CLUSTERING DE DOCUMENTOS WEB BASADO EN UNA HIBRIDACIÓN VIABLE DE DBSCAN Y K-MEANS

MAURICIO DOMINGUEZ DIAZ DEL CASTILLO

Director: Ph.D. (c) MSc. CARLOS ALBERTO COBOS LOZADA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
GRUPO DE I+D EN TECNOLOGÍAS DE LA INFORMACIÓN
LÍNEA DE INTERÉS EN GESTIÓN DE LA INFORMACIÓN – OPTIMIZACIÓN
POPAYÁN, Septiembre de 2013

Tabla de Contenido

Lista de Figuras

Lista de Algoritmos

Lista de Tablas

Lista de Código Fuente

Capítulo 1: Introducción

1 Planteamiento del Problema

En Internet es posible encontrar un sinnúmero de documentos, portales y páginas de cualquier tema o tópico. Debido a la amplitud de temas de los recursos disponibles en Internet, el crecimiento exponencial de la cantidad de dichos recursos y la dificultad para el acceso rápido a información relevante a las necesidades específicas de los usuarios, se han creado una amplia variedad de sistemas de recuperación de información (IR, Information Retrieval), conocidos como buscadores web, que le entregan al usuario resultados ordenados por su relevancia frente a la consulta (normalmente expresada por palabras clave) en el menor tiempo posible [1, 2].

Los motores de búsqueda tradicionales como Google, Yahoo!, Bing, Ask y Altavista entre otros, presentan los resultados en una lista ordenada de documentos, pero estos están mezclados respecto a las temáticas o tópicos que tratan, por ejemplo: cuando se consulta por Java se pueden recibir resultados del lenguaje de programación, de la isla en indonesia, de una clase de café, de una localidad en Estados Unidos, entre otros. Este hecho implica que los usuarios deben gastar más tiempo revisando secuencialmente los documentos de distintos tópicos, algunos de ellos, que no son de su interés, haciendo que este modelo de visualización no sea el mejor [1, 3] y que sólo se revisen los primeros documentos de la primera página de resultados [4].

Por lo anterior, continuamente se proponen alternativas a los buscadores tradicionales, las cuales intentan mejorar su rendimiento y la forma de presentar la información al usuario. Para alcanzar dicho objetivo, en 1999 se propuso una técnica de visualización conocida como agrupamiento de documentos por temas, clustering de documentos web, o clustering de resultados web [5]. Entre los sistemas que utilizan esta técnica se encuentran: Yippy (http://search.yippy.com/), iBoogie (http://www.iboogie.tv), Carrot (http://www.carrot2.org), SnakeT (http://snaket.di.unipi.it), Credo (http://credo.fub.it/), Grokker (http://grokker.com/), KartOO (http://www.kartoo.com/), CIIRarchies (http://www.cs.loyola.edu/~lawrie/hierarchies/), WebCAT (http://ercolino.isti.cnr.it/webcat/), AISearch (http://www.aisearch.de), SRC (http://rwsm.directtaps.net), EigenCluster (http://eigencluster.csail.mit.edu), WhatsOnWeb (http://gdv.diei.unipg.it/view/tool.php?id=wow) y WebClust (http://www.webclust.com); donde la

organización de los documentos resultado de una consulta se presenta por temas, tópicos o grupos relacionados, lo que permite a los usuarios encontrar resultados útiles de forma más rápida y eficiente [6-15]. Estos sistemas son también conocidos como Web Clustering Engines (WDC).

Para realizar el agrupamiento de documentos web, se han propuesto diversos algoritmos usando diversas técnicas, pero en la actualidad no son suficientemente eficientes. Estos algoritmos presentan una precisión y recuerdo que varía entre 60% y 84% dependiendo del conjunto de datos de prueba, lo que implica que entre un 16% y 40% de los documentos son clasificados de manera incorrecta [16]. Además, la gran mayoría de estos algoritmos presentan dificultades para manejar el ruido presente en los documentos[1] y tienen dificultades para definir correctamente el número de grupos haciendo que la revisión de los resultados pierda efectividad [16].

Una de las áreas que mayor influencia ha tenido en WDC ha sido la minería de datos. En minería de datos existe una gran variedad de algoritmos para realizar clustering, estos se dividen principalmente en las siguientes categorías [17, 18]: jerárquicos, particionales, basados en densidad, basados en grillas, basados en modelos, entre otros. Los algoritmos más usados en WDC han sido los jerárquicos y los particionales [19]. Los primeros presentan la desventaja de tener una complejidad de O(n^2), hecho que dificulta su uso en la agrupación de documentos web, ya que el tiempo de respuesta en estos sistemas debe ser muy rápida, normalmente menor a 1-2 segundos. Los particionales tienen dificultades como: la sensibilidad a valores atípicos y a la selección de los centroides iniciales, la necesidad de definir previamente el número de grupos, que estos sólo contemplan formas esféricas de agrupación [20] y que tienen dificultadas para manejar distribuciones sesgadas (unos grupos con gran cantidad de documentos y otros grupos con muy pocos documentos) en los grupos de documentos.

Por otro lado, en la a*grupación basada en densidad*, algoritmos como *DBSCAN, BIRCH, SNN* y *RDBC*, encuentran áreas de alta concentración de elementos que están separadas por áreas con baja densidad [21]. De estos algoritmos, DBSCAN (y variaciones del mismo) es uno de los más populares por su habilidad para encontrar clusters con formas aleatorias, su complejidad es O(n log(n)), su tolerancia al ruido y el hecho que no necesita conocer el número de grupos antes de iniciar el proceso de agrupación [22]. Sin embargo, este algoritmo tiene problemas para manejar data sets

[1] En el WDC, los documentos hacen referencia a los resúmenes retornados por los buscadores tradicionales.

donde los clusters tienen densidades muy variadas, ya que en este escenario es muy difícil establecer los parámetros del algoritmo. La calidad de los grupos resultantes depende de la medida de distancia que se utilice y tiene un problema adicional muy importante que se presenta en conjuntos de datos con alta dimensionalidad, donde su complejidad puede alcanzar $O(n^2)$ y necesita de un espacio en memoria considerable para cargar la totalidad de la información [21, 23].

Considerando las características de los algoritmos particionales y los basados en densidad, así como las características especifica del proceso de agrupación de documentos web, en este proyecto se planteó la siguiente pregunta de investigación: ¿Es posible crear un algoritmo que reporte mejores resultados de precisión, recuerdo y medida F en el proceso de agrupamiento de documentos Web a los reportados en el estado del arte, al realizar una hibridación entre las técnicas DBSCAN y K-Means?

2 Aportes

En términos de investigación, el proyecto buscó generar conocimiento nuevo y útil para la comunidad académica y científica de recuperación de información, buscando una nueva solución al problema del agrupamiento de documentos web. Este nuevo conocimiento de tipo exploratorio y descriptivo, respondió a las preguntas: ¿Es posible lograr una mejora general del Agrupamiento de documentos web con el uso de un algoritmo que utiliza una hibridación de K-Means y DBSCAN?, y ¿Cuáles son las condiciones que hacen que este algoritmo mejore la precisión, el recuerdo y la medida F [24] del agrupamiento de documentos web?

Desde la perspectiva descriptiva, con este proyecto se propuso un conjunto de algoritmos híbridos basados en K-Means y DBSCAN, que permitió medir las posibilidades de su uso real en el agrupamiento de documentos web, buscando definir el número de grupos de documentos, su precisión, recuerdo/exhaustividad y tiempos de respuesta sobre colecciones de datos usadas comúnmente en el agrupamiento de documentos web.

3 Objetivos

A continuación se presentan los objetivos tal como fueron aprobados por el Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca.

3.1 Objetivo General

Proponer un algoritmo para el clustering de documentos web basado en una hibridación de DBSCAN y K-Means, que aproveche las fortalezas de los dos enfoques de clustering (basado en densidad y particional), evitando las debilidades que cada uno de ellos tiene por separado.

3.2 Objetivos Específicos

- Definir un algoritmo de clustering de documentos web basado en DBSCAN, K-Means y el modelo espacio vectorial para la representación de los documentos, capaz de definir automáticamente el número de grupos, manejar agrupaciones con diferentes densidades y etiquetar los grupos con frases de sus propios documentos.

- Diseñar e implementar un prototipo software, que permita determinar la calidad del proceso de agrupación de documentos web utilizando colecciones cerradas[2] de prueba y realizar comparaciones (basado en precisión, recuerdo y medida F ponderada [24]) entre el algoritmo propuesto y los algoritmos Bisecting K-Means, STC y Lingo[3].

- Establecer el grado de calidad del proceso de agrupación de documentos web del algoritmo hibrido a través de un conjunto de pruebas con usuarios, quienes evaluaran la claridad y utilidad de las etiquetas, la pertinencia de los documentos a los grupos y el orden de los mismos en los grupos y comparar los resultados contra Carrot2 (lingo) [25].

4 Resultados

Como resultados de este proceso de investigación se tiene:

- Propuestas de algoritmos híbridos entre DBSCAN y K-Means que pueden ser probados en el entorno del Web Document Clustering, aplicación software que implementa los algoritmos propuestos sobre el framework de pruebas y validación para WDC e informes detallados de los resultados de ejecución de cada uno de los algoritmos para cada una de las consultas de los diferentes datasets de prueba.

[2] Datasets como los disponibles en http://www.unicauca.edu.co/~ccobos/wdc/wdc.htm
[3] Disponibles en la API de www.carrot2.org

- Monografía del trabajo de grado, que corresponde al presente documento, donde se detalla el trabajo realizado, los aportes, resultados obtenidos y las conclusiones y propuestas para trabajo futuro.
- Artículo científico donde se explica el trabajo de investigación realizado y los resultados obtenidos.

Capítulo 2: Marco Teórico

5 Recuperación de Información

"La recuperación de información es un área interdisciplinaria de estudio que busca las mejores formas de representar, almacenar, organizar y acceder ítems de información en forma automática [24]. Para entender mejor esta definición, es necesario pensar en ítems de información como documentos (normalmente desestructurados) que están relacionados con las solicitudes de búsqueda de un usuario [2].

Esta ofrece al usuario la posibilidad de realizar búsquedas sobre grandes cantidades de documentos teniendo en cuenta: concordancias parciales o las mejores concordancias frente a una solicitud de información, un mecanismo de inferencia basado en la inducción, un modelo de búsqueda probabilístico, la posibilidad de clasificar los documentos en múltiples temas, el uso de un lenguaje de consulta similar al natural implicando condiciones de consulta que son incompletas, y un despliegue de documentos ordenados por relevancia y con una alta posibilidad de equivocarse en el orden de presentación de dichos documentos [24, 26].

La recuperación de información ha tomado gran importancia desde 1940, y con el creciente uso de las computadoras se creó la posibilidad de manejar automáticamente grandes volúmenes de información. En este sentido, se ha definido una estructura general para un sistema de recuperación de información (SRI), compuesto principalmente por: Documentos (almacenados en bases de datos o directorios), Usuarios, Consultas (solicitudes), Resultados/Respuestas (documentos relacionados y ordenados por relevancia), Retroalimentación (del usuario al sistema) y el Proceso (software y hardware que realiza el proceso de recuperación de información) [2, 24, 26].

Los temas centrales de investigación en recuperación de información iniciaron con la definición de mecanismos eficientes de almacenamiento (índices, índices ponderados, índices invertidos, índices probabilísticas, clasificación automática de palabras claves, discriminación y representación), clasificación automática, estructuras de archivos, estrategias de búsqueda (modelo booleano, modelo de espacio vectorial, funciones de concordancia, búsqueda serial, agrupamiento representativo, retroalimentación, re-consultas, modelo probabilístico) y evaluación (rendimiento y

satisfacción del usuario) del sistema en una colección "controlada" de documentos [2, 24, 26, 27]. Con el tiempo, y específicamente el cambio que ha impreso Internet en la vida de todas las personas, la recuperación de información web o Búsqueda Web (uno de los servicios más esenciales de este ambiente [28-30]) ha tenido que tomar aportes conceptuales y metodológicos de una mayor cantidad de áreas de conocimiento. A este respecto, la Estadística y Probabilidad, la Inteligencia Artificial, el Reconocimiento de Patrones, el Procesamiento Paralelo y otras áreas han incorporado muchas otras técnicas "no tradicionales" de recuperación de información, entre ellas: redes bayesianas, lógica difusa, algoritmos genéticos, procesamiento de lenguaje natural, algoritmos concurrentes, almacenamiento distribuido; mientras que el estudio de datos multimedia, el manejo de múltiples idiomas, la navegación y la visualización de los datos ha tomado mucha mayor importancia [24, 31, 32].

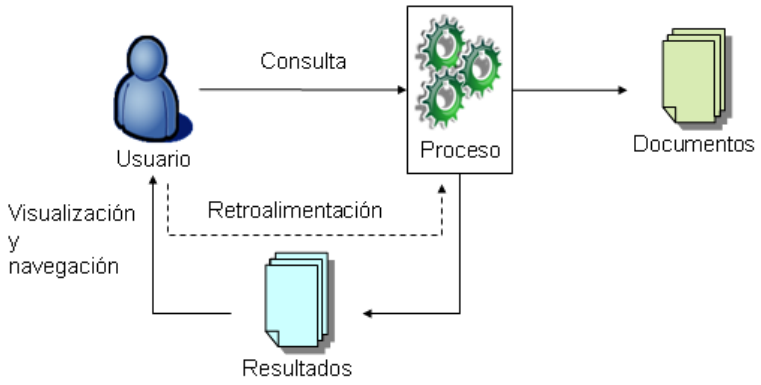

Figura 1 - Componentes de un SRI. Adaptado de [24, 26]

En la actualidad existen varios modelos de recuperación de información (RI), los más destacados [24, 26] son: el modelo booleano, el modelo de espacio vectorial y el modelo probabilístico. Además se encuentran algunas variaciones a estos tres primeros modelos, a saber: el modelo de conjuntos difusos, el modelo booleano extendido, el modelo del espacio vectorial generalizado, el modelo de indexado de semántica latente, el modelo de redes neuronales, el modelo de las redes bayesianas, el modelo de las redes de inferencia, el modelo de red de creencias, entre otros.

Como todo sistema software, los sistemas de recuperación de información deben ser evaluados antes de iniciar su funcionamiento en el ambiente real de producción. Dicha evaluación contempla aspectos como: análisis de funcionalidad, de unidad, de integridad, de tolerancia a fallos y de rendimiento (tiempo de respuesta al usuario, espacio requerido para almacenamiento adicional de

índices de búsqueda, la velocidad de los canales de comunicación, entre otros). Además, en los sistemas de recuperación de información se debe evaluar que tan precisa es la respuesta del sistema, conocida como la evaluación del rendimiento de la recuperación. Las medidas más conocidas para realizar esta última evaluación son la precisión, el recuerdo o la exhaustividad y la medida F [24].” (tomado de [33]).

6 Clustering de Documentos Web

“Esta alternativa de presentación de los resultados se basa en la denominada hipótesis del clúster [26], según la cual la agrupación de los documentos puede ser beneficiosa para los usuarios de un SRI, ya que es probable que los resultados relevantes se encuentren cerca el uno del otro en el espacio de los documentos [13], con lo que se puede reducir el tiempo que toma encontrar información útil. Este problema se diferencia del clustering general y del text clustering, ya que tiene un conjunto de restricciones y características que lo hacen único, ellos son:

- Definir automáticamente el número de clusters que serán creados.
- Generar clusters relevantes al usuario y asignar los documentos al clúster apropiado (los clusters en la colección de documentos tienen una distribución de tamaños extremadamente desigual)
- Definir etiquetas o nombres para los clusters que sean fácilmente comprensibles para los usuarios del sistema.
- Manejar clusters superpuestos (implicando que los documentos pueden pertenecer a más de un clúster).
- Reducir la alta dimensionalidad de la colección de documentos.
- Manejar datos dispersos que son muy comunes en las colecciones de documentos.
- Manejar el tiempo de procesamiento, esto es, el algoritmo debe ser capaz de trabajar con snippets y el tiempo de procesamiento debe ser menor o igual a 2.0 segundos.
- Manejar el ruido que es frecuente en los documentos.
- Opcionalmente, tener la habilidad de procesar los documentos de manera incremental a medida que el sistema los recibe o recupera.

En esta investigación, se parte del modelo de espacio vectorial donde los resultados de la búsqueda en Internet (en los buscadores tradicionales) se organizan en una Matriz de m términos x n documentos [24, 32]. Esta matriz es procesada y presentada al usuario en el orden más adecuado

(de los más relevantes a los menos). Aunque tradicionalmente la presentación de los resultados se realiza con una lista ordenada de documentos de acuerdo a un valor real que represente la relevancia del documento para el usuario, en los últimos años se ha considerado apropiado presentar los resultados en agrupaciones temáticas (clusters) como por ejemplo en Clusty (http://clusty.com), iBoogie (http://www.iboogie.tv), Carrot (http://www.carrot2.org), SnakeT (http://snaket.di.unipi.it/) y WebClust (http://www.webclust.com).

Otro aspecto que es importante al momento de estudiar o proponer un algoritmo que realice agrupación de documentos web, es el modelo de representación del documento. Los modelos más usados son [34]:

- Modelo de espacio vectorial [19, 24]: En este modelo se conciben los documentos como bolsas de palabras y la colección de documentos se representa con una matriz de M-términos por N-documentos. Cada documento se representa como un vector fila d en el espacio de términos tal que $d = \{w_1, w_2, \dots, w_n\}$, donde w_i es igual a la frecuencia del término (tf_i) normalizado en la colección multiplicado por la inversa de la frecuencia del documento para ese término, en lo que se conoce como el valor TF-IDF que se resume en la formula (1) o una variación de la misma. Adicionalmente en este modelo se usa la distancia de cósenos para medir el grado de similitud entre dos documentos o entre un documento y la consulta del usuario, calculado por la formula (2).

$$w_i = \frac{frecuencia_i}{\max(frequecia_i)} \times \log\left(\frac{N}{n_i}\right) \quad (1)$$

$$Sim(d,q) = \frac{\sum_{i=1}^{M} W_{i,d} \times W_{i,q}}{\sqrt{\sum_{i=1}^{M} W^2{}_{i,d}} \sqrt{\sum_{i=1}^{M} W^2{}_{i,q}}} \quad (2)$$

- Indexado semántico latente (Latent Semantic Indexing, LSI) [24, 35]: Es una variación del modelo de espacio vectorial en el que se usa teoría de descomposición de matrices como por ejemplo la descomposición en valores singulares (Singular Value Decomposition, SVD) [36]. Con la descomposición, se busca encontrar relaciones ocultas entre los términos de la colección y de esta forma encontrar los conceptos que representan mejor a los documentos. Además los *eigen* valores se puede usar para encontrar el número de grupos que deben ser generados.

- Modelo basado en ontologías [6, 11]: Es una variación del modelo de espacio vectorial en el que se usan ontologías como WordNet para encontrar las relaciones existentes entre los términos de la colección, como por ejemplo: sinónimos, hiperónimos, hiponimos, meronimos, entre otros. De esta forma la matriz que se construye es de M-Conceptos por N-Documentos.
- N-gramas [20]: En este modelo, el documento se representa como una secuencia de caracteres. Usando una ventana deslizante de tamaño n, recorre el documento para extraer todas las secuencias de n caracteres, denominadas n-gramas. Este modelo es tolerante a errores menores de ortografía y alcanza unos niveles menores de independencia del lenguaje cuando se usa con un algoritmo de *stemming*. La similitud se basa en el número de n-gramas compartidos entre los documentos.
- Modelo basado en frases [20]: En este modelo se buscan en los documentos los sufijos comunes de las frases y se construye un árbol de sufijos en el que cada nodo representa parte de una frase y se asocian con este los documentos que contienen ese sufijo. Otro enfoque es sintáctico, donde la información lingüística es usada para formar las frases. Por ejemplo, unir un adjetivo y un nombre para formar una frase [37].
- Modelo basado en conjuntos de palabras (términos) frecuentes [11, 38-41]: Un documento se representa como una transacción de términos que son frecuentes en una base de datos, similar al problema de encontrar reglas de asociación en minería de datos [42-47]. Usando algoritmos como Apriori o FP-growth se encuentran los términos frecuentes y cada documento tiene una similitud mayor o menor con esa lista de términos frecuentes, que luego se convierten en los nombres o etiquetas de los grupos.
- Representación enriquecida de documentos: "En este modelo, el documento está representado por un conjunto de términos lógicos y frases. Estos términos lógicos y frases describen las relaciones que han sido encontradas en el texto con una notación lógico cercana a la lógica multi-valuada de Michalski. Por ejemplo, una proposición tal como "para" en los fragmentos de la frase "... sistemas operativos para computadoras personales... ", sugiere una relación entre dos sustantivos "sistemas operativos" y "computadores personales". Entonces, estas relaciones se representan con un formato similar al de la lógica multi-valuada utilizada en la teoría de razonamiento plausible humano, es decir, sistema operativo (computadoras personales) " (traducción libre) [37].

Los algoritmos de agrupación se pueden clasificar en general en [17, 18]: jerárquicos, particionales, basados en densidad, basados en grillas, basados en modelos, entre otros. Los más usados para la agrupación de documentos web han sido los jerárquicos y los particionales [19].

Los algoritmos jerárquicos generan un dendograma o árbol de grupos, dicho árbol parte de una medida de similitud, entre las que están: single link, complete link y average link. En cuanto a la agrupación de documentos web, el algoritmo jerárquico que reporta mejores resultados en la precisión se llama UPGMA (Unweighted Pair-Group Method using Arithmetic averages) [18]. UPGMA es un algoritmo propuesto en 1990 [11, 48] que se basa en el modelo de espacio vectorial y usa un enlace promedio basado en la distancia de cósenos de las agrupaciones dividida por el tamaño de las dos agrupaciones que se están evaluando. UPGMA tiene las desventajas de tener un tiempo de ejecución O (n^3) y de ser estático en el proceso de asignación de los documentos a las agrupaciones." (tomado de [33]).

Para efectos de esta investigación, el trabajo se centró en dos enfoques, el agrupamiento particional y el basado en densidad. En la *Agrupación Particional,* los algoritmos realizan una división inicial de los datos en agrupaciones y luego mueven los objetos de un grupo a otro basado en la optimización de un criterio predefinido o función objetivo [17]. Los algoritmos más representativos que emplean esta técnica son: *K-Means (en la siguiente sección se explica en mayor detalle este algoritmo)*, *K-medoids* y *Expectation Maximization*. En *K-medoids [17]* cada agrupación queda representado por uno de los objetos que lo constituyen, al que se le llama medoid o "centroide real". De este modo, las agrupaciones serán subconjuntos de objetos que rodean al objeto medoid. Posteriormente, se define una función de distancia para medir la similitud entre un documento y un medoid. El algoritmo *K-medoids* presenta dos ventajas: Que no presenta limitaciones en los tipos de datos, y que es menos sensible a la presencia de valores atípicos (outliers) [49]. El algoritmo *Expectation-Maximization (EM)* es un algoritmo popular de refinamiento iterativo que asigna cada objeto a un grupo, de acuerdo a un peso que representa la probabilidad de admisión en el grupo.'

En WDC, el algoritmo particional más ampliamente usado es *K-Means*. Este algoritmo es muy popular ya que es fácil de implementar y su tiempo de complejidad es O(*n*); pero tiene desventajas muy importantes como su sensibilidad a valores atípicos y a la selección de los centroides iniciales, la necesidad de definir previamente el número de grupos y que los grupos obtenidos solo tienen formas esféricas [20].

La *Agrupación basada en densidad* [18] se utiliza principalmente para identificar clusters en bases de datos espaciales o geográficas. En este enfoque, el data set es particionado en celdas no-superpuestas y se construyen histogramas; las celdas que contienen una densidad de puntos relativamente alta[4] se consideran los posibles centros de clúster, mientras que las fronteras entre clusters se denominan "valles" del histograma. Esta técnica utiliza generalmente dos elementos para definir las agrupaciones: una distancia máxima entre los datos (ε) y un mínimo de puntos (*MinPts*) para definir los elementos densos dentro de un dataset.

Los principales representantes de este enfoque son OPTICS [50], DBRS [51] y DBSCAN (en una sección posterior se explica en mayor detalle este algoritmo) [52]. Estos han sido evaluados y comparados en sus principales campos de aplicación [22, 53]. OPTICS (Ordering Points to Identify the Clustering Structure) fue presentado por Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel y Jörg Sander en 1999 [54]. Su presentación general es similar a la de DBSCAN, pero ataca una de sus mayores debilidades: trabajar con agrupaciones de densidades variables. Para esto ordena (linealmente) los puntos en el dataset de manera que los que se encuentran linealmente cercanos se convierten en vecinos en el ordenamiento; además se almacena una distancia especial para cada uno, la cual representa la densidad requerida para que dos puntos pertenezcan al mismo clúster.

DBRS (Density-Based Spatial Clustering Method with Random Sampling) es un algoritmo propuesto por Wang y Hamilton en 2003 [55] para permitir encontrar de manera eficiente clusters en bases de datos espaciales con gran número de datos, al tiempo que considera en su proceso la información de tipo no-espacial y la pureza de los clusters. El proceso de DBRS se basa en la hipótesis que una agrupación puede ser definida con un mínimo de puntos centrales (llamados *skeletal points*) y sus vecinos. Un clúster puede tener más de *MinPts*, pero analizar sus vecindarios es inútil porque ya se sabe que estos pertenecen a un grupo, y si un punto sin clasificar pertenece a un grupo, seguramente será descubierto al analizarlo.

Para encontrar los puntos centrales, en lugar de realizar una búsqueda general del dataset (que es un problema NP-Completo) DBRS realiza una selección aleatoria de los puntos de muestra, explora sus vecindarios y, si estos se interceptan los combina en un solo grupo; aunque de esta manera los

[4] En base a los criterios de cada algoritmo, y los parámetros definidos para este.

puntos de muestra no sean los mismos puntos centrales, el número de consultas sobre los datos es bastante menor a la de DBSCAN, sobre todo en datasets con densidades muy variadas.

7 K-Means

K-Means es un algoritmo de clustering particional propuesto por Stuart Loyd en 1957 y fue publicado por primera vez en 1982 [56]. Debido a su baja complejidad [O(n)] y facilidad de implementación, es ampliamente utilizado para resolver una variedad de problemas de clustering, ente ellos en procesamiento de imágenes, reconocimiento de patrones y visión de máquina [57].

El algoritmo opera en un conjunto de vectores d-dimensionales, $D = \{x_i \,|i = 1, \dots , \ N\}$ donde $x_i \in R^d$ denota el *i*-esimo dato. El algoritmo inicia seleccionando *k* puntos como los representantes del clúster (centroides). Existen diversas formas para seleccionar estos puntos iniciales, entre ellas, se pueden seleccionar aleatoriamente puntos pertenecientes al dataset o usar el resultado de agrupar un sub conjunto de los datos. Una vez seleccionados el algoritmo itera en dos pasos básicos hasta alcanzar la convergencia. Estos son:

- *Paso 1 asignación de los datos:* cada punto se asigna al centroide más cercano.
- *Paso 2 Recalculo de los centroides:* cada centroide es reubicado en el centro (promedio) del todos los puntos asignados a él.

En la Figura 2 puede observarse un típico proceso de clustering con K-Means. Se puede ver como los centroides (representados por el símbolo +) son reubicados en cada iteración, y como cambia la pertenencia a los clusters (el color de cada punto) al avanzar el proceso de agrupamiento.

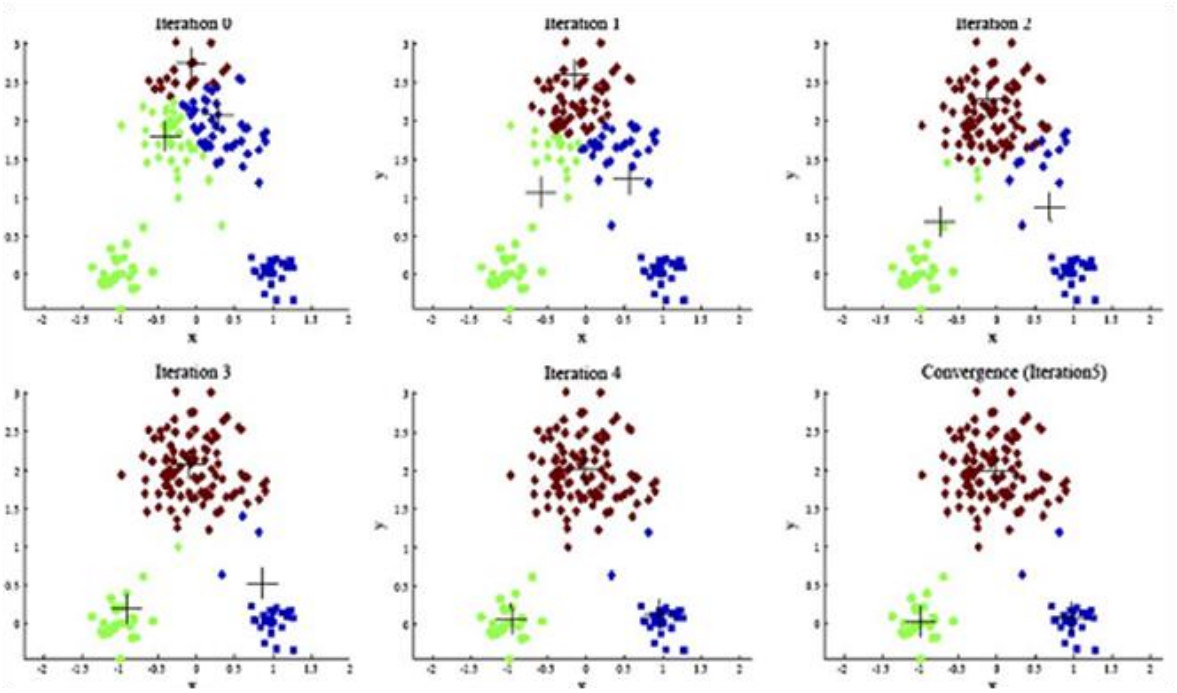

Figura 2 - Ejecución de K-Means. (tomado de [58])

Un detalle que debe resolverse al implementar este algoritmo es como definir la "cercanía" de los elementos en el paso de asignación. La forma típica de hacer esto es usando la distancia euclidiana[5], aunque es posible utilizar cualquier medida de distancia que cumpla con la desigualdad triangular. Además, debido a la forma de inicialización de los centroides, K-Means solo puede garantizar la selección de óptimos locales, más no globales, debido a que al inicializar los centroides, sus posiciones iniciales pueden generar soluciones equivocadas. Este problema es bastante común y puede ser contrarrestado en cierta medida al ejecutar el algoritmo en múltiples ocasiones con diferentes centroides iniciales, o haciendo una búsqueda local sobre la solución a la que convergió [58].

[5] Es la distancia entre dos puntos de un espacio euclidiano, la cual se deduce a partir del teorema de Pitágoras. Ej. en un espacio bidimensional, la distancia euclidiana entre dos puntos P1 y P2, de coordenadas (x1, y1) y (x2, y2) respectivamente, es: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

El algoritmo se muestra en mayor detalle a continuación en el Algoritmo 1.

1. Inicializar *k* puntos de manera aleatoria, o en base a un criterio previo.
2. Asignar cada dato al centroide más cercano.
3. Calcular el promedio de distancia dentro de cada clúster y reasignar los centroides.
4. Repetir 2 y 3 mientras los centroides cambien o no se cumpla el criterio de parada.

Algoritmo 1 - K-Means

En el ámbito del agrupamiento de documentos web (WDC), las principales desventajas de K-Means son: 1) requiere conocer de manera previa el número de clusters a generar y 2) su sensibilidad al ruido (valores atípicos). Debido a la naturaleza de las consultas en Internet y a la indeterminada cantidad de temas y combinaciones posibles, asignar un valor a priori al número de grupos y eliminar previamente todo el ruido es una tarea inviable. Por esto, y gracias a la velocidad del algoritmo, este suele ser utilizado como parte de una técnica más grande que realiza la inicialización de los centroides y explora el espacio de soluciones [59]. Esto consiste en realizar una cantidad predefinida de ejecuciones de K-Means, con diferentes valores para el parámetro *k*; los resultados obtenidos en cada ejecución se comparan, y el valor con la mejor aproximación a la solución es utilizado para refinar el proceso de clustering.

8 DBSCAN

Un conjunto abierto de puntos en el espacio euclidiano puede ser dividido en conjuntos de puntos interconectados. Para implementar esta idea de particionamiento de un conjunto finito requiere los conceptos de densidad, conectividad y fronteras. Un clúster, definido en términos de un conjunto denso de puntos interconectados se expande en dirección de esta densidad, por esto, las técnicas clustering basadas en densidad son capaces de descubrir clusters con formas arbitrarias y proveen una mayor protección a los valores atípicos, mostrando ventaja con respecto a los algoritmos basados en reubicación (ej. K-Means). En la figura 3 se muestra un ejemplo del tipo de agrupaciones que se pueden encontrar con este algoritmo [49].

Figura 3 - Clusters de formas irregulares

Con este concepto en mente, en 1996 Ester, Kriegel, Sander y Xu [60] propusieron el algoritmo DBSCAN (Density Based Spatial Clustering and Application with Noise). La idea principal detrás de esta propuesta es que, para cada objeto de un clúster, este debe contener un mínimo de vecinos (MinPts) en un radio (Eps) determinado.

Este algoritmo inicia tomando cualquier punto (*q*) del dataset y recuperando todos los vecinos dentro del radio de búsqueda, si *q* tiene al menos el mínimo de vecinos definido, estos se marcan como pertenecientes a un nuevo clúster y se procede a expandir de forma recursiva cada uno de los puntos encontrados dentro del vecindario. Durante el proceso de expansión, el algoritmo procede a evaluar los vecindarios de todos los puntos encontrados en el paso anterior, hasta que todos los puntos del clúster han sido explorados y agregados.

Si *q* no posee el mínimo número de puntos para ser considerado un conjunto denso, se procede a marcar el elemento como ruido y se prosigue con otro dato que aún no haya sido explorado, hasta que todo el dataset haya sido explorado. Cabe notar que un punto que haya sido marcado en una exploración inicial como ruido, puede ser integrado a un clúster, si este pertenece al vecindario de otro conjunto denso [61]. A continuación se detallan los pasos del algoritmo DBSCAN para realizar el proceso de clustering:

1. Seleccionar un punto (*p*) sin explorar.
2. Verificar el número de elementos dentro del vecindario eps de *p*. (n)
3. Sí n < MinPts Marcar *p* como ruido.
4. Si no:

5. Marcar cada punto dentro del vecindario como parte del cluster (*n*).
6. Explorar el vecindario de cada uno de estos puntos.
7. Agregar los puntos encontrados al cluster *n*.
8. *n* = *n* + 1.
9. Volver a 1 mientras haya puntos sin explorar.

Algoritmo 2 - DBSCAN

Debido a la naturaleza del enfoque usado por DBSCAN, su principal campo de uso ha sido en aplicaciones de análisis de datos geográficos, poblacionales y de baja dimensionalidad. Además, tiene como principales ventajas su facilidad de implementación, la flexibilidad para encontrar agrupaciones con formas arbitrarias, aún si estos se encuentran rodeados de otros clusters, es altamente tolerante a los valores atípicos y que únicamente requiere dos parámetros para su correcto funcionamiento.

Por otra parte, sus principales desventajas recaen en la dificultad para definir correctamente la función de distancia y los valores de Eps y MinPts. Además, este algoritmo tiene problemas para poder identificar correctamente clusters en un espacio donde estos poseen densidades variadas, y los elevados requisitos computacionales que necesita para datasets grandes [62]; es por esto que han surgido un conjunto de variaciones a este, tratando de compensar o corregir estas falencias, aplicándolas a un campo específico de la recuperación de información [63].

A pesar de su amplio uso en diversos campos de investigación, a la fecha no se encuentra una referencia de uso de DBSCAN en el ámbito del clustering de documentos web. Quiénes más se aproximan a este campo son Chehreghani et al con un algoritmo híbrido entre métodos basados en densidad, K-Means y técnicas de enlace simple, para mejorar el clustering de páginas web. Esta propuesta fue realizada en 2008 y se denomina "*Improving density-based methods for hierarchical clustering of web pages*" [64]. Este algoritmo toma primero todos los puntos del dataset y los organiza en una estructura especial que se encarga de ordenar los elementos de acuerdo a las distancias entre ellos. Posteriormente extrae estas distancias para con ellas realizar un proceso dinámico de clustering basado en densidad, en el que el parámetro de distancias varía según lo almacenado en esta estructura. Por último se realiza un proceso de enlace simple en el que se analizan los clusters obtenidos y se generan las jerarquías correspondientes según los hipervínculos y similitudes entre las diferentes agrupaciones.

El problema de esta propuesta radica en que los datasets utilizados para las validaciones del algoritmo no pertenecen al ámbito de WDC, por el contrario, estos pertenecen más a el campo del text clustering, por lo que, aunque presenta resultados muy positivos en comparación a K-Means, DBSCAN y otras técnicas, no se aplican al contexto del clustering de documentos web.

Capítulo 3: Híbridos Propuestos

9 Elaboración de los algoritmos

Para llegar a los algoritmos propuestos se inició realizando un análisis de las capacidades y características de DBSCAN y K-Means para encontrar la forma en que estos podrían realimentarse y permitir así un refinamiento de los resultados.

De esta forma se encontró que, mientras es posible definir los parámetros iniciales de K-Means utilizando una de las heurísticas existentes y aceptadas por la comunidad de recuperación de información y agrupamiento de documentos web, es necesario definir de manera dinámica según el dataset los valores de ejecución de DBSCAN debido a su sensibilidad a los valores iniciales y a que no hay precedente en la literatura que indique una forma adecuada de definir estos valores de manera automática.

Al llegar a una propuesta para el algoritmo se procede a implementarlo y probarlo utilizando un sub-conjunto de uno de los dataset de prueba de WDC para evaluar su desempeño y calidad de resultados. Para dichas evaluaciones, cada algoritmo es probado utilizando diferentes medidas para realizar el cálculo de los parámetros de DBSCAN.

Con los resultados preliminares del algoritmo se analiza que falencias presenta en su ejecución y se procede a tratar de subsanarlas; en caso de no encontrarse una mejor implementación del algoritmo, o de ver que las correcciones realizadas no están logrando generar una mejor calidad de los resultados del proceso de clustering, se procede a buscar un nuevo enfoque para combinar DBSCAN y K-Means que pueda ofrecer mejores resultados.

10 Conceptos Comunes

Comúnmente un algoritmo *hibrido* se refiere a un algoritmo que está hecho de dos o más algoritmos o enfoques de solución algorítmica diferentes y que son generalmente sencillos. Al combinar dos o más algoritmos se busca unir las ventajas o fortalezas de cada uno de ellos, al tiempo que se superan las desventajas que estos tienen por separado; pero los algoritmos o enfoques se pueden combinar

de muchas formas, resultando en una considerable variación de desempeño según como se combinen [65].

Con esto en mente se propone la hibridación de los algoritmos DBSCAN y K-Means, buscando obtener una mejora del proceso de Clustering de resultados web o Web Document Clustering.

Para lograrlo se necesitó realizar una modificación de los algoritmos originales, de manera que puedan realimentarse entre sí y puedan tener en cuenta los resultados obtenidos. En este caso, la modificación se hizo únicamente en K-Means (DBSCAN no fue modificado). A continuación se muestran los algoritmos usados para la hibridación:

10.1 Algoritmo y código fuente de K-Means modificado (C#)

1. Inicializar *k* puntos de manera aleatoria, o en base a un criterio previo.
2. Sí el punto (*p*) no se ha marcado como ruido:
3. Asignar cada punto al centroide más cercano.
4. Calcular el promedio de distancia dentro de cada clúster y reasignar los centroides.
5. Repetir 2 y 3 mientras los centroides cambien o no se cumpla el criterio de parada.

Algoritmo 3 - K-Means Modificado

11 Hibridación secuencial repetida

"Esta forma de hibridación utiliza un enfoque secuencial para la combinación de los algoritmos. De esta manera se inicia con un algoritmo de clustering y continúa con otro (que use una estrategia diferente al primero) para continuar el proceso.

Típicamente la hibridación secuencial consiste de un método que se caracterice por la exploración global, como por ejemplo un algoritmo genético, y concluye con una estrategia de refinamiento local. Se ha encontrado que este tipo de hibridación secuencial logra mejores resultados que los que se basan únicamente en la solución de 1 solo algoritmo, siempre que los algoritmos seleccionados y los parámetros de ajuste sean los adecuados.

La eficacia de una estrategia hibrida secuencial depende de los algoritmos específicos y los parámetros de ajuste que se utilicen en cada etapa del proceso. Debido a que cada algoritmo trabaja

de manera independiente, el progreso en cada fomento depende de que tan eficaz sea el método seleccionado para el problema que se está enfrentando, y la información recibida de los métodos ejecutados previamente.

Normalmente es imposible conocer de antemano que valores de los parámetros de ajuste funcionaran bien para un problema, y este problema se magnifica en las estrategias hibridas, ya que es necesario encontrar los parámetros adecuados para cada uno de los algoritmos que componen el híbrido. Aún más, se introducen nuevas incógnitas, como el orden de ejecución de los algoritmos y las estrategias para encontrar el criterio de parada." (Adaptado de [65])

Al tener en cuenta las características tanto de K-Means como de DBSCAN, en esta investigación se tomaron dos caminos para probar este enfoque; en el primero se realiza una realimentación continua entre K-Means y DBSCAN para buscar unos resultados más refinados y precisos.

El segundo método ejecuta varios K-Means diferentes entre los que selecciona el mejor usando el índice BIC, una vez seleccionado se procede a ejecutar DBSCAN tomando la información de esta solución y por último se ejecuta un nuevo K-Means usando los resultados arrojados por DBSCAN. A continuación se explican en mayor detalle ambas implementaciones:

11.1 Realimentación continua

Como se mencionó anteriormente, en esta implementación se ejecutan versiones de K-Means y DBSCAN en secuencia durante un tiempo o número de iteraciones predeterminado, utilizando los resultados de cada algoritmo para mejorar los resultados generales.

Para esto se inicia calculando un valor para *k* determinado por una heurística donde $k = \sqrt{\frac{numero\ de\ documentos}{2}} + 1$. Con este valor se ejecuta K-Means y, con sus resultados se busca el clúster más pequeño. El siguiente paso es encontrar el número de puntos que tiene este clúster, valor que es asignado a la variable *MinPts*, y encontrar la mayor distancia que separa dos puntos de este grupo (este valor es asignado a la variable *Eps*). Estos valores son utilizados para ejecutar DBSCAN para marcar los elementos de ruido; con estos resultados se vuelve a ejecutar K-Means,

pasando por alto los datos marcados como anómalos y se repite el ciclo hasta que se cumpla una condición de detención.

Al concluir cada ciclo se agrega el resultado obtenido a un listado. Cuando el algoritmo cumple el tiempo o número máximo de ejecuciones, se selecciona el mejor resultado como respuesta al problema. A continuación se muestra el algoritmo:

1. $k = \sqrt{((numero\ de\ documentos)/2) + 1}$
2. ResultadosKMeans <- Ejecutar K-Means
3. Mientras no se cumpla la condición.
4. MenorCluster <- Encontrar el clúster más pequeño (ResultadosKMeans)
5. MinPts <- ContarPuntos(MenorCluster)
6. Eps <- MáximaDistancia(MenorCluster)
7. Ejecutar DBSCAN (MinPts, Eps)
8. ResultadosKMeans <- Ejecutar K-Means
9. Agregar Resultado al listado.
10. Retornar el mejor resultado.

Algoritmo 4 - Refinamiento Continuo

A continuación se presenta el código fuente de este algoritmo.

```
var dtInicial = DateTime.Now;
var contador = 0;
var matrizDistancias = new MatrizSimilitud(matrizDeDocumentos);
var mejores = new List<VectorSolucion>();
const int maximoNumeroDeGeneraciones = 5;

if (MisParametros.Kmax == 0)
       MisParametros.Kmax = Kmax.Definir(matrizDeDocumentos.Count());

var k = MisParametros.Kmax;
var resultadosDBSCAN = new int[matrizDeDocumentos.Length];
var kMeans = new KMeans();
var resultadosKMeans = kMeans.Cluster(matrizDeDocumentos, k, resultadosDBSCAN,
                                      numeroAleatorio);

while (Continuar(dtInicial, contador, maximoNumeroDeGeneraciones))
{
       var smallestCluster =
              CalculosDBSCAN.SmallestCluster(resultadosKMeans.Resultados, k + 1);
       MisParametros.ParametrosDBSCAN.Epsilon =
              CalculosDBSCAN.MaximumEpsilon(matrizDeDocumentos,
                     resultadosKMeans.Resultados, smallestCluster);
       MisParametros.ParametrosDBSCAN.MinPts =
              CalculosDBSCAN.PuntosCluster(resultadosKMeans.Resultados,
```

```
                                                        smallestCluster);
        var dbscan = new DBSCAN(MisParametros.ParametrosDBSCAN.Epsilon,
                                MisParametros.ParametrosDBSCAN.MinPts,
                                matrizDistancias, matrizDeDocumentos);
        resultadosDBSCAN = dbscan.BuildCluster();

        resultadosKMeans = kMeans.Cluster(matrizDeDocumentos, k,
                                resultadosDBSCAN, numeroAleatorio);

        contador++;
        var resultado = new VectorRG(numeroAleatorio, MisParametros);
        var resultados = resultadosKMeans.Resultados;
        CalculosDBSCAN.OrdenarResultadosDBSCAN(ref resultados);
        resultado.Grupos = resultados;
        resultado.Centroides = resultadosKMeans.Centroides.ToList();

        mejores.Add(resultado);
}
mejores.Sort();
MejorRespuesta = mejores[0];
```

Código Fuente 1 - Refinamiento Continuo

11.2 Mejores Resultados

Para esta versión del algoritmo, se optó por realizar una combinación que permitiera buscar la mejor solución de K-Means, utilizando la habilidad de DBSCAN para encontrar ruido. A diferencia de la versión anterior, en este algoritmo solo se ejecuta DBSCAN una vez, para lograr marcar el ruido, y con esta información se inicia la ejecución de K-Means.

Para intentar mejorar el desempeño de DBSCAN, se cambia la forma de seleccionar los parámetros que este utiliza para su proceso de agrupamiento. Para esto se define el valor máximo del número de grupos, utilizando la misma heurística que en el algoritmo anterior. A continuación se procede a ejecutar diferentes procesos de K-Means con valores incrementales de k, empezando desde $k = 2 \dots kMax$; con cada ejecución se comparan los resultados de la ejecución actual con la anterior y utilizando el índice BIC, se selecciona la mejor.

Al terminar este paso se calculan los valores para los parámetros de DBSCAN (MinPts y Eps) usando las mismas técnicas que en la hibridación anterior. Con estos valores se procede a ejecutar una iteración de DBSCAN con lo cual se busca marcar los elementos de ruido y así poder excluirlos del proceso de K-Means.

Con los resultados obtenidos por DBSCAN se procede a hacer un último ciclo de iteraciones de K-Means, donde ahora se maneja el valor de *k* dentro de un rango alrededor del número de grupos encontrados por DBSCAN y se realizan varias ejecuciones del algoritmo con cada valor de *k*; se comparan los resultados de estas ejecuciones de K-Means midiendo su índice BIC y se seleccionan las mejores para cada valor de *k* y la que presente el mejor resultado es seleccionada como la respuesta.

A continuación se muestra el algoritmo general de este método y el código fuente del algoritmo.

1. $kMax = \sqrt{\frac{numero\ de\ documentos}{2}} + 1$
2. Para *k=2;* hasta *k=kMax*
3. resultados <- EjecutarKMeans(k)
4. bicActual <- CalcularBIC(resultados)
5. **si** bicActual > mejorBIC
6. mejorResultado <- resultados
7. MenorCluster <- Encontrar el clúster más pequeño (ResultadosKMeans)
8. MinPts <- ContarPuntos(MenorCluster)
9. Eps <- MáximaDistancia(MenorCluster)
10. resultadosDBSCAN <- Ejecutar DBSCAN (MinPts, Eps)
11. kMax <- ContarClusters(resultadosDBSCAN)
12. Para *k=kMax -1* hasta *k=kMax + 1*
13. Para *i = 0* hasta *i = 2*
14. resultados <- EjecutarKMeans(k)
15. bicActual <- CalcularBIC(resultados)
16. si bicActual > mejorBIC
17. mejorResultado <- resultados
18. Agregar el mejor resultado al listado.
19. Retornar el mejor resultado del listado.

Algoritmo 5 - Mejores Resultados

```
var kMax = Kmax.Definir(matrizDeDocumentos.Count());
var mejores = new List<VectorSolucion>();
var hilos = new List<Thread>();

for (var k = 2; k <= kMax; k++)
{
       var nuevosParametros = MisParametros.Clone();
       nuevosParametros.Kmax = k;
```

```
        var nuevaIsla = new Isla
                            {
                                    MiAlgoritmoHibrido =
                                            new ContinuousFeedback1(nuevosParametros),
                                    MatrizDeDocumentos = matrizDeDocumentos,
                                    NumeroAleatorio = numeroAleatorio,
                                    Mejores = mejores
                            };
        var hilo = new Thread(nuevaIsla.EjecutarIsla);
        hilo.Start();
        hilos.Add(hilo);
}
var enProceso = true;
while (enProceso)
{
        enProceso = false;
        foreach (var hilo in hilos.Where(hilo => hilo.IsAlive))
        {
                enProceso = true;
        }
        if (enProceso) Thread.Sleep(10);
}
mejores.Sort();
MejorRespuesta = mejores[0];
```

Código Fuente 2 - Mejores Resultados

12 Hibrido paralelo

“Los algoritmos híbridos paralelos están diseñados para superar muchas de las falencias de los algoritmos en serie. En esta estrategia, múltiples algoritmos de optimización trabajan de manera simultánea para resolver un problema de manera colaborativa. En vez de contribuir secuencialmente, estos métodos trabajan juntos para buscar e identificar soluciones óptimas.

Un algoritmo híbrido paralelo requiere de liderazgo, comunicación, coordinación y responsabilidad, atributos que deben ser integrados desde el principio al algoritmo. En lugar de explorar y refinar los resultados por separado, este método permite realizar estas etapas de manera concurrente y sinérgica. Esto no solo acelera el proceso sino que aumenta las posibilidades de encontrar el óptimo global.” (Adaptado de >Averill, 2012 #1379<)

En este proyecto se realizó la hibridación paralela tomando como base principal para el proceso, los algoritmos realizados para la hibridación secuencial. Lo que cambia es que se realiza la ejecución en paralelo de uno de estos algoritmos utilizando diferentes valores para *k*, desde 2 hasta la heurística definida al principio de esta sección.

Una vez completada la ejecución de los algoritmos, se ordenan los resultados obtenidos según su pertinencia y se selecciona el mejor resultado obtenido como respuesta de la solución. A continuación se presenta el algoritmo general para la hibridación en paralelo y el código fuente del algoritmo.

1. $kMax = \sqrt{\frac{numero\ de\ documentos}{2}} + 1$
2. Para *k = 2* hasta *k = kMax*
3. Ejecutar algoritmo híbrido secuencial con *k* como parámetro
4. Agregar resultado de la ejecución al listado
5. Seleccionar el mejor resultado como solución al algoritmo

Algoritmo 6 - Híbrido Paralelo

```
var mejor = new VectorRG(numeroAleatorio, MisParametros);
var mejorKMeans = new ResultadosKMeans();
var kMax = Kmax.Definir(matrizDeDocumentos.Length);
var clustersDBSCAN = new int[matrizDeDocumentos.Length];
var mejorBIC = double.MaxValue;
var mejorK = 0;
var matrizDistancias = new MatrizSimilitud(matrizDeDocumentos);
for (var k = 2; k <= kMax; k += 2)
{
       var kmeans = new KMeans();
       var resultados = kmeans.Cluster(
                     matrizDeDocumentos, k, clustersDBSCAN, numeroAleatorio);
       var bicActual = Calculos.CalcularIndiceBIC(
                           matrizDeDocumentos.Length, matrizDeDocumentos,
                           resultados.Centroides.ToList(),
                           resultados.Resultados);
       if (bicActual > mejorBIC) continue;
       mejorBIC = bicActual;
       mejorKMeans = resultados;
       mejorK = k;
}
var cluster = CalculosDBSCAN.SmallestCluster(mejorKMeans.Resultados, mejorK);
var eps = CalculosDBSCAN.MaximumEpsilon(matrizDeDocumentos, mejorKMeans.Resultados,
                                        cluster);
var minPts = CalculosDBSCAN.PuntosCluster(mejorKMeans.Resultados, cluster);

var dbscan = new DBSCAN(eps, minPts, matrizDistancias, matrizDeDocumentos);
clustersDBSCAN = dbscan.BuildCluster();
var numeroClustersDbscan = Math.Abs(clustersDBSCAN.Max());
mejorBIC = double.MaxValue;
mejorKMeans = new ResultadosKMeans();

for (var i = numeroClustersDbscan - 1; i < numeroClustersDbscan + 2; i++)
{
```

```
        if (i < 2) continue;
        for (var j = 0; j < 2; j++)
        {
                var kmeans = new KMeans();
                var res = kmeans.Cluster(matrizDeDocumentos, i, clustersDBSCAN,
                                         numeroAleatorio);
                var bicActual = Calculos.CalcularIndiceBIC(matrizDeDocumentos.Length,
                                                           matrizDeDocumentos,
                                                           res.Centroides.ToList(),
                                                           res.Resultados);
                if (bicActual > mejorBIC) continue;
                mejorBIC = bicActual;
                mejorKMeans = res;
        }
}
var resultadosFinales = mejorKMeans.Resultados;
CalculosDBSCAN.OrdenarResultadosDBSCAN(ref resultadosFinales);
mejor.Grupos = resultadosFinales;
mejor.Centroides = mejorKMeans.Centroides.ToList();
MejorRespuesta = mejor;
```

Código Fuente 3 - Híbrido Paralelo

13 Hibrido con distancias variables

Como se ha descrito anteriormente, DBSCAN es un algoritmo muy versátil y útil para encontrar agrupaciones densas, pero presenta dificultades para identificar de manera acertada agrupaciones con densidades muy variables.

Para esto se propone una nueva estrategia de hibridación basada en el trabajo de Chehreghani et.al >Chehreghani, 2008 #1350<. En esta propuesta ellos utilizan una hibridación entre K-Means, un algoritmo híbrido y un algoritmo de enlaces para refinar los resultados del proceso de clustering. En su propuesta utilizan una versión modificada de K-Means con un valor elevado para *K*, una vez concluido este proceso se calculan las diferentes distancias de cada clúster encontrado y estas se utilizan para definir los posibles valores para la medida de distancia del algoritmo de densidad. Una vez definidos se ejecuta un proceso de exploración del espacio cercano para realizar la construcción de los grupos y la unión de aquellos que se encuentren lo suficientemente cerca como para poder ser considerados como un mismo clúster.

Partiendo de esta propuesta se realizó una hibridación de K-Means y DBSCAN que usa distancias variables. En primer lugar se ejecuta K-Means sobre el dataset utilizando un valor de *K* alto para de esta manera obtener un alto número de grupos densos. Una vez concluida la ejecución de K-Means

se analizan los resultados obtenidos para encontrar la mayor distancia entre los miembros del clúster para cada uno de los grupos encontrados y se ordenan los resultados de forma ascendente.

Una vez ordenadas las distancias se procede a ejecutar un proceso incremental de DBSCAN con cada uno de las distancias encontradas. El proceso incremental consiste en realizar el proceso tradicional de DBSCAN con cada una de las distancias encontradas en el proceso de K-Means. En cada ejecución de DBSCAN se marcan los clusters encontrados con la distancia actual. Al finalizar cada exploración, los elementos que fueron marcados como ruido se etiquetan como no explorados y se inicia una nueva instancia de DBSCAN con la siguiente distancia, ignorando los elementos que ya han sido clasificados.

El objetivo de este algoritmo que se presenta a continuación es encontrar de manera más eficaz los clusters dentro del espacio de exploración al incrementar en cada iteración el rango de exploración del vecindario en el algoritmo basado en densidad. Al buscar con una distancia más grande, se busca compensar la debilidad de DBSCAN para manejar agrupaciones de diferentes densidades. A continuación se muestra el algoritmo y el código fuente delas dos aproximaciones que se implementaron de esta hibridación.

1. K <- Numero de Documentos/MinPts
2. Resultados K-Means <- Ejecutar K-Means(K)
3. Encontrar la distancia máxima de cada grupo encontrado por K-Means
4. Para cada distancia
5. Ejecutar DBSCAN
6. Marcar ruido como no explorado
7. Seleccionar el resultado de DBSCAN como respuesta del algoritmo.

Algoritmo 7 - Híbrido con Distancias Variables

```
var resultado = new VectorRG(numeroAleatorio, MisParametros);
var matrizSimilitudes = new MatrizSimilitud(matrizDeDocumentos);
var resultadosDBSCAN = new int[matrizDeDocumentos.Length];

var initK = matrizDeDocumentos.Length/MisParametros.ParametrosDBSCAN.MinPts;
var kmeans = new KMeans();
var firstResults = kmeans.Cluster(matrizDeDocumentos, initK, resultadosDBSCAN,
                                  numeroAleatorio);
var epsilons = GetClusterDistances(matrizDeDocumentos, firstResults);
foreach (var eps in epsilons)
```

```
{
       if (eps <= 0) continue;
       resultadosDBSCAN = new DBSCAN(eps, MisParametros.ParametrosDBSCAN.MinPts,
                            matrizSimilitudes,
                            matrizDeDocumentos).BuildCluster(-1, resultadosDBSCAN);
       CalculosDBSCAN.OrdenarResultadosDBSCAN(ref resultadosDBSCAN);
}

resultado.KsinCentroides = resultadosDBSCAN.Max() + 1;
resultado.Grupos = resultadosDBSCAN;
MejorRespuesta = resultado;
```

Código Fuente 4 - Híbrido con Distancias Variables 1

```
var resultado = new VectorRG(numeroAleatorio, MisParametros);
var matrizSimilitudes = new MatrizSimilitud(matrizDeDocumentos);
var resultadosDBSCAN = new int[matrizDeDocumentos.Length];

var initK = matrizDeDocumentos.Length / MisParametros.ParametrosDBSCAN.MinPts;
if (initK < 2)
       initK = Kmax.Definir(matrizDeDocumentos.Length);
var kmeans = new KMeans();
var firstResults = kmeans.Cluster(matrizDeDocumentos, initK, resultadosDBSCAN,
                                  numeroAleatorio);
var epsilons = GetClusterEpsilons(matrizDeDocumentos, firstResults);

var mayorCantidadGrupos = 0;
var mejorEpsilon = epsilons[0];
foreach (var eps in epsilons)
{
       resultadosDBSCAN = new DBSCAN(eps, MisParametros.ParametrosDBSCAN.MinPts,
                                  matrizSimilitudes,
                                  matrizDeDocumentos).BuildCluster(-1,
                                                          resultadosDBSCAN);

       CalculosDBSCAN.OrdenarResultadosDBSCAN(ref resultadosDBSCAN);
       var gruposGenerados = resultadosDBSCAN.Max() + 1;
       if (mayorCantidadGrupos < gruposGenerados)
       {
              mejorEpsilon = eps;
              mayorCantidadGrupos = gruposGenerados;
       }
}

resultadosDBSCAN =
       new DBSCAN(mejorEpsilon, MisParametros.ParametrosDBSCAN.MinPts,
                    matrizSimilitudes,
                    matrizDeDocumentos).BuildCluster(-1, resultadosDBSCAN);

CalculosDBSCAN.OrdenarResultadosDBSCAN(ref resultadosDBSCAN);
resultado.KsinCentroides = resultadosDBSCAN.Max() + 1;

resultado.Grupos = resultadosDBSCAN;
```

```
MejorRespuesta = resultado;
```

Código Fuente 5 - Híbrido con Distancias Variables 2

Capítulo 4: Evaluación

14 Datasets y Medidas de evaluación y comparación

14.1 Datasets

El algoritmo propuesto fue utilizado para realizar clustering de resultados web tomando cuatro datasets tradicionales de evaluación, estos son: DMOZ-50, AMBIENT, MORESQUE y ODP-239.

DMOZ-50 consiste de 50 consultas derivadas del Open Directory Project (acrónimo para el directorio de Mozilla). Cada consulta tiene un promedio de 129.14 documentos, 6.02 sub-tópicos (significados de temas diversos), y 22.62 resultados relevantes por cada sub-tópico recuperado. Cada consulta es una gran colección de documentos, cada uno con una cantidad de clases relativamente pequeña, y un número elevado de documentos en cada clase. En este dataset no están disponibles las palabras clave de la consulta; la colección está disponible para descargar en http://artemisa.unicauca.edu.co/~ccobos/wdc/wdc.htm.

El dataset AMBIENT (AMBIguous ENTries) consiste de 44 consultas extraídas de entradas ambiguas de Wikipedia, cada una de las consultas tiene un promedio de 50.55 resultados ordenados, recolectados de Yahoo! (anotados manualmente, con juicios de relevancia por sub-tópico a nivel de documento), 7.91 sub-tópicos y 7.72 resultados relevantes por cada sub-tema recuperado. La mayoría de las consultas en AMBIENT son de solo una palabra clave y están todas disponibles. Este dataset mide la habilidad de un algoritmo para recuperar subtemas que se encuentran dentro de los resultados de búsqueda (los documentos obtenidos por Yahoo!), no todos los posibles subtemas de una consulta. Este dataset se puede descargar de http://credo.fub.it/ambient.

El dataset MORESQUE (MORE Sense-tagged QUEry results) consiste de 114 consultas ambiguas que se desarrollaron como complemento a AMBIENT. Este dataset prueba el comportamiento de los algoritmos de búsqueda web en consultas de diversa longitud, variando de entre 1 y 4 palabras. MORESQUE contiene 114 búsquedas de longitud 2, 3 y 4 (todas disponibles), junto a un promedio de 53.54 resultados principales (documentos) de Yahoo!, 3.82 subtemas y 19.43 resultados relevantes por tema obtenido. Este dataset se puede encontrar en http://lcl.uniroma1.it/moresque.

El dataset ODP-239 consiste de 239 consultas derivadas del Open Directory Project (http://www.dmoz.org). Cada una tiene en promedio 106.95 documentos (cada uno compuesto de URL, título y una corta descripción), 9.56 subtemas y 11.38 resultados relevantes por cada subtema encontrado. Este dataset está compuesto por muchas colecciones pequeñas, cada una con una cantidad de clases relativamente grande, en vez de una colección grande con pocas clases. Los tópicos, sub-tópicos y documentos asociados fueron seleccionados de forma tal que la distribución de documentos en los subtemas refleje la importancia relativa de cada uno. Este dataset se puede encontrar en http://credo.fub.it/odp239.

14.2 Medidas

Para la evaluación de la calidad de los resultados de clustering de documentos se utilizan un conjunto de medidas de amplia aceptación y reconocimiento dentro de la comunidad de recuperación de información. A continuación se presenta la definición general de cada medida en el ámbito de la recuperación de información [66]. Para su uso en clustering de documentos que, la mayoría de ellas requiere su cálculo en cada clase o tema y un promedio general sobre el total de clases o temas:

- **Recall:** se refiere a la fracción de los documentos relevantes que son recuperados por la consulta. Por ejemplo en una búsqueda de texto, el recuerdo es el número de resultados correctos, dividido por el número de resultados que se deberían haber obtenido.

$$\boldsymbol{Recall} = \frac{|\boldsymbol{Documentos\ relevantes} \cap \boldsymbol{Documentos\ recuperados}|}{\boldsymbol{Documentos\ relevantes}}$$

- **Precision:** en el campo de recuperación de información, la precisión es la fracción de documentos recuperados que son relevantes a la búsqueda.

$$\boldsymbol{Precison} = \frac{|\boldsymbol{Documentos\ relevantes} \cap \boldsymbol{Documentos\ recuperados}|}{\boldsymbol{Documentos\ recuperados}}$$

- **F-measure:** es la medida con la cual se define la exactitud de los resultados obtenidos en el proceso de clustering. Esta medida considera la precisión y el recuerdo de la prueba para

calcular el puntaje. Esta medida puede ser considerada como el promedio ponderado de la precisión y el recuerdo, donde el valor está entre 0 (el peor) y 1 (el mejor).

$$F = 2 * \frac{precisión * recuerdo}{precisión + recuerdo}$$

- **ICC:** el coeficiente de correlación entre clases (ICC) es una estadística descriptiva que puede ser usada cuando se realizan medidas cuantitativas de unidades que se encuentran organizadas en grupos. Esta describe que tanto se parecen entre si los elementos dentro de cada grupo.

- **Fall-out:** es la fracción de documentos recuperados exitosamente que no son relevantes a la consulta, de entre todos los documentos no-relevantes disponibles. Esta medida puede considerarse como la probabilidad de que la consulta retorne un resultado irrelevante.

$$\boldsymbol{fall - out} = \frac{|\{\boldsymbol{Documentos\ irrelevantes}\} \cap \{\boldsymbol{documentos\ recuperados}\}|}{|\{\boldsymbol{documentos\ irrelevantes}\}|}$$

- **Accuracy:** esta es una medida estadística que valida si una clasificación incluye o excluye correctamente una condición. Esta es la proporción de resultados verdaderos (tanto positivos como negativos) en la población. Normalmente se extrae de la matriz de confusión.

$$\boldsymbol{accuracy} = \frac{\boldsymbol{true\ positives} + \boldsymbol{true\ negatives}}{\boldsymbol{true\ positives} + \boldsymbol{false\ positives} + \boldsymbol{true\ negatives} + \boldsymbol{false\ negatifes}}$$

- **NVP:** también conocido como el *valor negativo predictivo* **(negative predictive value)** es una estadística de resumen utilizada para describir el desempeño de un procedimiento de pruebas diagnósticas. Este se define como la proporción de elementos con resultados negativos que son correctamente clasificados como tal. Un valor alto de NVP implica que cuando una prueba arroja resultados positivos, seguramente está en lo correcto.

$$\boldsymbol{nvp} = \frac{\boldsymbol{true\ negatives}}{\boldsymbol{true\ negatives} + \boldsymbol{false\ negatives}}$$

- **FDR:** El *False Discovery Rate* (tasa de falsos descubrimientos) es una medida estadística utilizada para corregir en casos de múltiples comparaciones. En listados de hallazgos estadísticos significativos, el FDR es un método que permite controlar la cantidad esperada de falsos positivos [67].

Las anteriores formulas deben ser adecuadas al contexto de la agrupación de documentos web. Por ejemplo, precisión, recuerdo y medida F se calculan de la siguiente forma:

Dada una colección de clusters $\{C_1, C_2, \dots C_k\}$, se toman los siguientes pasos para evaluar su Precisión, Recuerdo y Medida-F ponderados respecto a la colección ideal de clusters $\{C_1^i, C_2^i, \dots C_n^i\}$: (a) para cada clúster ideal C_n^i encontrar un clúster único C_m que más se aproxime a él en la colección que está siendo evaluada, y medir $P(C, C^i)$, $R(C, C^i)$, y $F(C, C^i)$ como se define en (7) y (8). (b) Calcular la Precisión (P), Recuerdo (R) y Medida-F (F) ponderadas basado en (9).

$$P(C, C^i) = \frac{|C \cap C^i|}{|C|} \; y \; R(C, C^i) = \frac{|C \cap C^i|}{|C^i|} \tag{1}$$

Donde C es un cluster de documentos, y C^i es el grupo ideal de documentos.

$$F(C, C^i) = \frac{2 * P(C, C^i) * R(C, C^i)}{P(C, C^i) + R(C, C^i)} \tag{2}$$

$$P = \frac{1}{T}\sum_{j=1}^{h}|C_j^i| * P(C_m, C_j^i), \; R = \frac{1}{T}\sum_{j=1}^{h}|C_j^i| * R(C_m, C_j^i), \text{ y } F = \frac{2*P*R}{P+R} \text{ donde } T = \sum_{j=1}^{h}|C_j^i| \tag{3}$$

Finalmente, en Agrupación de Documentos Web durante los últimos años ha tomado fuerza el uso de una medida que refleja de mejor manera las operaciones mínimas (facilidad) que debe realizar un usuario para encontrar la información relevante en un motor que agrupa documentos web. Esta medida es conocida como **SSL_k:** (Subtopic Search Length under k document sufficiency). Esta se

define como el número promedio de elementos (resultados de búsquedas o etiquetas de agrupaciones) que deben ser examinadas antes de encontrar un número suficiente (*k*) de documentos relevantes a la consulta del usuario [68-70].

15 Experimentación

15.1 Comparación

En la literatura revisada se destacan algunos algoritmos de agrupación de documentos web, para la evaluación de esta investigación se toman los resultados de dichos algoritmos y se comparan con los obtenidos, a continuación se presentan dichos métodos de agrupación:

STC, Suffix Tree Clustering, es un algoritmo propuesto en 1998, toma el enfoque de frases frecuentes compartidas por los documentos, cuenta con tres pasos lógicos: 1. Limpieza del documento 2. Identificación de los grupos base mediante un árbol de sufijos 3. Combinación de los grupos base en los grupos finales. Una de sus mayores ventajas es que usa frases que proporcionan descripciones concisas y significativas de los grupos [71].

Bisecting k-means, algoritmo propuesto en el año 2000, combina la fortaleza de los métodos jerárquicos y particionales, inicialmente los datos son tratados como una sola agrupación, basado en una regla se selecciona una agrupación y esta se dividen en dos agrupaciones con la ayuda del algoritmo k-means, este proceso se repite hasta obtener las agrupaciones deseadas [72].

Lingo, algoritmo propuesto en el año 2003, usado por el buscador carrot2, se basa en frases completas, una de sus características es que primero define nombres descriptivos para sus agrupaciones y posteriormente organiza los documentos en los grupos. Su estrategia consiste en extraer las frases frecuentes en los documentos como fuente informativa de descripción de temas en los grupos, luego intenta reducir la matriz de términos por documentos, Lingo define el número de grupos k que se deben formar y relaciona las descripciones de los grupos con los documentos [73].

15.2 Resultados

La evaluación de los algoritmos se enfocó en ver que tan bien se desempeñaban los diferentes algoritmos obtenidos en el proceso de investigación, en un entorno controlado de Web Document

Clustering. Para esto, se ejecutó cada uno de los algoritmos 30 veces sobre cada uno de los datasets descritos anteriormente.

Para la evaluación se tomaron en cada ejecución las medidas detalladas en la sección anterior y a continuación se presentan los valores promediados del total de ejecuciones. Además, se realizaron las mismas mediciones sobre los algoritmos base de Bisecting K-Means, STC y Lingo. Fuera de esto se realiza también la comparación con los algoritmos tradicionales de DBSCAN y K-Means. Las pruebas de K-Means se realizaron tomando un valor de *K* dinámico, utilizando la heurística $k = \sqrt{\frac{numero\ de\ documentos}{2}} + 1$. Para DBSCAN se tomaron valores estáticos de Eps = 0.4 y MinPts = 6, después de realizar pruebas con un barrido desde Eps = 0.01 hasta 1 y MinPts = 2 hasta 10, y encontrar que estos eran los valores con los mejores resultados promedio.

En la Tabla 1 se encuentran los resultados de la ejecución de los diferentes algoritmos sobre el dataset de Ambient. Se observa que ninguno de los algoritmos propuestos logra mejorar o siquiera igualar los resultados obtenidos por los algoritmos de referencia. En todas las medidas son superados por STC y Lingo, donde STC presenta un mayor recuerdo y accuracy, logrando una mayor cantidad de documentos correctamente clasificados, mientras que Lingo gana en precisión, medida-F, fall-out y FDR, mostrando unos mejores resultados generales en el proceso de clustering.

	K ESTIMADO	DIFERENCIA CON K ÓPTIMO	TIEMPO	BUENOS	ICC-PURITY	PRECISION	RECUERDO	MEDIDA F	FALL-OUT	ACCURACY	NPV	FDR
BISECTING K-MEANS	11,242424	3,333333	0,072136	19,581818	40,646547	76,120994	40,646547	45,886374	0,037747	77,146563	0,768861	0,037747
STC	11,000000	3,090909	**0,006485**	**26,705303**	**53,152463**	72,420575	**53,152463**	55,403580	0,056368	**81,895935**	**0,809939**	0,056368
LINGO	20,856818	12,947727	0,026750	23,868182	50,145669	**86,678162**	50,145669	**58,607835**	**0,031911**	80,337060	0,787373	**0,031911**
K-MEANS	9,340909	**1,431818**	0,255661	24,636364	51,368309	72,740420	51,368309	55,460254	0,057202	80,293485	0,798962	0,057202
DBSCAN	1,000000	6,909091	0,135645	22,750000	41,763301	20,117508	41,763301	26,362097	0,417633	69,624157	0,495066	0,417633
REFINAMIENTO CONTINUO	3,000000	4,909091	0,145201	22,750000	41,763301	20,117508	41,763301	26,362097	0,417633	0,696242	0,495066	0,417633
MEJORES RESULTADOS	3,136364	4,772727	0,161190	23,022727	42,707657	24,849708	42,707657	28,791451	0,384699	70,928788	0,601597	0,384699
HÍBRIDO PARALELO	3,000000	4,909091	0,226416	22,750000	41,763301	20,117508	41,763301	26,362097	0,417633	69,624157	0,495066	0,417633

HÍBRIDO CON DISTANCIAS VARIABLES 1	1,818182	6,090909	0,136155	23,909091	44,377736	28,127895	44,377736	31,448718	0,382754	71,372524	0,612845	0,382754
HÍBRIDO CON DISTANCIAS VARIABLES 2	2,022727	5,886364	0,133341	23,431818	43,560056	29,825458	43,560056	31,622111	0,363123	71,265961	0,625127	0,363123

Tabla 1 - Resultados Ambient

A pesar de la correcta identificación del número de agrupaciones dentro del dataset DMOZ (ver Tabla 2) por parte de la primera y segunda versión del algoritmo híbrido con distancias variables, estos no logran identificar correctamente los documentos de cada cluster. En este dataset STC muestra una superioridad en todas las medidas de calidad tomadas para la evaluación. En este caso STC obtiene los mejores resultados en todas las medidas.

	K	**DIFERENCIA CON K ÓPTIMO**	**TIEMPO**	**BUENOS**	**ICC-PURITY**	**PRECISION**	**RECUERDO**	**MEDIDA F**	**FALL-OUT**	**ACCURACY**	**NPV**	**FDR**
BISECTING K-MEANS	11,039333	5,019333	0,113072	55,720667	42,900828	70,529208	42,900828	49,781533	0,047208	84,230644	0,863650	0,047208
STC	16,000000	9,980000	**0,010133**	**74,792000**	**57,852096**	**84,807878**	**57,852096**	**65,126958**	0,030884	**88,812131**	**0,900202**	0,030884
LINGO	34,273333	28,253333	0,153844	48,987333	37,881941	83,848866	37,881941	48,218643	0,047992	83,383942	0,851392	0,047992
K-MEANS	12,840000	6,820000	0,602326	67,360000	52,159366	80,365833	52,159366	60,578021	**0,029104**	0,874788	0,884080	**0,029104**
DBSCAN	1,120000	4,900000	0,529801	36,300000	28,281836	9,698986	28,281836	13,205139	0,277070	68,615506	0,631539	0,277070
REFINAMIENTO CONTINUO	3,000000	3,020000	0,658487	35,740000	27,883584	8,221604	27,883584	12,591592	0,278836	68,436101	0,602145	0,278836
MEJORES RESULTADOS	3,120000	2,900000	0,791821	36,560000	28,500981	10,033014	28,500981	13,661428	0,275393	68,761637	0,624511	0,275393
HÍBRIDO PARALELO	3,000000	3,020000	1,461693	35,740000	27,883584	8,221604	27,883584	12,591592	0,278836	68,436101	0,602145	0,278836
HÍBRIDO CON DISTANCIAS VARIABLES 1	6,060000	**0,040000**	0,478177	53,980000	41,782408	51,911865	41,782408	35,414009	0,171144	77,274918	0,872351	0,171144
HÍBRIDO CON DISTANCIAS VARIABLES 2	5,880000	0,140000	0,498595	52,400000	40,615179	47,916912	40,615179	33,738568	0,179241	76,390708	0,852403	0,179241

Tabla 2 - Resultados DMOZ

En el caso de MORESQUE, se ve un mejor desempeño de los híbridos propuestos respecto a los resultados anteriores, cabe notar que todos identifican de manera más acertada la cantidad correcta de temas retornados por la consulta, y los resultados en general están bastante próximos entre sí. Entre las propuestas se destacan el algoritmo híbrido paralelo y el híbrido con distancias variables. Ambos logran superar a Bisecting K-Means, STC y Lingo en las principales medidas, y el híbrido paralelo se destaca por obtener un mayor recuerdo, accuracy y elementos correctamente clasificados. Normalmente este es el dataset que presenta mayores dificultades para los algoritmos del estado del arte, pero es donde los algoritmos híbridos propuestos muestran un mejor desempeño. En principio esto se logra, porque los híbridos definen de una mejor manera el número de grupos a formar.

	K	DIFERENCIA CON K ÓPTIMO	TIEMPO	BUENOS	ICC-PURITY	PRECISION	RECUERDO	MEDIDA F	FALL-OUT	ACCURACY	NPV	FDR
BISECTING K-MEANS	10,519006	6,694444	0,063153	13,088596	29,902554	87,452945	29,902554	38,518087	**0,041316**	53,343266	0,469555	**0,041316**
STC	11,166667	7,342105	**0,006914**	22,921053	49,961329	82,857899	49,961329	57,182900	0,127507	65,441657	**0,506393**	0,127507
LINGO	20,159942	16,335380	0,030499	17,501754	39,543053	**90,577159**	39,543053	50,748104	0,062688	59,347128	0,492054	0,062688
K-MEANS	9,368421	5,543860	0,264544	18,578947	39,828421	85,076394	39,828421	50,002632	0,102957	58,663327	0,489282	0,102957
DBSCAN	1,035088	2,789474	0,144784	36,763158	67,851999	51,884368	67,851999	57,229675	0,629483	75,865662	0,260612	0,629483
REFINAMIENTO CONTINUO	3,000000	0,824561	0,167024	30,745614	58,538973	62,362876	58,538973	57,545649	0,382711	67,988116	0,494054	0,382711
MEJORES RESULTADOS	3,043860	0,780702	0,171417	36,438596	67,290641	53,880918	67,290641	57,654333	0,613562	75,235548	0,296087	0,613562
HÍBRIDO PARALELO	3,000000	0,824561	0,243076	**36,798246**	**68,022317**	50,938617	**68,022317**	57,177968	0,636364	**75,914905**	0,249763	0,636364
HÍBRIDO CON DISTANCIAS VARIABLES 1	3,736842	**0,087719**	0,139380	34,723214	64,764544	58,714183	64,764544	**58,411401**	0,551958	74,098554	0,345977	0,551958
HÍBRIDO CON DISTANCIAS VARIABLES 2	2,254386	1,570175	0,136146	33,543860	64,282360	59,655205	64,282360	57,941453	0,555190	0,733404	0,365059	0,555190

Tabla 3 - Resultados MORESQUE

En ODP239, nuevamente los resultados de los algoritmos propuestos distan mucho de aquellos reportados por los algoritmos líderes en el campo de WDC. En este caso se ve claramente a STC como el que presenta el mejor desempeño, superando por amplio margen a los demás algoritmos en prácticamente todas las mediciones. Todos los algoritmos propuestos muestran un desempeño bastante inferior al de los principales representantes de WDC.

	K	DIFERENCIA CON K ÓPTIMO	TIEMPO	BUENOS	ICC-PURITY	PRECISION	RECUERDO	MEDIDA F	FALL-OUT	ACCURACY	NPV	FDR
BISECTING K-MEANS	11,76904	**2,20837**	0,11169	34,08787	31,97410	55,58079	31,97410	34,74612	**0,06171**	78,09646	0,80137	**0,06171**
STC	15,98326	6,42259	**0,00836**	**42,70446**	**39,75536**	57,32636	**39,75536**	**41,81712**	0,09907	**80,65813**	**0,83146**	0,09907
LINGO	31,38550	21,82483	0,08914	35,17657	32,91862	**71,74880**	32,91862	41,00921	0,06551	79,15646	0,80020	0,06551
K-MEANS	12,01674	2,45607	0,36446	39,06276	36,65215	56,60429	36,65215	39,64013	0,06225	79,24339	0,81163	0,06225
DBSCAN	1,03347	8,52720	0,32798	40,69874	37,47420	16,73642	37,47420	22,30328	0,37261	71,85303	0,55915	0,37261
REFINAMIENTO CONTINUO	3,00000	6,56067	0,36934	40,90377	37,62292	16,48363	37,62292	22,26369	0,37623	71,90526	0,55422	0,37623
MEJORES RESULTADOS	3,05858	6,50209	0,42909	41,16736	37,88389	18,08239	37,88389	23,05303	0,36781	72,23254	0,59493	0,36781
HÍBRIDO PARALELO	3,00000	6,56067	0,71423	40,90377	37,62292	16,48363	37,62292	22,26369	0,37623	71,90526	0,55422	0,37623
HÍBRIDO CON DISTANCIAS VARIABLES 1	3,52301	6,03766	0,32305	40,25941	37,30573	31,04739	37,30573	27,55469	0,29702	73,33358	0,73849	0,29702
HÍBRIDO CON DISTANCIAS VARIABLES 2	3,67364	5,88703	0,32210	39,63180	36,72154	31,21361	36,72154	27,21084	0,29566	73,08136	0,72964	0,29566

Tabla 4 - Resultados ODP239

15.3 Comparación de las diferentes medidas

A continuación se presenta en detalle una comparación de las diferentes medidas tomadas para la evaluación de los algoritmos. Se muestra el comparativo para cada medida de los algoritmos propuestos y el resultado obtenido por el mejor algoritmo en cada categoría.

Figura 4 - Tiempo de ejecución

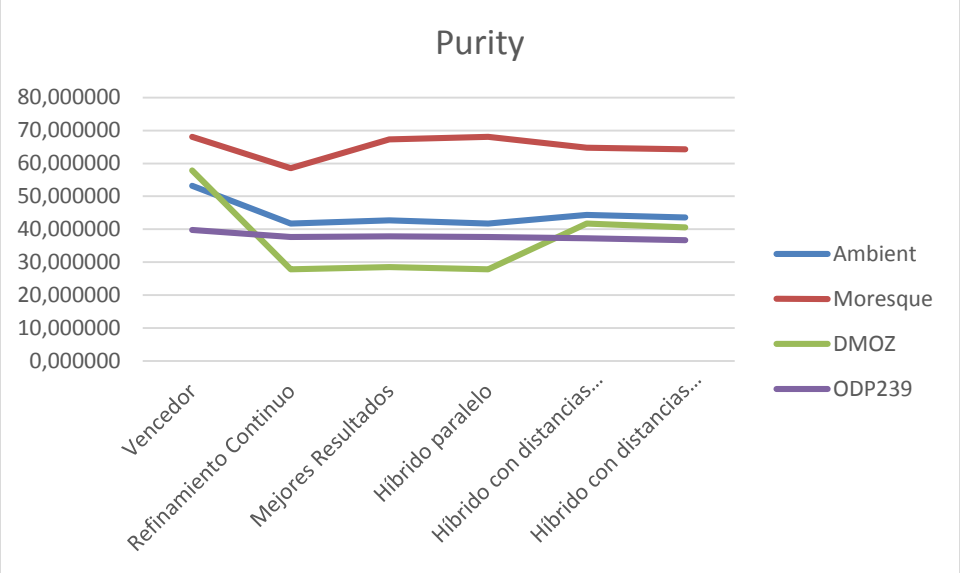

Figura 5 - Pureza

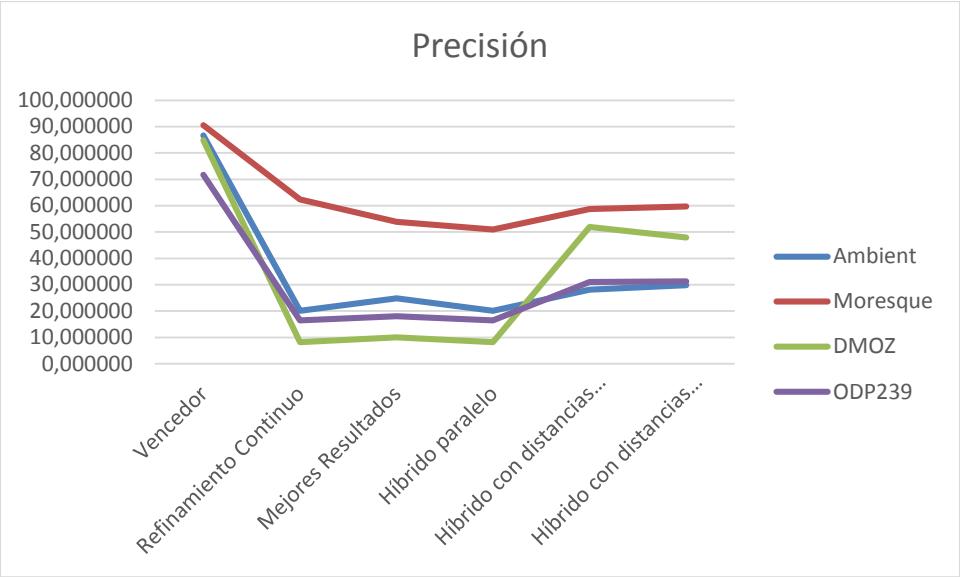

Figura 6 - Precisión

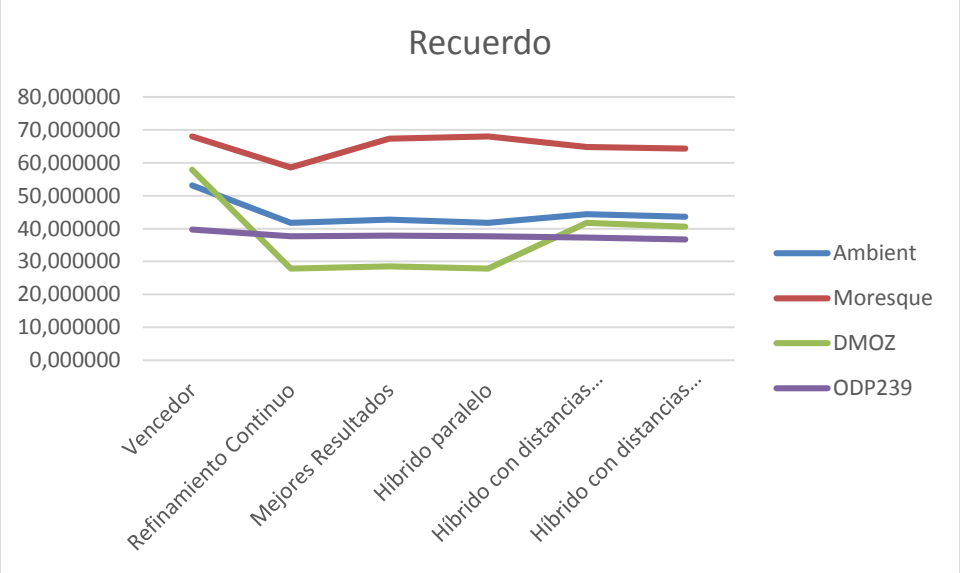

Figura 7 - Recuerdo

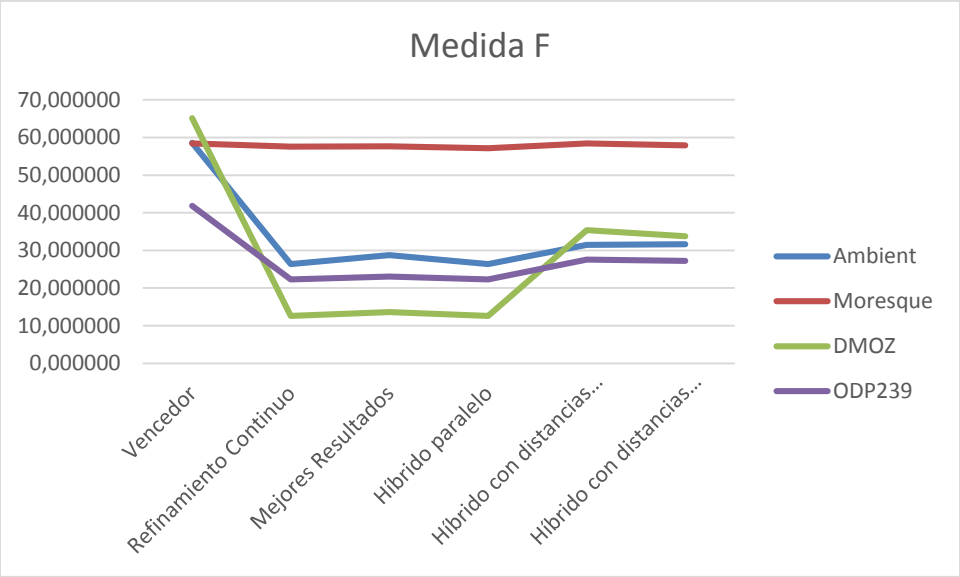

Figura 8 - Medida F

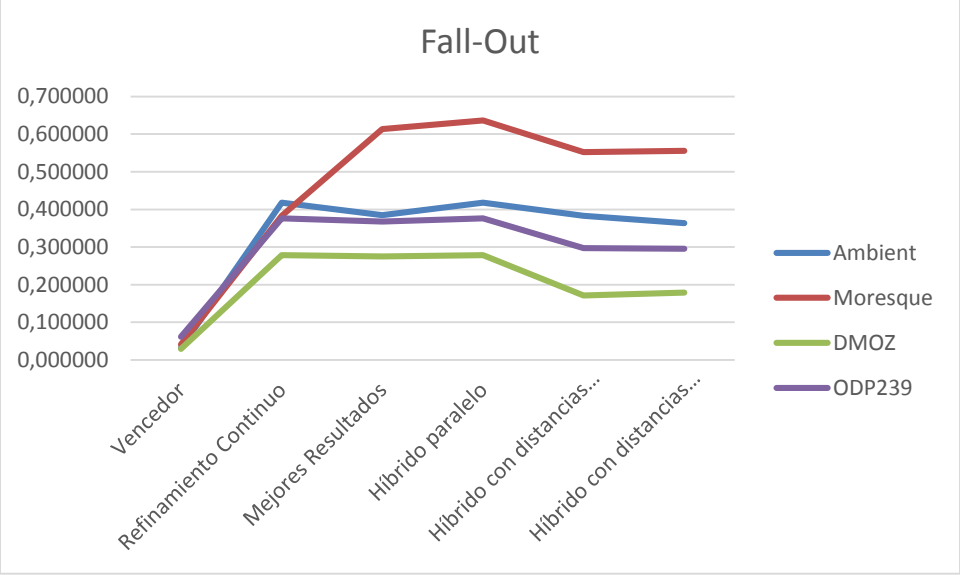

Figura 9 - Fall-Out

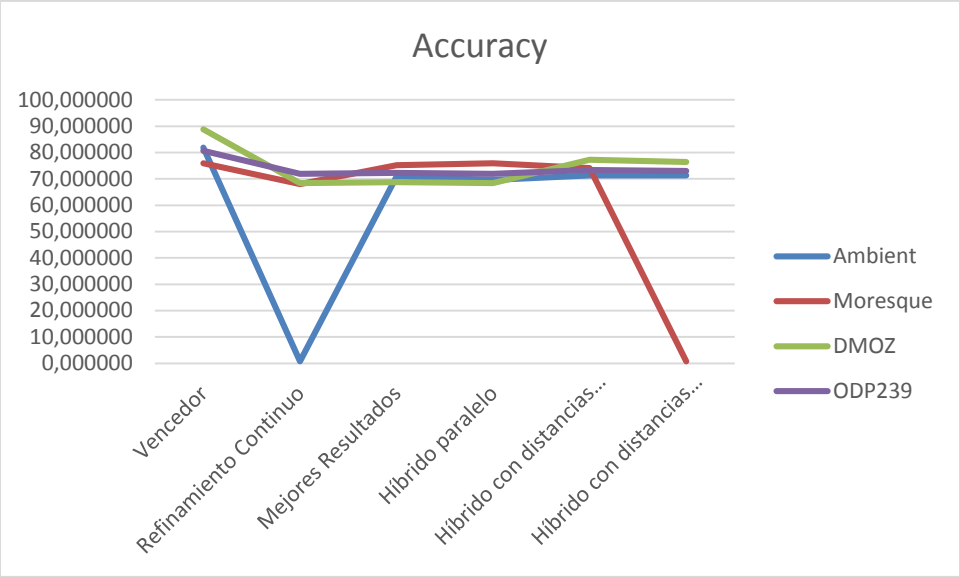

Figura 10 - Accuracy

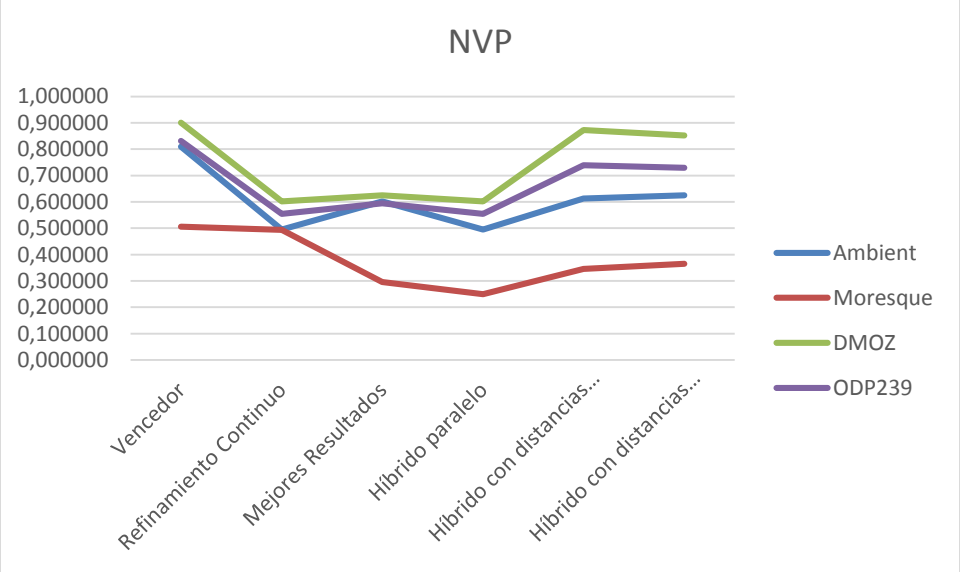

Figura 11 - NVP

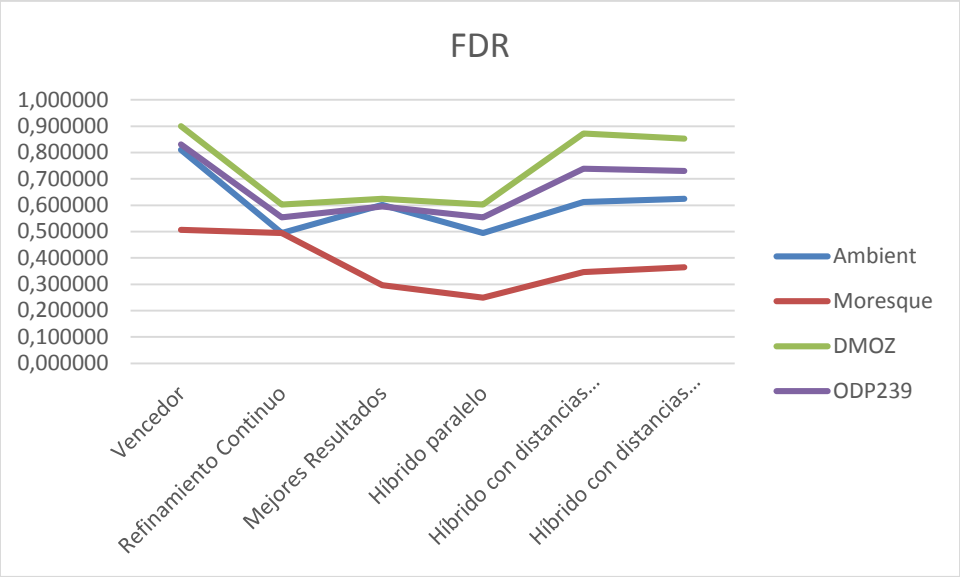

Figura 12 - FDR

15.4 Evaluación del comportamiento del usuario

La medida SSL_k se utilizó para evaluar la facilidad con la que los usuarios pueden utilizar los resultados de las búsquedas. Como se explicó anteriormente, esta medida evalúa la cantidad promedio de elementos (documentos) que un usuario debe examinar antes de encontrar una cantidad adecuada (k) de documentos relevantes al tema seleccionado. Se midió el desempeño de cada uno de los algoritmos en los diferentes datasets y los resultados fueron evaluados utilizando esta medida, con *k= 1, 2, 3 y 4.* Los resultados, promediados del total de las pruebas, se reportan en la Tabla 5.

	ALGORITMO	SSL1	SSL2	SSL3	SSL4	SSLK
AMBIENT	STC	27,261618	44,991796	54,794553	60,445658	187,493624
	Lingo	22,576827	36,725598	47,461419	54,625087	161,388931
	K-Means	14,928467	26,438738	32,801067	36,646649	**110,814920**
	DBSCAN	17,439584	37,109255	48,954276	56,890665	160,393780
	Refinamiento Continuo	17,517263	30,863786	40,960567	46,612878	135,954494
	Mejores Resultados	17,046307	29,767503	37,357188	42,244769	126,415766
	Híbrido paralelo	18,045245	32,752380	41,386220	47,245207	139,429052
	Híbrido con distancias variables 1	15,671549	30,777355	40,185614	46,261246	132,895764

	Híbrido con distancias variables 2	15,695390	30,592089	39,692101	45,832706	131,812286
DMOZ	STC	12,033688	16,391889	18,664139	21,422300	**68,512016**
	Lingo	13,863992	16,221121	18,031916	21,600839	69,717869
	K-Means	14,189508	18,312222	21,438690	24,454476	78,394897
	DBSCAN	17,583119	24,740571	30,986817	36,329341	109,639849
	Refinamiento Continuo	24,799317	30,779270	35,064659	39,550913	130,194159
	Mejores Resultados	29,485722	36,286460	41,242468	45,759016	152,773667
	Híbrido paralelo	26,585181	32,458733	37,510276	41,994891	138,549080
	Híbrido con distancias variables 1	17,460579	21,640563	25,825619	29,773286	94,700048
	Híbrido con distancias variables 2	17,517873	22,458119	26,578127	30,376873	96,930992
MORESQUE	STC	19,597470	32,234643	40,170703	45,202105	137,204921
	Lingo	16,531918	26,406226	33,864598	39,318383	116,121125
	K-Means	11,326424	19,429897	24,268794	27,536596	**82,561711**
	DBSCAN	11,644529	24,163137	32,619473	38,708552	107,135691
	Refinamiento Continuo	10,727316	20,141579	25,702777	30,022218	86,593891
	Mejores Resultados	11,707077	20,941075	26,554215	30,377298	89,579665
	Híbrido paralelo	11,986900	21,718576	28,141062	32,605844	94,452382
	Híbrido con distancias variables 1	11,675160	22,569377	28,695863	33,412882	96,353282
	Híbrido con distancias variables 2	11,459295	21,290883	27,429770	32,049456	92,229404
ODP239	STC	16,84519	27,45274	38,83384	52,58898	**135,72075**
	Lingo	26,31086	43,13864	60,69033	78,42594	208,56577
	K-Means	25,61667	38,13010	51,46479	66,53109	181,74266
	DBSCAN	19,77499	30,73301	41,60371	54,24557	146,35729
	Refinamiento Continuo	26,11196	39,64830	50,90702	61,59926	178,26653
	Mejores Resultados	26,04341	38,52722	49,85188	61,30684	175,72935
	Híbrido paralelo	26,85488	39,18413	49,82229	61,23835	177,09965
	Híbrido con distancias variables 1	25,62072	38,17261	49,31179	60,90021	174,00534
	Híbrido con distancias variables 2	22,70270	34,68441	45,07673	55,69475	158,15860

Tabla 5 - Evaluación del comportamiento de usuario

La evaluación de SSL_k en este apartado se realizó de forma automática apoyado en un procedimiento de comparación entre la etiqueta generada para cada grupo del algoritmo y las etiquetas ideales de los data sets. Este procedimiento usa la distancia de edición de Smith Waterman (http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm) para realizar esta comparación.

Bisecting K-means no se comparó porque usa muchas palabras clave como etiquetas de grupos, el uso excesivo de palabras claves distorsiona los resultados de la medida de Smith Waterman, y ya se ha probado en investigaciones previas que este esquema de etiquetado no es apropiado.

Al igual que con los resultados de clustering, los algoritmos tradicionales de WDC muestran un desempeño superior al de cualquiera de los algoritmos propuestos en este trabajo.

Capítulo 5: Conclusiones y Trabajo Futuro

16 Conclusiones

Con el fin de mejorar los resultados de los procesos de Web Document Clustering, se propuso cinco algoritmos que combinan a DBSCAN y K-Means utilizando las técnicas de hibridación planteadas en trabajos de investigación previos.

A través de la implementación de los algoritmos de Refinamiento Continuo, Mejores Resultados e Híbrido paralelo, sus respectivas pruebas con los dataset de WDC y la comparación con los resultados reportados por los algoritmos líderes de campo se encontraron resultados deficientes debido a la falta de flexibilidad de DBSCAN para trabajar con agrupaciones con densidades variables.

Para solventar la falencia presentada se propone una nueva forma de hibridación de la que se obtienen los Híbridos con Distancias Variables, que nuevamente fueron implementados y evaluados con los datasets.

Los algoritmos propuestos muestran un desempeño superior al de los métodos tradicionales en Moresque debido a que logran identificar de manera más acertada el número de grupos y las consultas retornan un grupo de documentos más reducido y coherente.

A pesar que se encuentra que algunas propuestas reportan un mejor desempeño que el de los algoritmos de referencia, este sigue siendo inferior al que se reportan actualmente los algoritmos líderes en el campo.

Se concluye que, a pesar que esta última propuesta logra una mayor precisión al encontrar los grupos existentes, tienen problemas para identificar de manera adecuada la pertenencia de los documentos a sus grupos debido a la dificultad que existe para definir de manera correcta los valores adecuados para la ejecución de DBSCAN.

Por esto se concluye que, debido a los resultados encontrados, la baja calidad de los clusters recuperados y el pobre desempeño general de los algoritmos propuestos, no es posible realizar una

hibridación viable entre DBSCAN y K-Means para el problema de Web Document Clustering, ya que no se encuentra una manera que permita definir los vecindarios de exploración de DBSCAN de manera adecuada, que además sea flexible a las variaciones presentes en las agrupaciones de este tipo, en el tiempo de que se dispuso para la investigación.

17 Trabajo Futuro

En el entorno de agrupamiento de documentos web, se recomienda reconocer la capacidad que tienen los algoritmos basados en densidad para detectar y manejar de manera eficiente el ruido. Teniendo en cuenta lo anterior, se propone buscar una alternativa al algoritmo DBSCAN como componente de la hibridación, por uno que posea la capacidad de manejar correctamente los espacios dimensionales en los que la densidad de los clusters varía como OPTICS, también podría presentarse como posibilidad utilizar un derivado de DBSCAN que ya tome en cuenta la necesidad de manejar este tipo de agrupaciones con densidades variables.

Capítulo 6: Referencias

18 Referencias

[1] Z. W. Geem and W. E. Roper, "Various continuous harmony search algorithms for web-based hydrologic parameter optimisation," *International Journal of Mathematical Modelling and Numerical Optimisation,* vol. 1, p. 14, 2010.

[2] C. Manning, P. Raghavan, and H. Schütze. (2008). *Introduction to Information Retrieval*. Available: http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html

[3] M. Jaberipour and E. Khorram, "Two improved harmony search algorithms for solving engineering optimization problems," *Communications in Nonlinear Science and Numerical Simulation,* vol. In Press, Corrected Proof, 22 January, 2010 2010.

[4] T. Joachims and F. Radlinski, "Search Engines that Learn from Implicit Feedback," *Computer,* vol. 40, pp. 34-40, 2007.

[5] C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A survey of Web clustering engines," *ACM Comput. Surv.,* vol. 41, pp. 1-38, 2009.

[6] W. Song, C. H. Li, and S. C. Park, "Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures," *Expert Systems with Applications,* vol. 36, pp. 9095-9104, 2009.

[7] M. Mahdavi and H. Abolhassani, "Harmony K-means algorithm for document clustering," *Data Mining and Knowledge Discovery,* vol. 18, pp. 370-391, 2009.

[8] R. M. Aliguliyev, "Clustering of document collection - A weighting approach," *Expert Systems with Applications,* vol. 36, pp. 7904-7916, 2009.

[9] M. Mahdavi, M. H. Chehreghani, H. Abolhassani, and R. Forsati, "Novel meta-heuristic algorithms for clustering web documents," *Applied Mathematics and Computation,* vol. 201, pp. 441-451, 2008.

[10] Y. Li, C. Luo, and S. M. Chung, "Text Clustering with Feature Selection by Using Statistical Data," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 20, pp. 641-652, 2008.

[11] Y. Li, S. M. Chung, and J. D. Holt, "Text document clustering based on frequent word meaning sequences," *Data & Knowledge Engineering,* vol. 64, pp. 381-404, 2008.

[12] R. Forsati, M. R. Meybodi, M. Mahdavi, and A. G. Neiat, "Hybridization of K-Means and Harmony Search Methods for Web Page Clustering," in *Web Intelligence and Intelligent*

Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on, 2008, pp. 329-335.

[13] G. Mecca, S. Raunich, and A. Pappalardo, "A new algorithm for clustering search results," *Data & Knowledge Engineering,* vol. 62, pp. 504-522, 2007.

[14] S. Osiński, "Improving quality of search results clustering with approximate matrix factorizations," in *28th European Conference on IR Research (ECIR 2006)*, London, UK, 2006, pp. 167-178.

[15] J. Moore, E.-H. S. Han, D. Boley, M. Gini, R. Gross, K. Hastings*, et al.*, "Web Page Categorization and Feature Selection Using Association Rule and Principal Component Clustering," in *Workshop on Information Technologies and Systems*, 1997.

[16] C. Cobos, C. Montealegre, M. Mejía, M. Mendoza, and E. León, "Web Document Clustering based on a New Niching Memetic Algorithm, Term-Document Matrix and Bayesian Information Criterion," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010, pp. 4629-4636.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.,* vol. 31, pp. 264-323, 1999.

[18] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*: Prentice-Hall, Inc., 1988.

[19] K. Hammouda. (2001, Web Mining: Clustering Web Documents A Preliminary Review. 1-13. Available: http://watnow.uwaterloo.ca/pub/hammouda/review-document-clustering.pdf

[20] Z. Oren and E. Oren, "Web document clustering: a feasibility demonstration," presented at the Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, Melbourne, Australia, 1998.

[21] Y. Tan, Y. Shi, K. Tan, H. Jiang, Y. Liu, and L. Zheng, "Design and Simulation of Simulated Annealing Algorithm with Harmony Search," in *Advances in Swarm Intelligence*. vol. 6146, ed: Springer Berlin / Heidelberg, 2010, pp. 454-460.

[22] H. Xu, X. Gao, T. Wang, and K. Xue, "Harmony Search Optimization Algorithm: Application to a Reconfigurable Mobile Robot Prototype," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 11-22.

[23] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of DBSCAN variations," in *2010 International Conference on Information and Emerging Technologies (ICIET)*, Karachi, 2010, pp. 1-6.

[24] R. Baeza-Yates, A. and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc., 1999.

[25] S. Osiński, J. Stefanowski, and D. Weiss, "Lingo: Search results clustering algorithm based on Singular Value Decomposition," in *Proceedings of the International Conference on Intelligent Information Systems (IIPWM)*, 2004, pp. 359-368.

[26] C. J. V. Rijsbergen, *Information Retrieval*: Butterworth-Heinemann, 1979.

[27] W. B. Frakes and R. A. Baeza-Yates, *Information Retrieval Data Structures & Algorithms* Prentice-Hall, 1992.

[28] J. Nielsen. (2004). *When Search Engines Become Answer Engines*. Available: http://www.useit.com/alertbox/20040816.html

[29] K. O'Hara and N. Shadbolt. (2004). *Knowledge Technologies and the Semantic Web* Available: http://eprints.ecs.soton.ac.uk/12469/

[30] A. Spink and J. L. Xu, "Selected results from a large study of Web searching: the Excite study," *Information Research,* vol. 6, 2000.

[31] S. Chakrabarti, "Web Search and Information Retrieval," in *Mining the Web*, ed San Francisco: Morgan Kaufmann, 2003, pp. 45-76.

[32] R. Baeza-Yates, C. Castillo, and B. Keith, "Web Searching," in *Encyclopedia of Language & Linguistics*, ed Oxford: Elsevier, 2006, pp. 527-538.

[33] C. Tung-Shou, T. Tzu-Hsin, C. Yi-Tzu, L. Chin-Chiang, C. Rong-Chang, L. Shuan-Yow*, et al.*, "A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray," in *Proceedings of 2005 International Symposium onIntelligent Signal Processing and Communication Systems, 2005. ISPACS 2005.*, 2005, pp. 405-408.

[34] L. Jing, "Survey of Text Clustering," ed, 2008.

[35] S. Osiński, "An Algorithm for clustering of web search results," Master, Poznań University of Technology, Poland, 2003.

[36] S. Dominich, *The Modern Algebra of Information Retrieval*: Springer-Verlag Berlin Heidelberg, 2008.

[37] W. B. Michael and C. Malu, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, 2008.

[38] L. Xiang-Wei, H. Pi-Lian, and W. Hui-Ying, "The research of text clustering algorithms based on frequent term sets," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 2352-2356 Vol. 4.

[39] X. Liu and P. He, "A Study on Text Clustering Algorithms Based on Frequent Term Sets," in *Advanced Data Mining and Applications*, ed, 2005, pp. 347-354.

[40] B. Fung, K. Wang, and M. Ester, "Hierarchical document clustering using frequent itemsets," in *Proceedings of the SIAM International Conference on Data Mining*, 2003, pp. 59-70.

[41] F. Beil, M. Ester, and X. Xu, "Frequent term-based text clustering," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, 2002, pp. 436-442.

[42] A. Spink, M. Park, B. J. Jansen, and J. Pedersen, "Multitasking during Web search sessions," *Information Processing & Management,* vol. 42, pp. 264-275, 2006.

[43] L.-C. Chen, C.-J. Luh, and C. Jou, "Generating page clippings from web search results using a dynamically terminated genetic algorithm," *Information Systems,* vol. 30, pp. 299-316, 2005.

[44] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed.: Morgan Kaufman Publishers, 2006.

[45] M. Kantardzic, *Data Mining: Concepts, Models, Methods and Algorithms*: John Wiley & Sons, 2003.

[46] D. T. Larose, *Discovering Knowledge in Data. An Introduction to Data Mining*: John Wiley & Sons, Inc., 2005.

[47] D. T. Larose, *Data Mining Methods and Models*: John Wiley & Sons, Inc., 2006.

[48] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD workshop on text mining*, Boston, MA, USA., 2000, pp. 1-20.

[49] P. Berkhin, "Survey Of Clustering Data Mining Techniques," Accrue Software, Inc.2002.

[50] O. Watanabe, T. Zeugmann, and X.-S. Yang, "Firefly Algorithms for Multimodal Optimization," in *Stochastic Algorithms: Foundations and Applications*. vol. 5792, ed: Springer Berlin / Heidelberg, 2009, pp. 169-178.

[51] C.-M. Wang and Y.-F. Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications,* vol. 37, pp. 2826-2837, 19 September 2009 2009.

[52] B. Panigrahi, V. Pandi, S. Das, and A. Abraham, "Population Variance Harmony Search Algorithm to Solve Optimal Power Flow with Non-Smooth Cost Function," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 65-75.

[53] P. Tangpattanakul, A. Meesomboon, and P. Artrit, "Optimal Trajectory of Robot Manipulator Using Harmony Search Algorithms," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 23-36.

[54] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure," in *International Conference on Management of Data*, 1999, pp. 49-60.

[55] X. Wang and H. J. Hamilton, *DBRS: A Density-Based Spatial Clustering Method with Random Sampling*, 2003.

[56] R. Forsati and M. Mahdavi, "Web Text Mining Using Harmony Search," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 51-64.

[57] G. Bo, M. Huang, W. Ip, and X. Wang, "The Application of Harmony Search in Fourth-Party Logistics Routing Problems," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 135-145.

[58] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda*, et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems,* vol. 14, pp. 1-37, 2008.

[59] M. Fesanghary, "An Introduction to the Hybrid HS-SQP Method and Its Applications," in *Recent Advances In Harmony Search Algorithm*. vol. 270, ed: Springer Berlin / Heidelberg, 2010, pp. 99-109.

[60] M. Ester, H.-p. Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996, pp. 226-231.

[61] H. Jiang, J. Li, S. Yi, X. Wang, and X. Hu, "A new hybrid method based on partitioning-based DBSCAN and ant clustering," *Expert Systems with Applications,* vol. 38, pp. 9373-9381, 2011.

[62] K. Mumtaz and D. K. Duraiswamy, "An Analysis on Density Based Clustering of Multi Dimensional Spatial Data," *Indian Journal of Computer Science and Engineering,* vol. 1, pp. 8 - 12.

[63] F. d. A. T. De Carvalho and Y. Lechevallier, "Partitional clustering algorithms for symbolic interval data based on single adaptive distances," *Pattern Recognition,* vol. In Press, Corrected Proof.

[64] M. H. Chehreghani, H. Abolhassani, and M. H. Chehreghani, "Improving density-based methods for hierarchical clustering of web pages," *Data & Knowledge Engineering,* vol. 67, pp. 30-50, 2008.

[65] R. Averill. (2012, 02/21/2013). *Series Hybrid vs. Parallel Hybrid* [Blog]. Available: http://www.redcedaru.com/blog/series-hybrid-vs-parallel-hybrid-04-18-2012

[66] Wikipedia. (2013). *Information Retrieval*. Available: http://en.wikipedia.org/wiki/Information_retrieval#Performance_and_correctness_measures

[67] Wikipedia. (2013). *False Discovery Rate*. Available: http://en.wikipedia.org/wiki/False_Discovery_Rate

[68] A. Bernardini, C. Carpineto, and M. D'Amico, "Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering," presented at the Web Intelligence, 2009.

[69] C. Carpineto and G. Romano, "Optimal meta search results clustering," presented at the Research and Development in Information Retrieval, 2010.

[70] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita, *Topical clustering of search results*, 2012.

[71] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," in *Research and Development in Information Retrieval*, 1998, pp. 46-54.

[72] C. Cobos and E. Leon, "State of the Art in Web Document Clustering Engines," ed. Popayán, 2011, p. 16.

[73] C. Cobos and E. León, "Metabuscador que Agrupa Documentos Web Enriquecido con una Taxonomía, Ontologías e Información del Usuario," presented at the Segundo Simposio de Socialización de la Investigación, Bogotá, Colombia, 2010.