres grados de relevancia (2, 1, y 0) de mayor a menor relevancia. Este campo está organizado por pares de columnas, cada par tiene una primera columna con el número de documento relevante y luego otra columna con cuatro números que identifican la relevancia por cada una de las fuentes evaluadoras. La colección incluye un fichero con más información.

Colección: TREC-DOE

La colección *"TREC-DOE"*, obtenida del TREC (*"Text REtrieval Conference"*) [105], contiene resúmenes (*abstracts*) del departamento de energía de EEUU (DOE -*Department Of Energy*-). El idioma de la colección es el inglés. La colección incluye 226087 documentos y 200 consultas de prueba con la consiguientes referencias a los documentos relevantes a las mismas.

La estructura de los ficheros es la siguiente. Todos los documentos se encuentran en un único fichero. Cada documento empieza con una línea con la etiqueta "<DOC>",

seguidamente incluye otra línea donde aparece el identificativo único del documento entre las etiquetas "<DOCNO>" y "</DOCNO>", luego en una nueva línea aparece la etiqueta que identifica el comienzo del texto "<TEXT>" y a continuación aparece el contenido propiamente dicho del documento usando todas las líneas necesarias, por último aparecen en dos nuevas líneas las etiquetas de fin del texto y de fin del documento, "</TEXT>" y "</DOC>" respectivamente.

El fichero de consultas es algo complejo, ya que cada consulta tiene varios campos de información algunos de ellos optativos. Lo importante es saber que cada consulta está delimitada por las etiquetas "<top>" y "</top>" y que el texto que se ha de utilizar como consulta es el identificado por la etiqueta "<title>" seguida de la palabra "Topic:". También se podría utilizar el texto que acompaña a la etiqueta "<desc>" y a la palabra "Description:". Destaca también el campo definido por la etiqueta "<narr>" seguida de "Narrative:" que explica el tema que deben tratar los documentos para que éstos sean relevantes. Y el campo identificado por "<num>" y la palabra "Number:" es uno de los más importantes ya que indica el número de la consulta.

El fichero con la relación de documentos relevantes para las consultas dadas tiene el siguiente formato. Una primera columna con el número identificativo de la consulta a la que hace referencia, una segunda columna que no se usa, normalmente suele ser cero, una tercera columna con el identificativo de un documento evaluado, y una cuarta y última columna con el resultado de la evaluación, cero si el documento no es relevante y uno si lo es. No tienen por que estar referenciados todos los documentos, de hecho sólo aparecen los que han sido evaluados, los demás han sido considerados irrelevantes. El orden en el que aparecen los documentos no indica ningún grado de relevancia, es decir, el primero en aparecer no tiene por que ser más o menos relevante que el segundo. La relevancia sólo se indica por el número de la última columna y un documento es o no es relevante.

2.6. Conclusiones

En esta tesis al hablar de sistema de recuperación de información se hace referencia a aquellos sistemas formados por documentos textuales, sin una estructura determinada y almacenados digitalmente, y además, se presupone que la colección no varía sustancialmente en su contenido.

De todas las posibles líneas de investigación que han surgido en la búsqueda de extraer la información de interés en el momento oportuno esta tesis se centra en la recuperación de información, dejando de lado la minería de datos (*Data Mining*), la navegación (*Browsing*), la clasificación y el filtrado (*Filtering*). El principal objetivo de los sistemas de recuperación de información será proporcionar aquellos documentos que se consideran relevantes y omitir los considerados irrelevantes respecto a una consulta dada.

El trabajo desarrollado en la tesis se centra en la organización, manipulación y evaluación automática de la información, a diferencia de la vertiente manual, muy utilizada hace mucho tiempo pero hoy en día en desuso (salvo algunas excepciones). También se omite la alternativa mixta, bastante poco habitual.

En general, existe una diferencia entre lo mucho que la gente espera de los sistemas de recuperación de información y los resultados normalmente no tan buenos, la cual se debe fundamentalmente a la problemática intrínseca de los sistemas de recuperación de información y a que siempre va a ver una diferencia, mayor o menor, entre la necesidad de información de un usuario y la expresión o representación de la misma en una consulta. Normalmente las consultas presentadas suelen ser vagas y poco precisas, mientras que en cambio se espera obtener respuestas concisas y bien organizadas.

Los problemas más destacables de los sistemas de recuperación son:

- La polisemia y la sinonimia, teniendo en cuenta que se pueden dar también a expresiones y no sólo a palabras sueltas.
- Las palabras relacionadas, que son conjuntos de palabras que por separado tienen un sentido, pero que si se toman como un grupo tienen otro distinto.
- La enormidad de la información a manejar, que hace fundamental encontrar métodos con muy buenas prestaciones de almacenamiento y computación.
- La heterogeneidad de los documentos, que dificulta la caracterización de los mismos de forma que se puedan comparar unos con otros para evaluar la información que ofrecen.
- La influencia de las distintas partes de un documento, que puede obligar a ponderar la importancia de un término en función de su localización en el texto.
- La interfaz de usuario, que puede provocar que un sistema se utilice o deje de utilizar independientemente de la calidad del sistema en la recuperación de información.

De todos ellos esta tesis se centra directa o indirectamente en unos pocos de ellos, concretamente en la polisemia y la sinonimia, las palabras relacionadas (indirectamente mediante la lematización semántica) y en la enormidad de la información a manejar.

El desarrollo de un sistema de recuperación de información comprende básicamente cuatro fases distintas. La cuarta fase sería la de utilización del sistema para la obtención de información relevante a partir de consultas, en la cual la tesis no entra. Las tres primeras corresponderían al desarrollo propiamente dicho del sistema y serían:

- La fase de preprocesamiento: conlleva las acciones necesarias para transformar los documentos de la colección en una estructura de datos con la información relevante de los documentos que será la base para la modelización. Una parte importante de la tesis ha estudiado esta fase, centrándose en la reducción al máximo de los datos y estructuras a manejar maximizando la información contenida en ellos. Concretamente se ha hecho especial hincapié en la reducción de los efectos negativos de la polisemia y sinonimia en los textos mediante el uso de técnicas como el *stemming*, la lematización semántica y reducciones heurísticas.

- La fase de modelización: es la que se encarga de definir la estructura y comportamiento del sistema de recuperación de información, tanto a nivel de colección, como de consulta, sin olvidar la presentación de resultados. Esta tesis se centra en el modelo vectorial, dejando a parte otros modelos como el probabilístico y el lógico, con todas sus variantes. Incluso dentro del modelo vectorial la tesis se centra en el modelo de indexación semántica latente (LSI) y en técnicas de *clustering* vectoriales.

- La fase de evaluación: es la encargada de determinar la calidad del sistema de recuperación de información y es una de las más complejas al intervenir multitud de aspectos subjetivos sobre la calidad del sistema. La tesis tampoco hace estudios centrados en esta fase, se limita a utilizar métodos ya definidos y ampliamente usados y corroborados, todos ellos basados en el uso directo o indirecto de la precisión (*precision*) y la cobertura (*recall*). La precisión sería la relación entre el número de documentos relevantes recuperados y el número total de documentos recuperados. Mientras que la cobertura (*recall*) sería la relación entre los documentos recuperados como relevantes y los realmente relevantes.

Durante el proceso de creación de un sistema de recuperación de información, la fase del preprocesamiento es la inicial. En ella se convierten los documentos de la colección en una estructura de datos que sintetiza la información relevante de los documentos. Las cinco transformaciones fundamentales que suelen tener lugar son:

- Análisis léxico para identificar los términos. En esta tesis se ha realizado mediante una extracción automática, obteniendo una lista inicial de palabras clave que posteriormente se refinan.

- Eliminación de las palabras carentes de significado (*stopwords*). Este es el primer refinamiento que se hace, obteniendo una reducción considerable de la lista de los términos índice, con la consiguiente mejora de las prestaciones computacionales, espaciales y temporales.

- Unificación de términos según su raíz sintáctica (*stemming*) o semántica (lematización). Siguiente refinamiento que se ha estudiado en la tesis, haciendo comparativas entre los mismos en relación a las prestaciones de recuperación, y obteniendo comportamientos parecidos aunque algo mejores con el *stemming*. Ambas reducen el número de términos índice, mejorando con ello también las prestaciones computacionales.

- Reducción de los términos identificativos de la colección (unificación de formatos, reducciones heurísticas, etc.). Dentro de las diferentes opciones de reducción la tesis se ha centrado en las reducciones heurísticas. Éstas se han basado en la eliminación de los términos que a priori podrían ser significativos pero que son muy infrecuentes o demasiado frecuentes y su eliminación compensa en prestaciones computacionales minimizando la pérdida de prestaciones de recuperación.

- Utilización de tesauros. Este último refinamiento no se ha tenido en cuenta en la tesis. Se basa en una lista precompilada de palabras a las que se asocia un conjunto de palabras relacionadas generalmente derivadas de una relación de sinonimia.

El enfoque de modelización utilizado en esta tesis está basado en el paradigma vectorial, tanto los documentos como las consultas se modelizan mediante vectores m-dimensionales que contienen el peso de cada término, superando así la restricción del uso de pesos binarios y posibilitando una correspondencia parcial con un grado de similitud entre consultas y documentos, el cual permite ordenar los documentos recuperados por similitud (*ranking*). Además, al trabajar en espacios vectoriales, facilita la implementación de métodos de retroalimentación de la consulta.

Concretamente, los documentos son representados como vectores de términos, teniendo cada elemento un peso que denota la importancia del término en el documento. El conjunto de los vectores documento forman la matriz que representa la colección, una matriz dispersa denominada matriz de pesos. Las consultas se definen análogamente a los documentos, como vectores de términos. Y para determinar los documentos relacionados con una consulta se mide el grado de similitud entre el vector consulta y los vectores documentos.

Un aspecto importante del método vectorial es el adecuado pesado de los términos. La función que determina el peso de cada elemento de la matriz está formada por tres componentes distintos: el peso local (importancia relativa de un término en un documento), el peso global (importancia de un término en relación a todos los documentos de la colección) y un factor de normalización (minimiza la influencia de la longitud de los documentos). La elección del primero depende de las características de la colección, principalmente de su vocabulario. El factor global está influido por lo habitual que sean los cambios en la colección. Esta tesis no se ha centrado en el estudio de las diferentes funciones de pesado, de hecho ha utilizado uno de los pesos más ampliamente usado y adecuado al tipo y características de las colecciones de prueba utilizadas (lenguaje general y vocabulario estático, con lo cual el esquema elegido es el tfn).

Existen muchos métodos para medir la similitud entre los documentos y la consulta, aunque el más común y más utilizada suele ser el coseno del ángulo que forman el vector consulta y los vectores documento. Si el pesado incluye normalización la medida de similitud pasaría a ser el producto escalar de ambos vectores. Esta ha sido la medida de similitud elegida a lo largo de toda la tesis.

Existen tres modelos como alternativas al paradigma algebraico: el modelo vectorial generalizado, en el que se asume que los vectores índice son linealmente independientes pero no ortogonales; el modelo de indexación semántica latente (LSI -*Latent Semantic Indexing*-); y el modelo de redes neuronales.

El modelo de Indexación Semántica Latente (LSI -*Latent Semantic Indexing*-) ha sido uno de los utilizados en los estudios presentados en la tesis. Éste se basa en una recuperación de información conceptual, fundamentándose en la asunción de que existe una estructura latente en los términos utilizados, la cual se encuentra oculta por la variabilidad en la elección de términos. Ya que en la LSI las búsquedas están

fundamentadas en los conceptos y no en los términos se pueden obtener documentos que no comparten ningún término con la consulta (sinonimia), e incluso no recuperar documentos que sí que comparten términos con la consulta (polisemia). El modelo LSI trabaja con aproximaciones de bajo rango, utilizando algoritmos tales como la factorización QR, el análisis de componentes principales (PCA -*Principal Component Analysis*-) y la descomposición en valores singulares (SVD -*Singular Value Decomposition*-), esta última herramienta es la utilizada en la tesis al usar el modelo LSI.

La idea principal del modelo LSI reside en la reducción de dimensionalidad, de rango, se proyecta cada documento dentro de un espacio dimensional más pequeño, el cual de forma explícita tendrá en cuenta la relación y dependencia entre términos. El rango de la matriz aproximada k será mucho menor que el rango de la matriz de pesos mín (m, n), es decir $k \ll$ mín (m, n), pero determinar el rango óptimo k es una cuestión abierta, aunque está bastante claro que es muy dependiente de la colección. Éste ha sido uno de los principales problemas que han aparecido al trabajar con este modelo. La complejidad del modelo LSI suele conllevar que sus prestaciones computacionales, de ejecución y respuesta, estén muy por debajo de los otros modelos vectoriales. Y este aspecto es el que ha llevado a estudiar otros modelos, concretamente el de *clustering*.

Aunque el modelo de *clustering* no está únicamente relacionado con el paradigma vectorial, muchos de los métodos de *clustering* (y todos los desarrollados en el marco de esta tesis) trabajan partiendo de una representación vectorial y utilizando técnicas de cálculo de distancias iguales o similares a las utilizadas en el modelo vectorial. Realmente las técnicas de *clustering* trabajan directamente con la colección de documentos y no con el texto de los documentos, por lo que no pertenecen a ninguno de los paradigmas de recuperación de información (lógico, vectorial o probabilístico).

Los métodos de *clustering*, básicamente, tratan de separar una colección de documentos en dos conjuntos disjuntos distintos apoyándose en una vaga descripción de uno de los conjuntos (consulta): el primero de los conjuntos estará formado por todos los objetos relacionados con dicha descripción vaga y el otro por todos los objetos que no están relacionados. Normalmente, parten inicialmente la colección en diferentes grupos (los *clusters*), donde los miembros de una agrupación son documentos más similares entre sí que a los de las demás agrupaciones. La hipótesis de base es que los documentos relevantes tienden a estar más cerca los unos de los otros que los documentos no relacionados.

El *clustering* puede encontrarse en muy diferentes contextos a parte de la recuperación de información: generación de hipótesis, prueba de hipótesis, reducción de información, predicción basada en grupos, negocios, análisis de datos espaciales, minería en la web, etc.; aunque también puede aparecer referenciado de distintas formas tales como aprendizaje no supervisado en reconocimiento de patrones, taxonomías numéricas en biología o ecología, tipologías en ciencias sociales, o particiones en teoría de grafos.

Toda esta multitud de métodos de *clustering* disponibles en la literatura puede ser clasificada fundamentalmente en los siguientes tipos: *clustering* particional, *clustering* jerárquico, *clustering* basado en densidades, *clustering* basado en *grid*, *clustering* basado en modelos y *clustering* de información categórica. En esta tesis se han utilizado dos tipos fundamentalmente: *clustering* jerárquico y *clustering* basado en densidades.

En este trabajo la consulta del usuario no se ha analizado, se han utilizado aquellas consultas que las colecciones de prueba aportaban. Las consultas se han traducido al modelo vectorial elegido y luego han sido utilizadas para la evaluación. Posibles mejoras de los resultados mediante la retroalimentación de la consulta no se han trabajado. La retroalimentación es un método iterativo de mejora de la consulta basándose en los documentos que se suponen a priori relevantes a dicha consulta. El procedimiento a seguir consistirá en reformular la consulta de tal forma que se acerque más a los términos de los documentos relevantes (se da más peso a estos términos) y si es posible se aleje más de los términos de los documentos no relevantes (se reduce el peso de estos términos). Con esta forma de actuar se pueden mejorar las prestaciones de recuperación en términos de precisión y cobertura (*recall*).

A la hora de valorar la calidad de un sistema de recuperación o algoritmo se plantea la difícil y delicada tarea de balancear tres aspectos fundamentales: velocidad, precisión y cobertura (*recall*); teniendo en cuenta que llegar a mejorar los tres factores es imposible, hay que llegar a un compromiso entre ellos. En esta tesis se ha dado más importancia a la precisión y la cobertura (*recall*) que a la velocidad de respuesta, aunque siempre manteniendo unos tiempos aceptables. De hecho no se ha compensado mejora de velocidad con calidad de respuesta, salvo en la fase de preprocesamiento cuando se ha utilizado un método heurístico para eliminar términos muy poco frecuentes, y aún así la perdida en recuperación ha sido prácticamente nula y la mejora de prestaciones computacionales enorme.

Aunque es importante no olvidar que la relevancia de un documento es una cuestión subjetiva, de hecho dos sistemas de recuperación ante la misma consulta pueden dar distinta relevancia al mismo documento, la evaluación de la calidad de las respuestas de un sistema de recuperación de información se suele basar en la medición directa o indirectamente de la precisión (los documentos devueltos son sólo los relevantes) y la cobertura (todos los documentos relevantes son devueltos) de dicho sistema en relación a una colección de prueba (conjunto de documentos y consultas de las cuales se conocen los documentos relevantes).

Las colecciones de prueba consisten en colecciones de documentos, las cuales llevan asociadas información añadida que permite hacer evaluaciones. Concretamente, suelen presentar un conjunto de consultas y los documentos relevantes a las mismas (decididos por especialistas). Estas colecciones permiten estudiar el comportamiento de diversos sistemas de recuperación de información, ya que facilitan enormemente la tarea de calcular medidas de calidad de recuperación. Este ha sido la principal herramienta de evaluación de la calidad de recuperación de los métodos desarrollados en la tesis, concretamente se han utilizado tres colecciones de prueba diferentes, de temas y tamaños distintos.

Un gráfico de precisión versus cobertura representa p puntos de precisión generados sobre la cobertura de forma uniformemente espaciada tomando valores entre 0 y 1, formando siempre una curva no incremental. Normalmente se basa en once niveles (11 puntos) de cobertura estándar (0 %, 10 %, 20 %,..., 100 %), los cuales son obtenidos por interpolación. Su utilidad radica en que permiten evaluar cuantitativamente tanto la calidad del conjunto de respuestas como la calidad de la cobertura del sistema de recuperación de información, permitiendo así comparar diferentes sistemas. Además,

son intuitivas y se pueden combinar en una única curva simple. Estos gráficos han sido los elegidos para representar los estudios de calidad de recuperación y comparar los diferentes algoritmos desarrollados. De todas formas, a veces, también es interesante utilizar otras medidas que combinen la precisión y la cobertura en una única medida simple, ya que éstas están relacionadas y capturan aspectos diferentes a tener en cuenta: *r-precision*, media harmónica, medida E (*E measure*), entropía, histogramas de precisión y ratios *coverage* y *novelty.*

Capítulo 3

Algoritmos de *clustering*

3.1. Tipos de métodos de *clustering*

El *clustering* es ampliamente reconocido como una herramienta útil en muchas aplicaciones de diferentes disciplinas, sin embargo, es difícil al combinar conceptos de diversos campos científicos (por ejemplo, bases de datos, reconocimiento de patrones, estadística, etc.) [5]. Lo cual ha llevado al desarrollo de una gran variedad de métodos de *clustering.* Aunque existen otras clasificaciones más modernas, por ejemplo en [107], la clasificación más común de los métodos utilizados en recuperación de información divide los métodos de *clustering* en particionales y jerárquicos [3, 4, 5, 6, 40]:

Particional: Construye particiones, donde cada *cluster* optimiza un criterio de *clustering*. Se caracterizan por su alta complejidad, algunos incluso enumeran exhaustivamente todas las posibles particiones buscando el óptimo global. Normalmente se empieza con una solución inicial aleatoria y luego se refina.

Jerarquico: Crea una descomposición jerárquica. Se subdividen a su vez en aglomerativos y divisivos. Los aglomerativos parten siendo cada documento un *cluster* independiente y sucesivamente los mezclan de acuerdo a una medida de distancia. Los divisivos parten de un único *cluster* formado por toda la colección y sucesivamente la divide en grupos más pequeños según algún criterio, hasta que cada documento forma un único *cluster*.

Un método jerárquico se puede utilizar como un método particional y el proceso inverso también es posible, de una técnica particional se pueden obtener resultados jerárquicos [38, 41]. En el primer caso basta con aplanar la jerarquía creada en el nivel adecuado para generar el número de clusters deseados, y en el segundo caso basta con mantener los resultados de diferentes repeticiones de la técnica particional construyendo así una jerarquía (sería algo parecido al método divisivo jerárquico).

A parte de estas dos principales categorías de algoritmos de clustering han surgido otros métodos, los cuales suelen estar enfocados a problemas o conjuntos de datos específicos [3, 5, 6]:

***Clustering* basado en densidades:** Los elementos vecinos se agrupan en un mismo conjunto basándose en funciones objetivo específicas de densidad (número de objetos en un vecindario particular).

***Clustering* basado en *grid*:** Están pensados principalmente para datos espaciales (por ejemplo, los que modelan estructuras geométricas). El procedimiento que siguen es dividir el espacio en un número finito de celdas y entonces trabajar con dichas celdas y su contenido.

***Clustering* basado en modelos:** Se utilizan para encontrar buenas aproximaciones de los parámetros del modelo que mejor define los datos. Están bastante cerca de los algoritmos basados en densidades aunque no tienen por que usar el mismo concepto de densidad.

***Clustering* de información categórica:** Están especialmente desarrollados para datos categóricos donde la medida de distancia euclídea u otras numéricas no pueden ser aplicadas.

Aunque un método de *clustering* específico puede trabajar bien en un conjunto de datos concreto, en cambio en otro puede obtener resultados muy pobres, dependiendo sobre todo de la talla y la dimensionalidad de los datos, así como de la función objetivo y de las estructuras usadas [6]. En lo que los investigadores están más o menos de acuerdo, independientemente del método elegido, es que una buena técnica de *clustering* debe verificar en un alto grado las siguientes características:

Escalabilidad: Debe funcionar bien tanto con pocos como con muchos elementos.

***Clusters* arbitrarios:** Debe ser capaz de encontrar los *clusters* independientemente de su forma. Muchos algoritmos de *clustering* sólo son capaces de encontrar *clusters* con una determinada forma, la más común es la esférica.

Parámetros de entrada mínimos: Debe exigir el mínimo de información sobre el sistema, ya que estos datos pueden ser desconocidos e influir en el resultado, por ejemplo el número de *clusters*.

Manejo del ruido: Debe ser capaz de manejar desviaciones de la normalidad en los datos, por ejemplo elementos aislados que no formarían parte de ningún *cluster*.

Insensibilidad al orden de entrada: No debe ser influido por el orden de reconocimiento de los datos. Muchos algoritmos, generalmente los que están dirigidos a soluciones locales, dan lugar a resultados distintos cuando los datos se analizan en orden diferente.

Dimensionalidad alta: Debe ser capaz de trabajar con alta dimensionalidad sin empobrecer la calidad del resultado ni deteriorar las prestaciones temporales de forma desproporcionada. Cuantos más atributos o características de los datos se tienen en cuenta, mayor es la dimensionalidad.

Interpretabilidad y usabilidad: Debe ofrecer los resultados de forma fácilmente interpretable y usable posteriormente.

3.1.1. *Clustering* particional

Los algoritmos de clustering particionales obtienen directamente tantas agrupaciones como se deseen sin construir una jerarquía y siguiendo los pasos tipo descritos en el algoritmo 1 [24, 38]:

Algoritmo 1 *Clustering particional*

Entrada: Colección de documentos y número de *clusters* a construir

Salida: *Clusters*

Paso-1: Formar tantos *clusters* como se quieran construir con semillas normalmente elegidas aleatoriamente.

Paso-2: Calcular un representante de cada *cluster* (por ejemplo el centroide que es la media de todos los elementos del *cluster*).

Paso-3: Asignar cada documento al *cluster* más cercano en función de una medida de distancia elegida (por ejemplo asignarlo al *cluster* cuyo centroide sea más similar al documento).

Paso-4: Si no ha habido movimiento de documentos entre *clusters* finalizar, si no refinar los *clusters* volviendo al paso-2.

Una característica importante de las técnicas particionales, es que cuando se trabaja con grandes conjuntos de datos los métodos jerárquicos conllevan un coste computacional prohibitivo, mientras que los particionales se pueden seguir utilizando [3]. En cambio, la principal desventaja que presentan los métodos particionales es que precisan como parámetro de entrada el número de *clusters* que se van a construir [3]. También, debido a su naturaleza de funcionamiento, las técnicas particionales presentan la desventaja de encontrar soluciones locales, ya que la búsqueda de soluciones globales conlleva demasiados costes computacionales. Por ello, lo que se suele hacer es ejecutar varias veces el algoritmo particional elegido con diferentes estados de partida (distintas semillas generadoras) y entonces seleccionar la mejor de todas las soluciones locales obtenidas [3].

En la tabla 3.1 en la página siguiente se muestran algunos de los algoritmos de *clustering* particionales más importantes [5, 6]. Se destacan características como los parámetros de entrada que precisan, para qué están optimizados, qué tipo de estructura de *cluster* detectan y qué complejidad computacional presentan.

Algoritmo	**Parámetros de entrada**	**Optimizado para**	**Estructura del *cluster***	**Complejidad**
K-MEANS	Número de *clusters*	*Clusters* separados	Esférico	$\mathcal{O}(Ikn)$
PAM	Número de *clusters*	*Clusters* separados Conjuntos de datos pequeños	Esférico	$\mathcal{O}\left(Ik\,(n-k)^2\right)$
CLARA	Número de *clusters*	Conjuntos de datos relativamente grandes	Esférico	$\mathcal{O}\left(ks^2+k\,(n-k)\right)$
CLARANS	Número de *clusters* Máximo número de vecinos	Conjuntos de datos espaciales	Esférico	$\mathcal{O}\left(kn^2\right)$
n = número de elementos ; k = número de *clusters* ; s = talla de la muestra ; I = iteraciones				

Tabla 3.1: Algoritmos de *clustering* particionales importantes.

3.1.2. *Clustering* jerárquico

Las técnicas jerárquicas dan lugar a una secuencia anidada de particiones con un único *cluster* en la zona más alta que engloba a todos los elementos y *clusters* unitarios formados por los elementos individuales en la parte más baja. Toda esta secuencia jerárquica de particiones se puede representar gráficamente como un árbol que provee distintas vistas en diferentes niveles de abstracción, el cual recibe el nombre de dendograma [3, 38, 40, 41] y se presenta en la figura 3.1.

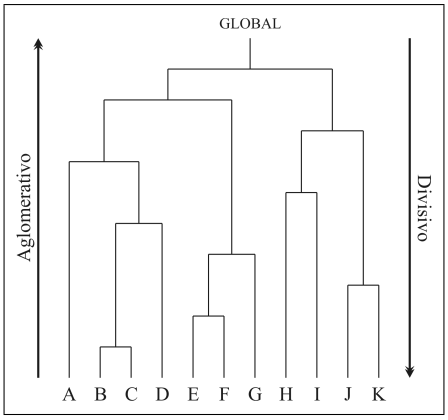

Figura 3.1: Árbol jerárquico o dendograma.

Cada nivel intermedio puede ser interpretado desde dos puntos de vista, como la agrupación de dos *clusters* de un nivel más bajo o como la división de un *cluster* de un nivel más alto, ambas opciones dan lugar a los dos procedimientos básicos dentro de los métodos jerárquicos: aglomerativo (se precisa una función de similitud o distancia entre *clusters*) y divisivo (se precisa un criterio de elección del *cluster* a dividir y el método para realizar tal división) [3, 5, 38, 40, 41, 43, 44].

La selección del *cluster* a dividir en un procedimiento jerárquico divisivo es un aspecto muy importante ya que tiene un gran impacto en los resultados globales de la técnica de *clustering*. Existen diversos métodos para realizar dicha elección, entre ellos destacan los siguientes [24, 41, 43]:

- Seleccionar el *cluster* más grande de todos los disponibles, suele dar soluciones razonablemente buenas y balanceadas, aunque su principal limitación es que no funciona bien cuando los *cluster* naturales son de distintos tamaños.
- Seleccionar todos los *clusters*, realmente no existe elección al dividirlos todos obteniendo un árbol binario completo ignorando la calidad de los *clusters*.
- Seleccionar el *cluster* con mayor varianza respecto a su centroide, basándose en la útil propiedad del esparcimiento de un *cluster*, lo que le lleva a ser el criterio comúnmente más usado.

En la tabla 3.2 se muestran algunos de los algoritmos de *clustering* jerárquicos más importantes [5, 6, 108, 109, 110]. Se destacan características como los parámetros de entrada que precisan, para qué están optimizados, qué tipo de estructura de *cluster* detectan y qué complejidad computacional presentan.

Algoritmo	**Parámetros de entrada**	**Optimizado para**	**Estructura del *cluster***	**Complejidad**
BIRCH	Factor de ramificación Umbral del diámetro	Conjuntos de datos grandes	Esférica	$\mathcal{O}(n)$
CURE	Número de *clusters* Número de *clusters* representativos	Conjuntos de datos relativamente grandes	Arbitraria	$\mathcal{O}(n^2 \log n)$
n = número de elementos				

Tabla 3.2: Algoritmos de *clustering* jerárquicos importantes.

3.1.3. *Clustering* basado en densidades

La idea subyacente en los método de *clustering* basados en densidades es agrupar aquellos elementos que son vecinos, determinando la cualidad de vecindad utilizando

una función objetivo que trabaje específicamente sobre densidades [5, 6]. Concretamente la función de densidad establece valores muy distintos entre los elementos de un *cluster* y los de los demás *clusters*, así como de los elementos considerados ruido (no pertenecen a ningún *cluster*) y los datos sueltos (tampoco pertenecen a ningún *cluster*) [40, 66]. En otras palabras, la idea clave es que los objetos densos se agrupan juntos dentro de un mismo *cluster*, entendiendo que un *cluster* es una región que presenta una densidad de objetos más alta que sus regiones vecinas [111].

En la tabla 3.3 se muestran algunos de los algoritmos de *clustering* basados en densidades más importantes [5, 6, 40, 58]. Se destacan características como los parámetros de entrada que precisan, para qué están optimizados, qué tipo de estructura de *cluster* detectan y qué complejidad computacional presentan.

Algoritmo	**Parámetros de entrada**	**Optimizado para**	**Estructura del *cluster***	**Complejidad**
DBSCAN	Radio del *cluster* Mínimo número de elementos en el *cluster*	Formas de *cluster* arbitrarias Conjuntos de datos grandes	Arbitraria	$\mathcal{O}(n \log n)$
DENCLUE	Radio del *cluster* Mínimo número de elementos	Formas de *cluster* arbitrarias Conjuntos de datos grandes	Arbitraria	$\mathcal{O}(n \log n)$
n = número de elementos				

Tabla 3.3: Algoritmos de *clustering* basados en densidades importantes.

Las principales ventajas de las técnicas basadas en densidades son [9, 62, 66, 112]:

1. Los parámetros iniciales son pocos y lo más importante, no hay que indicar cuántos *clusters* generar. Concretamente se suele definir inicialmente el radio de los *clusters* (en sentido amplio es el criterio de densidad y define que elementos pertenecen a un *cluster*) y el mínimo número de objetos que debe contener.

2. La estructura de los *clusters* no es importante ya que son detectados independientemente de su forma. Se encuentran clusters de tallas y formas arbitrarias.

3. La complejidad de estos métodos suele ser más que aceptable.

4. Pueden ser usados para todo tipo de espacios métricos y no están confinados a los espacios vectoriales.

5. Son robustos en relación a los datos sueltos (*outliers*) y filtran el ruido.

6. Han probado ser muy eficientes y efectivos agrupando toda clase de datos.

7. Las versiones paralelas y distribuidas amplían la eficiencia.

3.2. Métodos básicos

3.2.1. *K-Means*

El algoritmo *K-Means* es la técnica iterativa particional más ampliamente usada [43, 44], y el algoritmo 2 presenta el *K-Means* típico [3, 6, 38, 45]:

Algoritmo 2 *K-Means típico*

Entrada: Colección de documentos y número de *clusters* a construir (k)

Salida: *Clusters*

Paso-1: Elegir k centros de *clusters* generalmente de forma aleatoria.

Paso-2: Asignar cada documento al *cluster* de centro más cercano.

Paso-3: Recalcular los centros de los *clusters* a partir de los elementos del *cluster*.

Paso-4: Si se cumple el criterio de convergencia (normalmente que no haya movimiento de documentos entre *clusters* o que éste sea mínimo) entonces finalizar, si no refinar los *clusters* volviendo al paso-2.

Las principales razones detrás de la popularidad del algoritmo *K-Means* son [3]:

- Su complejidad temporal es $\mathcal{O}(Ikn)$ donde n es el número de documentos, k es el número de *clusters* e I es el número de iteraciones necesarias para converger. Típicamente, k e I son fijas y se establecen con anterioridad, con lo cual el algoritmo tendría una complejidad temporal lineal con la talla de la colección (en [21, 23] se analiza también asumiendo que la matriz de pesos es dispersa).
- Su complejidad espacial es $\mathcal{O}(k+n)$ más el espacio necesario para la matriz que modeliza la colección (en [21, 23] se analiza también asumiendo que la matriz de pesos es dispersa).
- No influye el orden en el que se analizan los documentos para una semilla dada, aunque sí influye la semilla inicial (el *K-Means* actúa de forma determinista respecto a los parámetros de entrada obteniendo una solución óptima local [46]).
- Aprovecha la dispersión típica de las matrices de pesos del modelo vectorial.

El cuello de botella del algoritmo *K-Means* es la computación de la medida de distancia entre los documentos y el centro de los *clusters* (paso-2 del algoritmo 2), pero como sólo es en las primeras iteraciones cuando hay mucho movimiento de documentos entre *clusters*, se puede mejorar el algoritmo evitando calcular distancias utilizando distancias estimadas y comparándolas con distancias límites [21].

Una mejora típica es intentar alcanzar el mínimo global, en lugar de uno local intentando para ello mejorar la elección de los centros iniciales [3]. Otra mejora importante es permitir separar y juntar *clusters* en función de dos umbrales, separando

cuando la varianza del *cluster* supera el umbral determinado y juntando cuando los centros de dos *clusters* superan el umbral de cercanía [3]. Otra posible mejora es actualizar los centros de los *clusters* incrementalmente, es decir a medida que los documentos se asignan a los *clusters* [38].

La definición de centroide o vector concepto el cual determina el centro de un *cluster* es muy importante (algunos autores utilizan vector concepto para definir el centroide normalizado). Si se denota por x_i el documento i-ésimo del *cluster* j-ésimo (π_j), y n_j denota el número de documentos de π_j, entonces el centroide o vector concepto c_j del *cluster* π_j se define como [23]:

$$c_j = \frac{\frac{1}{n_j} \sum_{x_i \in \pi_j} x_i}{\left\| \frac{1}{n_j} \sum_{x_i \in \pi_j} x_i \right\|} \tag{3.1}$$

Definiendo c_j como el centroide del *cluster* π_j, k como el número de *clusters* y x un documento del cluster π_j; entonces la función objetivo del algoritmo *K-Means* que intenta minimizar la distancia de cada documento con el centroide del *cluster* al que pertenece (normalmente se utiliza la distancia eEuclídea o el coseno del ángulo que forman -expresión 3.3-), se puede formular como [5, 21, 46]:

$$f\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k} \sum_{x \in \pi_j} d\left(x, c_j\right) \tag{3.2}$$

$$f\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k} \sum_{x \in \pi_j} x^{\mathrm{T}} \bullet c_j \tag{3.3}$$

El algoritmo *K-Means* es un proceso iterativo que suele tender a la convergencia de la función objetivo, por lo tanto es interesante disponer de un criterio de parada, así como contemplar un número máximo de iteraciones. Siendo ϵ un valor muy pequeño, el criterio de convergencia normalmente utilizado es [21, 23]:

$$\left| f\left(\{\pi_j^t\}_{j=1}^{k}\right) - f\left(\{\pi_j^{t+1}\}_{j=1}^{k}\right) \right| \leq \epsilon \cdot \left| f\left(\{\pi_j^{t+1}\}_{j=1}^{k}\right) \right| \tag{3.4}$$

La variante más importante del algoritmo *K-Means* es la que utiliza como medida de distancia el coseno del ángulo que forman los vectores representantes de los documentos y los centroides de los *clusters*. Cuando se hace uso del coseno como medida de cercanía los *clusters* resultantes toman una forma geométrica de esfera m dimensional. De ahí que esta variante reciba el nombre de *Spherical K-Means*:

Algoritmo 3 *Spherical K-Means*

Entrada: $M \in \mathbb{R}^{m \times n}$, ϵ, *maxiter*, $k \in \mathbb{N}$.

Salida: k *clusters* disjuntos $\{\pi_j\}_{j=1}^{k}$.

Paso-1: Construir arbitrariamente k *clusters* iniciales $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$, calcular sus centroides normalizados $\left\{c_j^{(0)}\right\}_{j=1}^{k}$ e inicializar t=0.

Paso-2: Construir una nueva partición $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ inducida por $\left\{c_j^{(t)}\right\}_{j=1}^{k}$ según:

$$\pi_j^{(t+1)} = \left\{x \in M :\ x^{\mathrm{T}} \cdot c_j^{(t)} > x^{\mathrm{T}} \cdot c_l^{(t)}\ ,\ 1 \leq l \leq k\ ,\ j \neq l\right\},\ 1 \leq j \leq k$$

Paso-3: Calcular los vectores concepto $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$ de la nueva partición.

Paso-4: Finalizar si el criterio de parada (expresión 3.4 en la página anterior) se cumple o si t es igual a *maxiter,* sino incrementar t y volver al paso-2.

3.2.2. DBSCAN

El método DBSCAN [40] presenta dos parámetros de entrada ϵ, $MinEle$; el primero define la distancia máxima entre dos elementos para considerarlos vecinos, y el segundo define el mínimo número de elementos que deben ser vecinos para formar un cluster [40]. La salida de este algoritmo son *clusters* disjuntos, el número de éstos no es un parámetro, viene determinado por los datos de entrada; también puede dar lugar a un conjunto de elementos definidos como ruido, ya que no se pueden incluir en ninguno de los *clusters* construidos.

El *DBSCAN* (algoritmo 4 en la página siguiente) comprueba qué elementos están a una distancia máxima ϵ de cada uno de los elementos del conjunto de datos de entrada (es decir, qué documentos están a una distancia ϵ de cada uno de los documentos de la colección). Y también determina si en dicha área de "radio" ϵ existen al menos $MinEle$ elementos (es decir, si en dicha área existen al menos $MinEle$ documentos). Si un elemento presenta esos mínimos elementos dentro de una distancia ϵ se le considera un elemento núcleo, el cual será el corazón de un *cluster*, y a partir de este tipo de elementos crecerán los *clusters* añadiendo aquellos otros que sean directamente alcanzables por densidad desde el punto corazón (a partir de un documento corazón se podrán alcanzar al menos $MinEle$ documentos en un área de "radio" ϵ los cuales formarán un *cluster* junto con todos los documentos que estén a una distancia ϵ de cualquier elemento que ya forme parte del *cluster*). Si dos puntos corazón están a una distancia ϵ el uno del otro entonces sus clusters se unen en uno sólo. Los elementos incluidos en un *cluster* que en un área de "radio" ϵ no incluyen al menos

$MinEle$ elementos se consideran elementos borde. El algoritmo termina si no existen más elementos (documentos de la colección) que puedan ser asignados a un *cluster*. Aquellos elementos que no pueden ser asignado a ningún *cluster* durante el proceso son tratados como ruido (los documentos que al finalizar el algoritmo no han sido incluidos en ningún *cluster* se consideran ruido dentro de la colección). [40, 112, 113]

Algoritmo 4 *DBSCAN*

Entrada: $M \in \mathbb{R}^{m \times n}$, $\epsilon \in \mathbb{R}$, $MinEle \in \mathbb{N}$.

Salida: k *clusters* disjuntos $\{\pi_j\}_{j=1}^{k}$, R (Ruido)

Paso-1: $j = 1$

Paso-2: Para todo documento $d \in M$ todavía por clasificar ($d \notin \{\pi_j\}_{j=1}^{k}$ y $d \notin R$):

Paso-2.1: $S = EpsVecindario(d, M, \epsilon)$ (elementos de M que están a una distancia ϵ de d).

Paso-2.2: Si $|S| < MinEle$ Entonces $R = R \cup d$

Paso-2.3: Sino ...

Paso-2.3.1: $\pi_j = \pi_j \cup S$

Paso-2.3.2: $S = S - d$

Paso-2.3.3: Mientras existan elementos en S

Paso-2.3.3.1: Elegir $x \in S$

Paso-2.3.3.2: $Z = EpsVecindario(x, M, \epsilon)$ (elementos de M que están a una distancia ϵ de x).

Paso-2.3.3.3: Si $|Z| \geq MinEle$ Entonces

Paso-2.3.3.3.1: $Z_{NC} = \left\{ z \in Z \, : \, z \notin \{\pi_l\}_{l=1}^{j} \right\}$

Paso-2.3.3.3.2: $Z_R = \{z \in Z \, : \, z \in R\}$

Paso-2.3.3.3.3: $S = S \cup Z_{NC}$

Paso-2.3.3.3.4: $\pi_j = \pi_j \cup Z_{NC} \cup Z_R$

Paso-2.3.3.4: $S = S - x$

Paso-2.3.4: $j = j + 1$

Destacan las siguientes características del DBSCAN [6, 40, 62, 66, 111, 112, 113]:

- Fue diseñado para descubrir eficientemente los *clusters* y el ruido de bases de datos.
- Funciona igual de bien para espacios euclídeos de 2 ó 3 dimensiones como para espacios de dimensionalidad alta.

- Busca regiones de densidad alta que están separadas por regiones de densidad baja.
- Encuentra *clusters* con talla y forma arbitraria, pero influida por la función utilizada para el cálculo de la distancia entre dos objetos.
- No es sensitivo al orden de entrada de los objetos.
- No es determinístico en un sentido estricto (la ejecución depende del orden de los objetos), pero tanto el tiempo de ejecución como el resultado (los *clusters* formados) son siempre los mismos.
- Es incremental, ya que los nuevos objetos insertados sólo afectan a un cierto vecindario.
- Es un algoritmo de *clustering* muy eficiente y efectivo, entre otros motivos porque sólo precisa de una evaluación a través de la base de datos.
- Es robusto respecto al ruido.
- Trabaja en cualquier espacio métrico, no sólo en el espacio vectorial.
- Puede ser fácilmente implementado.
- Requiere dos parámetros de entrada: el radio de los *clusters* y el número mínimo de objetos que deben formarlos. No requiere un número predeterminado de *clusters*.
- Presenta una complejidad computacional $\mathcal{O}\left(n^2\right)$, la cual se puede mejorar utilizando índices espaciales a $\mathcal{O}\left(n \log n\right)$.
- No permite trabajar con densidades variables, por lo cual no puede generar *clusters* de distintas densidades.

El algoritmo DBSCAN a dado lugar a diferentes variantes y mejoras destacando las siguientes:

GDBSCAN: Versión generalizada del DBSCAN que permite trabajar con datos numéricos y con atributos categóricos [57].

DBCLASD: Variante del DBSCAN que permite trabajar sin ningún parámetro de entrada, al asumir que los objetos dentro de un *cluster* son distribuidos uniformemente [58, 59]. Estudios muestran que el DBSCAN es más rápido (tendiendo al doble) que el DBCLASD [59].

DBSCAN Incremental: Se ha desarrollado una versión incremental eficiente [60].

DBSCAN Paralelo: Existe una versión paralela del DBSCAN la cual empieza con el conjunto de datos residente en un servidor central y entonces distribuye la información a los diferentes clientes [61]. También hay desarrollada una versión distribuida, que parte de la colección ya distribuida, entonces hace un *clustering* local y envía información del modelo local a un servidor central, donde se genera un modelo global a partir de los locales, el cual es distribuido para actualizar los modelos locales [62, 63].

OPTICS: Extensión del DBSCAN que relaja el requerimiento estricto de los parámetros de entrada, computando una descomposición jerárquica de *clusters* basada en un rango de configuración de los parámetros [65, 66]. El DBSCAN y el OPTICS son similares en estructura y presentan la misma complejidad computacional [6]. Este método aunque tiene el proposito del análisis de *clusters* no construye explícitamente los *clusters*, en su lugar crea un orden que representa la estructura de *clustering* basada en densidades [65].

Otras mejoras más recientes están sobre todo enfocadas a la reducción del tiempo de respuesta del DBSCAN. Entre ellas se puede destacar la utilización de diversos métodos para trabajar con muestras, como se presenta como por ejemplo en [70, 71, 78]. Este enfoque se pretende estudiar en el futuro para reducir también la necesidad de comunicación en la versión paralela (ver apartado 6 en la página 159). También destaca la reducción de comprobaciones sobre el vecindario de los elementos [72, 73].

También es interesante observar como en la literatura cada vez aparecen más métodos para diferentes campos que se basan en el DBSCAN principalmente [67, 68, 74, 76, 79, 80, 81, 82, 83, 84] (y en ciertos casos de alguna de sus variantes, sobre todo del OPTICS [74, 75, 85]).

3.3. Variante del *K-Means:* α*-Bisecting Spherical K-Means*

3.3.1. α*-Bisection*

El método de bisección consiste en recursivamente dividir un *cluster* en dos *subclusters* partiendo de un conjunto inicial formado por todos los elementos, este método es uno de los más comunes y básicos en campos como la recuperación de documentos, la minería de datos, el análisis de patrones, la segmentación de imágenes, la toma de decisiones, etc. [43, 44]. Al realizar diversas divisiones recursivas se pueden obtener tantos *clusters* como se deseen, obteniendo de este modo un árbol binario jerárquico (o taxonomía binaria), lo que lo hace tan atractivo para muchas aplicaciones [43, 44].

Aunque el método más común de bisección utiliza como criterio de reparto de documentos la distancia de los elementos a los centroides de los *subclusters* (ver algoritmo 5 en la página siguiente [43, 44]), se presenta también el α*-bisection* [114], una versión basada en la creación de los *subclusters* a partir de los elementos que se encuentran dentro de un radio α (ver algoritmo 6 en la página siguiente).

Algoritmo 5 *Bisección*

Entrada: $M \in \mathbb{R}^{m \times n}$, *maxiter.*

Salida: $\pi_l \in \mathbb{R}^{m \times nl}$ y $\pi_r \in \mathbb{R}^{m \times nr}$ donde π_l es el *cluster* izquierdo, π_r es el *cluster* derecho con $nl + nr = n$.

Paso-1: Calcular dos vectores concepto normalizados, $c_l \in \Re^m$y $c_r \in \Re^m$, de dos conjuntos arbitrarios de documentos.

Paso-2: Dividir M en dos *subclusters* π_l y π_r de acuerdo a:

$$d \in \pi_l \text{ si } \|d - c_l\| \leq \|d - c_r\|$$
$$d \in \pi_r \text{ si } \|d - c_l\| > \|d - c_r\|$$

Paso-3: Calcular los nuevos vectores concepto normalizados de π_l y π_r, c'_l *y* c'_r respectivamente, definidos como:

$$t = \frac{1}{nl} \sum_{d \in \pi_l} d \; ; \; c'_l = \frac{t}{\|t\|} \qquad t = \frac{1}{nr} \sum_{d \in \pi_r} d \; ; \; c'_r = \frac{t}{\|t\|}$$

Paso-4: Si $c'_l == c_l$ y $c'_r == c_r$ o *maxiter* $== 0$ entonces finalizar, sino decrementar *maxiter* en una unidad, hacer $c_l = c'_l$ y $c_r = c'_r$ e ir al paso-2.

Algoritmo 6 α*-Bisection*

Entrada: $M \in \mathbb{R}^{m \times n}$, α, ϵ, *maxiter.*

Salida: $\pi_l \in \mathbb{R}^{m \times nl}$ y $\pi_r \in \mathbb{R}^{m \times nr}$ donde π_l es el *cluster* izquierdo, π_r es el *cluster* derecho con $nl + nr = n$.

Paso-1: Calcular α usando la expresión 3.5 en la página siguiente. Y el vector concepto normalizado, $c \in \mathbb{R}^m$, de un conjunto de documentos seleccionado.

Paso-2: Dividir M en dos *subclusters* π_l y π_r de acuerdo a:

$$d \in \pi_l \text{ si } d^{\mathrm{T}} \bullet c \geq \alpha$$
$$d \in \pi_r \text{ si } d^{\mathrm{T}} \bullet c < \alpha$$

Paso-3: Calcular el nuevo vector concepto normalizado de π_l, c', definido como:

$$t = \frac{1}{nl} \sum_{m \in \pi_l} m \; ; \; c' = \frac{t}{\|t\|}$$

Paso-4: Si el criterio de parada ($\|c' - c\| \leq \epsilon$) ha sido alcanzado o *maxiter* $== 0$ entonces finalizar, sino decrementar *maxiter*, hacer $c = c'$ e ir al paso-2.

Una colección puede ser vista como una gran esfera, cuyo volumen está determinado por su radio que aumenta en relación a la dimensión de los documentos (número de términos). Pero se pretende crear k *clusters* y por tanto dividir la colección en k subesferas de volumen aproximadamente igual. Entonces el valor α representa el radio de cada una de esas k subesferas. Aunque para determinar el valor α se han considerado estos aspectos teóricos también se ha refinado experimentalmente la fórmula que representa el valor α [114]:

$$\alpha = 1 - \frac{\overline{x} - \sigma}{\sqrt[m]{k}} \tag{3.5}$$

En la expresión 3.5, $\overline{x}$ define la media de las distancias de coseno al centroide (relacionando las ecuaciones 2.2 en la página 46 y 3.1 en la página 78), σ la desviación estándar entre cada documento de la colección y el centroide de ésta, y m es el número de términos. Basándose en este parámetro α y en el algoritmo típico de bisección (algoritmo 5 en la página anterior) se ha desarrollado el α-*Bisection*, que se presenta en el algoritmo 6 en la página anterior.

3.3.2. α-*Bisecting Spherical K-Means*

El método de bisección permite crear una simple pero eficiente variante del *K-Means*, el *Bisecting K-Means* [38, 43, 44], el cual da lugar a *clusters* mejores que los producidos por el método básico del *K-Means*, debido principalmente a que genera *clusters* de talla relativamente uniforme. También genera *clusters* tan buenos o incluso mejores que los obtenidos con técnicas de *clustering* jerárquicas aglomerativas [38].

Algoritmo 7 α-*Bisecting K-Means*

Entrada: $M \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$.

Salida: k *clusters* disjuntos$\{\pi_j\}_{j=1}^{k}$.

Paso-1: Seleccionar M como el *cluster* a dividir, establecer $t = 1$ y α usando la expresión 3.5.

Paso-2: Encontrar dos *clusters* π_t y π_{t+1} con el algoritmo α-*Bisection* (algoritmo 6 en la página anterior).

Paso-3: Si $t == k-1$ entonces finalizar, sino seleccionar π_{t+1} como *cluster* a dividir, incrementar t e ir al paso-2.

El algoritmo 7 presenta una variante del bien conocido *K-Means* (algoritmo 2 en la página 77) el α-*Bisecting K-Means*. El cual utiliza el α-*Bisection* (algoritmo 6 en la página anterior) para realizar todas las particiones requeridas aprovechando las buenas características de este método (el α no se recalcularía). No se utiliza directamente el método de bisección puesto que éste sólo da lugar a dos *clusters* y lo que interesa es obtener k *clusters*, lo cual lo permite el *K-Means*.

Cuando se utiliza como medida de similitud el coseno, el algoritmo *K-Means* se convierte en el *Spherical K-Means* (algoritmo 3 en la página 79). Lo mismo ocurre cuando hacemos uso como medida de distancia el coseno en el α-*Bisecting K-Means* (algoritmo 7 en la página anterior), obtenemos el α-*Bisecting Spherical K-Means:*

Algoritmo 8 α-*Bisecting Spherical K-Means*

Entrada: $M \in \mathbb{R}^{m \times n}$, ϵ, *maxiter*, $k \in \mathbb{N}$.

Salida: k *clusters* disjuntos$\{\pi_j\}_{j=1}^{k}$.

Paso-1: Calcular k *clusters* iniciales $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$ aplicando el α-*Bisecting K-Means*, algoritmo 7 en la página anterior y sus vectores concepto normalizados $\left\{c_j^{(0)}\right\}_{j=1}^{k}$. Inicializar $t = 0$.

Paso-2: Construir una nueva partición $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ inducida por $\left\{c_j^{(t)}\right\}_{j=1}^{k}$ según:

$$\pi_j^{(t+1)} = \left\{x \in M : \ x^{\mathrm{T}} \bullet c_j^{(t)} > x^{\mathrm{T}} \bullet c_l^{(t)} \ , \ 1 \leq l \leq k \ , \ j \neq l\right\}, \ 1 \leq j \leq k$$

Paso-3: Calcular los nuevos vectores concepto $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$ de la nueva partición.

Paso-4: Finalizar si el criterio de parada, expresión 3.4 en la página 78, se cumple o t es igual a *maxiter*, sino incrementar t en una unidad e ir al paso-2.

La idea subyacente del α-*Bisecting Spherical K-Means* (algoritmo 8) es utilizar el α-*Bisecting K-Means* (algoritmo 7 en la página anterior) para obtener una solución inicial aproximada y más tarde refinarla con el *Spherical K-Means* (algoritmo 3 en la página 79), intentando tomar ventaja de ambos algoritmos. Esta misma idea se utilizó en una versión previa del α-*Bisecting Spherical K-Means* (desarrollada y analizada en [115]), la cual se basaba en una versión de bisección más estándar que no contemplaba el parámetro α.

3.4. Variante del DBSCAN: VDBSCAN

El algoritmo del DBSCAN se basa en dos parámetros, el mínimo número de elementos que pueden formar un *cluster* y la proximidad entre dos elementos para considerarlos pertenecientes al mismo *cluster*. Ambos parámetros dependen exclusivamente del conjunto de datos con el que se trabajen y no se conocen a priori. Ahora bien, realizando una pequeña variación en el algoritmo DBSCAN y al trabajar en recuperación de información sobre colecciones de documentos se puede minimizar la variabilidad de dichos parámetros e incluso eliminarla del todo.

3.4.1. El algoritmo VDBSCAN

En el algoritmo DBSCAN original [40] (algoritmo 4 en la página 80) el parámetro del mínimo número de elementos se utiliza cada vez que se buscan los vecinos de un elemento. Si el número de vecinos es menor que dicho parámetro entonces ese elemento o se considera ruido o no se tienen en cuenta sus vecinos. Cambiar este comportamiento es la idea principal del VDBSCAN (algoritmo 9).

Algoritmo 9 *VDBSCAN*

Entrada: $M \in \mathbb{R}^{m \times n}$, $\epsilon \in \mathbb{R}$, $MinEle \in \mathbb{N}$.

Salida: k *clusters* disjuntos $\{\pi_j\}_{j=1}^{k}$, R (Ruido)

Paso-1: $NoU = M$ y $j = 1$

Paso-2: Mientras existan elementos en NoU

- **Paso-2.1:** Incluir en S, un elemento aleatorio de NoU.
- **Paso-2.2:** Inicializar $U = \{\}$
- **Paso-2.3:** Mientras existan elementos en S
  - **Paso-2.3.1:** Elegir $x \in S$
  - **Paso-2.3.2:** $U = U \cup \{x\}$
  - **Paso-2.3.3:** $NoU = NoU - S$
  - **Paso-2.3.4:** $V = EpsVecindario(x, NoU, \epsilon)$ (elementos de NoU que están a una distancia ϵ de x).
  - **Paso-2.3.5:** $S = (S \cup V) - U$
- **Paso-2.4:** Si $|U| < MinEle$ Entonces $R = R \cup U$
- **Paso-2.5:** Si $|U| \geq MinEle$ Entonces $\pi_j = U$ y $j = j + 1$
- **Paso-2.6:** $NoU = M - \{\pi_i\}_{i=1}^{j} - R$

En el VDBSCAN el número mínimo de elementos sólo se utiliza para determinar si el cluster construido debe considerarse ruido o *cluster* real. Es decir, se tiene en cuenta siempre el vecindario de un elemento, sea cual sea el tamaño de éste. Ahora bien, cuando partiendo de una semilla ya se han agrupado todos los posibles elementos teniendo en cuenta todos sus vecinos, es entonces cuando se determina si dicha agrupación da lugar a un *cluster* o dicha agrupación pasa a considerarse ruido.

El procedimiento seguido en el VDBSCAN (algoritmo 9) es que mientras hayan elementos sin analizar (Paso-2, algoritmo 9) se elegirá una semilla, la cual dará lugar a un *cluster*. Ahora, a partir de dicha semilla se irán incluyendo en la agrupación sus vecinos y los vecinos de éstos continuamente (Paso-2.3, 2.3.1,...,2.3.5, algoritmo 9). Primero elegimos un elemento (Paso-2.3.1, algoritmo 9) y calculamos su vecindario (Paso-2.3.4, algoritmo 9) e incluimos todos ellos como posibles elementos de expansión

(para valorar sus posibles vecinos) (Paso-2.3.5, algoritmo 9 en la página anterior). Cuando ya no quedan elementos a incluir (todos los vecinos ya están incluidos) se comprueba cuantos elementos forman la agrupación y en función de ello se elegirá su naturaleza (Paso-2.4 y Paso-2.5, algoritmo 9 en la página anterior)

Con este cambio en la forma de actuar conseguimos tres aspectos muy importantes:

1. Reducir de dos a una el número de veces que se calcula el vecindario en cada iteración del algoritmo, y sobre menos elementos.

2. Poder determinar un valor constante de forma general para el mínimo número de elementos que forman un *cluster*.

3. Poder obtener heurísticamente un buen valor para la distancia mínima entre dos elementos de un mismo *cluster*.

Por lo tanto, conseguimos reducir el coste computacional y, por consiguiente, el temporal del algoritmo (ver resultados del apartado 3.4.3 en la página 89). Realmente en el mejor de los casos computacionalmente hablando, se conseguiría teóricamente un coste lineal $\Omega(n)$, que coincidiría con el caso de formar un único *cluster*. Y en el peor de los casos, todos los elementos son ruido, el coste sería $\mathcal{O}\left(\frac{n^2}{2}\right)$, al menos teóricamente. Aunque no se ha hecho el estudio, al igual que el DBSCAN, la complejidad se podría mejorar mediante índices espaciales.

También, conseguimos reducir uno de los parámetros del algoritmo, ya que, al trabajar en recuperación de información, se puede establecer el número mínimo de elementos que forman un *cluster* a dos elementos (ver el estudio más detallado en el apartado 3.5.1 en la página 93). Y, una vez determinado uno de los parámetros, el otro, la distancia entre dos elementos, se puede calcular de forma heurística para obtener un valor que da lugar a un buen comportamiento, aunque no sea el óptimo (ver más detalles en el apartado 3.5.2 en la página 95).

Con todo ello alcanzamos el objetivo de reducir la variabilidad del algoritmo determinando a priori uno de los parámetros y el otro mediante un método heurístico. Claro está que todo lo conseguido sólo es aceptable si el VDBSCAN presenta iguales prestaciones al DBSCAN o al menos las diferencias son mínimas. En el apartado 3.4.2 se presenta la demostración de que las prestaciones de recuperación son iguales.

3.4.2. Estudio de prestaciones de recuperación

Los resultados del estudio de prestaciones de recuperación del VDBSCAN (algoritmo 9 en la página anterior) se han presentado en las figuras 3.2 en la página siguiente y 3.3 en la página siguiente. En ellas se observa al VDBSCAN con el número mínimo de elementos fijado a 2 y el valor de la distancia mínima obtenido heurísticamente (ver apartado 3.5 en la página 90) comparado con el DBSCAN (algoritmo 4 en la página 80) con varios valores para el número mínimo de elementos y con la distancia mínima igual a la usada en el VDBSCAN. También se incluyen como referencia los resultados de la matriz de pesos. En dicha comparación se han usado dos colecciones

de prueba: la '*Times Magazine 1963*' y la '*Cystic Fibrosis Database*' (ver más detalles en el apartado 2.5.5 en la página 62).

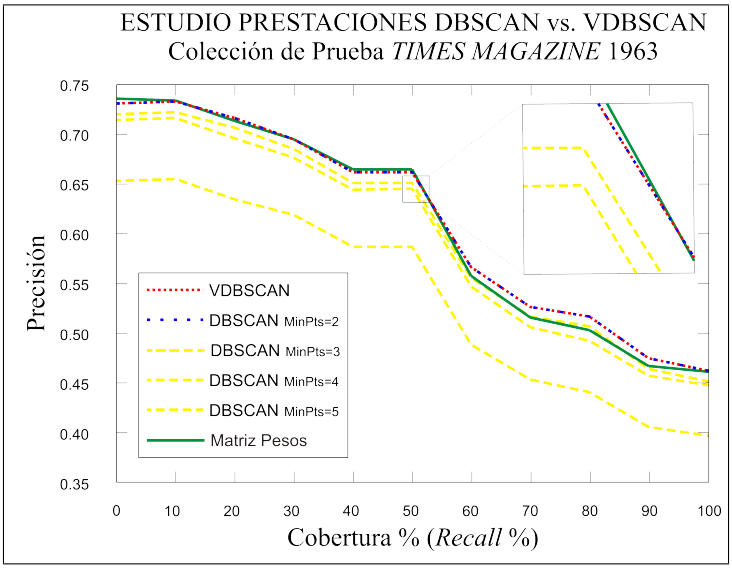

Figura 3.2: Estudio de prestaciones DBSCAN vs. VDBSCAN - Colección *Times* -.

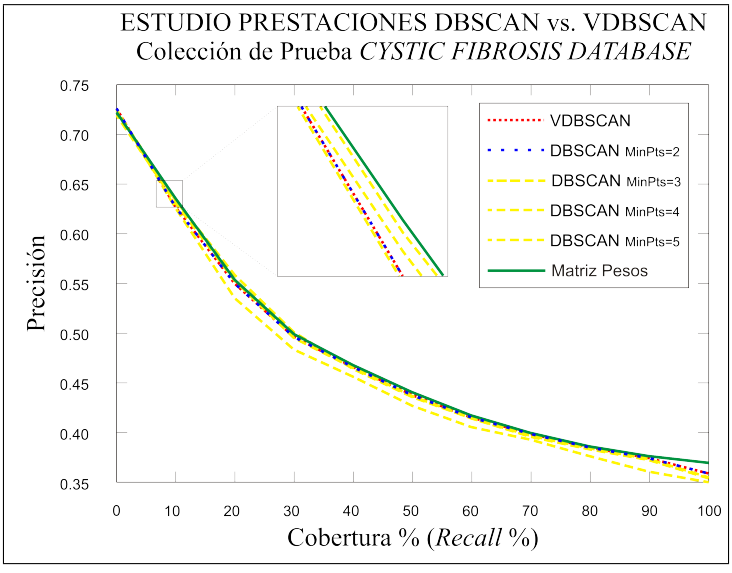

Figura 3.3: Estudio de prestaciones DBSCAN vs. VDBSCAN - Colección *Cystic* -.

En la figura 3.2 en la página anterior se observa que el DBSCAN para otros valores del mínimo número de elementos empeora en prestaciones. De hecho a medida que dicho valor aumenta las prestaciones disminuyen, esto se aprecia más claramente en la zona de aumentada.

En la figura 3.3 en la página anterior en cambio se observa que el algoritmo DBSCAN para otros valores del mínimo número de elementos, las prestaciones incluso llegan a mejorar en algunos casos. Así y todo el comportamiento del algoritmo DBSCAN con un valor de dos y el algoritmo VDBSCAN es muy bueno, e incluso mejor globalmente.

En ambas figuras se observa que tanto el comportamiento del VDBSCAN como el comportamiento del DBSCAN con valor de dos para el mínimo número de elementos es prácticamente igual (los resultados numéricos aseguran que ambos métodos obtienen valores iguales). Además, las dos curvas son muy similares a la de la matriz de pesos, de hecho en algunos momentos la mejoran.

De todo este estudio se puede concluir que el nuevo algoritmo, VDBSCAN, tiene las mismas prestaciones que el algoritmo DBSCAN para un valor de dos en el número mínimo de elementos de un *cluster*. Además, en las pruebas realizadas con otros valores el comportamiento obtenido en ambos algoritmos era igual. Aunque es posible que exista algún caso particular en el que no se de dicha igualdad, ya que el algoritmo DBSCAN puede dar lugar en condiciones concretas a resultados distintos si la semilla inicial varía [40] y en cambio el algoritmo VDBSCAN da el resultado independiente de la semilla elegida.

3.4.3. Estudio temporal

Es interesante ver como influye en los costes temporales la eliminación de un cálculo de vecindario. Al ser, en el caso del VDBSCAN (algoritmo 9 en la página 86), el cálculo de los vecinos dependiente del número de elementos todavía no analizados y, en cambio, en el DBSCAN (algoritmo 4 en la página 80), independiente pero influido por el número de veces que se calcula el mínimo número de elementos que forman un *cluster*, entonces las diferencias temporales no serán proporcionales. Por ello en la figura 3.4 en la página siguiente se presenta el estudio temporal realizado comparando el DBSCAN para varios valores del mínimo número de elementos y VDBSCAN en las colecciones de prueba: la '*Times Magazine 1963*' y la '*Cystic Fibrosis Database*' (ver más detalles en el apartado 2.5.5 en la página 62).

Para ambos algoritmos el valor de la distancia entre dos elementos es el mismo, el obtenido mediante el método heurístico desarrollado para el algoritmo VDBSCAN (ver apartado 3.5.2 en la página 95).

La figura 3.4 en la página siguiente destaca dos aspectos muy importantes. Primero, el tiempo obtenido por el algoritmo DBSCAN es independiente del parámetro del mínimo número de elementos. Y segundo, el algoritmo VDBSCAN siempre es considerablemente más rápido que el DBSCAN, incluso llegando a tardar menos de la mitad del tiempo con la colección '*Cystic Fibrosis Database*'.

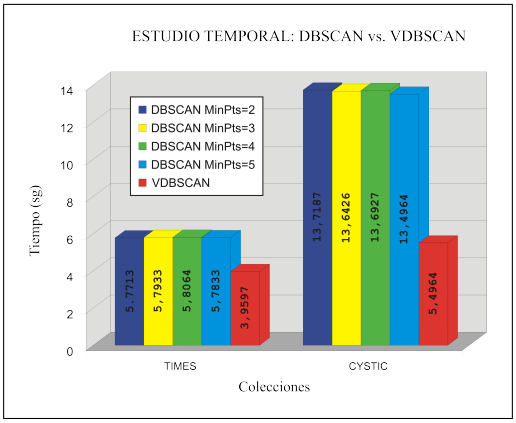

Figura 3.4: Estudio temporal: DBSCAN vs. VDBSCAN.

3.5. Eliminación de parámetros del VDBSCAN

Como se muestra en este apartado, el número mínimo de elementos se puede fijar, dejando así de ser un parámetro. También se presenta en esta sección un método heurístico que se ha desarrollado para determinar el valor de proximidad a elegir. Dicho valor permite obtener una buena calidad de recuperación aunque no asegure ser el óptimo global, el cual daría lugar a la mejor calidad de recuperación posible.

Es cierto que los autores del DBSCAN aportan una solución simple para determinar los parámetros de entrada del algoritmo [40]. Pero dicha solución sólo es factible para pequeños conjuntos de datos, ya que hay que realizar costosas operaciones para cada dato (k-vecinos más próximos), lo cual es ineficiente para grandes conjuntos de datos [40, 59]. Además, la evaluación de los parámetros se basa en la visualización de un gráfico el cual está limitado a conjuntos de datos relativamente pequeños debido a las limitaciones de resolución de las pantallas [40, 59]. Esta limitación puede ser suavizada utilizando muestras de los datos, pero entonces la precisión de la estimación de los parámetros puede decrecer significativamente [59].

Por otro lado, se podría ver innecesario estudiar la fijación de los parámetros del DBSCAN ya que existe una variante, el DBCLASD [59], el cual no precisa parámetros de entrada. Sin embargo, esta variante del DBSCAN se basa en la asunción de que los elementos dentro de un *cluster* están uniformemente distribuidos [59], situación que no tiene por que darse en los sistemas de recuperación de información. Y aunque en algunos casos, como los catálogos de terremotos, también funciona bien aún sin estarestrictamente los datos distribuidos uniformemente [59], en los sistemas de recuperación de información los documentos de las colecciones están distribuidos aleatoriamente. Además, hay que tener en cuenta también el coste temporal de ambos algoritmos. Con conjuntos de datos pequeños el DBSCAN es ligeramente más rápido que el DBCLASD, pero a medida que crece el tamaño del conjunto de datos esta diferencia crece tendiendo a ser el doble e incluso el triple [59].

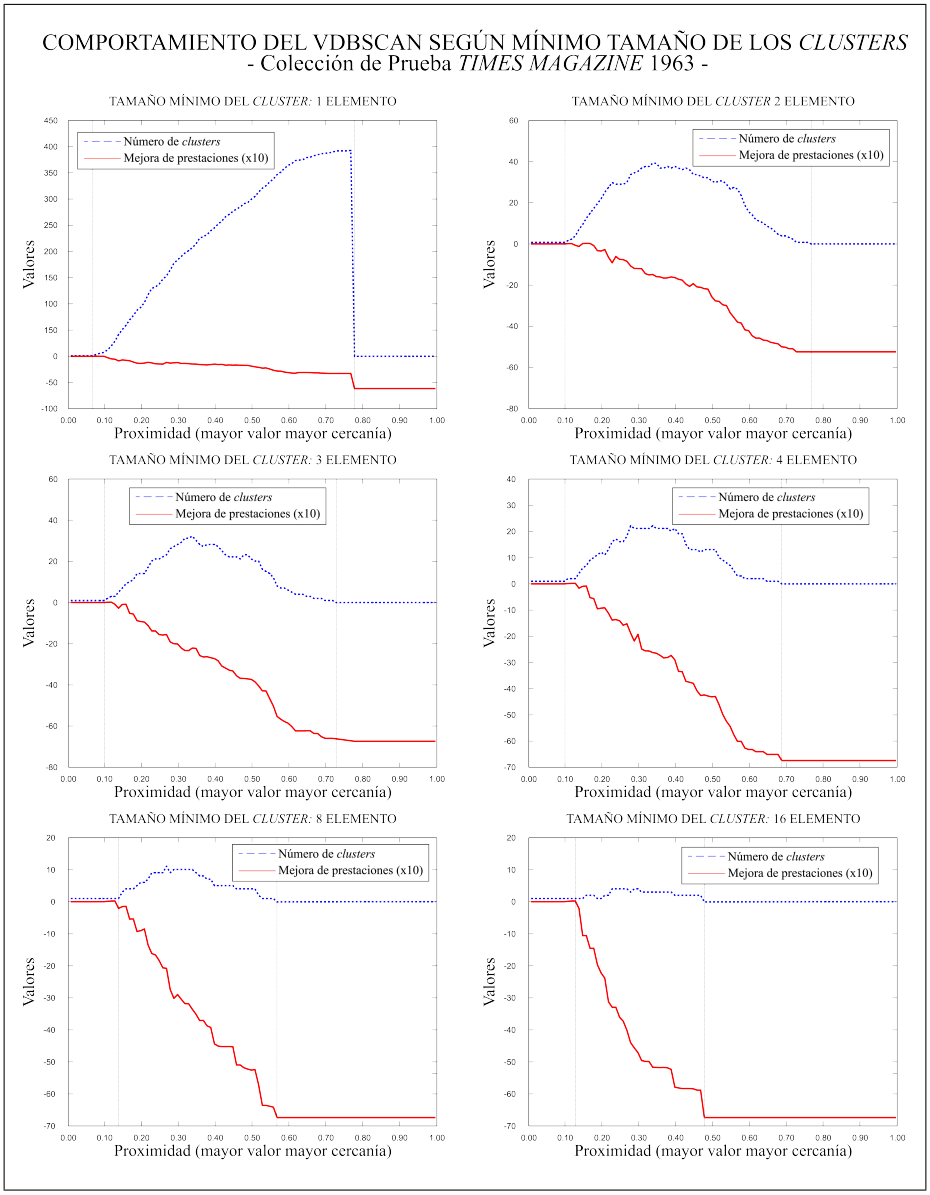

Figura 3.5: Mínimo número de elementos por *cluster* - Colección *Times* -.

En las figuras 3.5 y 3.6 en la página siguiente se presenta un estudio sobre el comportamiento del algoritmo VDBSCAN para un amplio abanico de proximidades (casi total) respecto a diferentes números mínimos de elementos por *cluster*, y todo ello para dos colecciones de documentos de prueba: '*Times Magazine 1963*' y '*Cystic Fibrosis Database*' (ver más detalles en el apartado 2.5.5 en la página 62).

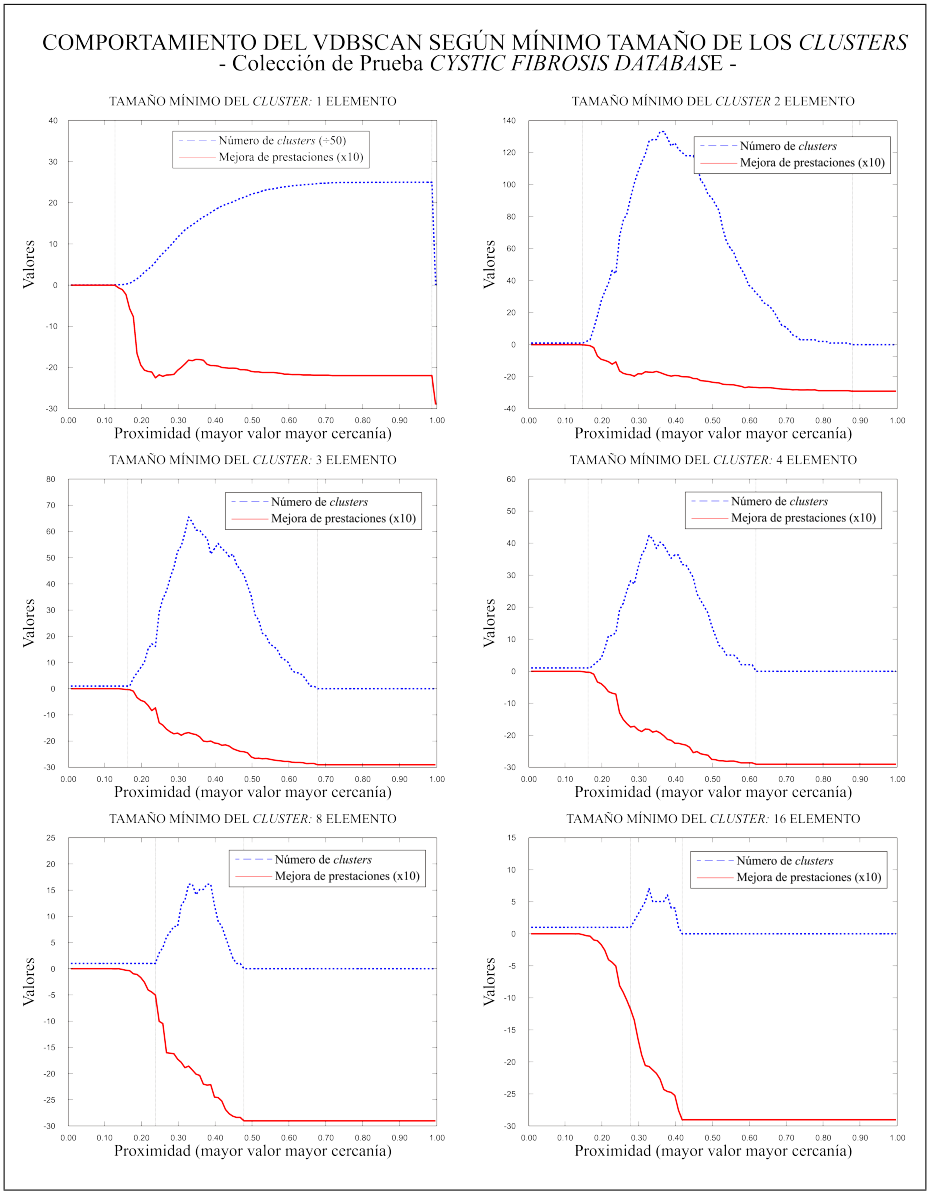

Figura 3.6: Mínimo número de elementos por *cluster* - Colección *Cystic* -.

Con estas gráficas se pretende observar y estudiar el comportamiento del algoritmo VDBSCAN respecto a sus dos parámetros (por ello el eje vertical representará el número de clusters o la mejora de prestaciones según corresponda a una u otra curva). En todas las gráficas se muestran dos curvas, una con el número de *cluster* que genera el algoritmo respecto a las distintas proximidades estudiadas. Y otra con

la relación de mejora (o empeoramiento) de la calidad de recuperación obtenida respecto a la matriz de pesos, también en función de diferentes proximidades. Destaca, además, una zona delimitada por dos líneas verticales, que determinan el intervalo de proximidades que dan lugar a más de un cluster ya que si no estaríamos en una situación equiparable a trabajar con la matriz de pesos, pero perdiendo prestaciones de calidad de recuperación (fundamentalmente por los documentos que el VDBSCAN considera ruido).

3.5.1. Fijación del mínimo número de elementos por *cluster*

La fijación del mínimo número de elementos por *cluster* conlleva la eliminación de un parámetro y por tanto la eliminación de variabilidad. Dicha variabilidad es perjudicial ya que interesa encontrar una solución buena, si no óptima, valorando el mínimo número de casos posibles.

De las gráficas de las figuras 3.5 en la página 91 y 3.6 en la página anterior se deduce que hay dos tipos de comportamiento. Uno cuando el tamaño mínimo de *cluster* es uno y otro cuando es mayor de uno. Esto se observa claramente tanto en la curva del número de *clusters* generados como en la mejora de prestaciones (en relación con la matriz de pesos). De todas formas se resalta con mayor claridad en la curva del número de *clusters* generados. En ésta se aprecia que mientras al ser el tamaño mínimo igual a un elemento, la curva nunca es decreciente, al ser éste al menos de dos elementos por *cluster* la curva hace una campana (con tendencia hacia una distribución normal, estadísticamente hablando).

Mediante un análisis bastante sencillo y lógico se puede descartar la utilización de un elemento como tamaño mínimo de *cluster*. Esto es debido principalmente a varios motivos:

- Todos los elementos pertenecerán siempre a un *cluster*, aunque sea el formado por ellos mismos. Con lo cual no habrá ningún elemento que se pueda considerar ruido, perdiendo así una de las características potenciales del algoritmo VDBSCAN.
- A poco que se estudie la separación de los documentos en *clusters* cuando se genera más de uno, se verá que la tendencia es a construir un *cluster* con prácticamente todos los documentos y luego el resto con prácticamente ninguno, muchas veces un sólo elemento por *cluster*. Con ello básicamente lo que se está haciendo es construir un único *cluster* más ruido, más algún pequeño *cluster* prácticamente irrelevante de cara a las consultas en general.
- Cuando se generan muchos *clusters* se tiende cada vez más a crearlos unitarios, indicando más que dichos elementos son ruido y no *clusters*. Llegando incluso al extremo de que toda la colección está dividida en tantos *clusters* como documentos la forman, situación bastante inútil de cara a la recuperación de información.
- Al comparar la calidad de las recuperaciones con la calidad ofrecida por un sistema modelado únicamente con la matriz de pesos, se observa como el deterioro

de dicha calidad es grande y muy rápido. Prácticamente desde el momento que se generan más de un *cluster* el empeoramiento es considerable, y aunque llega un momento que se estabiliza, lo hace a un nivel inaceptable.

Cuando en el VDBSCAN se varía el tamaño mínimo de *cluster*, siendo éste siempre superior a uno, se observa que el comportamiento para los posibles tamaños mínimos de *cluster* es análogo, aunque con diferencias significativas. Destacan, por un lado, que a mayor tamaño mínimo de *cluster* el número máximo de *clusters* generados disminuye. Llegando incluso, en algunos casos, a diferencias considerables con poco aumento del tamaño mínimo de *cluster* (por ejemplo, para la colección *Cystic* figura 3.6 en la página 92, variando de un tamaño mínimo de dos a tres elementos, el número máximo de *clusters* pasa de aproximadamente 135 a 65, se reduce más o menos a la mitad).

Por otro lado, también es destacable que, el intervalo de proximidades entre las cuales se encuentra la campana de la curva del número de *clusters* disminuye a medida que el tamaño mínimo aumenta. Esto directamente implica, que cuanto más estrecha es la campana pequeñas variaciones en la proximidad elegida dan lugar variaciones grandes en el número de *clusters.*

La ventana de proximidades delimitada por la obtención de más de un *cluster*, la cual delimita la campana de la curva de número de *clusters*, también es la que establece el intervalo de estudio de la mejora o empeoramiento de la calidad de recuperación. Al ser esta ventana más estrecha, compacta más el comportamiento de la curva de mejora de calidad, con lo cual la caída de prestaciones se hace más pronunciada y los desniveles se van haciendo más agudos. Por lo tanto, pequeñas variaciones en la proximidad elegida dan lugar a grandes variaciones en la calidad de la recuperación, lo que indicaría que el problema está más condicionado.

La elección de un número mínimo de elementos por *cluster* tiene que ser tomada teniendo en cuenta los siguientes aspectos de interés:

- Los *clusters* deben de ser lo más compactos y específicos que se pueda, al menos debe existir esa posibilidad aunque luego se estime oportuno no llegar al límite. Con compactos se quiere que los elementos de un mismo *cluster* estén lo más cerca posible entre ellos, y con específicos, se desea que la temática que une a los documentos de un *cluster* sea más concreta y menos general.

- No limitar arbitrariamente el tamaño mínimo de los *clusters* ya que en general no se tiene información suficiente, ni siquiera, para saber el número de *clusters* existentes y menos para conocer el tamaño mínimo de los mismos. Aunque en algunos casos se puede disponer de dicha información y con ella tomar una decisión apropiada para mejorar las prestaciones, no es un conocimiento presente normalmente.

- La calidad de recuperación de los algoritmos de *clustering* suele ser peor que la alcanzada con la matriz de pesos. Pero con la utilización de ellos se pretende fundamentalmente mejorar las prestaciones temporales de la consulta a cambio de empeoramientos aceptables en la calidad de recuperación. Por ello interesa el mayor número de *clusters* posible mientras no cambie sustancialmente las

prestaciones de recuperación, ya que el tiempo de consulta disminuirá más, al tener que trabajar sólo con los documentos de un *cluster* y no con todos los de la colección.

Basándose en todos estos aspectos de interés se puede elegir un número mínimo de dos elementos por *cluster* como valor para fijar el correspondiente parámetro. Permite *clusters* más específicos al permitir menos elementos por *cluster* y además los hace más compactos ya que para la misma proximidad genera mayor número de *clusters* (como se puede ver en las figuras 3.5 en la página 91 y 3.6 en la página 92). La limitación arbitraria de tamaño es mínima sino nula, ya que limita a dos elementos, pero menos de dos está claro que no es viable. Para la misma proximidad el número de *clusters* es mayor y la calidad de recuperación empeora menos, e incluso en algunos casos puede mejorar a la obtenida con la matriz de pesos (por ejemplo, con la colección *Times*, tamaño mínimo de 2 documentos y una proximidad más o menos de 0.15 la calidad de recuperación mejora respecto al sistema modelizado con la matriz de pesos).

3.5.2. Heurística para seleccionar una buena proximidad

La elección de la proximidad a utilizar como parámetro es también una cuestión delicada. Ya que en función del valor elegido los resultados pueden ser estupendos o desastrosos (en las figuras 3.5 en la página 91 y 3.6 en la página 92 se puede observar dicho comportamiento). Por lo tanto sería interesante poder dar un valor a este parámetro que permita obtener buenos resultados, es decir, buena calidad de recuperación y un buen número de *clusters*.

A diferencia del número de elementos mínimos por *cluster* la proximidad no se puede fijar, al depender directamente de los documentos de la colección. Pero lo que sí se puede hacer es utilizar una heurística que ayude a seleccionar un valor de proximidad bueno, si no óptimo, con un coste computacional razonable. En este sentido se ha desarrollado un método heurístico (plasmado en el algoritmo 10 en la página 97), el cual determina un valor de proximidad máximo local, basándose en el siguiente proceso iterativo y sustentado por las siguientes propiedades:

Propiedad 1: Sea $f(x)$ la curva diferencia entre dos curvas de comportamiento de precisión vs. cobertura para once puntos estándares. $f(x)$ es continua en un intervalo $[a, b]$ y derivable en dicho intervalo abierto $]a, b[$.

Propiedad 2: Sea $g^{\mathrm{n}}(y) = \{r(c_0, c_1) \cup \ldots \cup r(c_{n-1}, c_n)\}$, donde $c_0 = a$, $c_n = b$ y $r(c_j, c_k)$ la recta que pasa por los puntos c_j y c_k. Sea $c_i = c_j + \frac{c_k - c_j}{2}$ y $g^{\mathrm{n}}(c_i) = f(c_i)$. Entonces $f(x) = \lim_{n \to \mathrm{ínf}} g^{\mathrm{n}}(y)$.

Propiedad 3: Dada una recta r entre dos puntos, $r = recta(c_i, c_j)$ con $c_i < c_j$, entonces r es creciente si $r(c_i) < r(c_j)$ y decreciente si $r(c_i) > r(c_j)$.

Proceso Iterativo: De la función $f(x) \simeq g^4(y) = \{r(c_0, c_1) \cup r(c_1, c_2) \cup r(c_2, c_3) \cup r(c_3, c_4)\}$ parte el proceso iterativo, donde $c_0 = a$, $c_2 = c_0 + \frac{c_4 - c_0}{2}$, $c_1 = c_0 + \frac{c_2 - c_0}{2}$, $c_3 = c_2 + \frac{c_4 - c_2}{2}$ y $c_4 = b$, cumpliéndose además que $f(c_0) > f(c_3)$

(decreciente). Además, $LI = c_0$, $PMI = c_1$, $PM = c_2$, $PMD = c_3$ y $LS = c_4$ que delimitan las rectas de trabajo.

Paso 1: Reordenar si hiciese falta los índices para obtener $g^{\mathrm{k}}(y) = \{r(c_0, c_1) \cup \ldots \cup r(c_{k-1}, c_k)\}$ y seleccionar las rectas de trabajo $g^{\mathrm{k}}(y) \supseteq h(y) = \{r(LI, PMI) \cup r(PMI, PM) \cup r(PM, PMD) \cup r(PMD, LS)\}$.

Paso 2: Si la recta $r(PMI, PM)$ es decreciente entonces ésta recta se subdivide en dos subrectas de la siguiente forma: $c_i = c_{LI} + \frac{c_{PMI} - c_{LI}}{2}$, $g^{\mathrm{k}}(c_i) = f(c_i)$, $c_j = c_{PMI} + \frac{c_{PM} - c_{PMI}}{2}$, $g^{\mathrm{k}}(c_j) = f(c_j)$ y $LI = LI$, $LS = PM$, $PM = PMI$, $PMI = c_i$, y $PMD = c_j$. Y se volvería al paso 1.

Paso 3: Si la recta $r(PMI, PM)$ es creciente entonces la recta $r(PM, PMD)$ se estudiará. Si ésta es decreciente finalizariamos y el punto elegido sería PM. Sino se crearían las rectas de trabajo de la siguiente forma: $c_i = c_1 + \frac{c_2 - c_1}{2}$, $g^{\mathrm{k}}(c_i) = f(c_i)$, $c_j = c_2 + \frac{c_3 - c_2}{2}$, $g^{\mathrm{k}}(c_j) = f(c_j)$ y $LI = PMI$, $LS = PMD$, $PM = PM$, $PMI = c_i$, y $PMD = c_j$. Y se volvería al paso 1.

Básicamente el algoritmo 10 en la página siguiente divide el intervalo de proximidades en el cual se trabaja en dos mitades y elige sus puntos medios. Comprueba si en la proximidad media de la mitad izquierda se mejora la calidad respecto al punto medio actual que divide el intervalo en dos. Si mejora las prestaciones de recuperación se define un nuevo intervalo siendo ahora el límite superior el que hasta ahora era el punto medio del intervalo. Si la calidad hubiese empeorado entonces se elegiría el punto medio derecho. Si éste mejorara entonces se redefiniría el intervalo siendo el límite inferior el punto medio izquierdo y el superior el punto medio derecho. Si, en cambio, no hubiese mejora entonces el punto medio actual se elegiría como valor de proximidad máxima local, que suele ser un buen valor, que incluso está cerca del óptimo en muchos casos. Cuando una proximidad es peor que otra anterior porque el número de *clusters* no es mayor de dos, se asume que la mejora en esa proximidad es menos infinito, ya que así la siguiente proximidad aunque empeore la curva de la matriz de pesos puede considerarse mejor que la proximidad que no genera suficientes *clusters*.

La prioridad que se le da a la zona izquierda se basa en que a medida que la proximidad crece la calidad de recuperación empeora, al menos como tendencia general de comportamiento (en las figuras 3.5 en la página 91 y 3.6 en la página 92 se aprecia este hecho claramente). Esto permite asumir con más seguridad que cuanto más a la izquierda en la curva esté el valor de proximidad elegido, hasta cierto umbral, mayor será la mejora de calidad.

Algoritmo 10 *Selección de buena proximidad*

Entrada: $M \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{m\times nq}$, $R \in \mathbb{R}^{mr\times nq}$, $APR_M \in [0,1]^{11\times 1}$, $MinEle \in \mathbb{N}$.

Salida: *Clusters* disjuntos π, $Prox \in \mathbb{R}$, $APR \in [0,1]^{11\times 1}$.

Paso-1: Inicializar: $Li = 0$ y $Ls = 1$ (Límite inferior y superior del intervalo)

Paso-2: Calcular datos del punto medio:

Paso-2.1: $PM = Li + \frac{Ls-Li}{2}$

Paso-2.2: $\pi_{PM} = \text{VDBSCAN}\,(M, PM, MinEle)$

Paso-2.3: $APR_{PM} = \text{PrecisionCobertura*}\,(M, Q, R, \pi_{PM})$

Paso-2.4: Calcular $v_{PM} = \sum_{i=1}^{11}(APR_{PM} - APR_M)$

Paso-3: Calcular datos del punto medio izquierdo:

Paso-3.1: Calcular $PMI = Li + \frac{PM-Li}{2}$

Paso-3.2: $\pi_{PMI} = \text{VDBSCAN}\,(M, PMI, MinEle)$

Paso-3.3: $APR_{PMI} = \text{PrecisionCobertura*}\,(M, Q, R, \pi_{PMI})$

Paso-3.4: Calcular $k_{PMI} = |\pi_{PMI}|$ (Número de *clusters* en π_{PMI})

Paso-3.5: Calcular $v_{PMI} = \sum_{i=1}^{11}(APR_{PMI} - APR_M)$

Paso-4: Si PMI mejora** PM Entonces

Paso-4.1: $Ls = PM$

Paso-4.2: Ir al Paso-2

Paso-5: Calcular datos del punto medio derecho:

Paso-5.1: Calcular $PMD = PM + \frac{Ls-PM}{2}$

Paso-5.2: $\pi_{PMD} = \text{VDBSCAN}\,(M, PMD, MinEle)$

Paso-5.3: $APR_{PMD} = \text{PrecisionCobertura*}\,(M, Q, R, \pi_{PMD})$

Paso-5.4: Calcular $k_{PMD} = |\pi_{PMD}|$ (Número de *clusters* en π_{PMD})

Paso-5.5: Calcular $v_{PMD} = \sum_{i=1}^{11}(APR_{PMD} - APR_M)$

Paso-6: Si PMD mejora** PM Entonces

Paso-6.1: $Li = PMI$ y $Ls = PMD$

Paso-7: Finalizar: $\pi = \pi_{PM}$, $Prox = PM$, $APR = APR_{PM}$.

* Calcula la curva media de precisión para 11 niveles estándar de cobertura (ver apartado 2.5.2).

** Mejorar es cumplir $v_{PMI} > v_{PM}$ Y $k_{PMI} > 2$, empeorar es no mejorar.

Lógicamente en el criterio de mejora, además de la calidad de recuperación, se contempla también el hecho de que se generen un mínimo número de *clusters*, en este caso el límite se ha establecido a dos para que sirva en general para cualquier caso (si un *cluster* fuese viable directamente trabajaríamos con la matriz de pesos y no con *clustering*). Pero si se tiene información suficiente sobre la colección dicho límite se puede subir.

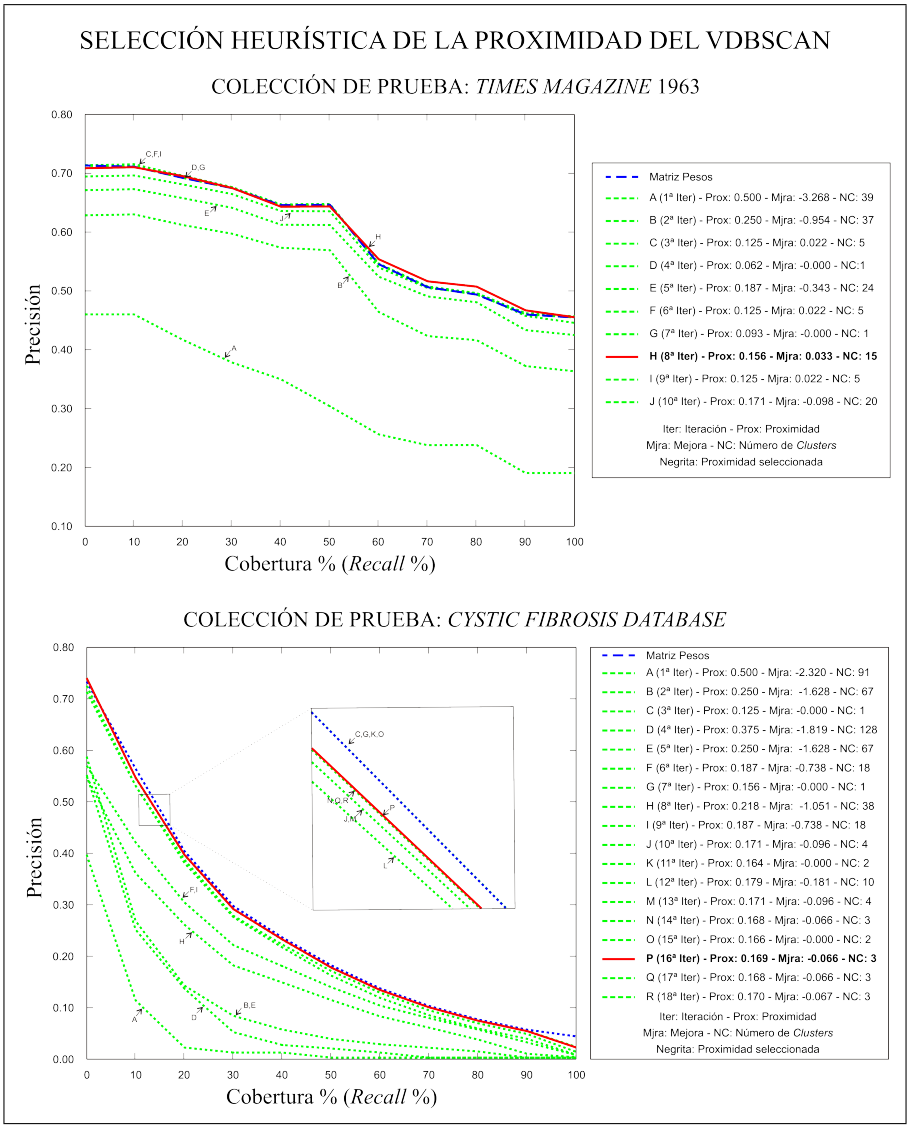

Figura 3.7: Proceso heurístico de selección de la proximidad para el VDBSCAN.

En la figura 3.7 en la página anterior se presentan el comportamiento del proceso heurístico de selección de una buena proximidad para el VDBSCAN, descrito en el algoritmo 10 en la página 97. Aparecen dos gráficos, uno correspondiente a la colección de prueba *Times Magazine* 1963 y el otro a la colección de prueba *Cystic Fibrosis Database*. En ambos casos se observa un comportamiento similar obteniendo además, buenos resultados finales, también en ambos casos.

Las dos gráficas de la figura 3.7 en la página anterior presentan tres tipos de curvas. Una discontinua gruesa y azul que corresponde con la curva obtenida mediante la matriz de pesos, la cual se utiliza como curva de referencia. Otra curva discontinua fina y verde que identifica las distintas curvas intermedias obtenidas durante el proceso heurístico, cada una de las cuales identificada con la etiqueta correspondiente a la iteración en la que se genera. Cuando éstas se corresponden con la curva obtenida mediante la matriz de pesos significa que el número de *clusters* generados no es superior al umbral elegido de dos *clusters*. Y por último, una curva lisa y roja que se corresponde con la curva obtenida con la proximidad seleccionada, también está identificada con una etiqueta identificativa de la iteración en la que se produce. En la leyenda se presentan también datos importantes que ayudan a entender el proceso y justificar porque la proximidad elegida es una buena selección.

Aunque a priori no se establece ninguna cota superior en el número de iteraciones, las pruebas realizadas están entre diez y veinte iteraciones. Es cierto que en cada iteración se debe realizar una evaluación completa de las consultas de prueba, pero el coste computacional que conlleva es aceptable en la fase de modelización y puesta en marcha de un sistema de recuperación de información. De todas formas se podría plantear estudiar el trabajar con un subconjunto de las consultas de prueba, e incluso con una muestra de la colección. Tampoco hay que olvidar que en la implementación se han hecho optimizaciones para evitar algunas evaluaciones completas de las consultas de prueba. Por ejemplo, cuando se debe volver a evaluar una proximidad ya elegida anteriormente (Pasos 2.1, 3.1 y 5.1 del algoritmo 10 en la página 97). Dichas optimizaciones no se han incluido en la versión presentada en la tesis (algoritmo 10 en la página 97) porque se desea destacar la metodología del mismo.

En el algoritmo 10 en la página 97 a la hora de determinar la mejora de una proximidad respecto a otra sólo se tiene en cuenta la calidad de recuperación, pero se podrían contemplar otros factores, dependiendo de los intereses del sistema. Por ejemplo, una posible optimización que se podría plantear es compensar la pérdida de calidad de recuperación aumentando el número de *clusters*. De esta forma se podría trabajar con una función cuyos parámetros fuesen la mejora de calidad y el número de *clusters* generados.

3.6. Conclusiones

La comunidad investigadora está más o menos de acuerdo, independientemente del método de *clustering* elegido, en que una buena técnica de *clustering* debe verificar en un alto grado las siguientes características: escalabilidad, *clusters* arbitrarios, parámetros de entrada mínimos, manejo de ruido, insensibilidad al orden de entrada,

dimensionalidad alta e interpretabilidad y usabilidad. Los principales métodos desarrollados en la tesis el *α-Bisecting Spherical K-Means* y el VDBSCAN han buscado presentar estar características. Aunque el VDBSCAN destaca más en algunas de las características, como encontrar *clusters* arbitrarios, manejar el ruido o la sensibilidad al orden de entrada.

La clasificación más común de los métodos de *clustering* es dividirlos en particionales y jerárquicos; estos últimos a su vez se dividen en aglomerativos y divisivos. A parte de estas dos principales categorías de algoritmos de clustering han surgido otros métodos, los cuales suelen estar enfocados a problemas o conjuntos de datos específicos: *clustering* basado en densidades, *clustering* basado en grid, *clustering* basado en modelos y *clustering* de información categórica. La tesis se ha centrado en dos tipos concretos, en *clustering* particional y en *clustering* basado en densidades.

El *clustering* particional construye particiones, donde cada *cluster* optimiza un criterio de *clustering*. Se caracterizan por su alta complejidad, algunos incluso enumeran exhaustivamente todas las posibles particiones intentando encontrar el óptimo global. Normalmente se empieza con una solución inicial aleatoria y luego se refina. Su principal desventaja es que precisan como parámetro de entrada el número de *clusters* que se van a construir. También presenta la desventaja de encontrar soluciones locales, ya que, debido a su naturaleza de funcionamiento, la búsqueda de soluciones globales conlleva demasiados costes computacionales. Por ello, se suele seleccionar la mejor solución obtenida tras varias ejecuciones con distintas semillas.

El algoritmo *K-Means* [3, 6, 38, 45] es un proceso iterativo que suele tender a la convergencia de la función objetivo, por lo tanto es interesante disponer de un criterio de parada, así como contemplar un número máximo de iteraciones. Es la técnica iterativa particional más ampliamente usada siendo las principales razones [3, 21, 23, 43, 44, 46]:

- Su complejidad temporal es $\mathcal{O}(Ikn)$ donde n es el número de documentos, k es el número de *clusters* e I es el número de iteraciones que el algoritmo necesita para converger. Típicamente, k e I son fijas y se establecen con anterioridad, con lo cual el algoritmo tendría una complejidad temporal lineal con la talla de la colección.

- Su complejidad espacial es $\mathcal{O}(k+n)$ más el espacio necesario para la matriz que modeliza la colección.

- No influye el orden en el que se analizan los documentos para una semilla dada, aunque si influye la semilla inicial.

- Aprovecha la dispersión típica de las matrices de pesos del modelo vectorial.

El método de bisección recursivamente divide un *cluster* en dos, partiendo de un *clusters* formado por todos los elementos, este método obtiene recursivamente tantos *clusters* como se deseen. Este método permite crear una simple pero eficiente variante del *K-Means*, el *Bisecting K-Means*, que da lugar a *clusters* de documentos mejores que los producidos por el método básico del *K-Means*, debido principalmente a que

genera *clusters* de talla relativamente uniforme. Y análogamente a éste, en la tesis se presenta el α-*Bisecting K-Means* (incluido en el apartado 3.3.2 en la página 84) a partir del α-*bisection* (apartado 3.3.1 en la página 82).

Cuando se utiliza como medida de similitud el coseno del ángulo que forman los vectores representantes de los documentos y los centroides de los *clusters*, el algoritmo *K-Means* se convierte en el *Spherical K-Means*, su variante más importante, la cual genera *clusters* cuya forma geométrica es de esfera m dimensional. Lo mismo ocurre cuando hacemos uso como medida de distancia el coseno en el α-*Bisecting K-Means,* obtenemos el α-*Bisecting Spherical K-Means* (apartado 3.3.2 en la página 84), cuya idea subyacente es utilizar el α-*Bisecting K-Means* para obtener una solución inicial aproximada y más tarde refinarla con el *Spherical K-Means*.

La idea subyacente en los métodos de *clustering* basados en densidades es agrupar aquellos elementos que son vecinos, determinando esto mediante una función objetivo que trabaje específicamente sobre densidades. En otras palabras, la idea clave es que los objetos densos se agrupan juntos dentro de un mismo *cluster*, entendiendo que un *cluster* es una región que presenta una densidad de objetos más alta que sus regiones vecinas.

Las principales ventajas de las técnicas basadas en densidades son:

1. El número de parámetros iniciales son pocos y lo más importante, no hay que indicar el número de *cluster* que se van a generar.
2. Se encuentran clusters de tallas y formas arbitrarias.
3. La complejidad de estos métodos suele ser más que aceptable.
4. Pueden ser usados para todo tipo de espacios métricos.
5. Son robustos en relación a los datos sueltos (*outliers*) y filtran el ruido.
6. Han probado ser muy eficientes y efectivos agrupando toda clase de datos.
7. Las versiones paralelas y distribuidas amplían la eficiencia.

El algoritmo DBSCAN [40], representante típico de los métodos basados en densidades, presenta dos parámetros de entrada ϵ, $MinEle$; el primero define la distancia máxima entre dos elementos para considerarlos vecinos, y el segundo define el mínimo número de elementos que deben ser vecinos para formar un cluster. La salida de este algoritmo son *clusters* disjuntos, el número de éstos no es un parámetro, viene determinado por los datos de entrada; también puede dar lugar a un conjunto de elementos definidos como ruido, ya que no se pueden incluir en ninguno de los *clusters* construidos.

Destacan las siguientes características del DBSCAN [6, 40, 62, 66, 111, 112, 113]:

- Fue diseñado par descubrir eficientemente los *clusters* y el ruido de bases de datos.
- Funciona igual de bien para espacios euclídeos de 2 ó 3 dimensiones como para espacios de dimensionalidad alta.

- Busca regiones de densidad alta que están separadas por regiones de densidad baja.
- Encuentra *clusters* con talla y forma arbitraria, pero influida por la función utilizada para el cálculo de la distancia entre dos objetos.
- No es sensitivo al orden de entrada de los objetos.
- No es determinístico en un sentido estricto (la ejecución depende del orden de los objetos), pero tanto el tiempo de ejecución como el resultado (los *clusters* formados) son siempre los mismos.
- Es incremental, ya que los nuevos objetos insertados sólo afectan a un cierto vecindario.
- Es un algoritmo de *clustering* muy eficiente y efectivo, entre otros motivos porque sólo precisa de una evaluación a través de la base de datos.
- Es robusto respecto al ruido.
- Trabaja en cualquier espacio métrico, no sólo en el espacio vectorial.
- Puede ser fácilmente implementado.
- Requiere dos parámetros de entrada: el radio de los *clusters* y el número mínimo de objetos que deben formarlos. No requiere un número predeterminado de *clusters*.
- Presenta una complejidad computacional $\mathcal{O}\left(n^2\right)$, la cual se puede mejorar utilizando índices espaciales a $\mathcal{O}\left(n \log n\right)$.
- No permite trabajar con densidades variables, por lo cual no puede generar *clusters* de distintas densidades.

En esta tesis se ha desarrollado el algoritmo VDBSCAN (apartado 3.4.2 en la página 87), realizando una variación en el algoritmo DBSCAN. De esta forma y al trabajar en recuperación de información sobre colecciones de documentos, se puede minimizar la variabilidad de sus dos parámetros (desconocidos a priori) e incluso eliminarla del todo.

En el algoritmo DBSCAN original el parámetro del mínimo número de elementos se utiliza cada vez que se buscan los vecinos de un elemento. Si el número de vecinos es menor que dicho parámetro entonces ese elemento o se considera ruido o no se tienen en cuenta sus vecinos. Cambiar este comportamiento es la idea principal del VDBSCAN donde el número mínimo de elementos sólo se utiliza para determinar si el cluster construido debe considerarse ruido o *cluster* real. Es decir, se tiene en cuenta siempre el vecindario de un elemento, sea cual sea el tamaño de éste, y cuando ya se han agrupado todos los posibles elementos teniendo en cuenta todos sus vecinos se decide si dicha agrupación da lugar a un *cluster* o no.

Con este cambio en la forma de proceder conseguimos tres aspectos muy importantes:

1. Reducir de dos a una el número de veces que se calcula el vecindario en cada iteración del algoritmo, consiguiendo reducir el coste computacional y por tanto temporal del algoritmo.

2. Poder determinar un valor constante de forma general para el mínimo número de elementos que forman un *cluster*, reduciendo el número de parámetros del método.

3. Poder obtener heurísticamente un buen valor para la distancia mínima entre dos elementos de un mismo *cluster*.

Las prestaciones de recuperación del VDBSCAN, con el número mínimo de elementos fijado a 2 y el valor de la distancia mínima obtenido heurísticamente, comparadas con el DBSCAN, con varios valores para el número mínimo de elementos y con la distancia mínima igual a la utilizada en el VDBSCAN, presentan un comportamiento igual (más detalles en el apartado 3.4.2 en la página 87). Tras el estudio temporal pertinente entre ambos métodos (apartado 3.4.3 en la página 89) destacan dos aspectos muy importantes. Primero, el tiempo del DBSCAN es independiente del parámetro del mínimo número de elementos. Y segundo, el VDBSCAN siempre es considerablemente más rápido que el DBSCAN, incluso llegando a tardar menos de la mitad del tiempo en algunos casos.

Aunque los autores del DBSCAN aportan una solución simple para determinar los parámetros de entrada de dicho algoritmo, ésta sólo es factible para pequeños conjuntos de datos, ya que hay que realizar costosas operaciones para cada dato (k-vecinos más próximos), y además, la evaluación de los parámetros se basa en la visualización de un gráfico limitado a la resolución de las pantalla. Existe el DBCLASD, una variante del DBSCAN que no precisa parámetros de entrada, sin embargo se basa en la asunción de que los elementos dentro de un *cluster* están uniformemente distribuidos, situación que no tiene por que darse en los sistemas de recuperación de información. Además, hay que tener en cuenta que el DBCLASD es más lento que el DBSCAN tendiendo a ser el doble e incluso el triple.

El número mínimo de elementos del VDBSCAN se puede fijar, dejando así de ser un parámetro y mediante el método heurístico presentado en esta tesis se puede elegir un buen valor de proximidad (más detalles en el apartado 3.5 en la página 90). Aunque a priori en la heurística no se establece ninguna cota superior en el número de iteraciones y en cada iteración se debe realizar una evaluación completa de las consultas de prueba, el coste computacional que conlleva es aceptable en la fase de modelización y puesta en marcha de un sistema de recuperación de información. De todas formas se podría plantear estudiar el trabajar con un subconjunto de las consultas de prueba, e incluso con una muestra de la colección.

Capítulo 4

Clustering en sistemas distribuidos

4.1. Introducción

Hay un hecho que está claro, se quiere que las cosas se hagan más veces o más rápido. Ahora bien, por un lado la velocidad de un procesador está limitada por la velocidad de la luz, cuando se haya alcanzado la máxima miniaturización de los microchips. Y por otro el reducir el tamaño de los microchips tiene un coste muy alto, tanto en diseño como en manufacturación [116]. Con lo cual la paralelización se presenta como una importante alternativa para hacer más cosas o hacerlas más rápido.

Concretamente la paralelización se puede utilizar para mejorar los siguientes aspectos [116]:

El tiempo de respuesta: Aunque un sistema secuencial de recuperación de información tenga unas buenas prestaciones, cuando el número de usuarios que quiere utilizarlo simultáneamente es alto, el tiempo de respuesta del mismo se incrementa enormemente. Existen fundamentalmente dos alternativas para mejorar el tiempo de respuesta. Usar métodos más rápidos a costa de peores prestaciones, o utilizar paralelismo y obtener las mismas buenas prestaciones acelerando el tiempo de respuesta individual.

Las grandes cantidades de datos: El tiempo de respuesta de un sistema de recuperación de información pequeño, que maneje poca cantidad de datos, debería ser el mismo que el de un sistema de recuperación de datos grande, que maneje una gran cantidad de datos. En general la escalabilidad de un sistema secuencial es mucho peor que la de un sistema paralelo. Es decir, normalmente un sistema secuencial empeora mucho más y más rápido en sus prestaciones al aumentar la cantidad de datos que maneja que un sistema paralelo.

La gran demanda computacional de los algoritmos: Muchas veces la posibilidad de mejorar las prestaciones utilizando algún otro método no se tiene en cuenta porque las alternativas presentan unos costes computacionales elevados. Y tampoco se plantea la posibilidad a priori de la paralelización, ya que las características del sistema es de un usuario y los datos a manejar son pocos. De esta forma se está perdiendo la posibilidad de mejorar las prestaciones y la solución estaría en utilizar la paralelización, pero para reducir (mejor dicho repartir) los costes computacionales de los métodos alternativos.

Al igual que en la actualidad, la tendencia futura es a que cada vez hayan más usuarios utilizando un mismo sistema, que además, hayan más datos que manejar, y que el coste computacional de los métodos aumente a medida que se quieran contemplar más aspectos para mejorar las prestaciones de recuperación. Todo ello indica claramente la utilidad de aprovechar las características de la paralelización, entendiéndola de la forma más amplia. Incluso, muchas veces, la propia naturaleza del problema obliga a la paralelización (a la distribución) de los datos. Ya que estos se encuentran de por si distribuidos y no es admisible su centralización.

Una vez se tiene claro en que puede ayudar la paralelización hay que entender los sistemas paralelos. Éstos se clasifican básicamente en cuatro tipos distintos:

SISD (*Single Instruction, Single Data*): Un mismo flujo de instrucciones se aplica a un mismo conjunto de datos. Es una estructura Von Newman secuencial.

MISD (*Multiple Instruction, Single Data*): Varios flujos de instrucciones se aplican a un mismo conjunto de datos. Es una estructura muy rara, destacarían los computadores sistólicos.

SIMD (*Single Instruction, Multiple Data*): Un mismo flujo de instrucciones (programa) se aplica a varios conjuntos de datos. A la hora de programar se suele utilizar como interfaz de paso de mensajes el MPI [117, 118].

MIMD (*Multiple Instruction, Multiple Data*): Varios flujos de instrucciones (programas) se aplican a varios conjuntos de datos. A la hora de programar se suele utilizar como interfaz de paso de mensajes el PVM [119, 120].

El concepto "elementos de proceso" (o "unidades de proceso") utilizado en esta tesis se refiere a cualquier unidad con capacidad de procesar información de forma paralela al estar comunicada con otras unidades análogas. Ejemplos claros y comunes son: los procesadores de un multiprocesador, los distintos PCs de un *Cluster* de PCs, los diferentes ordenadores conectados a una red, etc.

Los algoritmos paralelos diseñados en esta tesis están basados en el modelo instrucción simple y múltiples datos (SIMD -*Single Instruction, Multiple Data*-). No se trabaja con arquitecturas de memoria compartida ya que de esta forma existe la posibilidad de escalar fácilmente el algoritmo al número de elementos de proceso que se desee. La implementación de los algoritmos se ha desarrollado en un *cluster* de PCs, principalmente por la disponibilidad y la versatilidad a la hora de hacer las pruebas. Para las comunicaciones se ha elegido como interfaz de paso de mensajes el

MPI [117, 118] debido a que es un estándar y es muy portable (está desarrollado para multitud de plataformas). Los algoritmos desarrollados no son exclusivos para multiprocesadores o *clusters* de PCs, están también pensados para sistemas distribuidos. La principal diferencia a la hora de la implantación en un tipo u otro de sistemas se presenta en las comunicaciones, ya que para los sistemas distribuidos la comunicación debería ser utilizando TCP/IP. La transformación del uso de MPI a TCP/IP es de fácil extrapolación, sólo habría algo de complejidad a la hora de implementar las operaciones de reducción.

La principal diferencia entre un ordenador paralelo y un sistema distribuido es el coste de las comunicaciones entre unidades de proceso, el cual es considerablemente más alto en los entornos distribuidos. En los primeros se tiende hacia un granulado fino, mientras que en los segundos la tendencia es hacia uno granulado grueso [11]. En esta tesis se han desarrollado los algoritmo pensando en sistemas distribuidos, es decir con un alto coste en las comunicaciones.

Existen dos formas de aprovechar el concepto de paralelización dentro de la recuperación de información. Creando nuevos algoritmos cuya naturaleza tienda directamente a la paralelización (por ejemplo, mediante redes neuronales). O adaptando los algoritmos de recuperación de información actuales que se ha confirmado que son buenos. Esta segunda opción es la que se ha elegido en esta tesis, concretamente se van a crear versiones distribuidas del α-*Bisecting Spherical K-Means* (algoritmo 8 en la página 85) y del VDBSCAN (algoritmo 9 en la página 86).

Hay que distinguir cuando se hace una misma consulta sobre varias colecciones distintas y distribuidas de cuando se considera una única colección distribuida. En el primer caso se obtendrían varios resultados, asumiendo como tal un *ranking* de documentos, realmente uno por cada colección. Con lo cual, habrá que unificar en un único *ranking* cada uno de los *rankings* obtenidos, con los consiguientes problemas [4]. En cambio en el segundo caso se obtendría directamente un único *ranking*, lógicamente a cambio de un mayor coste inicial a la hora de modelizar la colección de forma distribuida.

4.2. Posibles distribuciones

Primero de todo hay que decir que la utilización de paralelismo no garantiza la mejora de prestaciones, dependerá de como se distribuyan las tareas y los datos [116]. La distribución elegida determinará el nivel de comunicación entre las unidades de proceso. La distribución de los datos podrá basarse en una granularidad fina, gruesa o variable. Y la distribución de las tareas determinará el paralelismo sobre la consulta, es decir elegir si cada consulta es distribuida entre los procesadores (la misma consulta trabaja en cada unidad de proceso con distintos datos, *Intra-query*), o múltiples consultas se realizan concurrentemente (en cada unidad de proceso trabaja una consulta distinta con todos los datos, *Inter-query*) [11, 116]. El primer caso se podría ver como si en cada elemento de proceso se ejecutase un sistema de recuperación de información separado e independiente, y únicamente habría que añadir un elemento de proceso que determinará a cual mandar la consulta. En este caso los elementos de

proceso no cooperarían al gestionar la consulta, aunque podrían buscarse formas de que compartieran recursos. A medida que se incluyesen más elementos de proceso, se podrían tratar más consultas concurrentemente, pero el tiempo de respuesta de cada consulta individual no cambiaría. En el segundo caso, cada elemento de proceso se encargaría de resolver parcialmente la consulta ya que dispondría de una parte de los datos (de la colección de documentos) y colaborarían todos ellos para dar un resultado final. En este caso la distribución de datos podría seguir básicamente dos líneas, o dividir los términos de la colección o repartir los documentos.

Los dos algoritmos que se van a paralelizar, *α-Bisecting Spherical K-Means* (algoritmo 8 en la página 85) y del VDBSCAN (algoritmo 9 en la página 86), se han desarrollado basándose en una matriz de pesos. Se sabe que existen tres tipos de distribuciones fundamentales cuando la estructura de datos subyacente es una matriz: distribución por filas, por columnas y por bloques. En este caso concreto equivaldría a una distribución por términos de la colección (por filas), una distribución por documentos (por columnas) o una distribución parcial de términos y documentos (por bloques).

Particionar la colección en función de los términos implica gran sobrecarga por comunicaciones durante el proceso de consulta, por ello es raramente empleada en sistemas distribuidos [11]. La operación que más aparece es el producto vector por vector, si la distribución fuese por filas la comunicación de menor coste se implementaría como una reducción con un coste igual al número de columnas. En cambio si la distribución fuese por columnas la comunicación de menor coste se implementaría como una difusión (*Broadcast*) de coste igual al número de filas. Entonces, por un lado es una característica inherente a los sistemas de recuperación de información que el número de filas (términos) es mucho menor al de columnas (documentos), y por tanto la distribución por columnas es de menor coste. Y por otro lado, a la hora de implementar estas comunicaciones en un sistema distribuido y por tanto con TCP/IP es más sencilla la operación de difusión que la de reducción (no olvidar que la situación real más común son colecciones distribuidas entre máquinas conectadas por redes TCP/IP: intranets, internet, etc.).

Particionar la colección por bloques conllevaría también gran sobrecarga de comunicaciones, además, los algoritmos a paralelizar no presentan ninguna característica que favorezca la utilización de bloques, y la división de los documentos daría lugar a una mayor complejidad.

De esta forma, las dos opciones de distribución por filas y por bloques, a priori se podrían descartar. Pero además, la idea es poder trabajar en un sistema distribuido donde la colección de documentos pueda estar formada por subcolecciones reales, las cuales muy posiblemente pueden estar separadas físicamente. Es decir, por definición del problema los documentos pueden estar ya distribuidos. Por lo tanto, la mejor elección es la distribución por documentos (por columnas).

En esta tesis se pretende que la distribución sea transparente al usuario, por lo cual, para éste, habrá una colección lógica global y única. Pero luego, cada elemento de proceso tendrá una subcolección disjunta. Por ello, el usuario podrá obtener como respuesta la referencia a cualquier documento que se encuentre en la colección global, aún cuando dichos documentos se encuentren en alguna subcolección distante.

4.3. α-*Bisecting Spherical K-Means* distribuido

El algoritmo *K-Means* ha sido paralelizado muchas veces (por ejemplo en [51, 52, 53, 54]) partiendo en la mayoría de los casos de un conjunto de datos centralizado o repetido. Es decir, o los datos residen en un servidor central y entonces se distribuyen entre los diferentes elementos de proceso, o cada elemento de proceso presenta una copia de todos los datos. La filosofía seguida en esta tesis parte de que la colección de documentos se encuentra ya distribuida entre los distintos elementos de proceso.

Estudiando un poco el algoritmo *K-Means* en seguida se encuentra una filosofía de paralelización básica y sencilla, operaciones de suma y producto vectoriales en paralelo y una operación de reducción global, la cual también sirve como método de sincronización. Esta es la línea que han seguido en general las versiones paralelas que se encuentran en la literatura (ver por ejemplo en [51, 52, 53]). En esta tesis también se ha elegido esta forma de paralelizar pero los algoritmos que se presentan son versiones paralelas de las modificaciones y mejoras realizadas al *K-Means* desarrolladas en esta tesis, concretamente: α-*Bisecting K-Means* paralelo (algoritmo 13 en la página 113) y α-*Bisecting Spherical K-Means* paralelo (algoritmo 14 en la página 114).

4.3.1. α-*Bisecting K-Means*

En la versión paralela del α-*Bisecting K-Means* (algoritmo 7 en la página 84) desarrollada se ha asumido una distribución por documentos, tal y como muestra la figura 4.1.

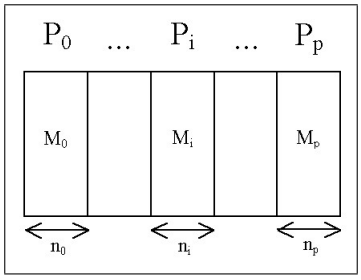

Figura 4.1: Distribución de la colección en el α-*Bisecting K-Means*.

En otras palabras, cada unidad de proceso tiene un pequeño número de documentos de la colección. Pero además, realizando las comunicaciones necesarias entre ellos, todas las unidades de proceso disponen del mismo vector concepto normalizado. Por tanto, es preciso desarrollar un algoritmo paralelo para crear un vector concepto normalizado (algoritmo 11 en la página 111), una versión paralela del α-*Bisection* (algoritmo 12 en la página 112) y finalmente la versión paralela del α-*Bisecting K-Means* (algoritmo 13 en la página 113).

Cuando se busca un *cluster*, éste no se construye explícitamente. En su lugar, se genera un vector que incluye el índice a cada documento incluido en el cluster. Este

vector de índices es usado como parámetro de entrada en el algoritmo paralelo de cálculo del vector concepto normalizado y en la versión paralela del α-*Bisection*.

La figura 4.2 muestra el proceso general seguido a la hora de calcular el vector concepto normalizado en paralelo. Destaca el vector *idx* (*nidx* será la cardinalidad de *idx*), el cual contine los índices de los documentos con los que se va a trabajar, es decir, incluye la referencia de las columnas que intervienen en el cálculo del vector concepto (las que pertenecerían al *cluster* con el que se está trabajando). Localmente cada elemento de proceso suma todas las columnas de la parte de la matriz de pesos que guarda y que su vector de índices le indica, obteniendo su vector concepto local (ncv_i). Luego mediante una reducción global se construye el vector concepto global (*ncv*) a partir de los vectores concepto locales. Por último, cada elemento de proceso normaliza el vector concepto.

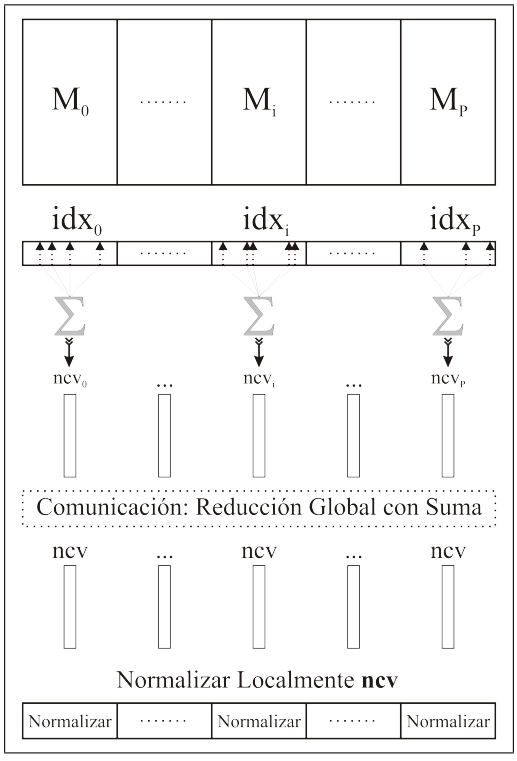

Figura 4.2: Proceso de construcción del vector concepto normalizado en paralelo.

El proceso se presenta detallado en el algoritmo 11 en la página siguiente. La implementación del cálculo de la 2-norma del *ncv* también se realiza en paralelo, concretamente aprovechando que el vector *ncv* se encuentra en todos los elementos de proceso cada uno de ellos realiza una parte del cálculo y luego se utiliza una reducción global para unificarlos.

Algoritmo 11 *Vector concepto normalizado en paralelo*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, $idx_i \in \mathbb{R}^{nidx_i}$.

Salida: $ncv \in \mathbb{R}^m$.

Paso-1: $ncv_i = \sum_{j=1}^{nidx_i} m_{idx_i(j)} \ : \ m_{idx_i(j)} \in M_i$.

Paso-2: Ejecutar la reducción global sumando todos los $nidx_i$ en N. Ahora N, en todas las unidades de proceso, es igual a la suma de todos los $nidx_i$.

Paso-3: Ejecutar la reducción global sumando todos los ncv_i en ncv. Ahora ncv, en todas las unidades de proceso, es igual a la suma de todos los documentos procesados.

Paso-4: $ncv = \frac{ncv}{N}$

Paso-5: $ncv = \frac{ncv}{\|ncv\|}$

En la figura 4.3 se muestra un diagrama representando el comportamiento de la versión paralela del α-*Bisection* presentado en el algoritmo 12 en la página siguiente.

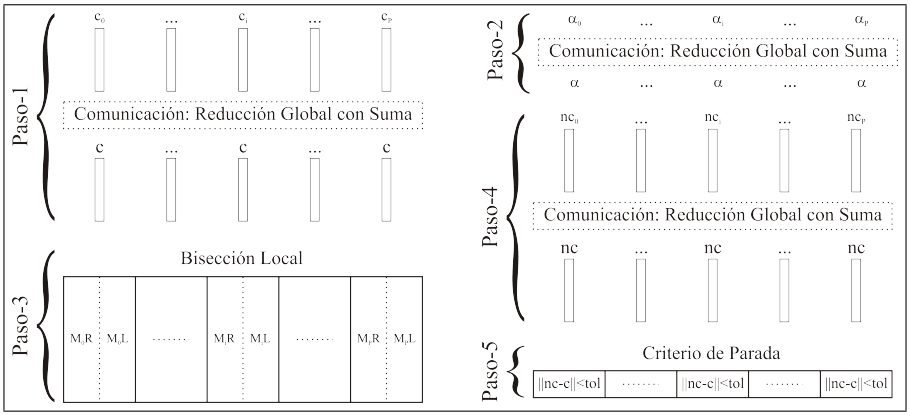

Figura 4.3: Diagrama del α-*Bisection* paralelo.

El método de α-*Bisection* da lugar a dos *clusters*, uno se identifica como izquierdo y el otro como derecho. Ahora bien, cuando el proceso se realiza en paralelo cada elemento de proceso tiene una parte de la colección y obtiene los *clusters* generados de su parte de la colección. Por tanto, luego habría que difundir esta información para que cada elemento de proceso conozca la distribución global de los *clusters* en relación a la colección. En el algortimo 12 en la página siguiente la parte del *cluster* izquierdo de la i-ésima unidad de proceso se denota por πl_i y análogamente la derecha por πr_i.

Algoritmo 12 *α-Bisection paralelo*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, ϵ, *maxiter.*

Salida: $\pi l_i \in \mathbb{R}^{m \times nl_i}$ y $\pi r_i \in \mathbb{R}^{m \times nr_i}$ donde $nl_i + nr_i = n_i$.

Paso-1: Seleccionar un conjunto de documentos locales (crean $idx_i \in \mathbb{R}^{nidx_i}$) y calcular en paralelo el vector concepto normalizado $c \in \Re^m$ (algoritmo 11 en la página anterior).

Paso-2: Calcular α usando la expresión 3.5 en la página 84.

Paso-3: Dividir M_i en dos *subclusters* πl_i y πr_i de acuerdo a:

$$d \in \pi l_i \text{ si } d^{\mathrm{T}} \bullet c \geq \alpha$$
$$d \in \pi r_i \text{ si } d^{\mathrm{T}} \bullet c < \alpha$$

Paso-4: Calcular nc, el nuevo vector concepto normalizado de πl, usando el algoritmo 11 en la página anterior, (idx_i referenciará los documentos que incluye πl_i).

Paso-5: Si el criterio de parada ($\|nc - c\| \leq \epsilon$) ha sido alcanzado o $maxiter == 0$ entonces finalizar, sino decrementar *maxiter*, hacer $c = nc$ e ir al paso-3.

Es importante remarcar dos aspectos importantes. Primero, el cálculo de α se realiza en paralelo (paso-2 del *α-Bisection* paralelo, algoritmo 12). Concretamente la paralelización es alcanzada obteniendo la media y la desviación típica. Cada unidad de proceso trabaja con su propia parte de la colección, entonces todas las unidades de proceso realizan una reducción global añadiendo las sumas parciales locales. Y segundo, la implementación de la 2-norma $\|nc - c\|$ (paso-4, algoritmo 12) es realizada también en paralelo, usando una reducción global, análogamente al cálculo de la 2-norma del *ncv* en el algoritmo 11 en la página anterior.

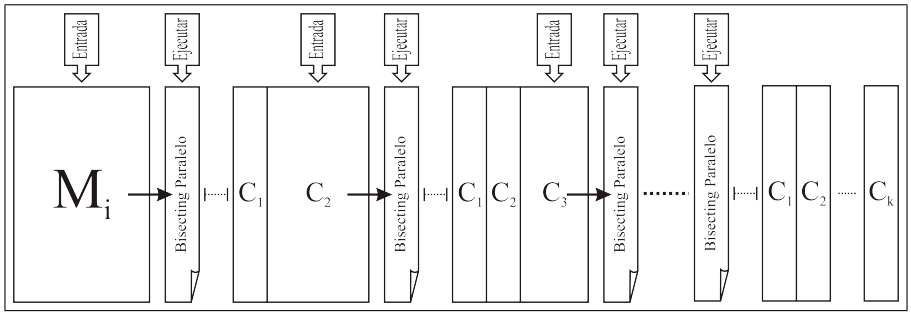

Figura 4.4: Diagrama del *α-Bisecting K-Means* paralelo.

La figura 4.4 en la página anterior presenta un diagrama esquemático del comportamiento del α-*Bisecting K-Means* paralelo (algoritmo 13).

Algoritmo 13 α-*Bisecting K-Means paralelo*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, $k \in \mathbb{N}$.

Salida: k *clusters* disjuntos $\{\pi_j\}_{j=1}^{k}$.

Paso-1: Seleccionar M_i como el *cluster* a dividir, establecer t=1 y calcular α usando la expresión 3.5 en la página 84.

Paso-2: Encontrar los *clusters* πt_i y $\pi t_i^{'}$ usando el α-*Bisection* paralelo, algoritmo 12 en la página anterior.

Paso-3: Construir π_t y π_{t+1} mediante reducción global a partir de πt_i y $\pi t_i^{'}$ respectivamente.

Paso-4: Si t es igual a k-1 entonces devolver todos los *clusters* sino seleccionar π_{t+1} como el *cluster* a dividir, incrementar t e ir al paso-2.

La idea del α-*Bisecting K-Means* paralelo (algoritmo 13) es aprovechar el paralelismo implementado en el α-*Bisection* paralelo (algoritmo 12 en la página anterior), ya que éste es el grueso del paralelismo que presenta. También hay comunicación para obtener la información global de los *clusters* y en el cálculo del α que también se realiza en paralelo, de igual forma a la descrita para el α-*Bisection* paralelo (algoritmo 12 en la página anterior). Ahora bien, cuando el método del α-*Bisection* paralelo se utiliza como parte del α-*Bisecting K-Means* paralelo el cálculo del α sólo se realiaza una vez, en este último. También, es importante destacar que cada unidad de proceso selecciona aleatoriamente un documento de su subcolección, entonces todas las unidades de proceso calculan el vector concepto normalizado de todos los documentos elegidos.

4.3.2. α-*Bisecting Spherical K-Means*

La distribución elegida para el desarrollo de la versión paralela del α-*Bisecting Spherical K-Means* (algoritmo 14 en la página siguiente) es la misma que se ha utilizado en el α-*Bisecting K-Means* paralelo (algoritmo 13). Dicha distribución es por documentos, la cual consiste en que una porción de la colección se encuentra presente en cada una de las unidades de proceso, y se presenta en la figura 4.1 en la página 109.

La evaluación del criterio de parada también se realiza en paralelo. De hecho, la evaluación de la función objetivo se hace en paralelo. Básicamente, cada unidad de proceso calcula la suma parcial con los documentos que contiene, entonces todas las unidades de proceso realizan un reducción global añadiendo la sumas parciales, obteniendo así el resultado final.

Algoritmo 14 *α-Bisecting Spherical K-Means paralelo*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, ϵ, *maxiter*, $k \in \mathbb{N}$.

Salida: k *clusters* disjuntos$\{\pi_j\}_{j=1}^{k}$.

Paso-1: Calcular k *clusters* $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$ aplicando el *α-Bisecting K-Means paralelo*, algoritmo 13 en la página anterior y calcular también en paralelo su vector concepto normalizado $\left\{c_j^{(0)}\right\}_{j=1}^{k}$, algoritmo 11 en la página 111. Inicializar t=0.

Paso-2: Construir una nueva partición $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ inducida por $\left\{c_j^{(t)}\right\}_{j=1}^{k}$ según:

$$\pi_j^{(t+1)} = \left\{x \in M_i : \ x^{\mathrm{T}} \bullet c_j^{(t)} > x^{\mathrm{T}} \bullet c_l^{(t)} \ , \ 1 \leq l \leq k \ , \ j \neq l\right\}, \ 1 \leq j \leq k$$

Paso-3: Calcular en paralelo el nuevo vector concepto normalizado $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$ asociado con la nueva partición mediante el algoritmo 11 en la página 111.

Paso-4: Finalizar si el criterio de parada, expresión 3.4 en la página 78, se cumple o si t es igual a *maxiter*, sino incrementar t e ir al paso-2.

4.4. VDBSCAN distribuido

Hoy en día, gran cantidad de información heterogénea y compleja reside en ordenadores diferentes, trabajando independientemente, los cuales están conectados vía redes de área local o ancha (LAN o WAN). Existen varias razones para que dicha información no pueda ser transmitida a un sitio central, por ejemplo, ancho de banda limitado o aspectos de seguridad [62, 63]. Además, la transmisión de ingentes cantidades de información a un sitio central es en algunas áreas de aplicación casi imposible, debido a la cantidad de información recolectada en poco tiempo, la cual puede ser no sólo excesiva para ser transmitida sino analizada en un único sitio central [62, 63].

4.4.1. Versiones paralelas del DBSCAN

El algoritmo PDBSCAN [61] es una versión paralela del DBSCAN. Este algoritmo parte de un conjunto de datos el cual reside completamente en un servidor central y entonces distribuye los datos entre los diferentes elementos de proceso. El algoritmo PDBSCAN presenta un programa principal (*master*) que inicializa los programas secundarios (*slave*) en cada uno de los elementos de proceso disponibles, y distribuye entre éstos el conjunto de datos.

Cada elemento de proceso genera *clusters* basándose sólo en sus datos locales. Las comunicaciones necesarias se realizan por paso de mensajes. Es el programa principal el que se encarga de balancear la carga dinámicamente y de mezclar los resultados producidos por los distintos elementos de proceso [61].

Los resultados del algoritmo PDBSCAN son bastante buenos según presenta sus autores [61], pero las pruebas han sido realizadas con conjuntos de datos de baja dimensionalidad (máximo 5). De todas formas hay que tener en cuentas que el enfoque del algoritmo precisa una distribución inicial la cual no es una situación natural e incluso inviable en muchos casos, sobre todo por el volumen de datos que debe manejar un único elemento de proceso (el que ejecute el programa principal).

El algoritmo DBDC (*Density Based Distributed Clustering*) [62, 63, 64] es una versión distribuida del DBSCAN. Este algoritmo parte de que la información se encuentra distribuida en diferentes elementos de proceso, que se consideran independientes los unos de los otros. Lo primero que hace es realizar localmente en cada elemento de proceso el *clustering* mediante la utilización del DBSCAN (algoritmo 4 en la página 80), este proceso se realiza independientemente en cada elemento de proceso sin comunicación entre ellos. Entonces se extrae información sobre los *clusters* localmente creados y ésta se envía a un servidor central, siendo el coste de transmisión mínimo al ser dicha información representativa, es sólo una fracción de la información original.

En el servidor central, basándose en la información representativa enviada desde los elementos de proceso locales, se genera un modelo global, el cual se devuelve a los elementos de proceso locales. Esta fase es la más delicada y difícil, ya que pueden existir muchas dependencias entre los datos localizados en los distintos elementos de proceso, que no se han tenido en consideración durante la creación del modelo local.

De todas formas, la fase de generación del modelo global en el servidor central es bastante rápida, ya que no trabaja con todo el conjunto de datos. De hecho hay que decidir qué información es la enviada llegando a un compromiso entre la complejidad y la precisión. Cuanta más información se envíe mayor será la complejidad de realización del modelo global y mayores serán los costes de transmisión. Pero también mayor será la precisión de los resultados. Lo usual es elegir un conjunto de representantes para cada *cluster* encontrado localmente.

El modelo global se genera normalmente analizando los representantes locales, lo cual es similar a ejecutar un nuevo método de *clustering* sobre el conjunto de representantes (una nueva ejecución del DBSCAN). El resultado del *clustering* global es enviado a los elementos de proceso locales, los cuales actualizan sus *clusters* basándose en dicho modelo global generado en el servidor central (por ejemplo, mezclando dos *clusters* locales en uno o asignando ruido local a un *cluster* global).

Según se presenta en [62, 63, 64] los resultados del algoritmo DBDC son bastante buenos, al menos en prestaciones computacionales. De todas formas presenta una carencia importante, aunque en determinados contextos pueda ser aceptada. Los *clusters* resultantes no coinciden con los obtenidos mediante una versión secuencial del propio DBSCAN. Es cierto que la variación entre el contenido de los *clusters* es mínima, pero también es cierto que el conjunto de datos con los que se han hecho las pruebas son bidimensionales, cuyo comportamiento es diferente al de una colección de documentos (vectores m-dimensionales).

4.4.2. Versión distribuida del VDBSCAN

El algoritmo VDBSCAN distribuido está enfocado a la recuperación de información en colecciones de documentos, aunque no está limitado únicamente a dicho campo de acción y debería funcionar en cualquier área al mismo nivel que el DBSCAN. Tal y como se ha argumentado en el apartado 4.2 en la página 107 la distribución elegida es por columnas quedando ésta como se muestra en la figura 4.5.

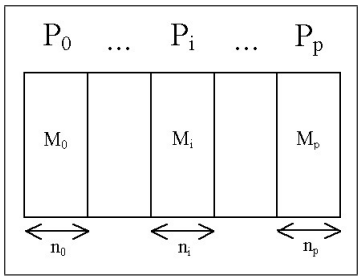

Figura 4.5: Distribución de la colección en el VDBSCAN paralelo.

La sincronización en el VDBSCAN distribuido (algoritmo 15 en la página siguiente) se produce al mantener de forma global la misma información de elementos no usados ($NoUG$), de semillas (SG) y de elementos usados (UG). Y también a la hora de calcular el vecindario de forma paralela, algoritmo 16 en la página siguiente.

Cada elemento de proceso trabaja con la parte de la colección formada por los documentos que almacena. También, cada elemento de proceso, guarda la información global de que documentos ya han sido clasificados, en que *cluster*, cuales todavía no han sido evaluados y cuales lo están siendo. El grueso de la comunicación se da a la hora de mantener esta información y cuando se distribuye el documento del cual se busca el vecindario. Más concretamente en la implementación del algoritmo VDBSCAN paralelo se utilizan operaciones de distribución y de reducción.

En el Paso-2.1 del algoritmo 15 en la página siguiente la elección del elemento aleatorio conlleva una sincronización, ya que todos los elementos de proceso deben usar ese mismo elemento. Luego cuando se empiezan a elegir elementos del conjunto de semillas (SG) Paso-2.3.1 del algoritmo 15 en la página siguiente, vuelve a haber otra sincronización ya que todos los elementos de proceso deben conocer el elemento seleccionado (el documento). La unidad de proceso que guarda dicho documento lo distribuye al resto de elementos de proceso, ya que todos deben calcular cual es el vecindario de éste dentro de sus propios documentos que todavía no han sido usados (Paso-2.3.4 del algoritmo 15 en la página siguiente). Las siguientes comunicaciones son reducciones para actualizar el número global de elementos que pertenecen a cada conjunto: elementos no usados, usados y semillas. Éstas se producen principalmente en el Paso-2.3.5, en el Paso-2.4, en el Paso-2.5 y en el Paso-2.6, ya que cada elemento de proceso actualiza dichos conjuntos en función de los elementos que guarda, pero todos necesitan al menos saber cuantos elementos de forma global se incluyen en cada conjunto.

Algoritmo 15 *VDBSCAN Distribuido*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, ϵ, $MinEle \in \mathbb{N}$.

Salida: k *clusters* disjuntos $\{\pi_j\}_{j=1}^{k}$, R (Ruido)

Paso-1: $NoUG = M_i$ y $j = 1$

Paso-2: Mientras existan elementos en $NoUG$

Paso-2.1: Incluir en SG, un elemento aleatorio de $NoUG$ (Sincronización, el mismo en todos los procesadores).

Paso-2.2: Inicializar $UG = \{\}$

Paso-2.3: Mientras existan elementos en SG

Paso-2.3.1: Elegir $x \in SG$ (Sincronización, el mismo x en todos los procesadores).

Paso-2.3.2: $UG = UG \cup \{x\}$

Paso-2.3.3: $NoUG = NoUG - SG$

Paso-2.3.4: $V = PEpsVecindario(x, NoUG, \epsilon)$ (cálculo en paralelo de los elementos de $NoUG$ que están a una distancia ϵ de x, según el algoritmo 16).

Paso-2.3.5: $SG = (SG \cup V) - UG$

Paso-2.4: Si $|UG| < MinEle$ Entonces $R = R \cup UG$

Paso-2.5: Si $|UG| \geq MinEle$ Entonces $\pi_j = UG$ y $j = j + 1$

Paso-2.6: $NoUG = M_i - \{\pi_l\}_{l=1}^{j} - R$

Algoritmo 16 *PEpsVecindario*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, ϵ, x y $NoUG \subseteq M$.

Salida: $V \subseteq M_i$.

Paso-1: Inicializar $V = \{\}$

Paso-2: Para toda columna $y \in M_i$ hacemos

Paso-2.1: Si $y \in NoUG$ y $x^t * y < \epsilon$ entonces $V = V \cup \{y\}$

Para que el algoritmo 16 en la página anterior se entienda mejor y sea más clara su notación se ha supuesto que la matriz M y las respectivas submatrices M_i se comportan como conjuntos de documentos.

En el algoritmo 16 en la página anterior se incluye la versión paralela del cálculo del vecindario de un elemento. Ésta está basada directamente en la versión del cálculo de vecindarios incluida en el artículo original donde se presento el DBSCAN [40]. Como se puede observar la paralelización no conlleva ninguna comunicación, puesto que cada elemento de proceso únicamente debe realizar productos vectoriales con los datos que ya posee y con un vector que recibe como entrada. Aunque este algoritmo no precise comunicaciones, no hay que olvidar que el vector de entrada x en la mayoría de los elementos de proceso se ha recibido tras una comunicación (ver algoritmo 15 en la página anterior).

4.5. Evaluación distribuida de consultas

Una vez realizado el desarrollo distribuido de los métodos de *clustering* también hay que paralelizar el proceso de evaluación. El método de evaluación (presentado en el algoritmo 17) es común a todos los métodos de *clustering* ya que trabaja únicamente con la colección, la consulta y los *clusters* construidos, independientemente de cómo se hayan creado éstos.

Algoritmo 17 *Evaluación de Clusters*

Entrada: $M \in \mathbb{R}^{m \times n}$, $q \in \mathbb{R}^m$, $\{\pi_j\}_{j=1}^k$ y $\{c_j\}_{j=1}^k$ (según la ecuación 3.1 en la página 78).

Salida: Un *ranking* R de los documentos.

Paso-1: Seleccionar el centroide (y *cluster* asociado) más cercano a la consulta q, el que maximice la ecuación 2.2 en la página 46.

$$c_h \ ; \ c_h^t \cdot q \geq c_j^t \cdot q, \ \ j = 1, ..., k \ , \ j \neq h$$

Paso-2: Calcular la relevancia de cada documento del *cluster* seleccionado ($d_l \in \pi_h$) respecto a la consulta q usando el coseno (ecuación 2.2 en la página 46).

$$Rel_l = d_l^t \cdot q \ ; \ \forall d_l \in \pi_h$$

Paso-3: Calcular el *ranking* R ordenando los documentos según su relevancia Rel_l.

Los centroides se construyen en la fase de modelización, una vez creados los diferentes *clusters*. Su cálculo es obtener la media de una serie de documentos (ver ecuación 3.1 en la página 78). En su versión paralela los documentos se encuentran distribuidos. El proceso es básicamente que cada elemento de proceso suma los documentos que guarda de cada uno de los *clusters*, luego mediante una operación de

reducción todos los elementos de proceso obtienen la suma de todos los documentos de cada *cluster*. Otra operación de reducción permite que todos los elementos de proceso conozcan el número total de documentos por *cluster*. Luego independientemente, cada elemento de proceso hace el resto de operaciones necesarias, incluyendo el cálculo de normas.

La versión distribuida de la evaluación de los *clusters* (algoritmo 17 en la página anterior) se presenta en el algoritmo 18.

Algoritmo 18 *Evaluación distribuida de Clusters*

Nota: Este código es ejecutado por la i-ésima unidad de proceso.

Entrada: $M_i \in \mathbb{R}^{m \times ni}$, $q \in \mathbb{R}^m$, $\{\pi_j\}_{j=1}^k$ y $\{c_j\}_{j=1}^k$ (según la ecuación 3.1 en la página 78).

Salida: Un *ranking* R global de los documentos.

Paso-1: Seleccionar el centroide (y *cluster* asociado) más cercano a la consulta q, el que maximice la ecuación 2.2 en la página 46.

$$c_h \ ; \ c_h^t \cdot q \geq c_j^t \cdot q, \quad j = 1, ..., k \ , \ j \neq h$$

Paso-2: Calcular la relevancia de cada documento local del *cluster* seleccionado ($d_l \in M_i$ y $d_l \in \pi_h$) respecto a la consulta q usando el coseno (ecuación 2.2 en la página 46).

$$Rel_l = d_l^t \cdot q \ ; \ \forall d_l \in \pi_h$$

Paso-3: Difundir las relevancias parciales para obtener las relevancias globales.

Paso-4: Calcular el *ranking* R ordenando los documentos del *cluster* según su relevancia Rel_l.

En el Paso-3 del algoritmo 18 cada elemento de proceso tiene las relevancias de los documentos que guarda, pero necesita las relevancias de todos los documentos. Por tanto, cada elemento de proceso difunde sus relevancias al resto de elementos de proceso y recibe las relevancias del resto, construyendo así la relación de relevancias globales. Éstas se usarán en el siguiente paso para generar el *ranking* de documentos.

4.6. Conclusiones

Hay un hecho que está claro, se quiere que las cosas se hagan más veces o más rápido. Ahora bien, mediante las mejoras tecnológicas hay un límite más o menos cercano, ya que la velocidad de un procesador está limitada por la velocidad de la luz, con lo cual la paralelización se presenta como una importante alternativa. Concretamente la paralelización se puede utilizar para mejorar aspectos como: el tiempo

de respuesta, las grandes cantidades de datos o la gran demanda de computación de algunos algoritmos.

Al igual que en la actualidad, la tendencia futura es a que cada vez hayan más usuarios utilizando un mismo sistema, que además, hayan más datos que manejar, y que el coste computacional de los métodos aumente a medida que se quieran contemplar más aspectos para mejorar las prestaciones de recuperación. Incluso, muchas veces, la propia naturaleza del problema obliga a la paralelización (a la distribución) de los datos. Ya que estos se encuentran de por si distribuidos y no es admisible su centralización.

Los algoritmos paralelos diseñados en esta tesis están basados en el modelo instrucción simple y múltiples datos (SIMD -*Single Instruction Multiple Data*-). La implementación de los algoritmos se ha desarrollado en un *cluster* de PCs. Para las comunicaciones se ha elegido como interfaz de paso de mensajes el MPI al ser éste un estándar y muy portable. Los algoritmos desarrollados no son exclusivos para multiprocesadores o *clusters* de PCs, están también pensados para sistemas distribuidos. La transformación del uso de MPI a TCP/IP es de fácil extrapolación, sólo habría algo de complejidad a la hora de implementar las operaciones de reducción. La principal diferencia entre un ordenador paralelo y un sistema distribuido es el coste de las comunicaciones entre unidades de proceso, el cual es considerablemente más alto en los entornos distribuidos. En esta tesis se han desarrollado los algoritmo pensando en sistemas distribuidos, es decir con un alto coste en las comunicaciones. Concretamente se han desarrollado las versiones distribuidas del α-*Bisecting Spherical K-Means* (Algoritmo 8 en la página 85) y del VDBSCAN (Algoritmo 9 en la página 86).

Existen tres tipos de distribuciones fundamentales cuando la estructura de datos subyacente es una matriz: distribución por filas (por términos de la colección), por columnas (por documentos) y por bloques (parcial de términos y documentos). En los algoritmos desarrollados la operación que más aparece es el producto vector por vector. Si la distribución fuese por filas la comunicación de menor coste se implementaría como una reducción con un coste igual al número de columnas. En cambio si la distribución fuese por columnas la comunicación de menor coste se implementaría como una difusión (*Broadcast*) de coste igual al número de filas. Entonces, por un lado es una característica inherente a los sistemas de recuperación de información que el número de filas (términos) es mucho menor al de columnas (documentos), y por tanto la distribución por columnas es de menor coste. Y por otro lado, a la hora de implementar estas comunicaciones en un sistema distribuido y por tanto con TCP/IP es más sencilla la operación de difusión que la de reducción. Particionar la colección por bloques conllevaría también gran sobrecarga de comunicaciones, además, la división de los documentos daría lugar a una mayor complejidad. Pero además, la idea es poder trabajar en un sistema distribuido donde la colección de documentos pueda estar formada por subcolecciones reales, las cuales muy posiblemente pueden estar separadas físicamente. Es decir, por definición del problema los documentos pueden estar ya distribuidos. Por lo tanto, la mejor elección es la distribución por documentos (por columnas) y es la que se ha utilizado a lo largo de toda la tesis.

α-*Bisecting Spherical K-Means* distribuido

El algoritmo *K-Means* ha sido paralelizado muchas veces, partiendo en la mayoría de los casos de un conjunto de datos centralizado (posteriormente se distribuyen entre los diferentes elementos de proceso) o repetido (cada elemento de proceso presenta una copia de todos los datos). La filosofía seguida en esta tesis parte de que la colección de documentos se encuentra ya distribuida entre los distintos elementos de proceso.

Estudiando un poco el algoritmo *K-Means* en seguida se encuentra una filosofía de paralelización básica y sencilla, operaciones de suma y producto vectoriales en paralelo y una operación de reducción global, la cual también sirve como método de sincronización. Esta es la línea que han seguido en general las versiones paralelas del *K-Means* que se encuentran en la literatura, en esta tesis también se ha elegido esta forma de paralelizar pero con las variantes desarrolladas en esta tesis.

En la paralelización del α-*Bisecting Spherical K-Means* cada unidad de proceso tiene un subconjunto de documentos de la colección y una copia del mismo vector concepto normalizado. Por tanto, es preciso desarrollar un algoritmo paralelo para crear un vector concepto normalizado, una versión paralela del α-*Bisection* y finalmente la versión paralela del α-*Bisecting K-Means*. Todas estas versiones paralelas suponen además el cálculo en paralelo de la 2-norma (distribuyendo el vector y usando una reducción global), el cálculo también en paralelo de α (se utiliza una reducción global de las sumas parciales locales) y la evaluación paralela del criterio de parada (la función objetivo se calcula parcialmente en cada unidad de proceso y luego éstas se reducen globalmente).

VDBSCAN distribuido

Principalmente existen dos versiones paralelas del DBSCAN: el PDBSCAN y el DBDC. Pero ambas, además, de haberse probado únicamente con datos de dimensionalidad baja, presentan una carencia fundamental, los *clusters* se generan basándose únicamente en información local. No dan lugar a los mismos *clusters* que presentaría la versión secuencial del DBSCAN, ahí es donde aparece la necesidad de desarrollar el VDBSCAN distribuido.

El algoritmo PDBSCAN parte de un conjunto de datos residente en un servidor central y entonces los distribuye entre los diferentes elementos de proceso. Cada uno de los cuales genera *clusters* basándose sólo en sus datos locales. Las comunicaciones necesarias se realizan por paso de mensajes. Es el programa principal el que se encarga de balancear la carga dinámicamente y de mezclar los resultados producidos por los distintos elementos de proceso. Según sus autores presenta buenos resultados, pero los datos de las pruebas son de baja dimensionalidad (máximo 5). De todas formas hay que tener en cuenta que el enfoque del algoritmo precisa una distribución inicial la cual no es una situación natural e incluso inviable en muchos casos.

El algoritmo DBDC (*Density Based Distributed Clustering*) es una versión distribuida del DBSCAN que parte de que la información se encuentra distribuida en diferentes elementos de proceso. Éstos realizan local e independientemente el *clustering* mediante el DBSCAN. Entonces se extrae información sobre los *clusters* localmente creados y se envía a un servidor central que genera un modelo global basándose en

ella (una nueva ejecución del DBSCAN), dicho modelo se devuelve a los elementos de proceso. Según los autores los resultados del algoritmo DBDC son bastante buenos, al menos en prestaciones computacionales (las pruebas se realizaron con datos bidimensionales).

En la versión distribuida del VDBSCAN cada elemento de proceso trabaja con la parte de la colección que almacena y guarda la información global de que documentos ya han sido clasificados, en que *cluster*, cuales todavía no han sido evaluados y cuales lo están siendo (estos puntos sirven de sincronización). El grueso de la comunicación se da a la hora de mantener esta información, ya que conllevan difusiones o reducciones, y cuando se distribuye el documento del cual se busca el vecindario.

La versión paralela del cálculo del vecindario de un elemento está basada directamente en la versión del cálculo de vecindarios incluida en el artículo original del DBSCAN [40]. Dicha paralelización no conlleva ninguna comunicación, puesto que cada elemento de proceso únicamente debe realizar productos vectoriales con los datos que ya posee y con el vector que recibe como entrada. Pero no hay que olvidar que el vector de entrada se ha recibido tras una comunicación.

Evaluación distribuida de consultas

El método de evaluación de consultas es común a todos los métodos de *clustering* ya que trabaja únicamente con la colección y los *clusters* construidos, independientemente de cómo se hayan creado éstos. En su versión distribuida cada elemento de proceso trabaja con sus documentos locales obteniendo las relevancias de éstos. Ahora bien, para construir el *ranking* global precisa de las relevancias de todos los documentos, por ello se hace una distribución de las relevancias locales (paso diferenciador de la versión secuencial).

Capítulo 5

Estudios experimentales

5.1. Colecciones de prueba y entornos utilizados

Con el objetivo de verificar el comportamiento de los métodos desarrollados en este trabajo, se ha hecho uso de tres colecciones de prueba distintas. Todas ellas consisten en colecciones de documentos, con un conjunto de consultas de prueba.

Las colecciones presentan muy distintas dimensiones (desde 425 documentos hasta más de 226000) y también muy diferentes temáticas (desde la política, pasando por la fibrosis cística, hasta resúmenes de energía).

Los estudios experimentales se han realizado en dos *clusters* de PCs: Kefren y Odin (los resultados expuestos son principalmente de Odin). Ambas máquinas están formadas por nodos biprocesador y presentan internamente una red con topología en malla. El sistema operativo que las gestiona es Linux y destacan las librerias numéricas BLAS, LAPACK y SCALAPCK, y la librería de comunicaciones MPI.

5.1.1. Colección: *Times Magazine* 1963

La colección *"Times Magazine 1963"* contiene 425 artículos escritos en inglés, todos extraidos del "*Times Magazine*" de 1963 y cuyo tema principal versa sobre noticias de política internacional. Viene acompañada de 83 consultas de prueba, los documentos relevantes a dichas consultas. Más detalles en el apartado 2.5.5 en la página 62.

Una vez construida la matriz de pesos, como modelo de referencia y base para los métodos de *clustering*, ésta presenta las siguientes dimensiones: 6545 filas (términos) por 425 columnas (documentos). Y la dispersión de la matriz de pesos es de 0.027017.

5.1.2. Colección: *Cystic Fibrosis Database*

La colección "*Cystic Fibrosis Database*" trata sobre aspectos de fibrosis cística, está escrita íntegramente en inglés y los 1239 documentos que la forman fueron publicados entre 1974 y 1979. También incluye 100 consultas con sus respectivos documentos relevantes. Ver más detalles en el apartado 2.5.5 en la página 62.

Una vez construida la matriz de pesos, como modelo de referencia y base para los métodos de *clustering*, ésta presenta las siguientes dimensiones: 3518 filas (términos) por 1239 columnas (documentos). Y la dispersión de la matriz es de 0.013698.

5.1.3. Colección: TREC-DOE

La colección "TREC-DOE" incluye resúmenes del departamento de energía de EEUU, lógicamente utiliza el idioma inglés. Concretamente contiene 226087 documentos y 200 consultas de prueba con la consiguientes referencias a los documentos relevantes a las mismas. Ver más detalles en el apartado 2.5.5 en la página 62.

Una vez construida la matriz de pesos, como modelo de referencia y base para los métodos de *clustering*, ésta presenta las siguientes dimensiones: 87417 filas (términos) por 226087 columnas (documentos). Y la dispersión de la matriz es de 0.00053025.

5.1.4. *Cluster* de PCs: KEFREN

Kefren consta de 20 nodos biprocesadores Pentium Xeon a 2 Ghz con 1 Gigabyte de memoria RAM. Debido a ciertas reconfiguraciones el *cluster* a pasado a 16 nodos, interconectados mediante una red SCI con topología de Toro 2D en malla de 4x4. La red se reconfigura automáticamente en caso de fallo de algún nodo.

Utiliza como sistema operativo un Linux CentOS 5.3 con kernel 2.6.18. Tiene instalados los compiladores: gcc, g++ version 4.1.2 y g77 version 3.4.6. Las librerias numéricas que ofrece son: BLAS Version 3.0-37, LAPACK Version 3.0-37 y SCALAPACK Version 1.8; y la librería de comunicaciones MPI que presenta es la NMPI 1.3.3.1 (implementación completa del estándar MPI-2, optimizada para la red SCI).

5.1.5. *Cluster* de PCs: ODIN

Odin consta de 51 nodos biprocesadores Intel(R) Xeon(TM) CPU a 2.80GHz, interconectados mediante una red SCI con topología de Toro 2D en malla de 10x5. Cada nodo consta de 2 Gigabyte de memoria RAM. El nodo principal es el punto de acceso al cluster y el resto de nodos (50) están disponibles para cálculo cientifico. La red se reconfigura automáticamente en caso de fallo de algún nodo.

Utiliza como sistema operativo un Linux CentOS 5.3 con kernel 2.6.18. Tiene instalados los compiladores: gcc, g++ version 4.1.2 y g77 version 3.4.6. Las librerias numéricas que ofrece son: BLAS Version 3.0-37, LAPACK Version 3.0-37 y SCALAPACK Version 1.8; y la librería de comunicaciones MPI que presenta es la NMPI 1.3.3.1 (implementación completa del estándar MPI-2, optimizada para la red SCI).

5.2. Comparación de calidad de recuperación

En este apartado se va a presentar la comparación entre los distintos métodos desarrollados tomando de referencia los resultados obtenidos directamente con la matriz de pesos. Dicha comparación se centra la calidad de recuperación, estudiando posteriormente otros aspectos también importantes, como el tiempo de computación.

Las medidas en las que se basan los estudios son la precisión y la cobertura (*recall*), concretamente se ha utilizado la curva para 11 niveles estándares que compara ambas. La elección se fundamenta en ser uno de los métodos más ampliamente usado, su sencillez, su claridad visual y su amplio rango de estudio.

5.2.1. Matriz de Pesos vs. *Spherical K-Means*

En este apartado se presentan las prestaciones del método *Spherical K-Means* (algoritmo 3 en la página 79) junto con las prestaciones de la matriz de pesos como referencia. Se incluyen cuatro gráficos por colección de prueba (figuras 5.1 y 5.2 en la página siguiente para la *Times* y la *Cystic* respectivamente), mostrando cada uno el comportamiento con dos, cuatro, ocho y dieciséis *clusters*. De esta forma se puede observar la evolución del comportamiento del método según el número de *clusters* elegido. En las pruebas realizadas también se estudiaron valores intermedios, los cuales siguen el comportamiento que se ve con los valores presentados.

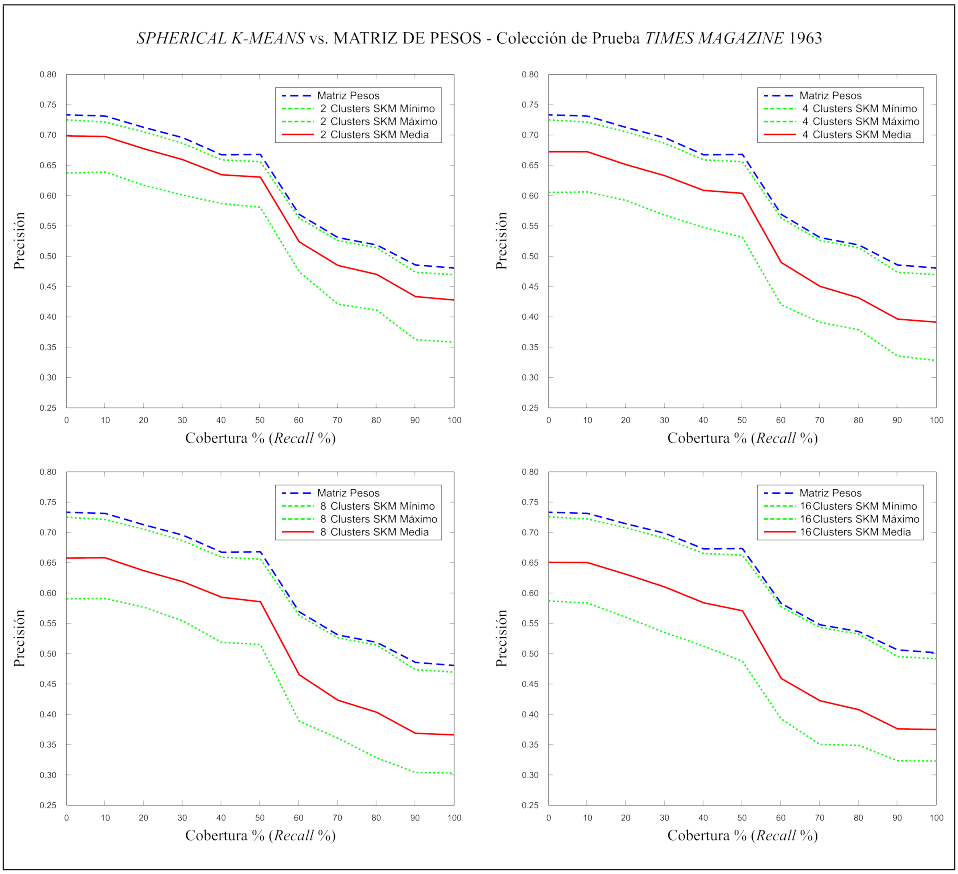

Figura 5.1: Comportamiento del *Spherical K-Means* con la colección *Times*.

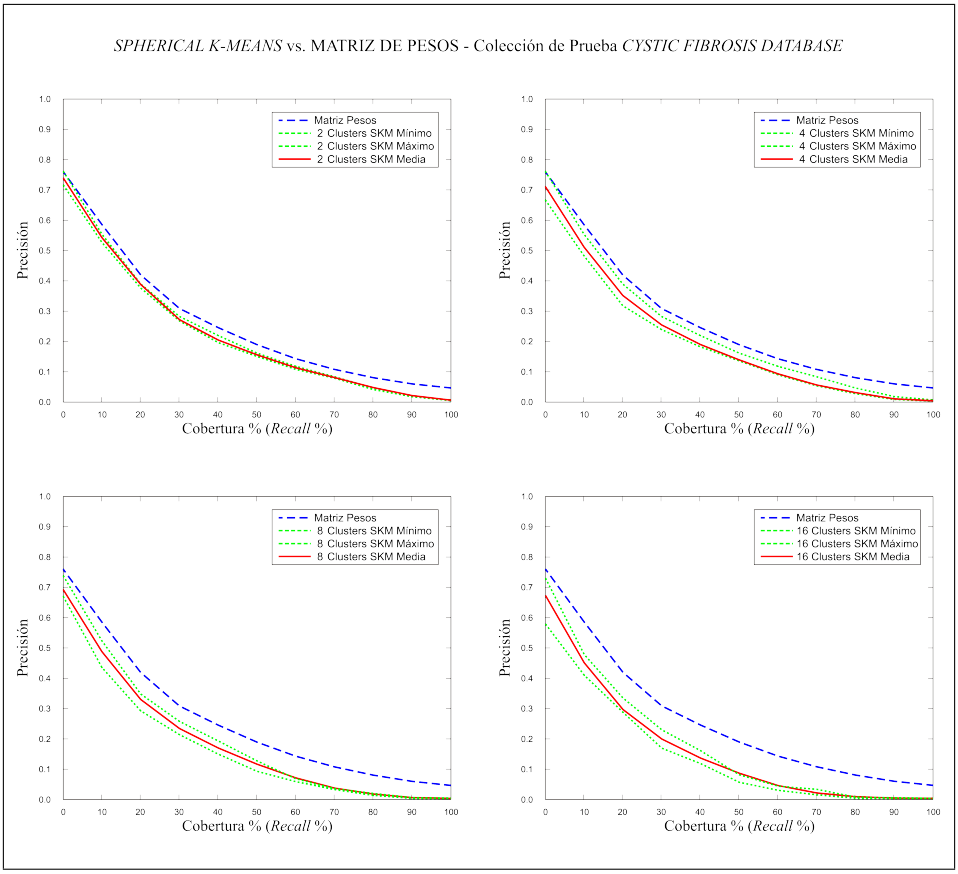

Figura 5.2: Comportamiento del *Spherical K-Means* con la colección *Cystic*.

Hay que tener en cuenta que el método *Spherical K-Means* depende mucho de la solución inicial elegida, es decir, de los centroides iniciales seleccionados. Por ello cada gráfico muestra cuatro curvas distintas, una corresponde a la matriz de pesos, como curva de referencia, y tres correspondientes al método de estudio. Dos de éstas son de acotación y corresponden al mejor y peor caso de todos los estudiados (un mínimo de cien pruebas). La otra presenta el comportamiento medio de todas las pruebas.

En la figura 5.1 en la página anterior, estudio realizado con la colección *Times*, se observa claramente que al aumentar el número de *clusters* generados la calidad de la recuperación empeora, excepto, como es lógico, la de la curva de máxima calidad que es prácticamente igual a la curva de la matriz de pesos, aunque siempre un poco menor. De ello podemos deducir que generar más de dos *clusters* es contraproducente.

Aunque generemos sólo dos *clusters* la calidad de recuperación será peor que con la matriz de pesos. Dependiendo de la suerte o la buena elección de la inicialización del algoritmo (si tuviésemos información previa) la calidad aumentará o disminuirá. Pero a no ser que afinemos mucho en general la calidad empeorará significativamente.

El comportamiento de las curvas obtenido con la colección *Cystic* (figura 5.2 en la página anterior) es análogo a los resultados presentados al utilizar la colección *Times* (figura 5.1 en la página 125). A medida que aumenta el número de *clusters* a generar empeora la calidad de recuperación, pero además, en este caso también se observa que la curva máxima baja de prestaciones significativamente sobre la matriz de pesos.

Con la colección *Cystic* el comportamiento del método no está muy influenciado por la selección inicial. De hecho las curvas media, máxima y mínima están prácticamente solapadas. Esto se explica en parte por el comportamiento tipo de las curvas, el cual ya se observa en la matriz de pesos: las prestaciones empeoran rápidamente a medida que aumenta la cobertura. Este comportamiento es intrínseco a la colección y a las consultas de prueba. Aunque la inicialización del algoritmo no influye de forma significativa sí se observa que las prestaciones finales obtenidas siempre son peores a las presentadas por la matriz de pesos. Eso sí, a diferencia de la colección *Times* dicho empeoramiento es bastante bajo, incluso insignificante generando dos *clusters*.

El hecho que en ambos casos (figuras 5.1 en la página 125 y 5.2 en la página anterior para la colección *Times* y la colección *Cystic* respectivamente) el aumento del número de *clusters* a generar provoque que la calidad de recuperación se vea menoscabada sugiere pensar que dicho comportamiento puede deberse más a características propias del método *Spherical K-Means* que a aspectos propios de las colecciones. De todas formas aventurar dicha conclusión sin más datos ni más estudios con otras distintas colecciones es muy arriesgado.

En todos los casos la curva media está por debajo de la matriz de pesos, y la curva de calidad máxima también. Por lo tanto habría que estudiar que ventajas ofrece el *Spherical K-Means* respecto a la matriz de pesos que pudiesen justificar esa pérdida de calidad. La principal ventaja que puede argumentar el *Spherical K-Means* es mejorar las prestaciones computacionales, sobre todo la mejora de la velocidad de consulta. Es cierto que inicialmente, en la modelización, el coste computacional es mayor, pero al realizarse éste una única vez se puede despreciar y no considerar.

Mediante este método se han generado dos *clusters* tanto en la colección *Times* como en la *Cystic* con lo cual a la hora de realizar una consulta sólo se ha de evaluar los documentos de un *cluster*. Con lo cual en media se podría decir que se evaluarían la mitad de documentos por consulta respecto a la matriz de pesos. Dicha mejora en el tiempo de respuesta ante una consulta puede ser importante cuando la colección es muy grande, siendo el empeoramiento de prestaciones de recuperación aceptable. Decir que con las dos colecciones de prueba presentadas, compensaría la mejora del tiempo de respuesta respecto al empeoramiento en la calidad de recuperación.

5.2.2. Matriz de Pesos vs. α-*Bisecting Spherical K-Means*

Este punto presenta las prestaciones del método α-*Bisecting Spherical K-Means* (algoritmo 3 en la página 79) junto con las prestaciones de la matriz de pesos como referencia. Al igual que en las prestaciones del *Spherical K-Means,* se incluyen cuatro gráficos por colección de prueba (figuras 5.1 en la página 125 y 5.2 en la página anterior para la *Times* y la *Cystic* respectivamente), cada uno de los cuales muestra el comportamiento del α-*Bisecting Spherical K-Means* generando dos, cuatro, ocho y

dieciséis *clusters*. De esta forma se puede observar la evolución del comportamiento del método según el número de *clusters* elegido.

Hay que tener en cuenta que el método α-*Bisecting Spherical K-Means*, al igual que el *Spherical K-Means*, depende mucho de los centroides iniciales elegidos. Y de la misma forma, cada gráfico muestra cuatro curvas distintas, una para la matriz de pesos, como curva de referencia, y tres más relacionadas directamente con el método de estudio. Dos de éstas son acotaciones y muestran las prestaciones mínimas y máximas de todas las pruebas realizadas (un mínimo de cien). La otra presenta el comportamiento medio de todas las pruebas.

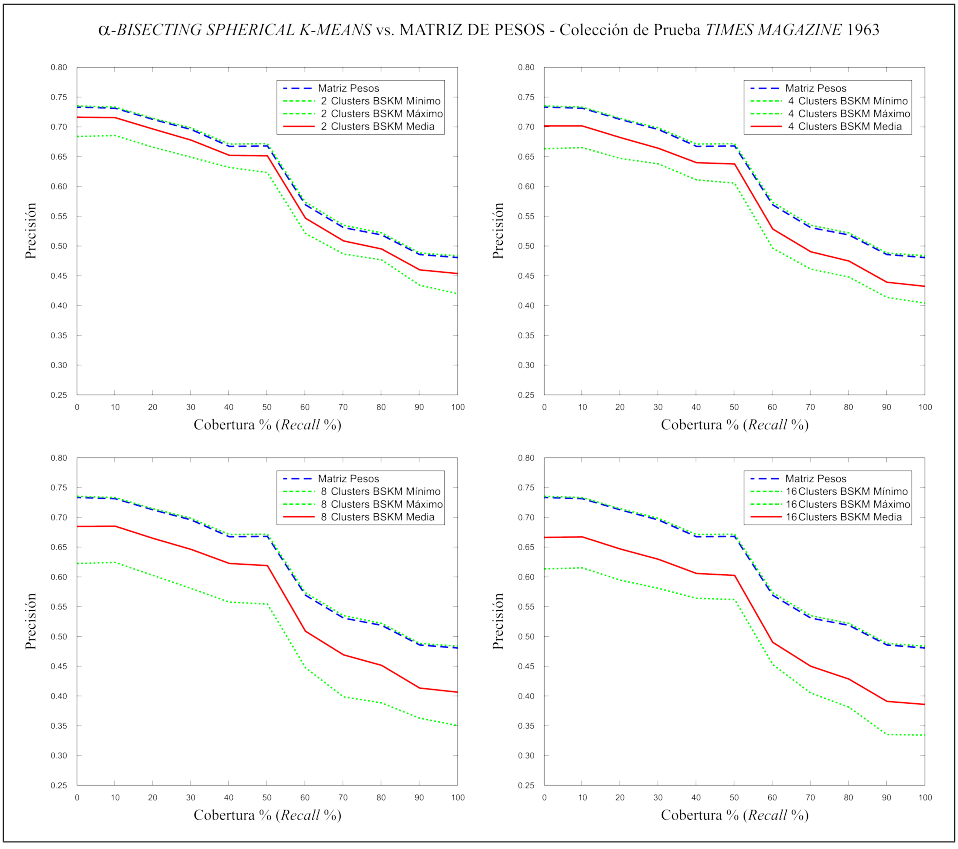

Figura 5.3: Comportamiento del α-*Bisecting Spherical K-Means*, colección *Times*.

En general el comportamiento del α-*Bisecting Spherical K-Means* con la colección *Times* (ver figura 5.3) es análogo al del *Spherical K-Means* (ver figura 5.1 en la página 125). Pero destacan tres aspectos importantes que se pueden resumir en uno, la calidad en recuperación de información mejora y además en todos los casos, tanto a nivel de *clusters* generados como de curvas mínima, máxima y media. El primer aspecto destacado es que la curva máxima es prácticamente igual que la curva

obtenida con la matriz de pesos, incluso se podría decir que la mejora. El segundo, la diferencia entre la curva mínima y la máxima es mucho menor que con el *Spherical K-Means*, el resultado no depende tanto de la solución inicial elegida. Y el tercero, y posiblemente más importante, es que la curva media mejora considerablemente con el *α-Bisecting Spherical K-Means* respecto al *Spherical K-Means*. También hay que decir, que generando dos *clusters* es cuando se obtienen mejores resultados en la recuperación de información.

El comportamiento del *α-Bisecting Spherical K-Means* con la colección *Cystic* (ver figura 5.4) sigue los mismos pasos que con la colección *Times*, pero las diferencias respecto al *Spherical K-Means* (ver figura 5.2 en la página 126) destacan menos, al tener un comportamiento mucho más regular. De hecho prácticamente no se aprecian diferencias de un caso al otro, sólo un poco en las curvas mínima y máxima que se ven mucho más cercanas a la curva media.

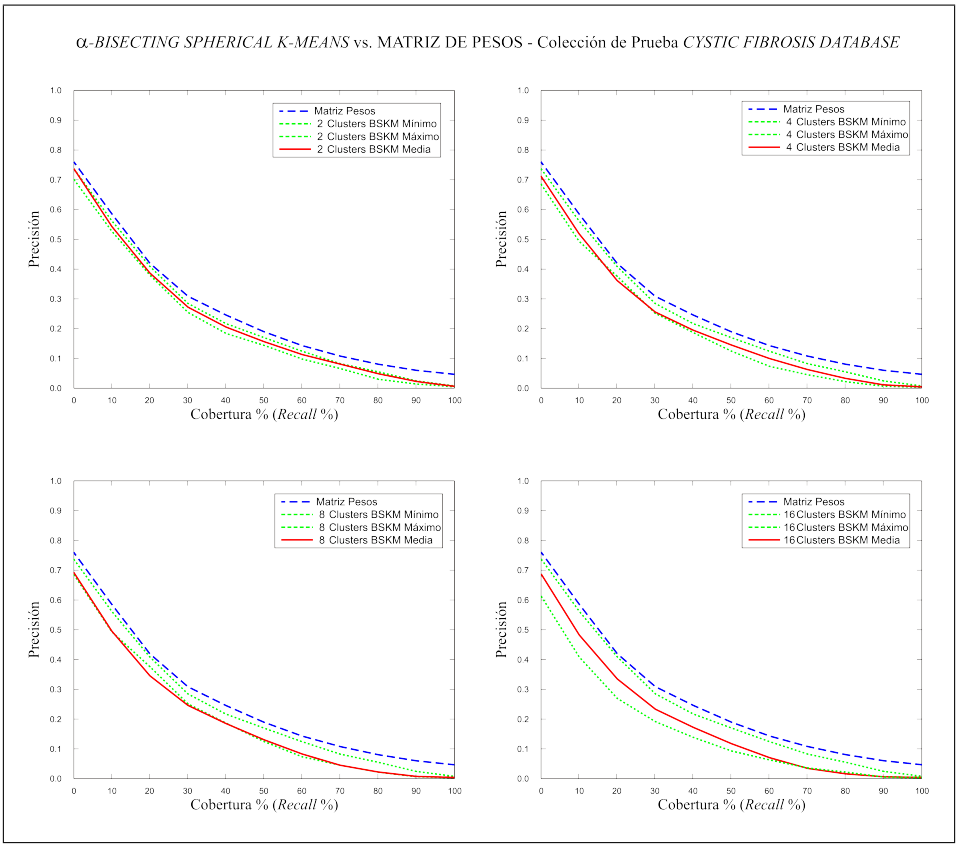

Figura 5.4: Comportamiento del *α-Bisecting Spherical K-Means*, colección *Cystic*.

5.2.3. *Spherical K-Means* vs. α-*Bisecting Spherical K-Means*

Del estudio del comportamiento del *Spherical K-Means* y del α-*Bisecting Spherical K-Means* se han seleccionado los mejores casos según el número de *clusters* seleccionados y se han incluido todos en un mismo gráfico (figura 5.5 y 5.6 en la página siguiente para las colecciones *Times* y *Cystic* respectivamente) para poder compararlos con mayor facilidad. En ambos casos se ha elegido las curvas medias, presentadas al generar dos *clusters* ya que son los casos óptimos para ambos métodos.

En el estudio con la colección *Times* (figura 5.5) se observa claramente que el método α-*Bisecting Spherical K-Means* presenta mejores prestaciones que el *Spherical K-Means* a lo largo de toda la curva. Ambos algoritmos son bastante regulares y siguen, aunque con peores prestaciones, la misma línea de comportamiento que la curva descrita por la matriz de pesos. Aunque se han presentado las curvas medias, según se observó en las figuras 5.1 en la página 125 y 5.3 en la página 128, el rango en el que se mueve el método α-*Bisecting Spherical K-Means* es mucho menor que el del *Spherical K-Means*. Por lo tanto la variación entre las curvas presentadas y las que realmente se obtendrían variarán mucho menos con el α-*Bisecting Spherical K-Means*.

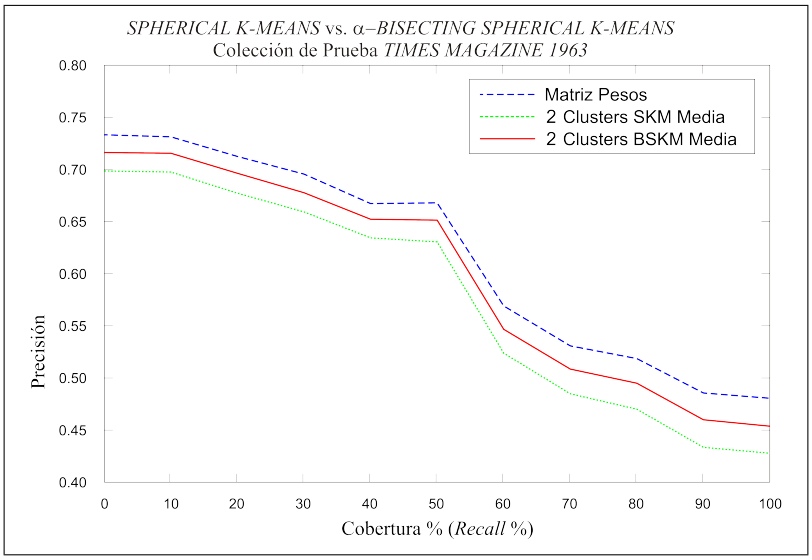

Figura 5.5: *Spherical K-Means* vs. α-*Bisecting Spherical K-Means*, con la *Times*.

Por otro lado el estudio con la colección *Cystic* (figura 5.6 en la página siguiente) da muy poca información, puesto que ambos métodos presentan diferencias entre sí despreciables con las curvas medias. Lo que sí está claro es que siempre están por debajo de las prestaciones de recuperación ofrecidas por la matriz de pesos. Como se observa en las figuras 5.2 en la página 126 y 5.4 en la página anterior la diferencia entre ambos métodos se aprecia en el rango de comportamiento, fundamentalmente entre la curva máxima y la mínima, pero así y todo estas diferencias son mínimas.

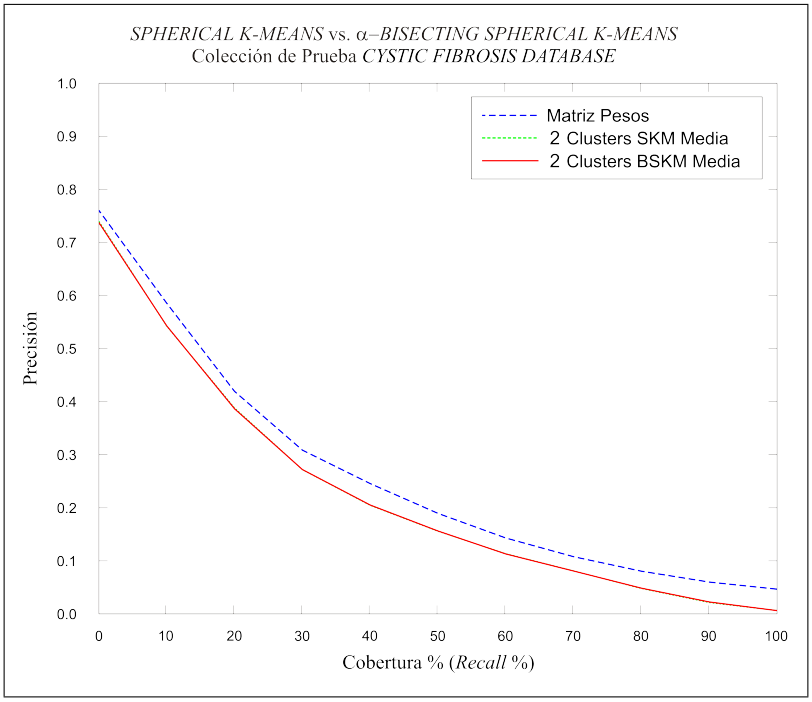

Figura 5.6: *Spherical K-Means* vs. *α-Bisecting Spherical K-Means*, con la *Cystic*.

5.2.4. Matriz de Pesos vs. VDBSCAN

En este apartado se presenta la comparación de comportamiento entre la matriz de pesos y el VDBSCAN (algoritmo 9 en la página 86 seleccionando el parámetro ϵ mediante el algoritmo 10 en la página 97). El estudio se ha realizado comparando las prestaciones de recuperación de información con la colección *Times* y la *Cystic* (figuras 5.7 en la página siguiente y 5.8 en la página siguiente respectivamente).

Con la colección *Times* (figura 5.7 en la página siguiente) se observa que el algoritmo VDBSCAN tiene un comportamiento con prácticamente las mismas prestaciones de recuperación de información que las obtenidas con la matriz de pesos. De hecho en algunos momentos la mejora y en otros la empeora, pero siempre con diferencias sutiles. Destaca que en la zona de cobertura inicial el VDBSCAN presenta menos prestaciones, pero enseguida (tras el 10 % de cobertura) ya ha igualado la matriz de pesos. Luego, a partir del 50 % de recuperación definitivamente mejora las calidad de recuperación de información.

El comportamiento del VDBSCAN con la colección *Cystic* (figura 5.8 en la página siguiente) también es prácticamente igual que el de la matriz de pesos. Aunque excepto en el momento inicial que la mejora de forma casi inapreciable, el resto la empeora pero con una diferencia despreciable (al final del todo, a partir del 90 % de cobertura, dicha diferencia se hace más evidente pero en esa zona es bastante indiferente).

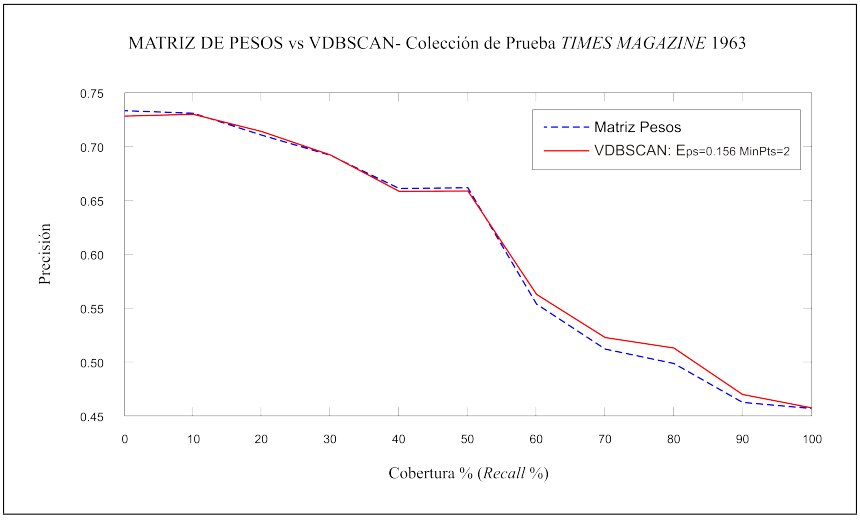

Figura 5.7: Comparación entre la matriz de pesos y el VDBSCAN, colección *Times*.

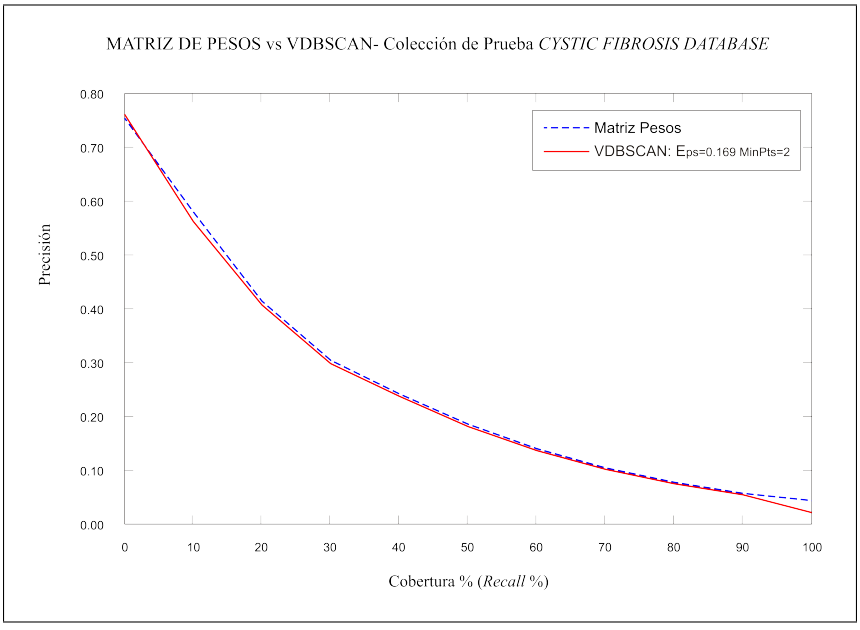

Figura 5.8: Comparación entre la matriz de pesos y el VDBSCAN, colección *Cystic*.

En ambos casos, con las dos colecciones utilizadas, se observa que el comportamiento del VDBSCAN habiendo seleccionado de forma adecuada sus parámetros, tal y como se presenta en el apartado 3.4 en la página 85, es tan similar al com-

portamiento de la matriz de pesos que podría perfectamente sustituirlo sin apreciar diferencias significativas en la calidad de la recuperación de información. Ahora bien, habrá que valorar otros aspectos también importantes, sobre todo las prestaciones computacionales.

Aunque el estudio computacional se trata más adelante hay un aspecto claro que se puede valorar analizando superficialmente ambos modelos. Utilizando la matriz de pesos se deben comprobar todos los documentos de la colección en cada consulta del usuario. Por el contrario con el VDBSCAN, al ser un método de *clustering*, la comprobación se limitará a los documentos de un *cluster*, que siempre son menos que la colección entera. De esta forma el coste computacional de la consulta se vería reducido en tiempo mediante el método VDBSCAN, y es obvio que el tiempo de respuesta ante el usuario es una característica fundamental. Además, el coste añadido en la modelización del VDBSCAN es bastante pequeño y al producirse ésta una única vez es por lo tanto aceptable.

5.2.5. α-*Bisecting Spherical K-Means* vs. VDBSCAN

Una vez comprobado el comportamiento individual de los dos métodos presentados en esta tesis (α-*Bisecting Spherical K-Means* y VDBSCAN respecto al modelo generado con la matriz de pesos, apartados 5.2.2 en la página 127 y 5.2.4 en la página 131 respectivamente), se realiza en este apartado la comparación entre ambos métodos nuevamente respecto a la matriz de pesos.

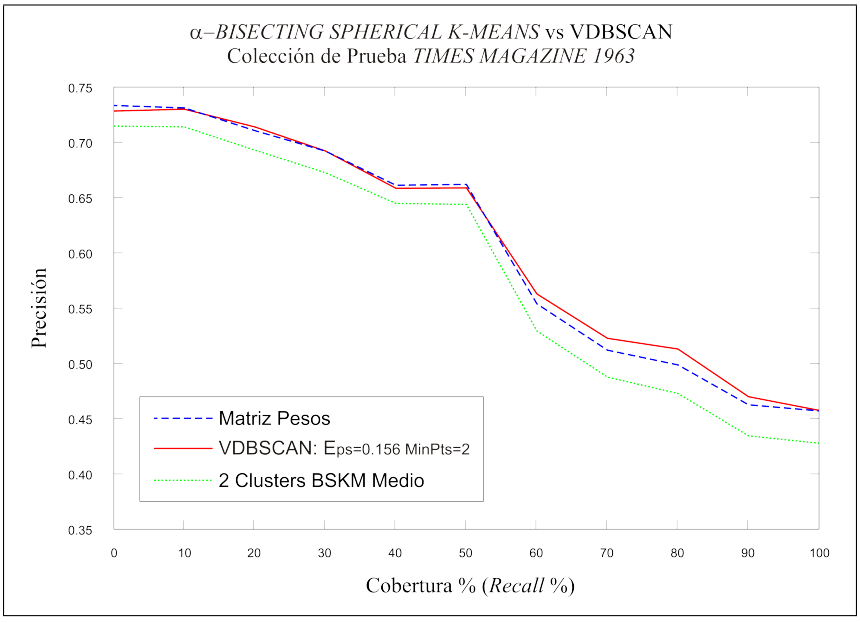

Figura 5.9: α-*Bisecting Spherical K-Means* vs. VDBSCAN, colección *Times*.

En la figura 5.9 en la página anterior se presenta el comportamiento de ambos métodos en relación a la colección *Times*. Y en la figura 5.10 se presentan ambos métodos respecto a la colección *Cystic*. Cuando se utiliza cualquiera de las dos colecciones de prueba, se observa claramente que el VDBSCAN supera en todos los aspectos al *α-Bisecting Spherical K-Means*, al menos en su curva media. De todas formas, si se comparase con la curva máxima también se podría apreciar que lo superaría.

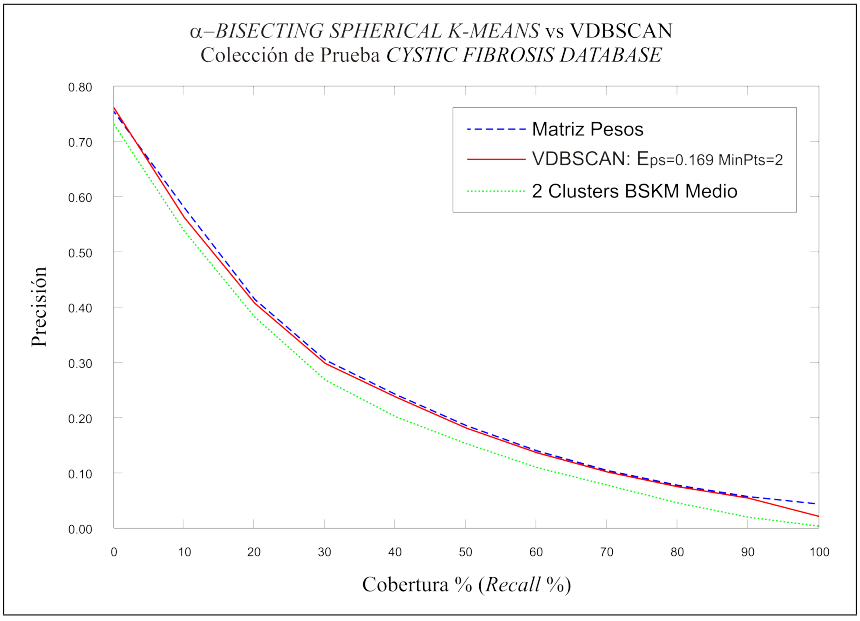

Figura 5.10: *α-Bisecting Spherical K-Means* vs. el VDBSCAN, colección *Cystic*.

5.3. Comparación de prestaciones computacionales

La evaluación de los algoritmos secuenciales la hemos realizado valorando la calidad de recuperación de información principalmente, y el coste temporal del algoritmo de forma superficial. Pero con los algoritmos paralelos esto no es suficiente, hay que valorar también otros factores destacando entre ellos el estudio de la reducción de tiempo a medida que se incrementa el número de unidades de proceso (evaluado estudiando el *speed up*) y el estudio de la relación entre los recursos utilizados y la mejora que estos suponen (evaluado estudiando la eficiencia).

El *speed up* se define como el ratio del tiempo de ejecución de un algoritmo en un procesador por el tiempo de ejecución en p unidades de proceso (relación entre el mejor tiempo secuencial y el tiempo paralelo). Es un resumen de la mejora temporal de un algoritmo paralelo cuando involucramos más unidades de proceso. Es decir, para una talla concreta del problema el *speed up* captura el decremento en el tiempo de ejecución que puede ser obtenido por el incremento del numero de procesadores.

$$S = \frac{\text{Mejor tiempo secuencial}}{\text{Tiempo paralelo}}$$

El valor ideal del *speed up* sería el número de elementos de proceso utilizados en la ejecución paralela, pero en la práctica es inalcanzable mientras no se pueda descomponer el problema secuencial en tantas subtareas como elementos de proceso hayan, evitando así los tiempos de sobrecarga necesarios para realizar la versión paralela (las comunicaciones necesarias).

La eficiencia se define como el ratio del *speed up* por el número de unidades de proceso utilizadas. Siendo, por tanto, un resumen de la calidad de utilización de nuevos recursos (unidades de proceso). La eficiencia capturaría el nivel de aprovechamiento de los recursos utilizados.

$$E = \frac{SpeedUp}{\text{Unidades de Proceso}}$$

El valor ideal de la eficiencia es 1, el 100 % de los recursos nuevos es utilizado en la mejora. Dicha eficiencia sólo se podría dar con un *speed up* perfecto, pero en situaciones reales es prácticamente inalcanzable un *speed up* ideal.

En definitiva cada caso de estudio se ha evaluado en función del coste temporal, del *speed up*, y de la eficiencia. Las prestaciones en recuperación de información son las mismas que la versión secuencial, ya que ambas dan lugar a los mismos *clusters*.

Todas los resultados presentados se han obtenido como medias de comportamiento ante diversas pruebas y repeticiones de las mismas para evitar posibles influencias puntuales. Por ejemplo, los tiempos de respuesta ante la evaluación son respecto a todas las consultas de prueba que las colecciones aportan, al igual que las curvas de precisión vs. cobertura (*recall*) también son medias de todas las consultas de prueba.

El algoritmo de evaluación de las consultas (ver apartado 4.5 en la página 118) es el mismo para todos los métodos de *clustering*, por lo cual se podría suponer que los costes computacionales de evaluación serían iguales independientemente del método de modelización. Dicha suposición es errónea, ya que en función de la modelización se generaran *clusters* de diferente talla y de diferente distribución, aspectos ambos que influyen directamente en el tiempo de respuesta de la fase de evaluación de consultas.

A modo de ejemplo ilustrativo se presenta en la tabla 5.1 los distintos métodos analizados con el número de documentos que incluye cada *cluster* generado y la distribución de las consultas de prueba, todo ello en base a la colección *Times*:

Spherical K-Means		
Documentos por *cluster*	201	224
Consultas por *cluster*	37	49

α-Bisecting Spherical K-Means		
41	384	Documentos por *cluster*
10	73	Consultas por *cluster*

	VDBSCAN (15 *clusters*)														
Documentos por *cluster*	324	12	3	2	9	5	4	2	5	2	2	2	2	2	4
Consultas por *cluster*	67	5	0	0	2	2	0	1	4	0	2	0	0	0	0

Tabla 5.1: Análisis de influencia de la modelización en los costes de evaluación.

Como se observa en la tabla 5.1 el método VDBSCAN genera muchos más *clusters* que los métodos *Spherical K-Means* y el *α-Bisecting Spherical K-Means*. Lógicamente

a la hora de evaluar una consulta requiere mayor tiempo decidir que *cluster* seleccionar entre 15 candidatos que entre 2 (hay que comparar la consulta con 15 centroides o sólo 2). También se puede ver en la tabla 5.1 en la página anterior que cada método genera los *clusters* con tallas diferentes. Obviamente no es lo mismo evaluar una consulta en un *cluster* de 384 documentos (el α-*Bisecting Spherical K-Means* evalúa 73 consultas) que en uno de 12 documentos (el VDBSCAN evalúa 5 consultas).

5.3.1. Modelización: VDBSCAN vs. α-*Bisecting Spherical K-Means* vs. *Spherical K-Means*

En este apartado se presenta el estudio temporal comparativo de la modelización de los dos métodos desarrollados en la tesis en relación al *Spherical K-Means*. También se incluye el estudio del *speed up* y de la eficiencia. Todos los estudios se presentan para las tres colecciones de prueba que se usan en esta tesis (ver apartado 5.1 en la página 123). No se ha incluido la matriz de pesos puesto que se supone que ésta es el punto de partida para las distintas modelizaciones.

Toma de Tiempos

Del estudio temporal de la modelización se pueden destacar tres aspectos importantes, independientemente de la colección de prueba utilizada. Primero, la modelización mediante el VDBSCAN es enormemente más costosa que el resto. Segundo, el *Spherical K-Means* y el α-*Bisecting Spherical K-Means* tienen un coste temporal muy parecido, pero el α-*Bisecting Spherical K-Means* es potencialmente más mejorable con paralelización. Y tercero, la mejora paralela se aprecia fundamentalmente en el VDBSCAN y cuando la colección empleada es de tamaño considerable.

Con la colección *Times* (figura 5.11 en la página siguiente) al ser pequeña, las diferencias temporales son también pequeñas. Además, la mejora en la paralelización sólo se da con hasta dos unidades de proceso, a partir de las cuales los beneficios se empiezan a perder, llegando incluso a empeorar los tiempos. Destaca también, que el VDBSCAN presenta una mejoría sustancial con pocas unidades de proceso pero al aumentar el número de éstas se producen perdidas considerables, debido a los costes de comunicación.

Con la colección *Cystic* (figura 5.12 en la página siguiente) se obtienen resultados prácticamente iguales que con la colección *Times*, debido principalmente a que ambas tienen un tamaño pequeño. Eso sí, como la colección *Cystic* presenta una talla algo superior (de 425 a 1239 documentos) destacan algo más los aspectos descritos con la colección *Times*.

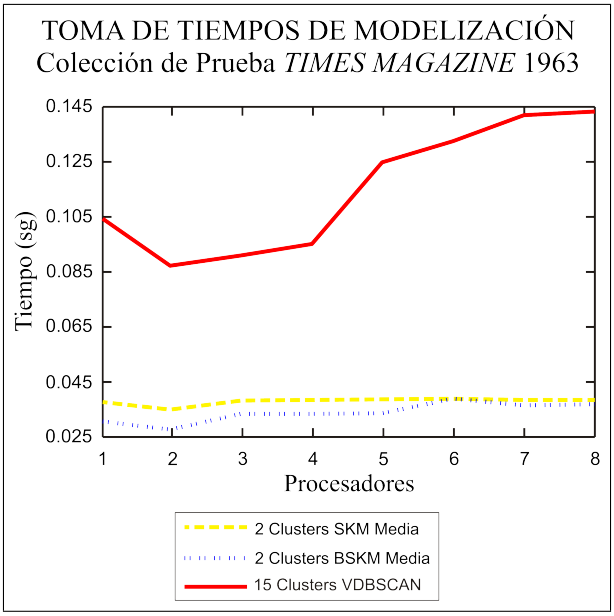

Figura 5.11: Comparación de toma de tiempos de modelización entre el *Spherical K-Means,* el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Times*.

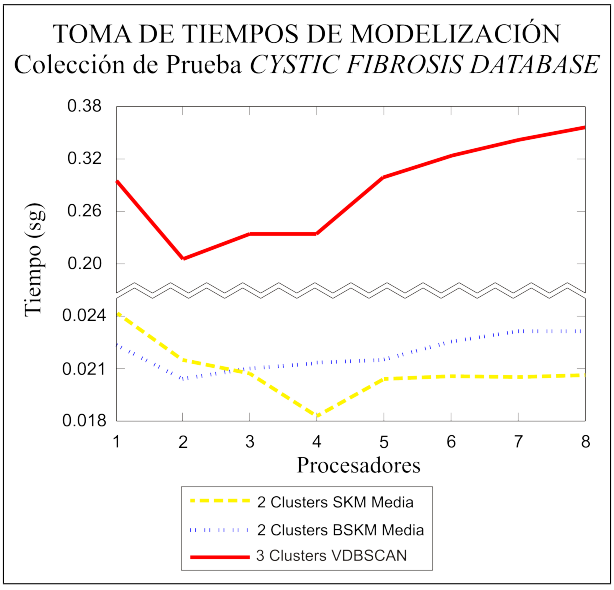

Figura 5.12: Comparación de toma de tiempos de modelización entre el *Spherical K-Means,* el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Cystic*.

La colección DOE (figura 5.13) conlleva un aumento del tamaño muy considerable (alcanzando unos 226000 documentos) y por lo tanto los resultados son más característicos, más claros y más extremos. El coste temporal en valores absolutos se diferencia de manera desmesurada, sobre todo el del método VDBSCAN. Ahora bien, la mejora de prestaciones con el paralelismo también es enormemente mejor en el VDBSCAN. También se observa con mucha claridad que los tres métodos mejoran con el paralelismo, al menos al utilizar hasta ocho unidades de proceso. Además, esta colección nos permite destacar otro aspecto importante, cuanto más elaborado es el método utilizado más costosa computacionalmente es su fase de modelización.

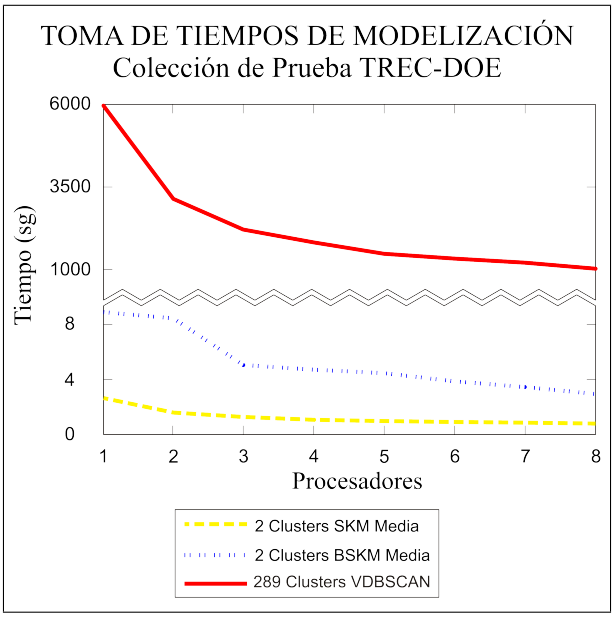

Figura 5.13: Comparación de toma de tiempos de modelización entre el *Spherical K-Means,* el α-*Bisecting Spherical K-Means* y el VDBSCAN con la colección DOE.

Estudio del *Speed Up* y de la Eficiencia

A partir del análisis del *speed up* y de la eficiencia con las tres colecciones de prueba se puede deducir, más o menos, lo esperable. Con las colecciones *Times* y *Cystic* los resultados del *speed up* son malos, hay una mínima mejora pero despreciable y enseguida un empeoramiento sustancial. Ello es debido al pequeño tamaño de las colecciones. Razón que también provoca que ninguno de los métodos obtenga una eficiencia aceptable a partir de dos unidades de proceso. En cambio al utilizar una colección de un tamaño ya considerable, la DOE, el comportamiento del *speed up* sufre una mejora continua, obteniéndose sólo buenos valores con el método VDBSCAN. En la eficiencia también mejoran las prestaciones plausiblemente, llegando con el VDBSCAN siempre a eficiencias superiores al 70 %.

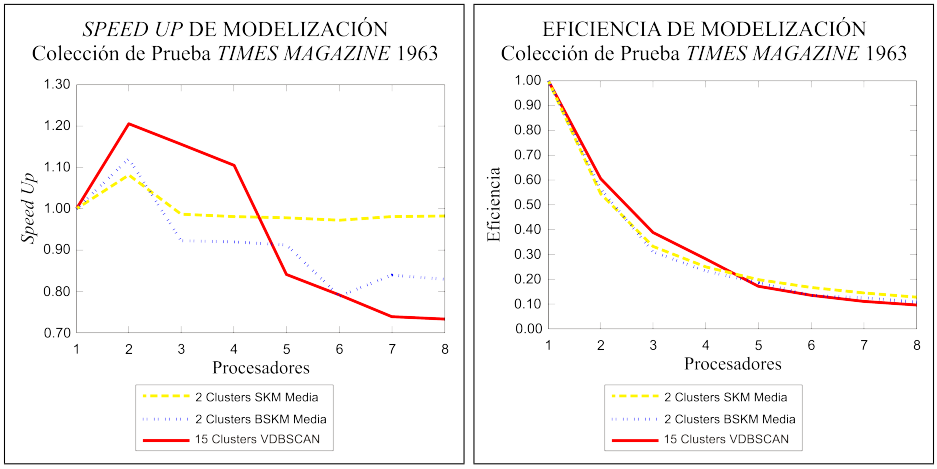

Figura 5.14: Comparación de *speed up* y eficiencia de modelización entre el *Spherical K-Means,* el α*-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Times*.

En el estudio del *speed up* con la colección *Times* (figura 5.14) destaca que es utilizando dos procesadores dónde parece que éste quiera mejorar, pero a partir de ahí el *speed up* decae alcanzando valores bastante malos. Por su lado, la eficiencia presenta una caída bastante pronunciada, aunque con dos procesadores consigue mantenerse cerca de un 60 %, la bajada luego continúa hasta alcanzar valores cercanos a la ineficiencia. Este comportamiento es común a los tres curvas obtenidas por los algoritmos comparados, siendo las diferencias entre unos y otros métodos prácticamente despreciables.

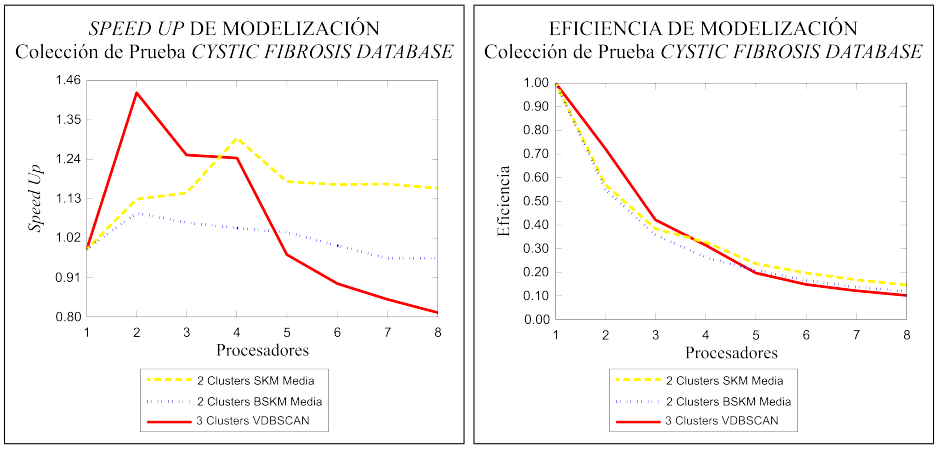

Figura 5.15: Comparación de *speed up* y eficiencia de modelización entre el *Spherical K-Means,* el α*-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Cystic*.

Al utilizar la colección *Cystic* (figura 5.15 en la página anterior) los resultados mejoran algo más, llegan a valores más altos y caen más tarde, así y todo los resultados alcanzados son malos. Nuevamente debido al tamaño reducido del problema.

Cuando ya utilizamos una colección de mayor embergadura, la DOE (figura 5.16), los resultados ya son más que aceptables, llegando incluso a ser bastante buenos con el algoritmo VDBSCAN. Al ser la colección de un tamaño ya considerable los algoritmos pueden aprovechar mejor la paralelización y eso se observa en los resultados.

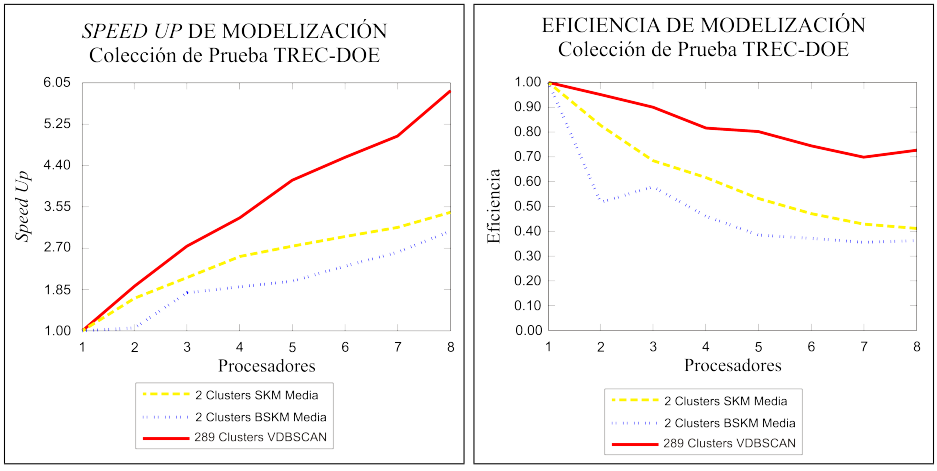

Figura 5.16: Comparación de *speed up* y eficiencia de modelización entre el *Spherical K-Means,* el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *DOE*.

Hasta ocho procesadores los tres métodos mejoran su *speed up* y el VDBSCAN además de mejorar continuamente obtiene valores cercanos a los óptimos con dos y tres procesadores (2 procesadores 1.90 y 3 procesadores 2.70), y alejándose progresivamente de los valores óptimos a medida que aumentan las unidades de proceso.

El *Spherical K-Means* y el *α-Bisecting Spherical K-Means* presentan eficiencias medio-altas hasta los tres o cuatro procesadores y a partir de ahí mantienen la eficiencia algo baja (rondan el 40 %). Por su lado el VDBSCAN mantiene buenos valores de eficiencia hasta las ocho unidades de proceso, no bajando en ningún caso del 70 %.

5.3.2. Evaluación: Matriz de Pesos vs. *Spherical K-Means* vs. *α-Bisecting Spherical K-Means*

En este apartado se presenta el estudio temporal comparativo de la evaluación de consultas mediante el *Spherical K-Means* y el *α-Bisecting Spherical K-Means* usando como referencia el coste temporal de la matriz de pesos. También se presentan los estudios del *speed up* y de la eficiencia. Todos los estudios se presentan para las tres colecciones de prueba que se utilizan a lo largo de esta tesis (ver apartado 5.1 en la página 123).

Toma de Tiempos

En general, independientemente de las colecciones, los métodos *Spherical K-Means* y *α-Bisecting Spherical K-Means* presentan en todo momento un tiempo de evaluación menor que la matriz de pesos, siendo siempre más rápido el *Spherical K-Means*. Ahora bien, ninguno de los dos métodos responde con mejoras claras a la paralelización.

En la figura 5.17, se observa claramente como con la colección *Times* se manifiesta el comportamiento general. Ambos métodos son más rápidos que la matriz de pesos utilizada como referencia. Además, la paralelización del *Spherical K-Means* es nula y con el *α-Bisecting Spherical K-Means* muy reducida.

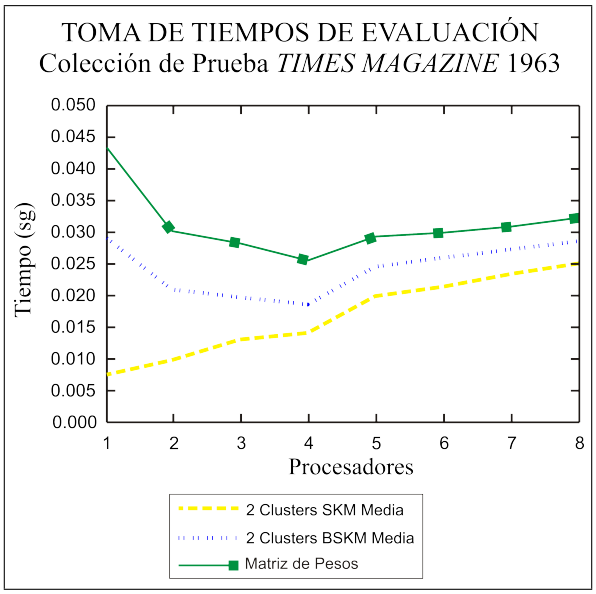

Figura 5.17: Comparación de toma de tiempos de evaluación entre el *Spherical K-Means,* el *α-Bisecting Spherical K-Means* y la Matriz de Pesos con la colección *Times*.

Con la colección *Cystic* (figura 5.18 en la página siguiente) el comportamiento obtenido es similar, la matriz de pesos en todo momento tiene un mayor coste temporal que los otros métodos. En este caso la respuesta a la paralelización es mala respecto a las prestaciones temporales, ya que en lugar de mejorar o mantenerse, empeoran de manera drástica llegando a perder casi toda la mejora que presentaban.

A medida que el tamaño de la colección crece, como se puede ver en la figura 5.19 en la página siguiente, proporcionalmente la mejora temporal del *Spherical K-Means* y el *α-Bisecting Spherical K-Means* se acrecenta, manteniendo siempre la misma estructura. También se observa que los costes de comunicación en la paralelización se compensan con la mejora temporal de usar más unidades de proceso, por ello el tiempo de respuesta de ambos métodos consiguen mantenerse en el mismo orden de magnitud.

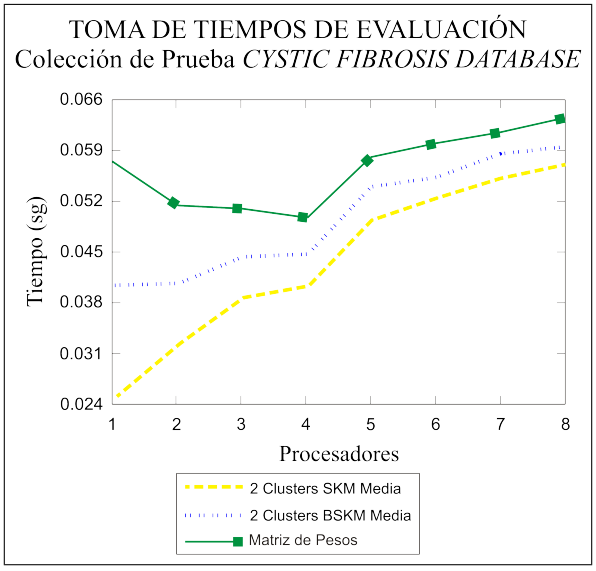

Figura 5.18: Comparación de toma de tiempos de evaluación entre el *Spherical K-Means,* el α-*Bisecting Spherical K-Means* y la Matriz de Pesos con la colección *Cystic*.

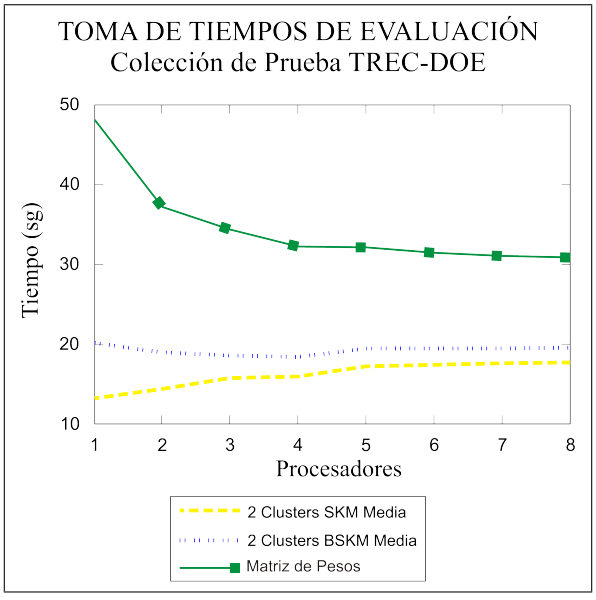

Figura 5.19: Comparación de toma de tiempos de evaluación entre el *Spherical K-Means,* el α-*Bisecting Spherical K-Means* y la Matriz de Pesos con la colección *DOE*.

Estudio del *Speed Up* y de la Eficiencia

La colección *Times* (figura 5.20) ofrece unos resultados buenos pero anormales y no generalizables, al menos con el α-*Bisecting Spherical K-Means* que sigue un comportamiento similar a la matriz de pesos, la cual se utiliza de referencia. Estos resultados se asocian a relación óptima entre la colección, el método y los parámetros de éste elegidos. Por su parte el *Spherical K-Means* presenta un *speed up* que empeora continuamente.

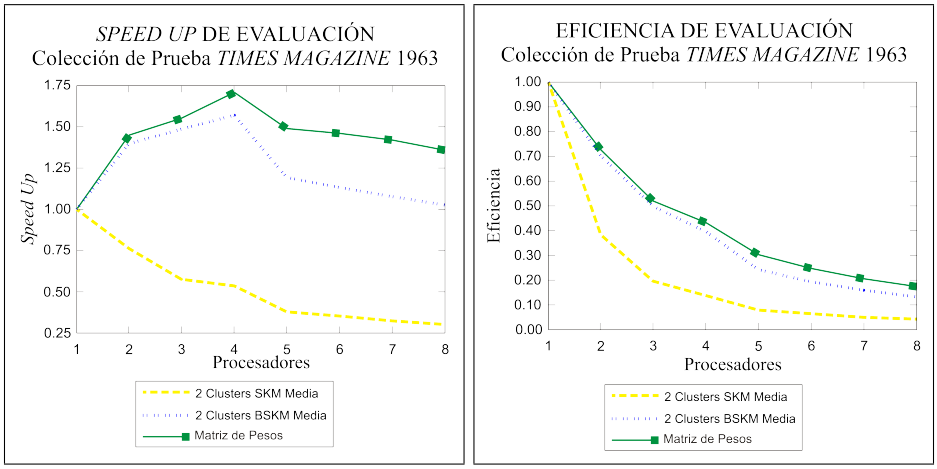

Figura 5.20: Comparación de *speed up* y eficiencia de evaluación entre el *Spherical K-Means,* el α-*Bisecting Spherical K-Means* y la Matriz de Pesos, colección *Times*.

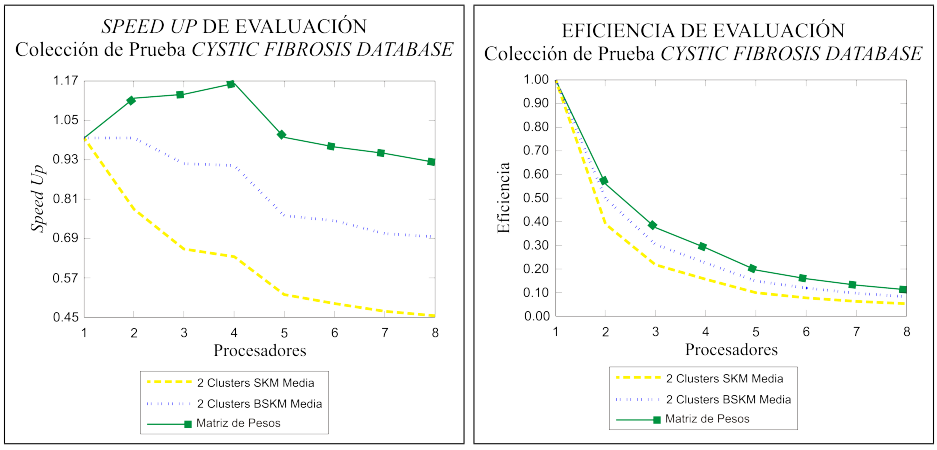

Figura 5.21: Comparación de *speed up* y eficiencia de evaluación entre el *Spherical K-Means,* el α-*Bisecting Spherical K-Means* y la Matriz de Pesos, colección *Cystic*.

El comportamiento presentado en la figura 5.21 en la página anterior (colección *Cystic*) representa mejor el comportamiento general de ambos métodos, las malas prestaciones computacionales al utilizar paralelismo. El *speed up* de ambos empeora al subir los procesadores usados.

Con la colección *Cystic* la curva de la eficiencia de ambos métodos es cercana a la de referencia, y ya que esta última de por sí no presenta buenas prestaciones, lógicamente los métodos presentados no obtienen buenas prestaciones. Así y todo los valores que muestran se reducen a una eficiencia entorno al 50 % con dos unidades de proceso y valores inaceptables a partir de ahí.

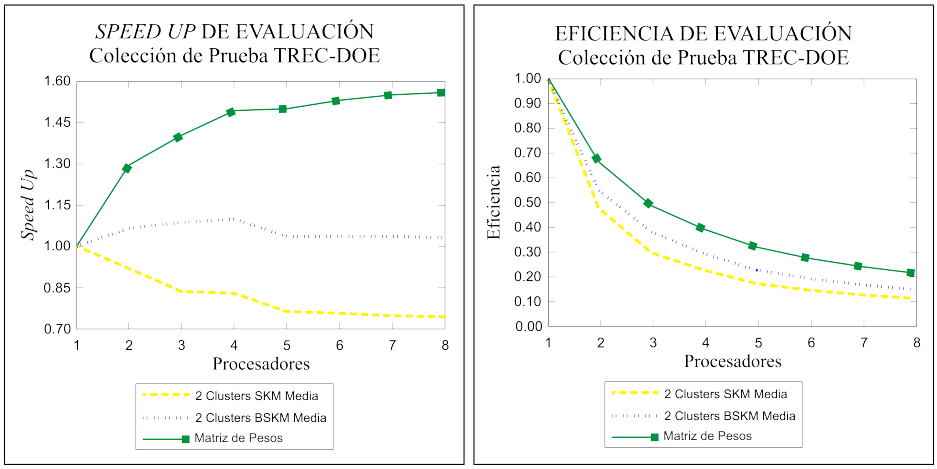

Figura 5.22: Comparación de *speed up* y eficiencia de evaluación entre el *Spherical K-Means,* el *α-Bisecting Spherical K-Means* y la Matriz de Pesos, colección *DOE*.

La utilización de una colección de mayor tamaño (colección DOE, figura 5.22) compensa los costes de comunicación con los costes de evaluación propios y las curvas del *speed up* y eficiencia se suavizan. Pero así y todo está claro que tanto el *Spherical K-Means* como el *α-Bisecting Spherical K-Means* no obtienen mejores prestaciones temporales al utilizar paralelismo.

El *speed up* viene a plasmar de forma más clara la respuesta ante la paralelización ya vista en el estudio temporal. El método *Spherical K-Means* presenta un *speed up* pésimo, puesto que está por debajo de uno siempre y a medida que se aumentan los elementos de proceso baja todavía más. Por su lado el *α-Bisecting Spherical K-Means* presenta un comportamiento también malo pero menos desastroso, llegando incluso a tener algunas mejoras con la colección *Times*, lo cual se explica por una situación óptima casual de la colección y los parámetros elegidos. Del estudio de la eficiencia destacan dos aspectos. Primero que el *α-Bisecting Spherical K-Means* tiene un comportamiento especial y relativamente bueno con la colección *Times*. Y segundo, salvo dicho caso, la eficiencia de ambos métodos sólo podría llegar a ser aceptable con dos unidades de proceso con un valor rondando el 50 %, a partir de ahí sufre un importante descenso.

5.3.3. Evaluación: Matriz de Pesos vs. VDBSCAN

En este apartado se hace la comparación de los costes temporales del VDBSCAN con la matriz de pesos como elemento de referencia. También se incluyen los estudios del *speed up* y de la eficiencia. Todos los estudios se presentan para las tres colecciones de prueba que se han usado en esta tesis (ver apartado 5.1 en la página 123).

Toma de Tiempos

El método VDBSCAN en todos los casos tiene un coste temporal menor que el método de referencia (la matriz de pesos). A medida que el tamaño del problema aumenta la diferencia entre uno y otro aumenta, siendo en comparación proporcionalmente cada vez más rápido el VDBSCAN. También destaca que parece que la mejora de prestaciones al utilizar paralelismo se pierde antes con el VDBSCAN, en otras palabras el número de unidades de procesamiento que hace mejorar el tiempo de evaluación es menor con el VDBSCAN que con la matriz de pesos. Esto es debido a que en el primero la complejidad del método conlleva mayor número de comunicaciones y por tanto éstas limitan antes el número de elementos de proceso que benefician las prestaciones.

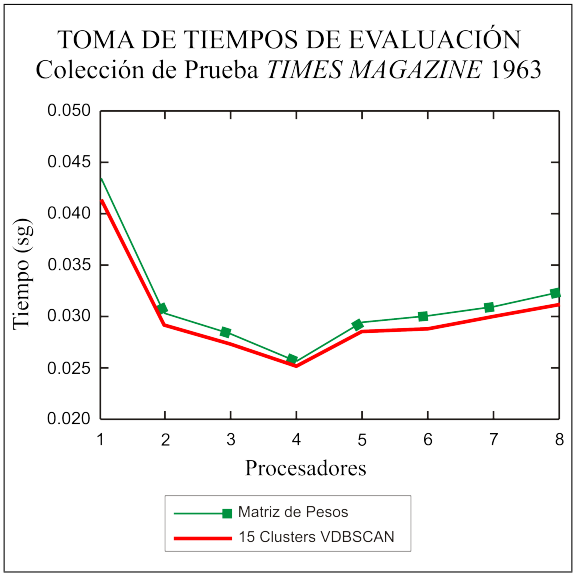

Figura 5.23: Comparación de toma de tiempos de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *Times*.

La colección *Times* (figura 5.23) conlleva un tiempo de respuesta ante la evaluación prácticamente igual tanto en el VDBSCAN como con la matriz de pesos, pero siempre el VDBSCAN por debajo. También se observa que a medida que aumentamos el número de unidades de proceso el tiempo de evaluación se reduce, hasta que

se alcanzan las cuatro unidades, punto a partir del cual el paralelismo deja de ser beneficioso y los tiempos empeoran.

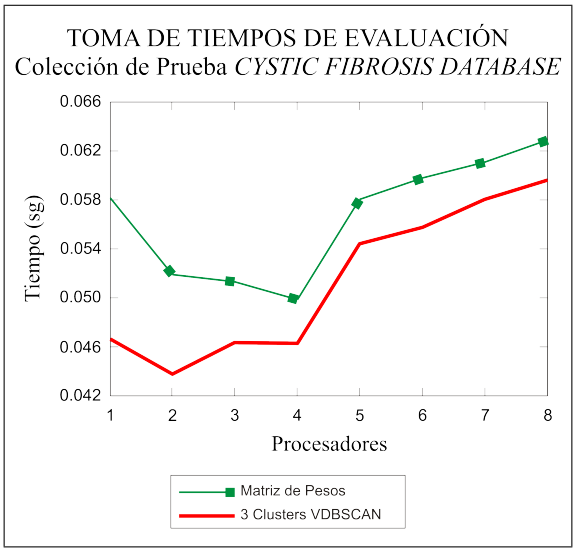

Figura 5.24: Comparación de toma de tiempos de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *Cystic*.

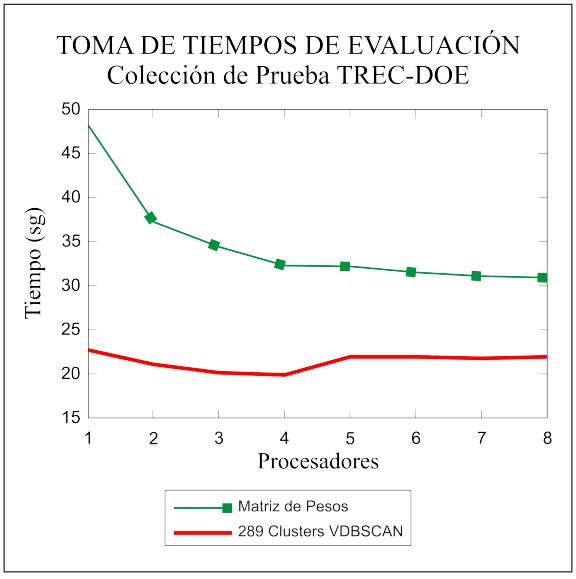

Figura 5.25: Comparación de toma de tiempos de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *DOE*.

Como muestra la figura 5.24 en la página anterior, con la colección *Cystic* el comportamiento es similar, aunque en este caso las diferencias temporales son mayores. Pero también es peor la respuesta ante la paralelización, la matriz de pesos tiene mejor comportamiento relativo. De hecho, el VDBSCAN a partir de dos unidades de proceso ya no presenta beneficios con el paralelismo, mientras que la matriz de pesos alcanza las cuatro unidades de proceso.

Con la colección DOE (figura 5.25 en la página anterior) las diferencias temporales de evaluación son todavía más significativas, el VDBSCAN es bastante más rápido en comparación. En cuanto a la respuesta ante la paralelización se observa que ambas curvas suavizan la mejora de prestaciones, sobre todo el VDBSCAN. Ahora bien, mientras la matriz de pesos aumenta las unidades de proceso que le mejoran las prestaciones, el VDBSCAN sigue manteniéndose en cuatro como número límite, aunque sí que alarga el límite en el cual empieza a realmente empeorar las prestaciones.

Estudio del *Speed Up* y de la Eficiencia

El VDBSCAN presenta un *speed up* no muy bueno, de hecho en todo momento está por debajo de la matriz de pesos (modelo de referencia) aunque ésta de por sí ya lo tiene malo. De hecho destaca que a medida que aumenta el tamaño de la colección el *speed up* empeora en relación con la matriz de pesos. Es obvio que no se aprovechan lo que debieran las posibilidades de mejora de prestaciones de la paralelización, existen demasiadas comunicaciones.

Tras lo visto en el estudio del *speed up* el estudio de la eficiencia no aporta mucho más. El VDBSCAN es menos eficiente en todos los casos respecto a la matriz de pesos, aunque no se va mucho de ésta. Así y todo, eficiencias aceptables sólo se consiguen con dos unidades de proceso y en algunos casos concretos se podrían aceptar hasta tres unidades de proceso.

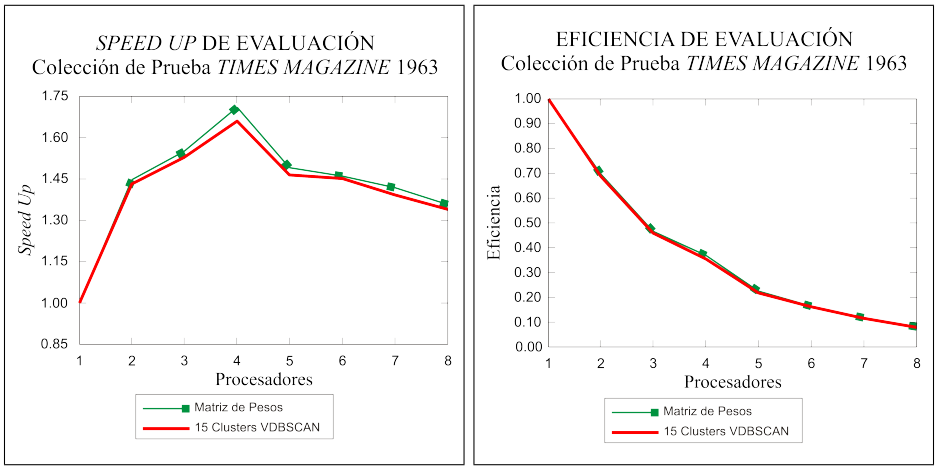

Figura 5.26: Comparación de *speed up* y eficiencia de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *Times*.

La colección *Times* (figura 5.26 en la página anterior) es la que presenta el mejor comportamiento, ya que se puede observar como el VDBSCAN obtiene prácticamente el mismo *speed up* que la curva de referencia (la matriz de pesos). No se puede hablar de un buen *speed up* puesto que está muy lejos de los valores ideales, pero al menos el VDBSCAN responde respecto a los valores de referencia. Análogamente presenta la mejor eficiencia, estando en todo momento pareja a la de la matriz de pesos. De hecho es la única que admitiría hasta tres unidades de proceso con una eficiencia que ronda el 50 %.

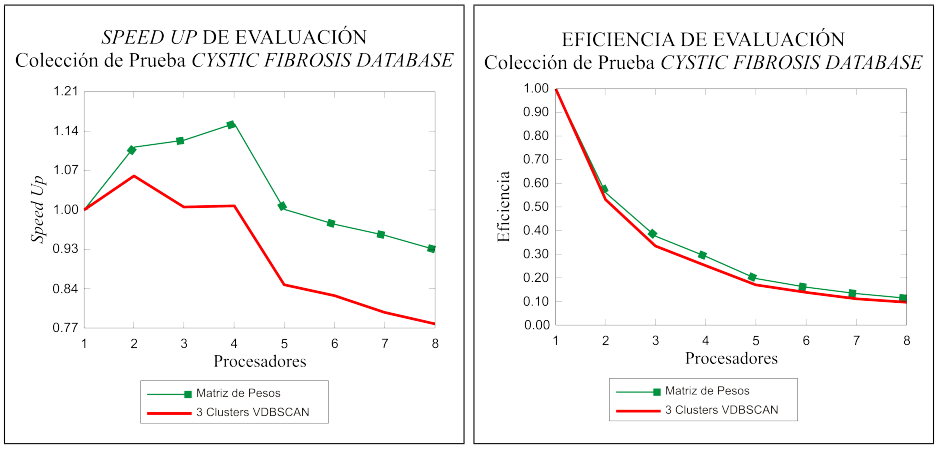

Figura 5.27: Comparación de *speed up* y eficiencia de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *Cystic*.

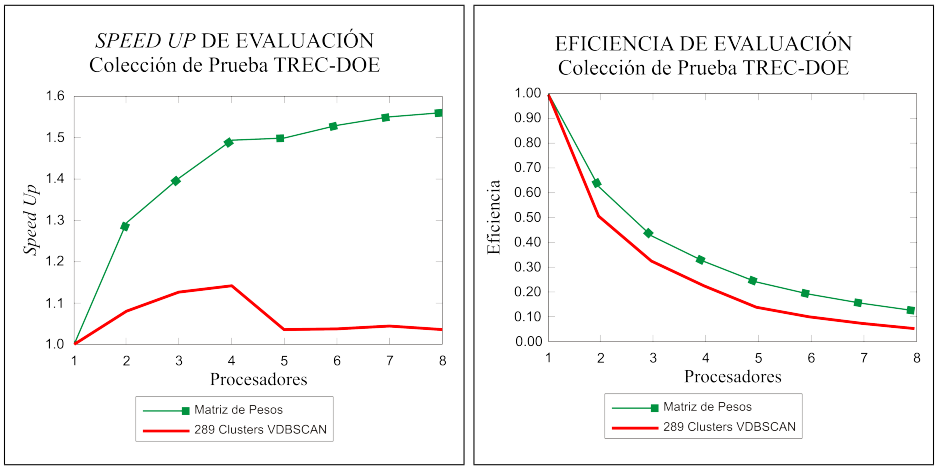

Figura 5.28: Comparación de *speed up* y eficiencia de evaluación entre el VDBSCAN y la Matriz de Pesos con la colección *DOE*.

Con la colección *Cystic* (figura 5.27 en la página anterior) las diferencias son mayores, pero también hay que decir que ya la matriz de pesos presenta un *speed up* malo. El comportamiento de la eficiencia es también similar, un poco menor el VDBSCAN respecto a la matriz de peso, pero ambos muy igualados. Eso si la eficiencia decae mucho más y ya con sólo dos unidades de proceso alcanza valores menores de un 60 %.

Con la colección DOE (figura 5.28 en la página anterior) sería de esperar obtener unas prestaciones mucho mejores debido a que la colección es mucho más grande, pero los resultados indican que las diferencias respecto a la matriz de pesos son mayores. Esta mala respuesta a la paralelización del algoritmo VDBSCAN tiene sentido ya que mientras que con la matriz de pesos se trabaja con toda la colección (muchos más documentos), el VDBSCAN evalúa sólo unos cuantos documentos, los que se encuentran dentro del *cluster* seleccionado. Además, cuando la colección de documentos crece, también lo hace la dispersión de la matriz que lo modeliza, con lo cual los costes computacionales también disminuyen pero las comunicaciones no.

5.3.4. Evaluación: VDBSCAN vs. α-*Bisecting Spherical K-Means*

Por último se presenta la comparación del tiempo de respuesta en la fase de evaluación de consultas de los dos métodos destacados: el α-*Bisecting Spherical K-Means* y el VDBSCAN. El estudio del *speed up* y de la eficiencia también son incluidos. Todos los estudios se presentan para las tres colecciones de prueba que se usan en esta tesis (ver apartado 5.1 en la página 123).

Toma de Tiempos

En el estudio temporal destaca que el α-*Bisecting Spherical K-Means* presenta en todo momento un tiempo de evaluación menor que el VDBSCAN, aunque esta diferencia es pequeña sobre todo a medida que aumentan las unidades de proceso utilizadas. Siempre teniendo en cuenta que se compara el comportamiento medio del α-*Bisecting Spherical K-Means* y que este depende de la elección de los parámetros iniciales, desconocidos a priori.

Con la colección *Times* (figura 5.29 en la página siguiente) el método α-*Bisecting Spherical K-Means* siempre presenta un tiempo de respuesta menor que el tiempo obtenido con el método VDBSCAN. Ambos métodos, además, reducen el tiempo de respuesta a medida que aumentan las unidades de proceso utilizadas, pero mientras que el algoritmo VDBSCAN a partir de cuatro empieza a dejar de mejorar, el algoritmo α-*Bisecting Spherical K-Means* a partir de cuatro empieza a empeorar rápidamente, perdiendo pronto todos los beneficios alcanzados, e incluso emperorando los tiempos secuenciales.

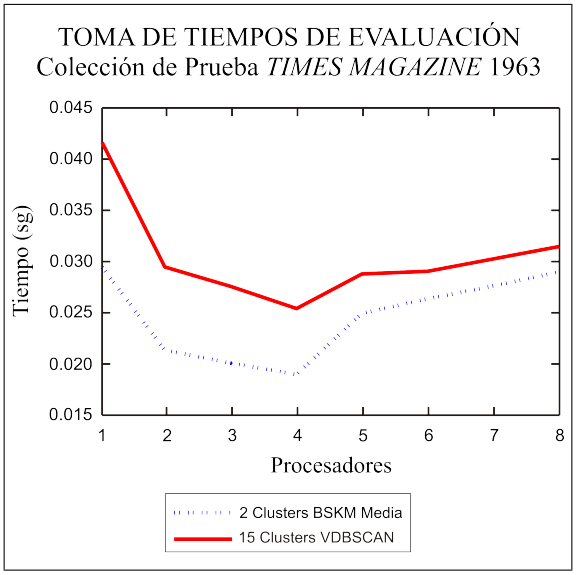

Figura 5.29: Comparación de toma de tiempos de evaluación entre el α-*Bisecting Spherical K-Means* y el VDBSCAN con la colección *Times*.

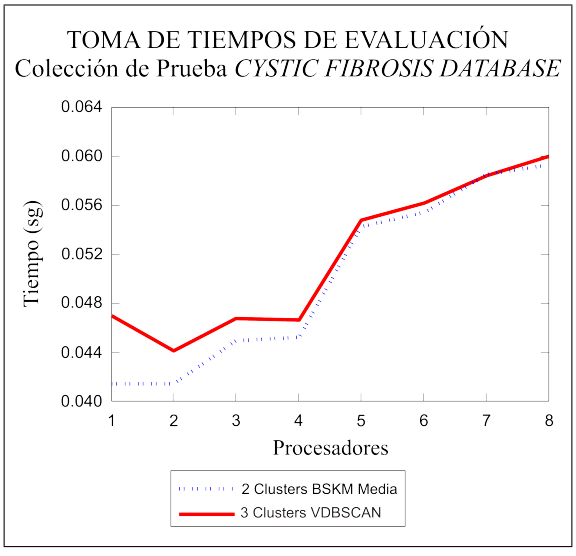

Figura 5.30: Comparación de toma de tiempos de evaluación entre el α-*Bisecting Spherical K-Means* y el VDBSCAN con la colección *Cystic*.

La colección *Cystic* (figura 5.30) conlleva un comportamiento mucho más similar entre ambos métodos, de hecho a partir de cuatro unidades de proceso el tiempo

es prácticamente el mismo, aunque excepto con siete unidades de proceso que el método VDBSCAN es más rápido, el método *α-Bisecting Spherical K-Means* en todo momento presenta menores tiempos. Lo que también destaca con esta colección es que con el paralelismo no se aprecia mejoría en los tiempos de respuesta, de hecho éstos empeoran a apartir de tres unidades de proceso (sólo mejoran un poco con dos unidades de proceso y con el método VDBSCAN).

Con la colección DOE (figura 5.31) se continúa viendo el mismo comportamiento general que con las demás colecciones. El método *α-Bisecting Spherical K-Means* tarda algo menos de tiempo en realizar la evaluación de la consulta, pero el método VDBSCAN tiene mayor mejora al aumentar el número de elementos de proceso.

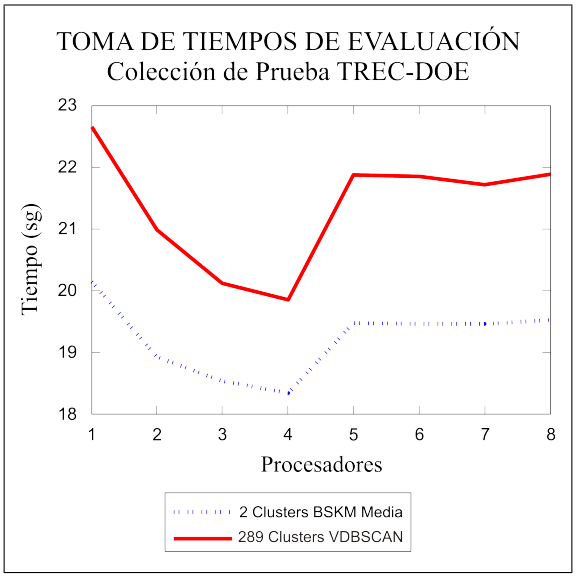

Figura 5.31: Comparación de toma de tiempos de evaluación entre el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *DOE*.

Estudio del *Speed Up* y de la Eficiencia

Como era de esperar al observar los costes temporales, el estudio del *speed up* confirma que ambos algoritmos no tienden a responder bien al paralelismo en la fase de evaluación (tal y como pasaba con el modelo de referencia, la matriz de pesos), pero así y todo el método VDBSCAN obtiene mejores prestaciones paralelas que el método *α-Bisecting Spherical K-Mean.*

En general se aprecia que la eficiencia de ambos métodos es muy similiar, prácticamente igual, y que en los dos casos es bastante mala, ya que a partir de dos unidades de proceso la eficiencia siempre está por debajo del 50 %. Aunque no hay que olvidar que este comportamiento ya lo tenía la matriz de pesos, lo cual implica que es inherente al problema.

La figura 5.32 muestra el estudio realizado con la colección *Times*, en ella se observa como en todo momento el método VDBSCAN mantiene mejores prestaciones, aunque éstas no sean buenas en valores absolutos (sí lo son en relación al modelo de referencia, la matriz de pesos). También se aprecia que a partir de tres unidades de proceso el método *α-Bisecting Spherical K-Means* se empieza a diferenciar mucho más del método VDBSCAN, y a partir de cuatro, su *speed up* cae en picado. Respecto a la eficiencia se observa cómo ambas curvas siguen prácticamente el mismo camino, teniendo una eficiencia aceptable ambos métodos con dos y tres unidades de proceso (casi del 70 % y del 50 %, respectivamente) y a partir de ahí bajando paulatinamente.

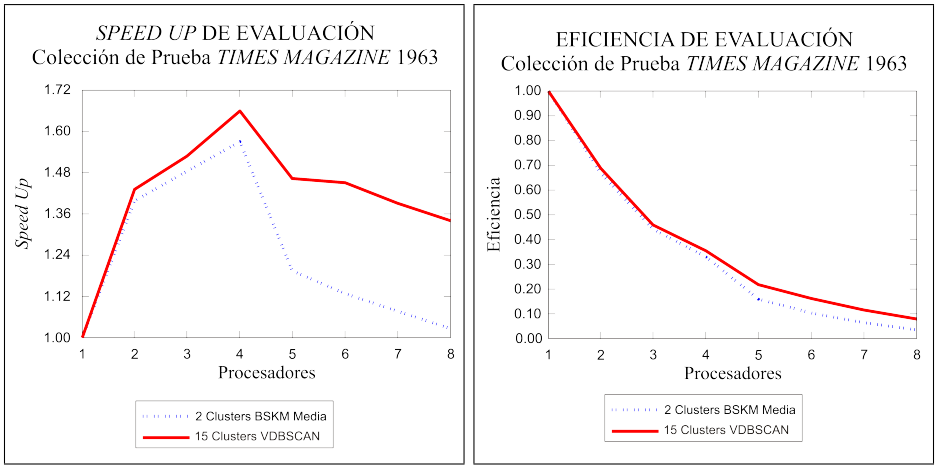

Figura 5.32: Comparación de *speed up* y eficiencia de evaluación entre el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Times*.

Con la colección *Cystic* (figura 5.33 en la página siguiente) el comportamiento es parecido al de la colección *Times*, el *speed up* del VDBSCAN es siempre superior al del *α-Bisecting Spherical K-Means*. Pero, en este caso, los resultados a nivel de paralelismo son desastrosos, el *speed up* cae desde el principio, a excepción del caso de dos unidades de proceso con el VDBSCAN que sube sutilmente. En eficiencia ambos métodos presentan los mismos valores de eficiencia, siendo bastante bajos, estando entorno al 50 % con dos unidades de proceso y luego siempre por debajo.

Al hacer el estudio con la colección DOE (figura 5.34 en la página siguiente) se observa de nuevo el comportamiento general, el *α-Bisecting Spherical K-Means* obtiene un *speed up* inferior al VDBSCAN, el cual, además, cada vez está más distante. Hasta cuatro unidades de proceso las prestaciones mejoran, aunque sea un poco, pero a partir de ahí éstas empiezan a empeorar drásticamente. El mayor tamaño de la colección DOE no hace mejorar las prestaciones de eficiencia de ninguno de los dos métodos. Lo único que hace es minimizar más las pequeñas diferencias de eficiencia de ambos métodos. Al igual que con la colección *Cystic* la eficiencia con dos unidades de proceso ronda el 50 % y ya con más elementos de proceso decae considerablemente.

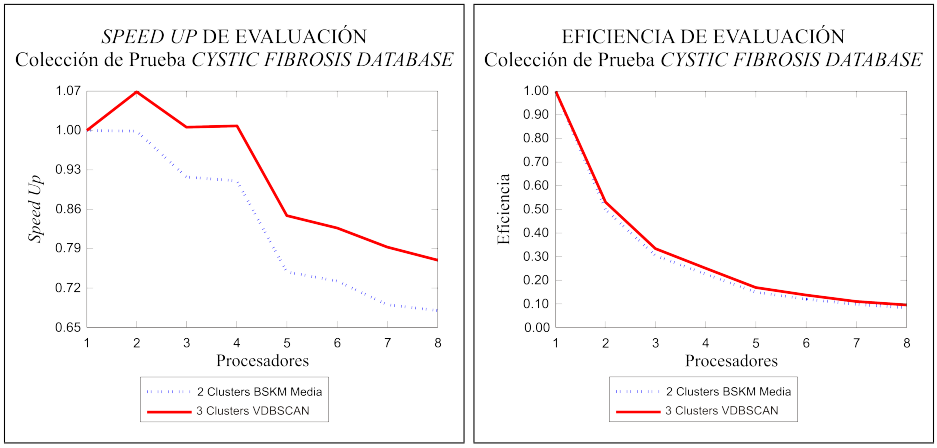

Figura 5.33: Comparación de *speed up* y eficiencia de evaluación entre el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *Cystic*.

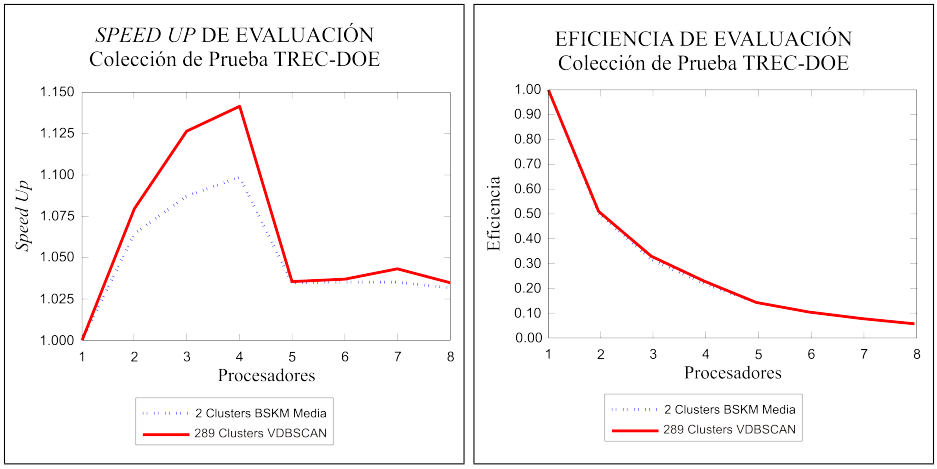

Figura 5.34: Comparación de *speed up* y eficiencia de evaluación entre el *α-Bisecting Spherical K-Means* y el VDBSCAN con la colección *DOE*.

5.4. Conclusiones

Se ha presentado el estudio experimental de la calidad de recuperación de información y de las prestaciones computacionales de los métodos *α-Bisecting Spherical K-Means* y VDBSCAN. Pero todo ello de forma extendida y por partes, en este apartado se destacan de manera resumida los resultados hallados y en función de los mismos se determina el mejor método.

5.4.1. Prestaciones en recuperación de información

Con el método *Spherical K-Means* se puede deducir que generar más de dos *clusters* es contraproducente, de hecho aunque se generen sólo dos *clusters* la calidad de recuperación será peor que con la matriz de pesos. El método *Spherical K-Means* depende mucho de la solución inicial elegida, por tanto según sea su elección la calidad aumentará o disminuirá, pero hay que afinar mucho si no en general la calidad empeorará significativamente.

El método *α-Bisecting Spherical K-Means*, al igual que el *Spherical K-Means*, depende mucho de los centroides iniciales elegidos (pero algo menos) y en general su comportamiento es análogo al del *Spherical K-Means*. Pero la calidad en recuperación de información mejora tanto en el peor, como en el mejor y medio de los casos. La curva media mejora considerablemente con el *α-Bisecting Spherical K-Means* respecto al *Spherical K-Means*, obteniéndose los mejores resultados en la recuperación de información generando dos *clusters*.

Comparando ambos métodos, el *Spherical K-Means* y el *α-Bisecting Spherical K-Means*, se observa que el segundo mejora las prestaciones de recuperación del primero. Ambos son bastante regulares y siguen la misma línea de comportamiento que la matriz de pesos.

El método VDBSCAN, con la selección adecuada de sus parámetros, tiene un comportamiento de prestaciones de recuperación de información muy similar al de la matriz de pesos. Comparando éste, con el método *α-Bisecting Spherical K-Means*, se aprecia que el algoritmo VDBSCAN obtiene una mejor calidad. Si en lugar de comparar con la curva media se hiciese con la curva óptima del algoritmo *α-Bisecting Spherical K-Means* el método VDBSCAN sigue siendo superior aunque las diferencias son menores.

5.4.2. Prestaciones computacionales

Al igual que ocurría en el estudio de la calidad de recuperación de información, en el estudio de costes temporales, también se ha utilizado la matriz de pesos como referencia de comparación. Esto es debido a que los métodos presentados pretenden dar mejores prestaciones que la matriz de pesos, que es el modelo base a partir del cual se construyen los demás. Teniendo en cuenta este aspecto, en ocasiones se observa que las prestaciones de un método pueden parecer que no sean buenas, pero cuando se comparan éstas con las prestaciones de referencia (las que da lugar el modelo de la matriz de pesos) se observa que éstas son relativamente muy buenas.

Respecto a la modelización de los métodos hay que decir que mientras el *Spherical K-Means* y el *α-Bisecting Spherical K-Means* presentan unos costes similares entre si, los obtenidos con el VDBSCAN son claramente mucho más costosos. Aunque éste también responde mejor a la paralelización. De todas formas, no hay que olvidar que la fase de modelización se hace una única vez al inicio de la puesta en funcionamiento del sistema de recuperación de información, por lo cual un coste elevado en la misma seguramente es más que aceptable si luego se compensa en la fase de evaluación o en la calidad de recuperación.

En el análisis de la modelización a partir del *speed up* y la eficiencia se observa que con las colecciones *Times* y *Cystic* los resultados alcanzados no son buenos, con dos unidades de proceso son aceptable pero a partir de ahí son realmente malos. Ninguno de los tres métodos con estas colecciones presenta un comportamiento destacado, los tres son muy similares. Cuando se utiliza la colección DOE, de un tamaño ya considerable las mejoras son claras. Son pequeñas y aceptables con los métodos *Spherical K-Means* y *α-Bisecting Spherical K-Means* pero bastante buenas con el VDBSCAN llegándose a *speed up* cercanos al óptimo en algunos casos y no bajando de una eficiencia del 70 % hasta con ocho unidades de proceso.

En relación a la comparativa de evaluación entre el *Spherical K-Means* y el *α-Bisecting Spherical K-Means* respecto a la matriz de pesos como referencia, hay que decir que ambos métodos presentan en todo momento un tiempo de evaluación menor que la matriz de pesos, siendo siempre más rápido el *Spherical K-Means*. Ahora bien, ninguno de los dos métodos responde con mejoras claras a la paralelización. Ésto se puede ver en el estudio del *speed up*. El método *Spherical K-Means* presenta un *speed up* por debajo de uno siempre y a medida que se aumentan los elementos de proceso baja todavía más. Por su lado el *α-Bisecting Spherical K-Means* presenta un comportamiento también malo pero más suavizado, éste además presenta un caso concreto en el que llega incluso a tener mejoras. Esta situación se da con la colección *Times* y se explica por una situación óptima casual de la colección y los parámetros elegidos.

El estudio de la eficiencia confirma lo observado en el estudio temporal y de *speed up*. El *α-Bisecting Spherical K-Means* presenta una situación anormal pero buena con la colección *Times*. Pero a excepción de dicha situación la eficiencia de ambos métodos sólo es aceptable con dos unidades de proceso con un valor en torno al 50 %.

Cuando se compara el tiempo de respuesta de la evaluación del método VDBSCAN respecto a la matriz de pesos se observa que el método VDBSCAN en todos los casos tiene un coste temporal menor. A medida que el tamaño del problema aumenta la diferencia entre uno y otro aumenta, siendo en comparación proporcionalmente cada vez más rápido el VDBSCAN. También destaca que el comportamiento del VDBSCAN ante la paralelización es parecido al de la matriz de pesos, pero algo peor, tal y como demuestra el estudio de su *speed up* y de la eficiencia.

El VDBSCAN presenta un *speed up* no muy bueno, en todo momento está por debajo de la matriz de pesos (modelo de referencia) aunque ésta de por sí ya lo tiene malo. Es obvio que no se aprovechan lo que debieran las posibilidades de mejora de la paralelización, existen demasiadas comunicaciones. El VDBSCAN es menos eficiente que la matriz de pesos en todos los casos, aunque no se va mucho de ésta. Así y todo, eficiencias aceptables sólo se obtienen para dos y a lo sumo tres unidades de proceso.

Para acabar y para aclarar se ha comparado el tiempo de respuesta del *α-Bisecting Spherical K-Means* y el VDBSCAN, donde destaca que el primero siempre es un poco más rápido que el segundo, haciéndose incluso más pequeñas estas diferencias a medida que aumentan el número de unidades de proceso. Mediante el estudio del *speed up* y la eficiencia se confirma lo que ya se había visto al estudiar los métodos respecto a la matriz de pesos, su paralelismo no aporta claras mejoras en las prestaciones temporales (aunque en relación a la referencia de la matriz de pesos, no son resultados tan malos,

ya que ésta ya presenta malas prestaciones de por si). Así y todo el VDBSCAN obtiene mejores valores de *speed up* y sensiblemente mejores resultados de eficiencia estando éstos enseguida por debajo del 50 %.

En general, para todos los métodos, se ha observado que las prestaciones temporales no sufren una mejoría clara y satisfactoria mediante la paralelización. Es interesante mejorar los métodos de paralelización, por ello en el apartado de trabajos futuros (apartado 6.3 en la página 170) se explica un poco más las posibles razones y formas de mejorar este aspecto.

5.4.3. Mejor método según sus prestaciones

La coparación se ha realizado entre los métodos estudiados: DBSCAN, VDBSCAN, *Spherical K-Means* y α-*Bisecting Spherical K-Means*. Y los criterios de evaluación son la calidad de recuperación de información y la complejidad computacional, teniendo ambas en cuenta de forma global.

Primero de todo una aclaración, las comparaciones siempre se han hecho sobre el comportamiento medio del *Spherical K-Means* y del α-*Bisecting Spherical K-Means* y éste depende de la elección de los parámetros iniciales, desconocidos a priori. Con lo cual, no se puede asegurar que los resultados sean mejores o peores, pero normalmente tenderán a no mejorar puesto que la elección de los parámetros iniciales no se hará bajo un conocimiento de apoyo.

No hay que olvidar que en calidad de recuperación el VDBSCAN y el DBSCAN obtienen los mismos resultados. En cuestión de coste temporal el VDBSCAN es casi el doble de rápido que el DBSCAN. Por todo ello, entre estos dos métodos es claramente superior el VDBSCAN. Así que desde este momento se descarta el DBSCAN del análisis.

Desde el punto de la calidad de la recuperación de información ha quedado bastante claro que el método *Spherical K-Means* es el peor de todos, ni siquiera se acerca al modelo de referencia (la matriz de pesos). El método α-*Bisecting Spherical K-Means* mejora sustancialmente la calidad de recuperación del *Spherical K-Means*, aunque sigue dependiendo mucho de la elección inicial de las semillas. Por su lado el método VDBSCAN, cuyos parámetros adecuados se pueden conocer con un preprocesamiento (ver apartado 3.5 en la página 90), presenta una calidad de recuperación muy similar al modelo de referencia, estando en algunos momentos por debajo y en otros por encima de éste. Además, gana con claridad en calidad de recuperación de información al método α-*Bisecting Spherical K-Means*, de hecho es superior incluso al compararse con la curva óptima del método α-*Bisecting Spherical K-Means*.

Por lo tanto, en cuestión de calidad de recuperación de información es claramente mejor el VDBSCAN. Con lo cual, a no ser que comparativamente fuese muy inferior en la cuestión temporal, esto sería suficiente para elegir globalmente el VDBSCAN como mejor método, ya que la calidad de recuperación de información es más importante, en general, que el tiempo de respuesta, mientras que éste se encuentre dentro de unos límites aceptables.

Respecto al estudio temporal se puede destacar que en cuestión de tiempo de modelización el VDBSCAN es mucho más costoso que el *Spherical K-Means* o el α-

Bisecting Spherical K-Means, aunque al incluir paralelización la mejoría del primero es también mucho mayor que los otros dos métodos, alcanzando valores de *speed up* cercanos al óptimo y eficiencias superiores al 70 %. Tampoco hay que olvidar que la fase de modelización se da una única vez al poner en funcionamiento el sistema y que por lo tanto no es tan importante el tiempo que conlleve, ya que el que más importancia tiene es el tiempo de evaluación de las consultas una vez el sistema de recuperación de información esté en funcionamiento.

Cuando se estudia el tiempo de evaluación de los distintos métodos se observa que todos ellos tienen tiempos de respuesta inferiores al del modelo de referencia (matriz de pesos). Y por orden, el más rápido es el *Spherical K-Means*, seguido por el α-*Bisecting Spherical K-Means* y por último, el más lento de los tres es el VDBSCAN. De todas formas las diferencias tampoco son muy grandes en valores absolutos. Además, estas diferencias se suavizan a favor del VDBSCAN cuando se acude a la paralelización. Y de hecho el comportamiento ante ésta es peor por parte del *Spherical K-Means* y el α-*Bisecting Spherical K-Means*. Aunque en ninguno de los casos se puede hablar de un buen aprovechamiento de las posibilidades del paralelismo a la mejora de prestaciones, aspecto que, de todas formas, ya es observado en el propio modelo de referencia.

Con todo ello y a modo de resumen tenemos que:

- El VDBSCAN obtiene claramente mejor calidad de recuperación de información, el aspecto más importante.
- Las diferencias entre los tiempos de respuesta ante la evaluación de las consultas es mínima y además se reduce a medida que aumentan las unidades de proceso en la paralelización (no olvidar que el sistema se plantea ya en un entorno distribuido de forma natural).
- El VDBSCAN presenta un mejor comportamiento ante la paralelización, sobre todo durante la modelización, aunque también conlleva unos costes computacionales mucho mayores que el α-*Bisecting Spherical K-Means.*

En conclusión, según los requisitos del sistema y sus características el mejor método será el VDBSCAN o el α-*Bisecting Spherical K-Means*. El primero de ellos será elegido siempre que se considere primordial la calidad de recuperación, ya que el VDBSCAN presenta las mejores prestaciones a este respecto. Ahora bien, si el sistema prevé que la fase de modelización se va a repetir muchas veces, por ejemplo porque cada cierto tiempo se incluyen nuevas subcolecciones al sistema, entonces se dará un mayor valor a los costes computacionales, en cuyo caso el α-*Bisecting Spherical K-Means* sería la opción elegida.

Capítulo 6

Conclusiones finales y trabajos futuros

6.1. Evolución de la tesis y produción científica

6.1.1. Evolución de la tesis

La idea original de la tesis surgió al estudiar la utilización de la descomposición en valores singulares como posible método de recuperación de información. Ello conllevó empezar a estudiar los distintos modelos de sistemas de recuperación centrándonos rápidamente en el modelo vectorial básico y en el enfoque LSI (*Latent Semantic Indexing*) más sofisticado. La razón principal fue la estructura de datos subyacente, una matriz dispersa de gran dimensión. Durante el desarrollo de la tesis se trabajó en el estudio de la fase de preprocesamiento, fundamentalmente en el *stemming*, la reducción heurísitica de términos y la lematización semántica. Y sobre todo se abordó la fase de modelización, estudiando diferentes métodos vectoriales y posibles mejoras de los mismos. Fue en esta fase cuando se descartó el enfoque LSI en pro del de *clustering* donde se perfeccionó el conocido método *Spherical K-Means* y luego al observar los problemas intrínsecos al mismo nos decantamos por trabajar en base al DBSCAN, llegando a desarrollar una variante con mejores prestaciones.

Al principio estudiamos la fase de preprocesamiento, donde empezamos por evaluar diferentes métodos de *stemming*, concretamente tres de los más importantes [121]: *Potter*, *Lovins* y *Paice*. Para su comparación se desarrollo una aplicación en el entorno Matlab que integraba los algoritmos de los distintos métodos desarrollados en C (desarrollados por otros autores). Nuestros estudios presentados en [115] y los presentados en [122] demostraron que el algoritmo *Paice* proporcionaba mejores prestaciones al obtener una mayor reducción de terminos.

También, siguiendo con la fase de preprocesamiento, se estudió en [115] la influencia de la reducción heurística de términos, eliminando aquellos que daban poca información, por aparecer mucho o prácticamente nada. Destacó el hecho que elimi-

nando aquellos términos que sólo aparecían en un documento se reducía en casi el 50 % los terminos que modelizaban la colección, y eliminando los que sólo aparecían en cinco documentos se obtenía más de un 75 % de reducción. Como además de la reducción del problema, es decir la reducción de términos, también es muy importante las prestaciones de recuperación se estudió ésta en función de las diferentes reducciones heurísticas. Se desarrolló una aplicación en Matlab que evaluaba un sistema de recuperación de información vectorial básico, y se evaluó dicho sistema con las distintas reducciones heurísticas, llegándose a la conclusión que eliminando heurísticamente aquellos términos que sólo aparecían en un único documento las prestaciones del sistema se mantenían practicamente intactas y el tamaño del problema se reducía casi a la mitad. La misma aplicación en Matlab desarrollada nos sirvió para estudiar la influencia de la utilización de una matriz de pesos respecto a la matriz de frecuencias, más primitiva y simple. Corraboramos en [115] que la matriz de pesos permitía obterner mejores prestaciones de recuperación que la matriz de frecuencias.

Tras los estudios de la fase de preprocesamiento pasamos a comparar el enfoque LSI mediante la utilización de la SVD como herramienta y una primera modificación del conocido método de *clustering Spherical K-Means*. La modificación dio lugar a una versión primitiva del *α-Bisecting Spherical K-Means* en la cual el parámetro α fue determinado empíricamente. Ambos métodos fueron desarrollados en el entorno Matlab para las primeras pruebas, más adelante se desarrollaron en C. Primero de todo se demostró que ambos métodos daban lugar a prestaciones de recuperación similares a la matriz de pesos. Al utilizar la SVD se estudió también cual era el número de valores singulares adecuado a calcular, y con *α-Bisecting Spherical K-Means* se ajustarón sus parámetros iniciales. Después se compararon entre sí ambos métodos, dando lugar a prestaciones de recuperación similares. Todo ello se puede ver con más detalle en [115].

Siguiendo en la línea de la investigación del preprocesamiento estudiamos la influencia de la lematización semántica. En [86] se presentó el estudio comparativo entre el *stemming* y la lematización semántica basándonos en el modelo vectorial básico, el enfoque LSI y el método de *clustering α-Bisecting Spherical K-Means*.

La comparación entre *stemming*, que sería agrupar los términos por su raíz morfológica, y la lematización semántica, que sería agrupar los términos por el significado de cada uno según el contexto en el que se encuentre, dio como resultado que las prestaciones obtenidas con el *stemming* son claramente mejores que las obtenidas con la lematización semántica. Estos resultados se dan tanto con la matriz de pesos (modelo vectorial básico), como con el enfoque LSI (mediante la técnica SVD) y el *clustering*. Y todo ello se puede ver en [86].

Esta línea de investigación se volvió a trabajar más adelante mediante variaciones en los sistemas de lematización [87, 88]. Se llegó a mejorar parcialmente la lematización, pero así y todo el *stemming* seguía ganando tanto mediante el modelo de *clustering* usando el *α-Bisecting Spherical K-Means*, como el modelo LSI con la SVD.

En un momento dado de la investigación decidimos estudiar cómo funcionarían las técnicas de recuperación de información desarrolladas en otros entornos distintos a los sistemas de información, concretamente en el análisis de imágenes de test mamográfico. Los resultados los presentamos en los trabajos [123] y [124].

Concretamente se desarrollo una aplicación en Matlab que interpreta la imagen que representa la zona de estudio de la resolución de un patrón mamográfico, el cual permite controlar la calidad de los mamógrafos. Dicha resolución vendrá dada por el número de agrupaciones que se puedan identificar [123, 124].

Después de interpretar dicha imagen se genera una matriz dispersa que la modeliza y que es equivalente a la obtenida por un sistema de recuperación de información. Entonces se aplica la modelización LSI, una indexación semántica latente basada en la descomposición en valores singulares (SVD), y se elige como consulta un patrón de las franjas que forman las agrupaciones. Una vez realizada la consulta se seleccionan los "documentos" (posibles franjas de las agrupaciones) por orden de relevancia y se van identificando las agrupaciones. Con esta forma de proceder en [123] y en [124] se presentaron los resultados obtenidos mediante este proceso, pudiendo observar cómo el modelo desarrollado es útil en el análisis de imágenes de test mamográfico. Incluso al compararlo con otro método ya desarrollado se concluyó que se obtenían mejores resultados.

Paralelamente se nos invitó a hacer una ponencia en el I Congreso Internacional de Computación Paralela, Distribuida y Aplicaciones, en Mexico [125]. Dicho trabajo expuso a nivel general los sistemas de recuperación de información, sus modelos y aplicaciones, sobre todo enfocado al los métodos de *clustering* y al uso de la SVD como método LSI. Se presentarón también detalles sobre la importancia de la paralelización y las posibilidades de ella en ambos enfoques. También se presentó un caso de estudio y otras posibles aplicaciones de los métodos en entornos diferentes a la recuperación de información.

En este punto observamos que los métodos de *clustering* tenían importantes ventajas respecto a los métodos basados en el modelo LSI. Los primeros trabajan con matrices dispersas que permiten utilizar diversos tipos de metodos de compresión mejorando los costes de almacenamiento. Mientras que la SVD y otros métodos similares (QR, etc.) precisan almacenar matrices densas, es decir conllevan mayores costes de almacenamiento. Además, los costes de modelización también dan lugar a mayores costes computacionales. Y al final de todo las prestaciones de recuperación no son destacablemente mejores. Por todo ello decidimos centrarnos más en los métodos de *clustering.*

En esta línea trabajamos en la paralelización de los métodos de *clustering* desarrollados, para así estudiar su comportamiento en sistemas distribuidos. En [114] se mejoró y paralelizó el método α-*Bisecting Spherical K-Means*, paralelizando también las versiones en las que éste se basa: cálculo del vector concepto normalizado, α-*Bisection* y α-*Bisecting K-Means.*

Para la paralelización se llegó a la conclusión de que la mejor distribución era por documentos (por columnas). Cada elemento de proceso tiene un pequeño conjunto de la colección sobre la que trabaja, pero todos el mismo vector concepto (lo que conlleva el grueso de las comunicaciones, basadas en reducciones globales).

Los experimentos de este trabajo se centraron en la comparación de costes computacionales de la modelización de un sistema de recuperación de información mediante las versiones paralelas del α-*Bisecting K-Means* y del α-*Bisecting Spherical K-Means.* Ambos métodos mejoraban sus prestaciones computacionales al aumentar

el número de elementos de proceso cuando se trabajaba con una matriz de grandes dimensiones, concretamente se utilizó la colección TREC-DOE (ver apartado 2.5.5 en la página 63). Es decir, se reduce el tiempo de generación de los *clusters* que modelizan el sistema de recuperación de información. Así y todo la eficiencia obtenida no es demasiado buena (con pocos elementos de proceso llega enseguida a un 50 %) lo cual se debe a la dispersión de las matrices que da lugar a muy pocas operaciones que compensen las comunicaciones [114]. También se obtuvo una importante conclusión, que la inicialización de éstos métodos es su principal inconveniente ya que influencia mucho el resultado (número de iteraciones y por tanto coste computacional).

En relación a los sistemas distribuidos en recuperación de información, realizamos una presentación sobre la paralelización de modelos algebraicos de recuperación de información en la Universidad de San Luis (Argentina), como iniciativa de acercamiento de colaboración entre universidades.

Tras todo esto decidimos buscar métodos de *clustering* alternativos a la familia del *K-Means*, ya que estos tienen el principal problema de que sus parámetros iniciales no son conocidos a priori. Lo cual fácilmente conlleva a resultados insatisfactorios. Ello nos condujo a empezar a trabajar con el DBSCAN, el cual presentaba buenos resultados de recuperación de información (y también en otros campos). Aunque no había muchos estudios sobre este método en recuperación de información y menos en entornos de gran dimensionalidad, estuvimos realizando estudios previos en el entorno Matlab que dieron grandes esperanzas.

El primer trabajo que desarrollamos utilizando el método DBSCAN fue adaptarlo para trabajar con nuestra estructura de datos, para que trabajara con matrices dispersas. En un primer momento hicimos las pruebas con las propias imágenes que estudiaba el artículo original del DBSCAN [40] y vimos que todo funciona correctamente.

El siguiente paso era estudiar cómo respondía el método con matrices de un tamaño importante, es decir extender el problema a datos de gran dimensionalidad. Nuestras pruebas obtuvieron buenos resultados. A nivel de recuperación de información se obtuvieron mejores resultados incluso que con la matriz de pesos (modelo de referencia) y computacionalmente los costes aumentaron considerablemente en la fase de modelización, el aumento de costes en esta fase eran compensados por la calidad de recuperación en la fase de evaluación. Todas estas pruebas se hicieron fundamentalmente a través del entorno Matlab.

Tras observar que el DBSCAN podía trabajar bien con matrices dispersas de gran dimensionalidad, pasamos a afrontar el siguiente obstáculo, la elección de los parámetros iniciales del método: mínimo número de elementos para formar un *cluster* y la distancia mínima para determinar que dos elementos son vecinos. Es cierto que este método no precisa determinar el número de *clusters* que se van a formar y que para los mismos parámetros iniciales no importa por qué elementos se empiece. Pero también es cierto que la elección adecuada de sus parámetros no es conocida a priori, y que en función de su elección los resultados pueden ser buenos o muy malos.

Los autores del algoritmo presentaron un método que permite determinar dichos parámetros, pero dicho método sólo es funcional con sistemas de muy poca dimensionalidad (se presenta para dos dimensiones), de tamaño reducido, y siempre hay un

factor de selección humano. Ya que se realiza un gráfico y de manera visual se determina qué valores son los que provocan un cambio fundamental. Lógicamente este sistema no es viable para matrices grandes donde se trabaja con cientos, miles o más dimensiones.

Tras varias ideas y pruebas poco fructíferas desarrollamos una pequeña variante al DBSCAN original, mediante una pequeña simplificación desarrollamos el VDBSCAN. Esta variante del DBSCAN deja de utilizar los conceptos de elementos núcleo y elementos borde, y se basa fundamentalmente en el de elementos alcanzables, y además, utiliza el mínimo número de elementos para decidir si un *cluster* se forma o pasa a ser ruido una vez ya está creado (ver más detalles en el apartado 3.4 en la página 85). Con estos cambios se obtuvo un método que generaba los mismos resultados que el DBSCAN original, pero mucho más rápido (casi la mitad de tiempo, ver estudios en el apartado 3.4.3 en la página 89) y el cual permitía fijar el mínimo número de elementos y encontrar heurísticamente la distancia mínima.

Realizamos un estudio exhaustivo de la influencia del mínimo número de elementos en relación al espectro de distancias mínimas y pudimos llegar a la siguiente conclusión. En sistemas de recuperación de información basados en colecciones de documentos (tales como las que se han trabajado en esta tesis) el mínimo número de elementos se puede fijar a dos (ver estudio detallado en el apartado 3.5.1 en la página 93). Ahora nos queda por concluir si esta situación se puede extender a otros sistemas de recuperación de información, o si en otros se puede fijar también aunque sea a otro valor.

A continuación, desarrollamos un método heurístico de coste aceptable que encuentra un máximo local para determinar el valor de la distancia mínima entre dos elementos para considerarlos vecinos (ver método en el apartado 3.5.2 en la página 95). Mediante este método que en pocas iteraciones da un buen valor para la distancia mínima, el VDBSCAN consigue mejorar las prestaciones de recuperación de los métodos de *clustering* de la familia *K-Means* que se han desarrollado en esta tesis.

Con todo ello se ha conseguido un método que presenta mejores prestaciones de recuperación de información, que computacionalmente conlleva unos costes de evaluación un poco mayores pero más que aceptables, y que sobre todo no tiene problemas a la hora de elegir sus parámetros de entrada.

Una vez desarrollado con éxito el nuevo método VDBSCAN pasamos a trabajar en una versión funcional en C que permitiera trabajar con colecciones de gran tamaño. Luego estudiamos su funcionamiento y prestaciones en entornos distribuidos y paralelos. Con lo cual se desarrollo una versión paralela del mismo y se estudio bajo los mismos casos que el α-*Bisecting K-Means* y del α-*Bisecting Spherical K-Means*.

Esta fase final de la tesis, que aborda los desarrollos del VDBSCAN, su fijación de parámetros, el estudio de su paralelización, y los estudios comparativos con los anteriores métodos de *clustering* descritos en la tesis, se ha reflejado en diferentes publicaciones. Concretamente en dos artículos ya publicados y otros que están pendiente de publicación. En [126] se presentó el estudio comparativo entre las versiones paralelas de los algoritmos VDBSCAN y el α-*Bisecting Spherical K-Means*, centrado en la comparación de prestaciones. Por su parte, en [127] se centró la comparación de los métodos α-*Bisecting Spherical K-Means* y VDBSCAN en los sistemas de re-

cuperación de información, y se enfocó al estudio de la calidad de recuperación y al estudio de los costes computacionales. Además, toda la parte de presentación y desarrollo del método VDBSCAN en sí mismo, así como la parte de fijación de sus parámetros, está en fase de publicación.

6.1.2. Publicaciones de la tesis

A Comparision of Experiments with the Bisecting-Spherical K-Means Clustering and SVD Algorithms

Datos importantes

AUTORES:	D. Jiménez ; V. Vidal ; C. F. Enguix
AÑO:	2002
PUBLICACIÓN:	Actas de las I Jornadas de Tratamiento y Recuperación de la Información

Resumen En este artículo proponemos una versión modificada del algoritmo de *clustering Spherical K-Means*, el *Bisecting Spherical K-Means*. El algoritmo de *clustering Bisecting* se usa para determinar el conjunto inicial del *Spherical K-Means*. Se han preparado un conjunto de experimentos para comparar la SVD con un números diferentes de valores singulares para encontrar una solución óptima. Análogamente, se ha estudiado el *Bisecting Spherical K-Means* con diferentes números de clusters. Y finalmente se han comparado ambas técnicas respecto a ratios de precisión vs cobertura (*recall*).

The Influence of Semantics in IR using LSI and K-Means Clustering Techniques

Datos importantes

AUTORES:	D. Jiménez ; E. Ferretti ; V. Vidal ; P. Rosso ; C.F. Enguix
AÑO:	2003
PUBLICACIÓN:	Proceedings of the International Symposium on Information and Communication Technologies

Resumen Se estudia la influencia de la semántica en la fase de preprocesamiento de un sistema de recuperación de información. Concretamente se compara las prestaciones alcanzadas con el *stemming* y la lematización semántica como métodos de preprocesamiento. Tres técnicas se han usado en el estudio: el uso directo de la matriz de pesos, la técnica SVD como herramienta dentro del modelo LSI y un método de *clustering* concretamente el *Bisecting Spherical K-Means*. Aunque los resultados no parecen ser prometedores, puede ser que en el futuro sean mejorados.

Utilización de una Aproximación SVD para el Análisis de la Resolución de un Fantoma Mamográfico

Datos importantes

AUTORES:	Daniel Jiménez ; Vicente Vidal
AÑO:	2003
PUBLICACIÓN:	XXIX Reunión Anual de la Sociedad Nuclear Española

Resumen Se presenta un método de análisis automático y objetivo de la resolución de un mamógrafo, a través de un fantoma. Dicho método está basado en la teoría de la recuperación de información, concretamente en el modelo de indexación semántica latente (LSI). Dentro del modelo LSI se ha elegido como herramienta la descomposición en valores singulares (SVD).

La calidad de una mamografía es fundamental para el diagnóstico de múltiples problemas, aunque principalmente el cancer de mama. La resolución del mamógrafo es uno de los principales factores que influyen en la calidad de la mamografía, por lo cual es fundamental evaluar su correcto funcionamiento. Para ello se utilizan los fantomas que presentan ciertas zonas para el análisis de la resolución del mamógrafo.

Análisis de imágenes de test mamográfico mediante la técnica SVD

Datos importantes

AUTORES:	V. Vidal ; D. Jiménez ; G. Verdú
AÑO:	2003
PUBLICACIÓN:	Actas del XVIII Congreso de Ecuaciones Diferenciales y Aplicaciones y VIII Congreso de Matemática Aplicada (CEDYA)

Resumen Este trabajo presenta un método para comprobar de forma automatizada parte del buen funcionamiento de un mamógrafo, en concreto la resolución que alcanza. El método se basa en técnicas de recuperación de información, específicamente se utiliza una aproximación de bajo rango calculada mediante el método SVD, dentro del modelo LSI. La aplicación de este método permite conseguir buenos resultados.

Aspectos Computacionales de Modelos Algebraicos de Recuperación de Información

Datos importantes

AUTORES:	V. Vidal ; D. Jiménez
AÑO:	2003
PUBLICACIÓN:	Libro de actas del I Congreso Internacional de Computación Paralela, Distribuida y Aplicaciones

Resumen En este trabajo se da a los sistemas de recuperación de información un enfoque más computacional que el punto de vista de los sistemas de información. Se presenta la creación de un sistema de recuperación de información basándose en tres modelos algebraicos: vectorial, LSI y *clustering*. Además, se introduce una extrapolación de estás técnicas a otros ámbitos distintos, concretamente la reconocimiento de patrones sencillos en imágenes. Y por último, se comenta posibles paralelizaciones de los tres modelos descritos.

Information Retrieval and Text Categorization with Semantic Indexing

Datos importantes

AUTORES:	Paolo Rosso ; Antonio Molina ; Ferran Pla ; Daniel Jiménez ; Vicente Vidal
AÑO:	2004
PUBLICACIÓN:	Computational Linguistics and Intelligent Text Processing (Lecture Notes in Computer Science)

Resumen Se presenta el efecto de la indexación semántica usando sentidos de WordNet en la recuperación de información y en tareas de categorización de textos. Los documentos han sido etiquetados semánticamente usando un sistema de desambiguación de sentidos de palabra basado en modelos de Markov (Specialized Hidden Markov Models - SHMMs -). Los resultados preliminares muestran una pequeña mejoría en las prestaciones que fueron obtenidas sólo en la tareas de categorización de textos.

Text Categorization and Information Retrieval using WordNet Senses

Datos importantes

AUTORES:	Paolo Rosso ; Edgardo Ferretti ; Daniel Jiménez ; Vicente Vidal
AÑO:	2004
PUBLICACIÓN:	Proceedings of the Second International WordNet Conference

Resumen Se estudia la influencia de la semántica en la categorización de textos y en la recuperación de información. El método de los k vecinos más próximos fue usado para realizar la categorización de textos. Los resultados experimentales fueron obtenidos teniendo en cuenta para un término relevante de un documento su correspondiente significado según WordNet. Para la tarea de recuperación de información, tres técnicas fueron investigadas: el uso directo de la matriz de pesos, la SVD como técnica del modelo de indexación semántica latente (LSI) y la técnica de *clustering Bisecting Spherical K-Means*. Los resultados experimentales obtenidos teniendo en cuenta la semántica de los documentos muestran una mejora de prestaciones en la categorización de textos mientras que no parecen muy prometedores en la recuperación de información.

Parallel Implementation of Information Retrieval Clustering Models

Datos importantes

AUTORES:	Daniel Jiménez ; Vicente Vidal
AÑO:	2004
PUBLICACIÓN:	High Performance Computing for Computational Science - Vecpar 2004 (Lecture Notes in Computer Science)

Resumen Se presenta el desarrollo en paralelo de dos algoritmos de *clustering*, concretamente del α-*Bisecting K-Means* y α-*Bisecting Spherical K-Means*, basados en un α (menos evolucionado que el desarrollado en esta tesis). También se presentó una comparación de prestaciones computacionales para los métodos desarrollados. El conjunto de experimentos se llevaron a cabo en un *cluster* de PCs con 20 nodos biprocesadores y dos colecciones diferentes.

Heuristic Fixation of the Distance between two Cluster Elements at DBSCAN Algorithm

Datos importantes

AUTORES:	Daniel Jiménez ; Vicente Vidal
AÑO:	2009
PUBLICACIÓN:	Enviado a *Information Science* y corrigiendo la revisión

Resumen Se presenta un algoritmo heurístico el cual elimina uno de los parámetros del algoritmo DBSCAN, cuando se trabaja sobre sistemas de recuperación de información sobre colecciones de documentos. Concretamente se elimina el estudio necesario para establecer la proximidad entre dos elementos para considerarlos pertenecientes al mismo cluster. Este parámetro depende exclusivamente del conjunto de datos y se desconoce a priori. Es importante porque el algoritmo DBSCAN es uno de los principales modelos de *clustering* y se usa en multitud de contextos, destacando la recuperación de información.

VDBSCAN and α-Bisecting Spherical K-Means in distributed information retrieval systems

Datos importantes

AUTORES:	Daniel Jiménez González ; Vicente Vidal Gimeno ; Leroy Anthony Drummond
AÑO:	2010
PUBLICACIÓN:	Vecpar 2010

Resumen Se presenta un estudio comparativo entre los algoritmos VDBSCAN y el α-*Bisecting Spherical K-Means*. El primero es una variante del algoritmo DBSCAN, el cual produce los mismos resultados pero añade la posibilidad de seleccionar heurísticamente uno de los parámetros del DBSCAN. El segundo algoritmo es una variación del bien conocido *K-Means* la cual mejora las prestaciones de éste. Se han implementado las versiones paralelas de ambos métodos y se han comparado sus prestaciones.

Una comparación entre el VDBSCAN y el α-*Bisecting Spherical K-Means* en sistemas de recuperación de información

Datos importantes

AUTORES:	Daniel Jiménez ; Vicente Vidal
AÑO:	2010
PUBLICACIÓN:	Jornadas de paralelismo 2010

Resumen Se presenta un estudio comparativo entre las versiones paralelas del método VDBSCAN y del α- Bisecting Spherical K-Means. Ambas variantes son mejoras del DBSCAN y del *K-Means* respectivamente. Destaca que la paralelización está condicionada a la naturaleza propia del problema, el cual ya se presenta distribuido al estar los sistemas de recuperación de información formados por subcolecciones de documentos reales repartidas por distintos lugares.

VDBSCAN: Variant DBSCAN without parameters in information retrieval systems

Datos importantes

AUTORES:	Daniel Jiménez ; Vicente Vidal
AÑO:	2011
PUBLICACIÓN:	Pendiente de publicación

Resumen Se presenta el método VDBSCAN, una variante del conocido del método DBSCAN. Éste, cambiando algunos detalles en la forma de proceder, permite generar los mismo *clusters* que el algoritmo DBSCAN pero en menos tiempos. Además, permite fijar sus dos parámetros. El mínimo número de elementos por *cluster* se puede fijar al trabajar con sistemas de recuperación de información, y la distancia mínima para considerar dos elementos del mismo *cluster* se establece mediante un método heuristico, el cual también es presentado.

6.2. Conclusiones

La tesis ha conseguido alcanzar en gran medida los objetivos propuestos. Se ha desarrollado una variante del DBSCAN, el VDBSCAN, la cual permite determinar buenos valores para los parámetros del algoritmo, fijando uno de ellos y obteniendo el

otro a partir de un método heurístico. Esta variante tiene un coste computacional de uso menor que el DBSCAN, llegando en ocasiones a ser menor de la mitad de éste, y una calidad de recuperación igual. Además, su variante paralela trabaja de forma aceptable, aunque mejorable, en entornos distribuidos.

También se ha desarrollado una variante del conocido *K-Means*, el *α-Bisecting Spherical K-Means*, la cual mejora las prestaciones de calidad de recuperación aunque no alcanza la calidad de la matriz de pesos. Respecto a los costes temporales el algoritmo *α-Bisecting Spherical K-Means* es algo más lento que el *Spherical K-Means* pero más rápido que la matriz de pesos. En paralelización ninguno de los dos presentaba buenas prestaciones ante sistemas de recuperación de información basados en colecciones de documentos. Además de todo eso, la influencia de los parámetros iniciales es muy importante y éstos no se conocen a priori. Por todo ello se desarrollo el método VDBSCAN, el cual mejora todos estos aspectos.

Cuando uno trabaja con sistemas de recuperación de información basados en colecciones de documentos es posible fijar uno de los dos parámetros que presenta el algoritmo VDBSCAN, y el otro se puede ajustar mediante un método heurístico (también desarrollado en esta tesis) a partir del cual se obtiene un buen valor de proximidad, aunque no sea el óptimo. Dicho método tiene uno coste computacional aceptable, sobre todo porque sólo es necesario utilizarlo una vez, durante la fase de modelización, con lo cual no afecta al tiempo de respuesta en la fase de utilización del sistema, cuando se evalúan las consultas al sistema. Mediante esta fijación de parámetros se consiguen principalmente dos aspectos importantes. Por un lado, cuando se desconocen suficientes características del sistema de recuperación de información para determinar a priori los parámetros de modelización (éstos dependerán del método de modelización), se evita la necesidad de un estudio exhaustivo del sistema para establecer dichos parámetros, o se evita en su lugar las pruebas, si son posibles, con distintos parámetros hasta obtener unos aceptables. Por otro lado, también se evita el riesgo de elegir unos parámetros que den lugar a prestaciones de calidad de recuperación malas, ya que en muchas ocasiones no se dispondrán de medios para evaluar si la modelización realizada da buenos resultados hasta que el sistema esté en uso.

El algoritmo VDBSCAN junto a la heurística para seleccionar la distancia mínima entre dos elementos podría verse como un método que trabaja sin parámetros basado en el algoritmo DBSCAN. En la literatura también existe una variante del método DBSCAN, el algoritmo DBCLASD, que permite trabajar sin parámetros, con lo cual parecería innecesario todo lo desarrollado. Ahora bien, mientras el VDBSCAN no añade ninguna restricción a las impuestas por el DBSCAN, el DBCLASD asume una distribución uniforme de los elementos, lo cual no es generalizable a los sistemas de recuperación de información. Además, mientras que el VDBSCAN conlleva un coste variable añadido en la fase de modelización (debido al método heurístico) y luego en la fase de uso presenta un tiempo de respuesta inferior al DBSCAN, el DBCLASD tiene un coste computacional que tiende a doblar el coste alcanzado por el DBSCAN e incluso llega a triplicarlo al utilizar bases de datos de gran tamaño.

En cuanto a calidad de recuperación de información el VDBSCAN presenta la misma que el DBSCAN, y éste alcanza una calidad muy similar al conseguido con la matriz de pesos, llegando en varios momentos a mejorarla. Por supuesto, también

mejora claramente a la calidad del α-*Bisecting Spherical K-Means*, al menos en la curva media, en la óptima tienen una calidad parecida.

Por lo que respecta al coste temporal se concluye que en general en tiempo de respuesta absoluto se obtienen buenas prestaciones pero en el campo de la paralelización no, hay que mejorarlo. De todas formas, estos resultados no son tan malos como a priori podrían parecer, por dos motivos. Primero las prestaciones comparadas con el modelo de referencia se ven que son parecidas a éste, es decir el sistema de por sí tiene una naturaleza que no responde bien a la paralelización. Y segundo, se supone que el problema de por sí es distribuido y no hay un gasto añadido para aumentar prestaciones, la situación es la que hay y los tiempos de respuesta mejoran. En trabajos futuros se abordará la mejora de las prestaciones paralelas con el objetivo de optimizar los recursos y poder ampliar el sistema de recuperación de información.

Las prestaciones ante la paralelización dependen mucho de la distribución de la colección y de la modelización final. Ya que en la consulta de un cluster depende mucho de cómo esté distribuido éste (el balanceo de la carga) y si la distribución no está equilibrada los tiempos de comunicación pueden no ser compensados por la reducción de tiempo de evaluación. Eso se puede ver en cómo cambia mucho de una colección a otra el tiempo de respuesta en la paralelización, incluso cuando la colección es muy grande (DOE) no se obtienen claras mejorías. Habría que confirmar este supuesto estudiando la distribución de las consultas y los clusters a los que afecta. No se ha hecho puesto que los resultados presentados son una media de todas las consultas de prueba que las colecciones aportaban.

6.3. Trabajos Futuros

Las líneas de trabajo que abre la tesis o que pueden permitir mejorar los resultados obtenidos en ella, se pueden agrupar de la siguiente forma:

- Mejorar la calidad de recuperación de información de los métodos desarrollados.
- Mejorar las prestaciones computacionales de los métodos presentados.
- Mejorar la respuesta de los algoritmos desarrollados ante la paralelización.
- Generalizar los resultados obtenidos a otros entornos y sistemas de información.

También sería interesante comparar en el futuro el método VDBSCAN presentado con otros métodos, preferíblemente también basados en densidades, que tampoco precisan parámetros de entrada, destacando el método DBCLASD [58, 59] y el más moderno método MajorClust [107]. Aunque ambos presentan peores tiempos de respuesta, parece ser que en algunos casos, sobre todo el MajorClust, obtiene mejores prestaciones de recuperación.

Mejorar la calidad de recuperación

Dentro de la línea de mejora de la calidad de recuperación de información se pueden destacar dos vías: la utilización de métodos híbridos que permitan obtener las

ventajas de ambos y minimizar las desventajas; y el refinamiento de los resultados obtenidos.

En la literatura ya aparecen métodos híbridos, por ejemplo en [70, 128], la idea es aplicar y acoplar los resultados obtenidos al método VDBSCAN. Por ejemplo, se podría buscar un método LSI de reducción de dimensiones para minimizar los problemas de sinonimia y polisemia, y luego utilizar el VDBSCAN para realizar el *clustering* final.

Por otro lado también se puede refinar el resultado que el método obtiene, ello se puede hacer reutilizando el mismo método tras filtrar los resultados o hacer uso de otro método distinto. Sea el método que sea el utilizado, todos se basan principalmente en solventar mejor los problemas de sinonimia y polisemia. Por ejemplo en [29] proponen la utilización de una comparación de términos con términos, o en [89, 90] hablan de un método iterativo de agrupación y separación con la intervención del usuario. Como trabajo futuro queda pendiente la identificación dentro del *cluster* que el VDBSCAN ofrece como resultado de una consulta, los subgrupos que existen debido a la polisemia de los términos de la consulta. Dicho de otro modo, tras la consulta el VDBSCAN genera un *ranking* con los elementos del *cluster* seleccionado, pero dentro de éste seguramente habrán elementos relacionados con los distintos significados que la consulta puede adoptar. Bien, pues el reto ahora es diseñar un método (posiblemente una variante del VDBSCAN) para obtener los distintos *subclusters* que determinen los distintos significados de los términos de la consulta y ofrecer éstos como resultados alternativos.

Mejorar las prestaciones computacionales

La primera línea de trabajo para reducir los costes computacionales del VDBSCAN se centra en reducir su coste de modelización, para lo cual es menester reducir el coste del método heurístico que se basa en varias ejecuciones del VDBSCAN. El planteamiento de partida es estudiar si la heurística seguiría dando buenos resultados si trabajase con menos consultas de prueba o sólo con una muestra de la colección. Este enfoque también se puede generalizar al propio método, el cual, como otras variantes del DBSCAN, puede trabajar con muestras de la colección para determinar los *clusters* en lugar de trabajar en todo momento con toda la colección.

Dentro de esta línea de trabajo se quiere también generalizar que el DBSCAN y el VDBSCAN dan lugar a los mismos resultados con los mismos parámetros de entrada (dan lugar a las mismas prestaciones de recuperación). Para ello se pretende hacer un estudio más riguroso y teórico de ambos métodos. Esto conllevaría la posibilidad de generalizar el uso de la heurística desarrollada en la tesis al método DBSCAN y por tanto, a todos sus usos y posiblemente a muchas de sus variantes.

Sería también interesante estudiar la importancia e influencia de las estructuras de datos utilizadas a la hora de implementar los métodos. Aunque en esta tesis se ha utilizado una estructura vectorial y alta dimensionalidad, en la literatura se suele hablar de los R*-Tree como una estructura óptima para el DBSCAN que reduce su coste de $\mathcal{O}\left(n^2\right)$ a $\mathcal{O}\left(n \log n\right)$ cuando se trabaja con baja dimensionalidad.

Mejorar la paralelización

Al buscar mejoras computacionales se ha hablado de utilizar muestras de la colección, bien pues esta línea de trabajo también influye en la mejora de la paralelización, ya que puede conllevar una disminución de las comunicaciones y por tanto minimizar el principal problema de la paralelización.

Para minimizar los tiempos de comunicación en la paralelización se podría pensar en crear un sistema de recuperación de información independiente en cada unidad de proceso, y luego una vez evaluado cada uno de ellos desarrollar un método que integre todos los *rankings* generados [4], en este caso el enfoque de trabajo futuro no sigue este camino, se basa más en desarrollar un método distribuido. De hecho, la línea de trabajo está más relacionada a aplicar el método de *clustering* a cada unidad de proceso y luego integrar los *clusters* generados para trabajar con ellos de forma distribuida, ya en [70] se menciona este tipo de enfoque para el DBSCAN.

Dentro del enfoque de la paralelización una línea importante es la de realizar los estudios en distintos entornos distribuidos, de hecho se tiende a una implementación donde las comunicaciones sean a través de Internet, y ver si entonces los comportamientos obtenidos permitirían llegar a conclusiones similares. Estos entornos describirían una situación mucho más real.

Generalizar los resultados

Existen tres líneas fundamentales de generalizar los resultados. Primero sería estudiar si el VDBSCAN es utilizable en todos los sistemas de información y entornos en los que el DBSCAN lo es, no sólo en la recuperación de información. Aspecto que parece cumplir debido a su similitud casi total al DBSCAN.

Segundo, extender la eliminación de parámetros utilizada en el VDBSCAN al propio DBSCAN y otras variantes de éste que funcionan bien en otros contextos, para así aprovechar al máximo las características de ambos.

Y tercero, hacer los estudios mediante otros métodos de evaluación, para así poder generalizar los métodos a sistemas o entornos en los que no existan consultas de prueba (o datos similares). Incluso habría que extender el método heurístico también a otros métodos de evaluación y así no depender de colecciones con consultas de prueba para poder aplicar dicha heurística.

Bibliografía

[1] Duran, B., Odell, P.: Cluster analysis: A survey. Lecture Notes in Economics and Mathematical Systems (1974)

[2] Faloutsos, C., Oard, D.W.: A survey of information retrieval and filtering methods. Technical report, Electrical engineering Department, University of Maryland (1995)

[3] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys **31** (1999) 264–323

[4] Greengrass, E.: Information retrieval: A survey. Technical report, UMBC Center for Architectures for Data-Driven Information Processing (2001)

[5] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17** (2001) 107–145

[6] Andritsos, P.: Data clustering techniques. Technical Report CSRG-443, Department of Computer Science, University of Toronto (2002)

[7] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: part i. SIGMOD Rec. **31** (2002) 40 – 45

[8] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Clustering validity checking methods: part ii. SIGMOD Rec. **31** (2002) 19 – 27

[9] Kotsiantis, S., Pintelas, P.: Recent advances in clustering: A brief survey. WSEAS Transactions on Information Science and Applications **1** (2004) 73 – 81

[10] Langville, A.N., Meyer, C.D.: A survey of eigenvector methods of web information retrieval. The SIAM Review **47** (2005) 135–161

[11] Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)

[12] Berry, M.W., Browne, M.: Understanding Search Engines: Mathematical Modeling and Text Retrieval. SIAM (2005)

[13] Skillicorn, D.B.: A generalisation of indexing for parallel document search. Technical report, Department of Computer Science, The Australian National University (1995)

[14] Kobayashi, M., Dupret, G., King, O., Samukawa, H.: Efficient estimation of singular values for searching very large and dynamic web databases. Technical report, IBM (2000)

[15] Tiun, S., Abdullah, R., Kong, T.E.: Automatic topic identification using ontology hierarchy. In: Proceedings of the 2nd International Conference on Intelligent Text Processing and Computational Linguistics. Lecture Notes on Artificial Intelligence, Springer-Verlag (2001)

[16] Grossman, D.A., Holmes, D.O., Frieder, O.: A parallel dbms approach to ir in trec-3. In: Proceedings of the Third Text Retrieval Conference (TREC-3), NIST Special publications. (1994) 279–288

[17] Fujii, A., Itou, K., Ishikawa, T.: A method for open-vocabulary speech-driven text retrieval. In: EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Association for Computational Linguistics (2002) 188–195

[18] Vailaya, A., Zhong, Y., Jain, A.K.: A hierarchical system for efficient image retrieval. In: Proc. Int. Conf. on Patt. Recog. (1996) 356–360

[19] Jain, A.K., Vailaya, A.: Image retrieval using color and shape. Pattern Recognition **29** (1996) 1233–1244

[20] Gupta, A., Jain, R.: Visual information retrieval. ACM **40** (1997) 70–79

[21] Dhillon, I.S., Fan, J., Guan, Y.: Efficient clustering of very large document collections. In: Data Mining for Scientifec and Engineering Applications. Kluwer Academic Publishers (2001) Invited Book Chapter.

[22] Ding, C.H.Q.: A similarity-based probability model for latent semantic indexing. In: SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM (1999) 58–65

[23] Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Technical report, IBM (2000)

[24] Karypis, G., Han, E.H.: Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical report tr-00-0016, University of Minnesota (2000)

[25] Baek, D., Lim, H., Rim, H.C.: Latent semantic indexing model for boolean query formulation. In: Research and Development in Information Retrieval. (2000) 310–312

[26] Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, Springer-Verlag New York, Inc. (1994) 173–181

[27] Deerweter, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41** (1990) 391–407

[28] Berry, M.W., Dumais, S.T., O'Brien, G.W.: Using linear algebra for intelligent information retrieval. SIAM Rev. **37** (1995) 573–595

[29] Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM **41** (1999) 335–362

[30] Dowling, J.: Information Retrieval using Latent Semantic Indexing and a Semi-Discrete Matrix Decomposition. PhD thesis, Monash University (2002)

[31] Kolda, T.G.: Limited-Memory Matrix Methods with Applications. PhD thesis, University of Maryland at College Park, Applied Mathematics Program. (1997)

[32] Berry, M.W., Fierro, R.D.: Low-rank orthogonal decompositions for information retrieval applications. Numerical Linear Algebra with Applications **1** (1996) 1–27

[33] Letsche, T.A., Berry, M.W.: Large-scale information retrieval with latent semantic indexing. Information Sciences **100** (1997) 105–137

[34] Kobayashi, M., Dupret, G., King, O., Samukawa, H.: Efficient estimation of singular values for searching very large and dynamic web databases. Technical report, IBM (2000)

[35] Aslam, J., Leblanc, A., Stein, C.: A new approach to clustering. In: Workshop on Algorithm Engineering. (2000)

[36] Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Knowledge Discovery and Data Mining. (2001) 269–274

[37] Choo, J., Jiamthapthaksin, R., Chen, C.S., Celepcikay, O.U., Giusti, C., Eick, C.F.: Mosaic: A proximity graph approach for agglomerative clustering. In: Data Warehousing and Knowledge Discovery, 9th International Conference. (2007) 231 – 240

[38] Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. KDD-2000 Workshop on Text Mining (2000)

[39] Fasulo, D.: An analysis of recent work on clustering algorithms. Technical report, Department of Computer Science & Engineering (1999)

[40] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: In proceedings of 2nd International Conference on Knowledge Discovery and Data Mining. (1996) 226–231

[41] Zhao, Y., Karypis, G.: Evaluation of hierarchical clustering algorithms for document datasets. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, ACM Press (2002) 515–524

[42] Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data, New York, NY, USA, ACM (2006) 281 – 286

[43] Savaresi, S., Boley, D.L., Bittanti, S., Gazzaniga, G.: Choosing the cluster to split in bisecting divisive clustering algorithms. Technical report, University of Minnesot (2000)

[44] Savaresi, S.M., Boley, D.L.: On the performance of bisecting k-means and pddp. First Siam International Conference on Data Mining (2001)

[45] Hartigan, J.: Clustering Algorithms. Wiley (1975)

[46] Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In: Proc. 15th International Conf. on Machine learning, Morgan Kaufmann, San Francisco, CA (1998) 91–99

[47] Pizzuti, C., Talia, D., Vonella, G.: A divisive initialization method for clustering algorithms. In: PKDD'99 - Third Europ. Conf. on Principles and Practice of Data Mining and Knowledge Discovery. Volume 1704 of Lecture Notes in Artificial Intelligence., Prague, Springer-Verlag (1999) 484–491

[48] Pelleg, D., Moore, A.: Accelerating exact k-means algorithms with geometric reasoning. In: Proceedings of Fifth International Conference of Knowledge Discovery adn Data Mining. (1999) 277–281

[49] Alsabti, K., Ranka, S., Singh, V.: An efficient k-means clustering algorithm. In: Proceedings of IPPS/SPDP Workshop on High Performance Data Mining. (1998)

[50] Modha, D., Spangler, S.: Feature weighting in k-means clustering. Machine Learning **52** (2003)

[51] Kantabutra, S., Couch, A.L.: Parallel k-means clustering algorithm on nows. NECTEC Technical Journal **1** (2000) 243–248

[52] Xu, S., Zhang, J.: A hybrid parallel web document clustering algorithm and its performance study. Technical report, Department of Computer Science, University of Kentucky (2003)

[53] Dhillon, I.S., Modha, D.S.: A parallel data-clustering algorithm on distributed memory multiprocessors. In: Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence. Volume 1759. Springer-Verlag, New York (2000) 245–260

[54] Li, X., Fang, Z.: Parallel clustering algorithms. Parallel Computing **11** (1989) 275–290

[55] Forman, G., Zhang, B.: Distributed data clustering can be efficient and exact. ACM SIGKDD Explorations Newsletter **2** (2000) 34–38

[56] Forman, G., Zhang, B.: Linear speed-up for a parallel non-approximate recasting of center-based clustering algorithms, including k-means, k-harmonic means, and em. Technical report, HP Labs Technical Reports (2000)

[57] Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery **2** (1998) 169 – 194

[58] Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Knowledge Discovery and Data Mining. (1998) 58 – 65

[59] Xu, X., Ester, M., Kriegel, H.P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. In: Proc. Int. Conf. on Data Engineering (ICDE'98). (1998) 324–331

[60] Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proc. of 24rd International Conference on Very Large Data Bases, New York. (1998) 323 – 333

[61] Xu, X., Jäger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. Data Mining and Knowledge Discovery **3** (1999) 263 – 290

[62] Januzaj, E., Kriegel, H.P., Pfeifle, M.: Towards effective and efficient distributed clustering. In: Workshop on Clustering Large Data Sets, 3rd Int. Conf. on Data Mining (ICDM 2003). (2003) 49 – 58

[63] Januzaj, E., Kriegel, H.P., Pfeifle, M.: Dbdc: Density based distributed clustering. In Springer, ed.: Advances in Database Technology - EDBT 2004, Lecture Notes in computer Science (2004) 88–105

[64] Januzaj, E., Kriegel, H.P., Pfeifle, M.: Scalable density-based distributed clustering. In: PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, New York, NY, USA, Springer-Verlag New York, Inc. (2004) 231 – 244

[65] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. ACM SIGMOD Record **28** (1999) 49 – 60

[66] Kailing, K.: New Techniques for Clustering Complex Objects. PhD thesis, Ludwig-Maximilians Universität München, Munich, Germany (2004)

[67] Kailing, K., Kriegel, H.P., Kroeger, P.: Density-connected subspace clustering for high-dimensional data. In: Proc. SDM. 2004. (2004) 246 – 257

[68] Jahirabadkar, S., Kulkarni, P.: Isc - intelligent subspace clustering, a desity based clustering approach for high dimensional dataset. World Academy of Science, Engineering and Technology **55** (2009) 69 – 73

[69] Kriegel, H.P., Kroger, P., Renz, M., Wurst, S.: A generic framework for efficient subspace clustering of high-dimensional data. In: ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society (2005) 250 – 257

[70] El-Sonbaty, Y., Ismail, M.A., Farouk, M.: An efficient density based clustering algorithm for large databases. In: ICTAI '04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, USA, IEEE Computer Society (2004) 673 – 677

[71] Borah, B., Bhattacharyya, D.: An improved sampling-based dbscan for large spatial databases. In: Proceedings of the 2004 International Conference on Intelligent Sensing and Information Processing, IEEE Computer Society (2004) 92 – 96

[72] Zhou, B., Cheung, D.W.L., Kao, B.: A fast algorithm for density-based clustering in large database. In: PAKDD '99: Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining, London, UK, Springer-Verlag (1999) 338 – 349

[73] Liu, B.: A fast density-based clustering algorithm for large databases. In: Proceedings of the 2006 International Conference on Machine Learning and Cybernetics, IEEE Computer Society (2006) 996 – 1000

[74] Achtert, E., peter Kriegel, H., Pryakhin, A., Schubert, M.: Hierarchical density-based clustering for multi-represented objects. In: Workshop on Mining Complex Data (MCD 2005) at ICDM05. (2005)

[75] Birant, D., Kut, A.: St-dbscan: An algorithm for clustering spatial-temporal data. Data Knowl. Eng. **60** (2007) 208 – 221

[76] Zhao, Y., Zhang, C., Shen, Y.D.: Clustering high-dimensional data with low-order neighbors. In: WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, Washington, DC, USA, IEEE Computer Society (2004) 103 – 109

[77] Viswanath, P., Pinkesh, R.: l-dbscan: A fast hybrid density based clustering method. In: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, Washington, DC, USA, IEEE Computer Society (2006) 912 – 915

[78] Biçici, E., Yuret, D.: Locally scaled density based clustering. In: ICANNGA '07: Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part I, Berlin, Heidelberg, Springer-Verlag (2007) 739 – 748

[79] Huang, J., Ertekin, S., Giles, C.: Efficient name disambiguation for large-scale databases. Knowledge Discovery in Databases: PKDD 2006 (2006) 536 – 544

[80] Liu, S., Dou, Z., Li, F., Huang, Y.: A new ant colony clustering algorithm based on dbscan. In: Proceedings of the 3rd International Conference on Machine Learning and Cybernetics. (2004) 1491 – 1496

[81] Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: SSTD, Springer (2005) 364 – 381

[82] Birant, D., Kut, A.: Spatio-temporal outlier detection in large databases. In: Proceedings of the 28th International Conference on Information Technology Interfaces. (2006) 179 – 184

[83] Roy, S., Bhattacharyya, D.K.: An approach to find embedded clusters using density based techniques. In: Distributed Computing and Internet Technology, Second International Conference. (2005) 523 – 535

[84] Ruiz, C., Spiliopoulou, M., Ruiz, E.M.: C-dbscan: Density-based clustering with constraints. In: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. (2007) 216 – 223

[85] Brecheisen, S., Kriegel, H.P., Kröger, P., Pfeifle, M.: Visually mining through cluster hierarchies. In: Proceedings of the Fourth SIAM International Conference on Data Mining. (2004) 400 – 412

[86] Jiménez, D., Ferretti, E., Vidal, V., Rosso, P., Enguix, C.F.: The influence of semantics in ir using lsi and k-means clustering techniques. Proceedings of International Symposium on Information and Communication Technologies (ISICT) (2003) 286–291

[87] Rosso, P., Molina, A., Pla, F., Jiménez, D., Vidal, V.: Information retrieval and text categorization with semantic indexing. Lecture Notes in Computer Science **1** (2004) 596–600

[88] Rosso, P., Jimenez, D., Vidal, V.: Text categorization and information retrieval using wordnet senses. Proceedings of Second International Wordnet Conference (GWC 2004) (2004) 299–304

[89] Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W.: Scatter/gather: a cluster-based approach to browsing large document collections. In: SIGIR'92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press (1992) 318–329

[90] Cutting, D.R., Karger, D.R., Pedersen, J.O.: Constant interaction-time scatter/gather browsing of very large document collections. In: SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (1993) 126 – 134

[91] Shankar, S., Karypis, G.: Weight adjustment schemes for a centroid based classifier. Technical report, University of Minnesota, Department of Computer Science (2000)

[92] Aslam, J., Pelekov, K., Rus, D.: Information organization algorithms. In: Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR). (2000)

[93] Hearst, M.A.: Tilebars: Visualization of term distribution information in full text information access. In: Proceedings of the Conference on Human Factors in Computing Systems, CHI'95. (1995)

[94] Porter, M.F.: An algorithm for suffix stripping. Program **14** (1980) 130–137

[95] Manning, C.D., Sch(ü)tze, H.: Foundations of Statistical Natural Language Processing. The (MIT) Press (1999)

[96] Miller, A.: Wordnet: lexical database for english. Communications of the ACM **38** (1995) 39–41

[97] Wang, B.B., Mckay, R.I.B., Abbass, H.A., Barlow, M.: A comparative study for domain ontology guided feature extraction. In: ACSC '03: Proceedings of the 26th Australasian computer science conference, Australian Computer Society, Inc. (2003) 69–78

[98] Gonzalo, G., Verdejo, F., Chugur, I., Cigarran, J.: Indexing with wordnet synsets can improve text retrieval. In: Proceedings of the COLING/ACL, 98 Workshop on Usage of WordNet for NLP (1998)

[99] Losada, D., Barreiro, A.: A logical model for information retrieval based on propositional logic and belief revision. The Computer Journal **44** (2001) 410–424

[100] Raghavan, V.V., Wong, S.K.M.: A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science **37** (1999) 279–287

[101] Jung, Y., Park, H., Du, D.: A balanced term-weighting scheme for improved document comparison and classification. In: Text Mine'01. (2001) 68

[102] Golub, G.H., Loan, C.F.V.: Matrix Computations. third edition edn. The Johns Hopkins University Press (1996)

[103] Halkidi, M., Vazirgiannis, M.: Clustering validity assessment: Finding the optimal partitioning of a data set. In: ICDM. (2001) 187–194

[104] Koenemann, J., Belkin, N.J.: A case for interaction: A study of interactive information retrieval behavior and effectiveness. In: CHI. (1996) 205–212

[105] http://trec.nist.gov/: (Text retrieval conference (trec))

[106] Shaw, W.M., Wood, J.B., Wood, R.E., Tibbo, H.R.: The cystc fibrosis database: content and research opportunities. Library and Information Science Research **13** (1991) 347–366

[107] Stein, B., Busch, M.: Density-based cluster algorithms in low-dimensional and high-dimensional applications. In: Second International Workshop on Text-Based Information Retrieval (TIR 05), Koblenz, Germany. (2005) 45–56

[108] Guha, S., Rastogi, R., Shim, K.: Cure: An efficient clustering algorithm for large databases. In: SIGMOD Conference, ACM Press (1998) 73–84

[109] Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: SIGMOD Conference'96. (1996) 103–114

[110] Zhang, T., Ramakrishnan, R., Livny, M.: Birch: A new data clustering algorithm and its applications. Data Min. Knowl. Discov. **1** (1997) 141–182

[111] Su, Z., Yang, Q., Zhang, H., Xu, X., Hu, Y.: Correlation-based document clustering using web logs. In: Proc. of the 34th Hawaii International Conference on System Sciences, IEEE Computer Society (2001) 5022

[112] Iváncsy, R., Babos, A., Legány, C.: Analysis and extensions of popular clustering algorithms. In: Proc. of the 6th International Symposium of Hungarian Researchers on Computational Intelligence. (2005) 390–400

[113] Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: SIAM International Conference on Data Mining (SDM'03). (2003)

[114] Jiménez, D., Vidal, V. In: Parallel Implementation of Information Retrieval Clustering Models. Volume 3402/2005 of Lecture Notes in Computer Science. Springer Berlin (2005) 129–141

[115] Jiménez, D., Vidal, V., Enguix, C.F.: A comparison of experiments with the bisecting-spherical k-means clustering and svd algorithms. Actas congreso JOTRI (2002)

[116] Macfarlane, A., Robertson, S., McCann, J.: Parallel computing in information retrieval - an updated review. Journal of Documentation **53** (1997) 274 – 315

[117] http://www.mcs.anl.gov/research/projects/mpi/index.htm: (The message passing interface (mpi) standard)

[118] http://www.mpi forum.org/: (Message passing interface forum)

[119] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.: PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing. MIT Press (1994)

[120] http://www.netlib.org/pvm3/book/pvm book.html: (Pvm: Parallel virtual machine a users' guide and tutorial for networked parallel computing)

[121] O'Neill, C., Paice, C.D.: What is stemming? www.comp.lancs.ac.uk/computing/research/stemming/ (2003)

[122] Paice, C.D.: An evaluation method for stemming algorithms. Proceedings of the 17th ACM-SIGIR Conference on R&D in Information Retrieval (1994) 42–50

[123] Jiménez, D., Vidal, V.: Utilización de una aproximación svd para el análisis de la resolución de un fantoma mamográfico. Actas congreso 29 Reunión Anual Sociedad Nuclear Española 2003 (2003)

[124] Vidal, V., Jiménez, D., Verdú, G.: Análisis de imágenes de test mamográfico mediante la técnica svd. Actas congreso XVIII CEDYA 2003 (2003)

[125] Vidal, V., Jimenez, D.: Aspectos computacionales de modelos algebraicos de recuperación de información. Actas Congreso Internacional de Computación Paralela, Distribuida y Aplicaciones (2003)

[126] Jiménez, D., Vidal, V., Drummond, L.A.: Vdbscan and a-bisecting spherical k-means in distributed information retrieval systems. In: VECPAR'10 9th International Meeting, High Performance Computing for Computational Science. (2010) http://vecpar.fe.up.pt/2010/lpapers.php

[127] Jiménez, D., Vidal, V.: Una comparación entre el vdbscan y el a-bisecting spherical k-means en sistemas de recuperación de información. In: XXI Jornadas de Paralelismo, JP (SARTECO). (2010) 889–894

[128] Tsai, C.F., Liu, C.W.: Angel: A new efficient data clustering algorithm. In: Artificial Intelligence and Soft Computing - ICAISC 2006. (2006) 702 – 711