HIBRIDACIÓN DE LA MEJOR BÚSQUEDA ARMÓNICA GLOBAL Y EL ALGORITMO K-MEANS PARA EL CLUSTERING DE DOCUMENTOS WEB

JENNIFER KATTERINE ANDRADE ROJAS
WILLIAM ANDRÉS CONSTAÍN DÍAZ

Director: Mag. CARLOS ALBERTO COBOS LOZADA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
POPAYÁN, FEBRERO DE 2010

AGRADECIMIENTOS

Nuestros agradecimientos a la Universidad del Cauca institución que nos forjó como personas, brindándonos la oportunidad a través del programa de Ingeniería de Sistemas para realizar nuestros estudios de pregrado.

A nuestro director, Mag. Carlos Alberto Cobos Lozada por su tiempo, dedicación, conocimientos y consejos compartidos durante la carrera y en la tesis de grado.

A nuestras familias por ser nuestro apoyo incondicional.

A nuestros compañeros, amigos y profesores por su acompañamiento durante el tiempo que estuvimos en la Universidad.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

En años recientes, el clustering de documentos web se ha convertido en un campo de investigación muy interesante. Esta representación alternativa de resultados está basada en la también llamada hipótesis de cluster [1], de acuerdo con lo cual el clustering de documentos podría ser beneficioso para los usuarios de un sistema de recuperación de información, debido a que es probable que los resultados que son relevantes para el usuario están cerca entre sí en el espacio de documentos, y por lo tanto tienden a caer dentro de un número relativamente reducido de clusters [2] y reducen el tiempo de búsqueda.

Actualmente se encuentran varias propuestas de algoritmos de clustering de documentos web que retoman algunos elementos importantes de los enfoques clásicos del clustering y tratan de dar solución a parte de los problemas encontrados en este campo, pero ninguno satisface completamente los requisitos específicos del clustering de documentos web [3, 4]. Por esta razón se presenta una propuesta de un nuevo algoritmo para el clustering de documentos web que hibrida la Mejor Búsqueda Armónica Global y el algoritmo de K-means.

A lo largo de este documento se presenta el proceso seguido para la realización del proyecto y los conceptos teóricos relevantes necesarios para el desarrollo del mismo. A continuación se hace una descripción general del contenido de este documento y la organización del mismo.

CAPÍTULO I – INTRODUCCIÓN

En este capítulo se presenta la problemática que originó el proyecto, la justificación, los objetivos y los principales resultados obtenidos.

CAPITULO II – CONTEXTO TEÓRICO

En este capítulo se describen las bases teóricas que enmarcan el proyecto, se tiene en cuenta los conceptos de clustering, sus tipos y se describen en detalle los algoritmos Búsqueda Armónica (Harmony Search - HS), Mejor Búsqueda Armónica Global (Global Best Harmony Search - GBHS) y las hibridaciones realizadas entre los algoritmos HS y K-means, así como los conceptos utilizados en el preprocesamiento de documentos.

CAPITULO III – ALGORITMO PARA CLUSTERING BASADO EN GBHS Y K-MEANS

En este capítulo se presenta en detalle los aspectos relacionados con el algoritmo propuesto para clustering de documentos web. Se describe la etapa de pre-procesamiento realizada, los parámetros y los pasos del algoritmo. Aquí, también se describen los aspectos relacionados con el desarrollo del meta buscador, entre ellos la metodología seguida, la arquitectura del sistema, el análisis, diseño y la

implementación. Además se describe en qué consistieron y cómo se desarrollaron las diferentes pruebas y los resultados de la evaluación.

CAPITULO IV – CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Aquí se describen las conclusiones que se generaron después de la culminación del proyecto y se establecen posibles mejoras o elementos adicionales que se puedan incluir en un trabajo futuro para la continuidad del proyecto.

CAPITULO V – REFERENCIAS BIBLIOGRÁFICAS Y GLOSARIO

Este capítulo contiene la bibliografía y documentación empleada en la realización del proyecto, incluye además el catálogo de palabras utilizadas en el contexto del proyecto con su respectiva definición.

PARTE I – INTRODUCCIÓN

1. PLANTEAMIENTO DEL PROBLEMA

En la actualidad la red Internet ofrece un gran número de servicios fundamentales a los internautas. Entre ellos se destacan el correo electrónico, la Web, el FTP, y los grupos de noticias. En la Web se alojan un gran número de páginas de múltiples contenidos. Desde sus orígenes en 1991, la Web ha crecido hasta abarcar diversos recursos de información [5]. El último estudio realizado por Netcraft Ltda. reportó un total de aproximadamente 186'727.854 sitios en Internet y este número aumenta cada día [6]. Se ha determinado que la segunda actividad más popular en Internet después del e-mail es la búsqueda de información, y debido a esta alta demanda existen cientos de buscadores de propósito general y miles de buscadores especializados [7]. Buscadores como Google (www.google.com), Yahoo! (www.yahoo.com) y MSN Live (www.live.com) son muy utilizados para localizar información en la Web. Cuando un usuario envía una consulta al buscador, los resultados son, normalmente, una larga lista de documentos ordenados. Es posible que los usuarios no encuentren lo que buscan en los 10 primeros documentos de la lista, y examinar los documentos resultantes uno por uno es un proceso que consume mucho tiempo y esfuerzo. Por lo tanto, cuando los usuarios no encuentran un documento que se ajuste a lo que necesitan después de 10 a 20 clicks, pueden desistir de la búsqueda. Por esta razón la precisión en la recuperación de información es un tema primordial para mejorar la eficiencia de los buscadores web [8].

Aunque muchos algoritmos y técnicas de recuperación de la información han sido publicados antes de la invención de la World Wide Web, hay características únicas de este nuevo medio que hacen que estas técnicas sean inapropiadas o insuficientes para la búsqueda web. Los algoritmos de recuperación de la información fueron desarrollados para colecciones de documentos relativamente pequeñas y coherentes, tales como artículos de periódicos o catálogos de libros en una biblioteca. La Web, por otra parte, es masiva, menos coherente, extremadamente cambiante, y está extendida sobre computadores distribuidos geográficamente [9].

Por lo anterior, es difícil para los usuarios realizar consultas eficientes con la ayuda de los buscadores tradicionales. Una forma de enfrentar esta necesidad, es organizar los grandes conjuntos de documentos en categorías por medio del clustering (agrupamiento) de documentos, dado que los resultados cubren normalmente varios temas y los usuarios pueden estar interesados solo en uno de ellos. Luego, agrupar los documentos similares dentro de grupos ayuda a los usuarios a encontrar la información más rápidamente y les permite enfocar su búsqueda en la dirección apropiada [3] [10].

En el clustering de documentos web se han desarrollado numerosos algoritmos con el fin de obtener cada vez mejores resultados en la calidad del agrupamiento, la cual debe tener en cuenta los principales requisitos del agrupamiento de documentos web [8] [4]. La mayoría de algoritmos de clustering de documentos web muestran una precisión en el agrupamiento que se encuentra entre el 60% y el 80% con respecto a los resultados esperados por el usuario. Esto muestra que existe mucho trabajo por desarrollar. Por otra parte el surgimiento de nuevos algoritmos en esta área aumentan las posibilidades de alcanzar resultados más precisos que los actuales, por ejemplo el surgimiento en 2008 de la Meta Heurística de la Mejor Búsqueda Armónica Global

[11], que promete una forma más eficiente que los algoritmos evolutivos (Genéticos, Meméticos) como estrategia de búsqueda global en problemas combinatoriales. Por lo anterior, en el presente proyecto se planteó desde su inicio la siguiente pregunta de investigación ¿Es posible encontrar resultados más precisos en el proceso de clustering de documentos web haciendo uso de un algoritmo inspirado en la mejor búsqueda armónica global?

2. JUSTIFICACIÓN

Partiendo de la pregunta de investigación previamente planteada y que a la fecha se han propuesto algoritmos de clustering de documentos web que ofrecen buenos resultados en el agrupamiento, existe la necesidad de superar ciertas deficiencias importantes, entre ellas la sensibilidad al ruido, la necesidad de definir previamente el número de grupos y la asignación de nombres adecuados a los grupos que se formen. Este proyecto presenta una alternativa de solución a través de la hibridación de los algoritmos de la Mejor Búsqueda Armónica Global [11] y de K-means [12], así como del uso del Criterio de Información Bayesiano (BIC) [13] y el índice de Davies-Bouldin (DB) [14] que permitió hallar automáticamente el número de grupos y un enfoque de frases frecuentes [15] y de términos estadísticamente representativos que permitieron asignar etiquetas descriptivas a los grupos obtenidos.

Desde un punto de vista práctico, este proyecto es conveniente puesto que buscó presentar al usuario resultados más precisos al realizar una consulta en Internet, evitándole de este modo, revisar una larga lista de resultados y permitiéndole enfocar mejor su consulta y encontrar los documentos relevantes de forma más rápida que con los buscadores tradicionales[1,2,3]. Además, el nuevo algoritmo se implementó en un meta-buscador web (disponible en http://spar.unicauca.edu.co/gruweb), que en primera instancia beneficiará a los docentes y estudiantes de la Universidad del Cauca.

Por otra parte el proyecto se encuentra enmarcado dentro del clustering de documentos web, tema que es de gran importancia internacional a nivel investigativo. Específicamente se hibridó la mejor búsqueda harmónica global y K-means, ya que no se encontró ninguna referencia que use la combinación mencionada, además los resultados obtenidos de la evaluación del algoritmo, tanto en laboratorio, como con usuarios, fueron mejores que los ya reportados por otros algoritmos. En las pruebas de laboratorio se utilizaron data sets estándar (Reuter21578[4] y DMOZ[5]) de prueba que permiten comparar los resultados con otros resultados obtenidos en la comunidad científica.

Las herramientas tecnológicas (Microsoft: Microsoft Visual Studio 2005, Microsoft SQL Server 2005 y Microsoft Project 2003) necesarias para la realización de este proyecto fueron las adecuadas; ellas fueron inicialmente seleccionadas con base en la experiencia que el GTI ha obtenido en los últimos seis (6) años de trabajo, específicamente a las experiencias exitosas en el desarrollo de proyectos relacionados

[1] www.google.com
[2] www.yahoo.com
[3] www.msn.com
[4] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[5] http://www.dmoz.org/

con Búsqueda Web (como por ejemplo: Buscador Inteligente Basado en Minería de Datos [16]) y finalmente a la disponibilidad del software, documentación y materiales de aprendizaje que se tiene en la Universidad del Cauca, gracias al programa MSDN Academic Alliance[6].

Durante el desarrollo del proyecto se aplicaron varios conceptos aprendidos en el transcurso de la carrera de ingeniería de sistemas, y se ha investigado sobre conceptos nuevos, necesarios para la consecución del producto final. Con lo anterior, los autores han podido demostrar su capacidad para plantear y resolver problemas relacionados con su formación como profesionales y como investigadores en formación del alma mater.

3. OBJETIVOS

A continuación se muestran los objetivos del proyecto, conforme fueron aprobados por el Comité de Investigaciones de la Facultad de Ingeniería Electrónica y Telecomunicaciones en el documento de anteproyecto.

3.1. OBJETIVO GENERAL

Proponer un algoritmo para el clustering de documentos web inspirado en la hibridación de la meta heurística conocida como la mejor búsqueda armónica global con el algoritmo k-means y evaluar sus resultados con medidas clásicas de recuperación de la información.

3.2. OBJETIVOS ESPECÍFICOS

- Seleccionar y/o adaptar algoritmos para el pre-procesamiento (análisis léxico, eliminación de palabras vacías, stemming e identificación del lenguaje) de snippets[7] resultados de consultas realizadas por Google, Yahoo y/o MSN Live para unificar la entrada del algoritmo de clustering de documentos Web.

- Modelar un algoritmo de clustering de documentos web que hibride la meta heurística de mejor búsqueda armónica global con el algoritmo k-means para definir el número de grupos (k) apropiados para el agrupamiento, teniendo en cuenta frases frecuentes en los documentos y un etiquetado de grupos que tenga sentido para los usuarios.

- Evaluar el algoritmo propuesto a través de dos medidas clásicas del área de la recuperación de la información, a saber, satisfacción del usuario (un grupo de estudiantes de Ingeniería de Sistemas de la Universidad del Cauca) y relevancia (a

[6] Acuerdo que vincula a Microsoft con entidades educativas universitarias, en la cual se permite tener acceso a software de desarrollo con propósitos académicos.
[7] Resumen de un documento

través de precisión[8], Exhaustividad[9] y medida F[10] en el data set de Reuters 21578[11], ampliamente usados en el área de recuperación de la información).

4. RESULTADOS OBTENIDOS

- Prototipo de un buscador. Una aplicación Windows para realizar clustering de documentos utilizando la colección de Reuters 21578, unos conjuntos de prueba basados en DMOZ, así como para realizar búsquedas en la web.

- Artículo: *Web document clustering based on global-best harmony search, k-means, frequent terms sets and Bayesian information criterion*. En proceso de evaluación en el congreso internacional denominado 2010 IEEE Congress on Evolutionary Computation (IEEE CEC 2010) a realizarse en Barcelona, España del 18 al 23 de Julio de 2010. Sitio Web: http://wcci2010.org/

- Meta-buscador Web. Aplicación web para la realización de búsquedas en Internet basado en los resultados de Google, Yahoo! Y MSN Live, código fuente e instaladores. Disponible en http://spar.unicauca.edu.co/gruweb.

- Monografía del trabajo de grado. Corresponde al presente documento, donde se describe el proceso seguido en el desarrollo del proyecto, los problemas que se presentaron, las respectivas soluciones, los aportes más sobresalientes, las conclusiones y recomendaciones para el desarrollo de futuras investigaciones.

[8] $precision = \frac{\left|\{relevant\_documents\} \cap \{retrieved\_documents\}\right|}{\left|\{retrieved\_documents\}\right|}$

[9] $recall = \frac{\left|\{relevant\_documents\} \cap \{retrieved\_documents\}\right|}{\left|\{relevant\_documents\}\right|}$

[10] $F = \frac{2 \times precision \times recall}{(precision \oplus recall)}$

[11] http://www.daviddlewis.com/resources/testcollections/reuters21578/

PARTE II – CONTEXTO TEORICO

5. CLUSTERING

Una de las soluciones adoptadas para facilitar la recuperación y la organización de los documentos en la web es utilizar el Clustering de documentos, que mediante el análisis de la información agrupa automáticamente los documentos entre los que existe una asociación notable y que son relevantes para las mismas consultas [17]. Buscadores como Clusty (http://clusty.com), iBoogie (http://www.iboogie.com), Carrot2 (http://www.carrot2.org), SnakeT (http://snaket.di.unipi.it) y WebClust (http://www.webclust.com) realizan clustering de documentos web y presentan en una jerarquía de grupos los documentos recuperados.

Para iniciar un proceso de clustering de documentos web, es preciso definir los requisitos básicos de esta tarea, entre los cuales se pueden tener en cuenta los siguientes [8] [3] [4]:

- ***Relevancia**:* El algoritmo de agrupamiento debe producir clusters que agrupen los documentos relevantes para la consulta del usuario separados de los documentos irrelevantes.

- ***Reducir la alta dimensión de los documentos de texto**:* Para procesar eficientemente un conjunto de documentos retornados de la consulta a un buscador tradicional, el algoritmo de agrupamiento de texto debe tener un modo de reducir la alta dimensionalidad de los documentos. Teniendo en cuenta que un documento se compone de muchos términos, una colección de documentos genera un conjunto muy amplio de términos (atributos), y el algoritmo debe poder definir cuáles son los términos o conceptos que describen mejor los grupos y los documentos.

- ***El solapamiento entre grupos de documentos**:* Los documentos no se relacionan únicamente con un tema, por esto se debe permitir que un documento pertenezca a más de un grupo (cluster).

- ***Asociar una etiqueta significativa a cada grupo final**:* De esta manera los usuarios pueden fácilmente saber sobre que trata el grupo, ya que la etiqueta puede proporcionar una descripción adecuada del grupo.

- ***El número de grupos es desconocido antes del agrupamiento:*** Es difícil especificar un número razonable de grupos para un conjunto de datos cuando se tiene poca información sobre este. El algoritmo de agrupamiento, debe encontrar por sí mismo el número de grupos.

- ***La velocidad del algoritmo***: El método debe ser capaz de crear los grupos de información en un tiempo tolerable por el usuario.

- ***Procesamiento incremental**:* Para ahorrar tiempo, el algoritmo debe comenzar a procesar cada snippet (resumen del documento) tan pronto como es recibido de la web.

- ***Tolerancia al ruido***: Un problema potencial que enfrentan muchos algoritmos de agrupamiento es la presencia de ruido y outliers (valores anómalos) en los datos. Un buen algoritmo de agrupamiento de documentos web debe ser bastante robusto para manejar este tipo de ruido y producir grupos de alta calidad que no sean afectados por este factor.

Otro aspecto que se debe tener en cuenta en el clustering de documentos web es el modelo para la representación de los documentos. Varios modelos se han creado, pero los más utilizados por los métodos de clustering son:

- ***Bolsas de Palabras***: Uno de los modelos clásicos para la representación de documentos de texto es el Modelo de Espacio Vectorial (Vector Space Model, VSM), el cual se basa en una matriz de términos por documentos, en donde cada celda representa la frecuencia observada del término por la importancia relativa del término en la colección de documentos, conocido como TF-IDF, la similitud de un documento con otro o con una consulta realizada por el usuario se hace usualmente con distancia de cósenos [9] [18]. Una variación del VSM es el Indexado Semántico Latente (Latent Semantic Indexing, LSI) que mediante técnicas de descomposición matricial como la Descomposición en Valores Singulares (Singular Value Decomposition, SVD) encuentra la esencia de la matriz (temas representativos) y las relaciones ocultas de los términos en la colección [18]. Otra variación que ha tomado gran importancia desde hace unos cinco años es la representación del documento como una *Bolsa de Conceptos*, lo que implica que la matriz se construye teniendo en cuenta las relaciones de los términos (sinonimia y polisemia) en base a una ontología como WordNet, lo que se puede conocer como un Modelo de Espacio Vectorial Extendido por Ontologías [8] [14].

- ***Términos Frecuentes***: Con base en la idea de *Frequent Itemsets* usada en la tarea de reglas de asociación en minería de datos [19] se ha propuesto el modelo de Términos Frecuentes [20] [21] para el clustering de documentos. Un término es cualquier palabra preprocesada dentro de un documento, y un documento puede ser considerado como un conjunto de términos que ocurren en ese documento al menos una vez. Un subconjunto bien seleccionado del conjunto de todos los términos frecuentes puede ser considerado como un cluster. Un conjunto de términos frecuentes se puede usar para la descripción de un cluster, el cual consiste del conjunto de documentos que contiene todos los términos del conjunto de términos frecuentes.

- ***Frases****:* En [8] proponen el modelo de *Frases Compartidas* para representar documentos, el cual conserva la relación secuencial entre las palabras. La idea de usar secuencias de palabras (frases) en el clustering de documentos web fue propuesta inicialmente en 1997 en [22]. Una frase es una secuencia ordenada de una o más palabras. Una frase frecuente es definida como una secuencia de palabras frecuentes que aparece en al menos cierto número o porcentaje de documentos. En el modelo de *Frases Compartidas* los documentos son tratados como una secuencia de palabras y si los documentos comparten frases o no, se usa como medida de la similitud entre los documentos. Existe también, el Modelo de Frases Compartidas extendido por Ontologías [8], el cual utiliza una ontología

como WordNet para convertir las palabras en significados de palabra explorando relaciones de sinonimia y polisemia entre las palabras o términos.

Varias categorías generales o enfoques de algoritmos de clustering han sido propuestos en la literatura, incluyendo: algoritmos jerárquicos, particionales, basados en densidad, basados en grillas [23] [24]. Los algoritmos de Clustering más usados en clustering de documentos son los jerárquicos [12] y los particionales [25]. Los *algoritmos jerárquicos* representan los documentos en una estructura multinivel y semejante a un árbol. En el clustering jerárquico los objetos de datos comprometidos a un grupo dado en la etapa temprana no pueden ser movidos a un grupo diferente, además ignoran la forma y tamaño global de los grupos, no permiten el solapamiento de los mismos y el tiempo de complejidad es cuadrático [26]. Los *métodos particionales* intentan dividir una colección de documentos dentro de un conjunto de grupos. Los clusters pueden ser superpuestos o no y los datos se agrupan en un solo nivel. El clustering particional ha sido durante mucho tiempo el clustering más popular, debido a que es dinámico, tiene buen desempeño y considera la forma y el tamaño global de los grupos [27].

En el *Clustering Jerárquico* dependiendo de la dirección en la que se construya la jerarquía se pueden identificar los *métodos Aglomerativos* [23] y los *métodos Divisivos* [23]. El enfoque aglomerativo es comúnmente el más usado en el clustering jerárquico. El *Clustering Jerárquico Aglomerativo* comienza con un conjunto de objetos como grupos individuales; luego en cada paso combina los dos grupos más similares. Este proceso se repite hasta que un número mínimo de grupos se ha alcanzado, o, si se requiere una jerarquía completa entonces el proceso continúa hasta que solo queda un grupo. Este método es muy simple pero necesita especificar como calcular la distancia entre dos grupos. Por lo tanto, se producen distintas clasificaciones dependiendo del método utilizado para calcular la distancia entre grupos.

Los métodos comúnmente usados para calcular esta distancia son: el método *Single Linkage* [12], en donde la distancia entre dos clusters es el mínimo de las distancias entre un objeto de un cluster y un objeto del otro. El método del *Complete Linkage* [12] donde la distancia entre dos clusters es el máximo de las distancias entre un objeto de un cluster y un objeto del otro. El Método de *Minimum Variance* [28] se basa en la dispersión dentro de los grupos, para lo cual se calcula la suma de los cuadrados de la distancia entre cada punto y el centroide del grupo. El método *Average linkage (UPGMA)* [3], donde la similitud entre dos clusters es calculada con base en la distancia promedio entre los elementos pertenecientes a los clusters correspondientes. Este método tiene en cuenta todos los posibles pares de distancias entre los objetos en los clusters, y es considerado más fiable y robusto a los outliers [3] [12]. Una de las desventajas que presentan los métodos jerárquicos aglomerativos es que son lentos cuando se aplican a colecciones grandes de documentos. Los métodos *Single Linkage* y *Average Linkage* tienen un tiempo de complejidad $O(n^2)$ y $O(n^3)$ respectivamente [4]. Otra desventaja, es que no producen grupos solapados.

En el *Clustering Particional* [25] los algoritmos asumen un conocimiento a priori del número de clusters en que debe ser dividido el conjunto de datos, realizan una división inicial de los datos en clusters y luego mueven los objetos de un grupo a otro según se optimice un criterio predefinido o función objetivo [12]. Los algoritmos más

representativos que emplean esta técnica son: *K-means y K-medoids*. El algoritmo *K-means* es muy popular porque es fácil de implementar, y su tiempo de complejidad es O(*n*) [12], donde *n* es el número de patrones o registros. La idea principal es definir k centroides (uno para cada grupo) y luego tomar cada punto de la base de datos y situarlo en la clase de su centroide más cercano. El próximo paso es recalcular el centroide de cada cluster y volver a distribuir todos los objetos según el centroide más cercano. El proceso se repite hasta que ya no hay cambio en los clusters de un paso al siguiente [12]. Entre las desventajas que presenta este algoritmo se encuentran: se debe definir antes del agrupamiento el número de clusters deseados y es sensible a los outliers (valores atípicos) [4]. El algoritmo *K-medoids* [23] [24] se basa en encontrar *k* objetos representativos del conjunto de datos (conocidos como medoids). Cada objeto *k* se convierte en el centro de un cluster y los objetos restantes se agrupan con el objeto representativo que le es más similar. Luego se ejecuta el método con base en el principio de minimizar la suma de las diferencias entre cada objeto y su correspondiente punto de referencia (criterio de valor absoluto). En general, el algoritmo itera hasta que, eventualmente, cada objeto representativo es en realidad el medio, o el objeto localizado más centralmente de su cluster. Entre las ventajas del *k-medoids* se encuentran: primero, no presenta limitaciones en los tipos de datos y segundo, es menos sensible a la presencia de outliers, puesto que para escoger los medoids tiene en cuenta las fracciones predominantes de puntos dentro de un cluster. Una de las desventajas que presenta el k-medoids es que su complejidad es O(n^2) por lo tanto no es adecuado para el agrupamiento de conjuntos de datos moderadamente grandes, y en comparación con el k-means es más lento en ejecución [29].

En el clustering de documentos web, se han realizado varias propuestas, la mayoría de ellas retoma algunos elementos importantes de los enfoques clásicos del clustering mencionados anteriormente, entre esas propuestas se destacan las siguientes:

En 1998 aparece el primer algoritmo que toma el enfoque de frases frecuentes compartidas por los documentos, llamado *Suffix Tree Clustering (STC)* [4]. STC tiene tres pasos lógicos: "limpieza" del documento, identificación de grupos base mediante un árbol de sufijos y combinación de los grupos base en agrupaciones. Una clara ventaja de STC es que usa frases que proporcionan descripciones concisas y significativas de los grupos. Sin embargo, los umbrales de STC desempeñan un papel importante en el proceso de formación de grupos, y resultan especialmente difíciles de afinar. Asimismo, la poda heurística de frases tiende a eliminar frases de alta calidad, dejando sólo las menos informativas y cortas. Además, si un documento no incluye ninguna de las frases extraídas este no se incluirá en los resultados aunque pueda ser pertinente [27]. Otra desventaja de STC es que no reduce la alta dimensionalidad de los documentos de texto, puesto que su complejidad es bastante alta para grandes bases de datos. Además, ignora las relaciones semánticas y léxicas entre los términos [8].

En el 2000 se da a conocer el algoritmo *Bisecting K-means* [27], un enfoque superior al método K-means básico en términos de precisión y eficiencia. En este algoritmo, inicialmente la base de datos entera es tratada como un cluster. Basado en una regla, selecciona un grupo para dividirlo en dos usando el algoritmo k-means básico. Este paso de bisección se repite hasta que se obtiene un número deseado de clusters. El algoritmo Bisecting K-means presenta las siguientes desventajas: no da una

descripción a cada cluster, no tiene pasos para reducir la alta dimensionalidad de los documentos de texto durante el agrupamiento y el usuario debe especificar el número deseado de grupos antes del proceso de agrupamiento [8].

En el 2001 se propone a *SHOC (Semantic, Hierarchical, Online Clustering)* [30], un enfoque de agrupamiento semántico, jerárquico en línea, que combina las técnicas de descubrimiento de frases y la Indexación Semántica Latente. El algoritmo SHOC utiliza una estructura de datos llamada arreglo de sufijos para identificar frases completas y sus frecuencias en tiempo O(*n*), siendo *n* la longitud total de los documentos procesados. La continua definición de grupos permite la superposición de categorías. Además, proporciona un método para ordenar los documentos dentro de los clusters. Uno de los inconvenientes de SHOC es que existen pocos detalles sobre los valores de los umbrales del algoritmo y del método que se utiliza para etiquetar los grupos resultantes. Además, en algunos casos el uso de SVD produce grupos continuos no intuitivos [31].

En el 2002, se propone el algoritmo *FTC (Frequent Term-Based Clustering)* [20], éste usa los conjuntos frecuentes de palabras compartidos entre los documentos para medir su proximidad en el clustering de texto. El algoritmo *FIHC (Frequent Itemset-based Hierarchical Clustering*) [21] mide la cohesividad de un cluster directamente usando conjuntos frecuentes de palabras, de modo que los documentos en el mismo cluster se espera que compartan más conjuntos frecuentes de palabras que con los que están en grupos diferentes. Una ventaja de los algoritmos FTC y FIHC es que asignan la etiqueta del grupo basado en el conjunto frecuente de palabras que son compartidas por los documentos en cada cluster. Un problema de FTC y FIHC es que dependen fuertemente de los conjuntos frecuentes de palabras, los cuales son desorganizados y en algunos casos no pueden representar bien los documentos [8].

En el 2003 se presenta el algoritmo *Lingo* [32] [33] como una mejora de SHOC y STC, el cual, a diferencia de la mayoría de los algoritmos, primero intenta descubrir nombres descriptivos para los clusters y solo después procede a asignar a cada grupo los documentos que coincidan. Específicamente, extrae frases frecuentes desde los documentos de entrada, esperando que ellos sean la fuente más informativa de descripciones apropiadas para los temas. Después, ejecutando la reducción de la matriz original de términos por documentos usando SVD, intenta descubrir cualquier estructura implícita existente de los diversos temas en los resultados de la búsqueda y define el número de grupos (valor de k) que se deben formar. Finalmente, relaciona las descripciones de grupo con los temas extraídos y asigna los documentos relacionados a ellos. Una de las desventajas que presenta este algoritmo es que la fase de separación de temas normalmente requiere transformaciones algebraicas computacionalmente costosas, por ejemplo, SVD.

En el 2004 se propone el algoritmo *Tolerance Rough Set Clustering (TRC)* [34], que se basa principalmente en una adaptación del método K-means. Esta adaptación hace que el algoritmo TCR se ejecute de forma rápida y conserve una buena calidad en los clusters. TRC utiliza un Tolerance Space (Espacio Tolerante) que está compuesto por las clases tolerantes las cuales agrupan los términos conceptualmente relacionados (frases). Para determinar las clases tolerantes se tiene en cuenta la co-ocurrencia de términos en todos los documentos. El uso del Tolerance Space y de una aproximación

superior para enriquecer la relación entre documentos y documentos-clusters permite al algoritmo descubrir semejanzas sutiles no detectadas de otro modo. El algoritmo TRC utiliza las frases recuperadas de los documentos como candidatas para la descripción de los clusters. TRC presenta las siguientes desventajas: no emplea una técnica adecuada para identificar los clusters iniciales, por lo tanto el número de grupos deber ser conocido antes de comenzar el agrupamiento, además, el enfoque de K-means adaptado por TRC tiene dificultades en manejar los objetos que se encuentran en los límites de la región, es decir, las partes del espacio de agrupamiento que están esparcidas no pueden ser clasificadas dentro de los clusters relacionados con la consulta del usuario, por lo tanto estos documentos se ubican en el cluster denominado “Otros documentos”, cuyo tamaño es relativamente grande.

En el 2007 se propone un algoritmo de agrupamiento de documentos, llamado *Dynamic Svd Clustering (DSC)* [2], el cual utiliza LSI en los documentos completos de contenido para realizar el clustering de documentos. DSC descubre incrementalmente el número óptimo de los valores singulares que se deben utilizar para fines del agrupamiento. Además, el paso de calcular la SVD en el algoritmo tiene un mejor desempeño en la práctica, puesto que no requiere calcular la SVD completa de la matriz original. El algoritmo se ha integrado en el buscador Noodles [2], una herramienta para buscar y agrupar los documentos Web y de escritorio. Una de las ventajas de DSC es que utiliza una estrategia para seleccionar el valor de k, es decir, el número de valores singulares que representan los “conceptos” en el espacio de los documentos, a diferencia de otras propuestas que asumen un valor fijo para k o establecen umbrales de aproximación fijos.

En el 2008 se proponen los algoritmos *CFWS* y *CFWMS* [8], que tratan los documentos de texto como secuencias de palabras (frases) frecuentes y secuencias de conceptos frecuentes, respectivamente. Estos algoritmos usan como medida de similitud si los documentos comparten o no frases o conceptos frecuentes. Además, muestran que al usar secuencias de conceptos frecuentes se obtienen mejores resultados, por esto CFWMS presenta mejores resultados. Para el pre-procesamiento de los documentos tiene en cuenta los sinónimos, hipónimos e hiperónimos proporcionados por la ontología WordNet, lo que hace que sea más preciso en capturar los temas de los documentos. Los dos algoritmos usan un árbol general de sufijos (una versión mejorada del árbol de sufijos de STC) para extraer las frases frecuentes, haciendo un análisis previo basado en el concepto de itemset frecuentes de las reglas de asociación.

En el 2009 se propone un *Algoritmo Genético Auto-Organizativo* [14] para el agrupamiento de texto basado en la ontología WordNet, la cual proporciona una evaluación más precisa de la similitud entre los documentos. Se propone también, un modelo transformado de LSI que captura apropiadamente las similitudes semánticas asociadas. Una de las ventajas del algoritmo es que mejora el rendimiento en comparación con el algoritmo Genético Estándar de clustering de documentos web [35] y K-means en entornos similares. Otra ventaja es que el modelo híbrido que combina la medida transformada basada en LSI con la medida de similitud basada en ontologías obtiene buen desempeño en términos de los parámetros *Precisión* y *Exhaustividad*. Además muestra que los resultados de LSI y del modelo ontológico son comparables. Una de las desventajas que presenta el algoritmo es que la ontología

WordNet que utiliza no es bastante precisa evaluando las similitudes semánticas en algunos dominios especializados.

6. BÚSQUEDA ARMÓNICA Y SUS VARIACIONES

6.1. BÚSQUEDA ARMÓNICA

En 2002 se desarrolla el algoritmo meta-heurístico llamado Búsqueda Armónica (Harmony Search - HS), el cual imita el proceso de improvisación de los músicos (donde los músicos improvisan los tonos de los instrumentos para obtener una armonía mejor) [36]. El algoritmo HS es fácil de entender y de implementar. HS ha sido aplicado exitosamente a muchos problemas de optimización: el problema del vendedor viajero [37], el envío económico de energía [38], la sincronización de sistemas caóticos de tiempo discreto [39], para el clustering de documentos web [26], entre otros. HS presenta varias ventajas con respecto a las técnicas de optimización, entre ellas:

- El algoritmo HS impone muy pocos requerimientos matemáticos y no requiere establecer valores iniciales para las variables de decisión;
- Como el algoritmo HS usa búsquedas aleatorias, la información derivada es también innecesaria;
- El algoritmo HS genera un nuevo vector, después de considerar todos los vectores existentes, mientras que métodos como el algoritmo genético (GA) solo consideran los vectores padres;
- HS no necesita codificar y decodificar las variables de decisión dentro de cadenas binarias;
- HS trata las variables continuas sin una pérdida de precisión.

Estas características incrementan la flexibilidad del algoritmo HS y producen mejores soluciones. El algoritmo HS funciona de la siguiente manera:

Paso 1. **Inicializar los parámetros del problema y los parámetros de HS**: El problema de optimización se define como Minimizar (o maximizar) $f(x)$ tal que $LB_i \leq x_i \leq UB_i$ donde, $f(x)$ es la función objetivo, x es una solución candidata que consiste de N variables de decisión, respectivamente. Además, los parámetros de HS se especifican en este paso. Estos parámetros son el tamaño de la memoria armónica (HMS), la tasa de consideración de la memoria armónica (HMCR), la tasa de ajuste del tono (PAR) y el número de improvisaciones (NI).

Paso 2. **Inicializar la memoria armónica**: La memoria armónica inicial es generada desde una distribución uniforme en los rangos $[LB_i, UB_i]$, donde $1 \leq i \leq N$. Esto se realiza de la siguiente manera: $x^j_i = LB_i + r \times (UB_i - LB_i)$, $j = 1,2,\ldots, HMS$ donde $r \sim U(0,1)$.

Paso 3. **Improvisar la nueva harmonía**: La generación de una nueva armonía es llamada *improvisación*. El nuevo vector armónico, $x^l = (x^l_1, x^l_2,\ldots, x^l_N)$, se genera utilizando las siguientes reglas: consideración de la memoria, ajuste del tono y la selección aleatoria.

Este procedimiento funciona de la siguiente manera:

para cada $i \in [1, N]$ **hacer**
  si $U(0,1) \leq HMCR$ **entonces** /*consideración de la memoria*/
    inicio
    $x^l_i = x^j_i$, donde $j \sim U(1,\ldots, HMS)$
    si $U(0,1) \leq PAR$ **entonces** /*ajuste del tono*/
    inicio
      $x^l_i = x^l_i \pm r \times bw$, donde $r \sim U(0,1)$ y bw es un ancho de banda arbitrario de la distancia
    fin _si
  sino /*selección aleatoria*/
    $x^l_i = LB_i + r \times (UB_i - LB_i)$
  fin_si
continuar_para

Paso 4. **Actualizar la memoria armónica**: El vector armónico generado, $x^l = (x^l_1, x^l_2,\ldots, x^l_N)$, reemplaza la peor armonía en HM, solo si el fitness (medido en términos de la función objetivo) es mejor que la peor armonía.

Paso 5. **Verificar el criterio de parada**: Terminar cuando el número máximo de improvisaciones se alcanza.

Los parámetros HMCR y PAR de la HM ayudan al método en la búsqueda de soluciones globales y locales mejoradas, respectivamente. PAR y bw tienen un profundo efecto en el desempeño de HS. Además, el ajuste de estos dos parámetros es muy importante.

En la Figura 1 se presenta el diagrama de flujo que resume los pasos del algoritmo de la Búsqueda Armónica (HS).

Existen dos mejoras importantes de HS reportadas recientemente, ellas son: la Búsqueda Armónica Mejorada y la Mejor Búsqueda Armónica Global. A continuación se explica cada una de ellas.

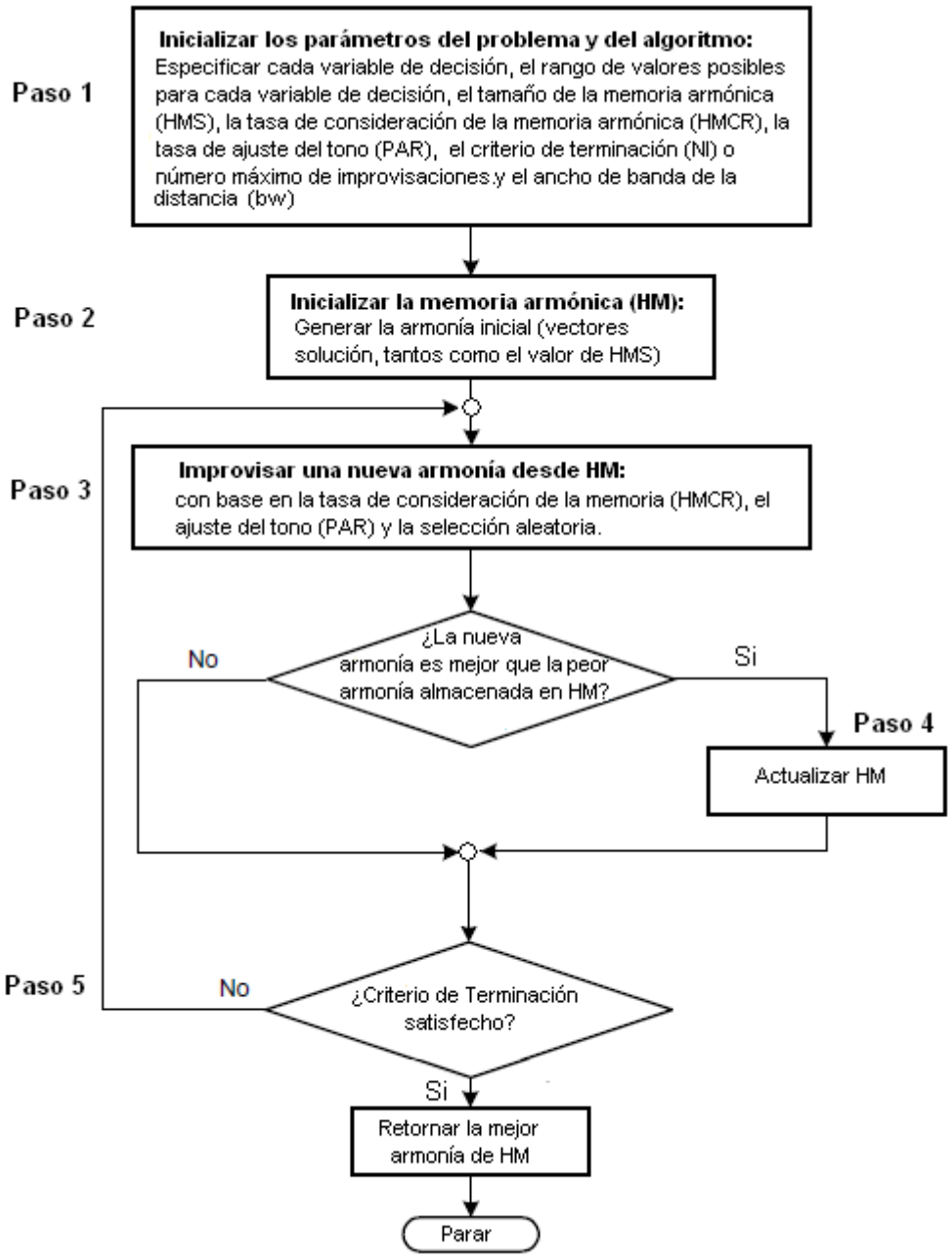

Figura 1. Diagrama de Flujo del algoritmo de la Búsqueda Armónica

6.2. BUSQUEDA ARMÓNICA MEJORADA

En el 2007 se propone una mejora a la Búsqueda Armónica, llamada Búsqueda Armónica Mejorada (Improved Harmony Search o IHS por sus sigla en inglés) [40], la cual modifica uno de los pasos del algoritmo HS para encontrar una mejor solución global y local. La diferencia más importante entre estos algoritmos está en la forma en la que ajustan los parámetros PAR (tasa de ajuste del tono) y bw (ancho de banda de

la distancia), puesto que en IHS estos parámetros son variables en el paso que corresponde a la improvisación de un nuevo armónico (paso 3). PAR y bw cambian dinámicamente con el número de generaciones como se muestra en la Figura 2 (figura tomada de [40]) y se calculan con las siguientes fórmulas:

$$\mathrm{PAR}(\mathrm{gn}) = \mathrm{PAR}_{\min} + \frac{(\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min})}{NI} \times \mathrm{gn}$$

donde,

PAR	tasa de ajuste del tono para cada generación
$\mathrm{PAR}_{\min}$	tasa mínima de ajuste del tono
$\mathrm{PAR}_{\max}$	tasa máxima de ajuste del tono
NI	número de generaciones de vectores solución
gn	número de generaciones

y

$$\mathrm{bw}(\mathrm{gn}) = \mathrm{bw}_{\max} \exp(c \cdot \mathrm{gn})$$

$$c = \frac{\mathrm{Ln}\left(\frac{\mathrm{bw}_{\min}}{\mathrm{bw}_{\max}}\right)}{\mathrm{NI}}$$

donde,

bw(gn)	ancho de banda de la distancia para cada generación
$\mathrm{bw}_{\min}$	ancho de banda de la distancia mínimo
$\mathrm{bw}_{\max}$	ancho de banda de la distancia máximo

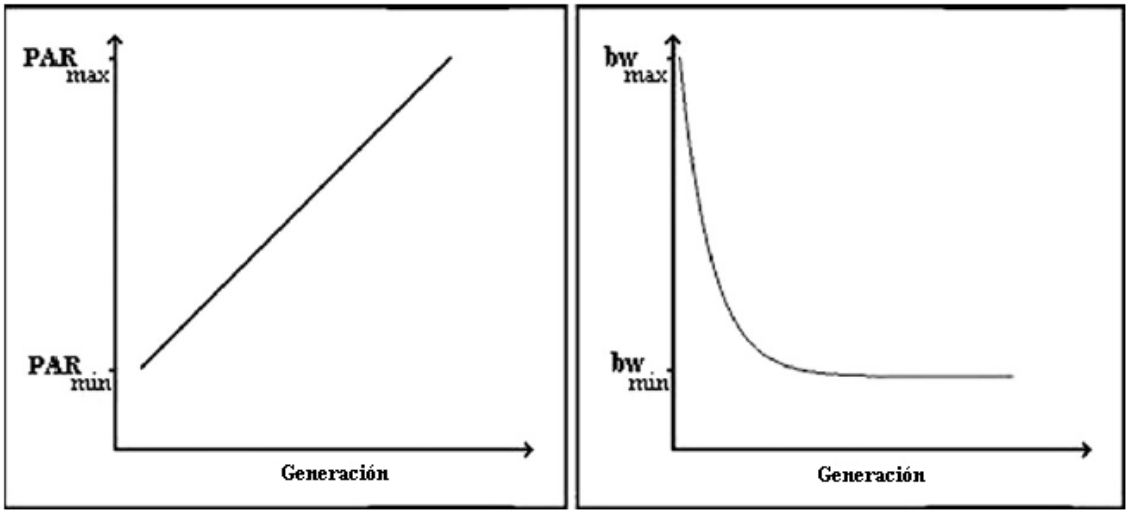

Figura 2. Variación de PAR y bw versus el número de generaciones

Con este cambio en los parámetros PAR y bw IHS logra mejorar el rendimiento de HS. La ventaja que presenta IHS es que es bueno encontrando soluciones óptimas tanto global como localmente.

6.3. MEJOR BÚSQUEDA ARMÓNICA GLOBAL

En el 2008 una nueva variante de la búsqueda armónica llamada Mejor Búsqueda Armónica Global (Global-Best Harmony Search, GHS) es propuesta en [11], donde los conceptos de inteligencia de enjambre se utilizan para mejorar el rendimiento de HS. GHS está inspirado en el concepto de inteligencia de enjambre propuesto en Particle Swarm Optimization (PSO) [11], donde un enjambre de individuos (llamados partículas) vuela a través de la búsqueda de espacio. Cada partícula representa un candidato a la solución del problema de optimización. La posición de una partícula está influenciada por la mejor posición visitada por sí misma (es decir, su propia experiencia) y la posición de las mejores partículas en el enjambre (es decir, la experiencia de enjambre). GHS modifica el paso de ajuste del tono en HS de modo que el nuevo armónico puede imitar el mejor armónico en la memoria armónica. Esto permite a GHS trabajar eficientemente en problemas continuos y discretos. En general GHS es mejor que IHS y HS, por ejemplo cuando se aplica a problemas de gran dimensionalidad y cuando hay presencia de ruido [11].

El GHS tiene exactamente los mismos pasos que IHS con la salvedad de la modificación del paso (3) que corresponde a la improvisación de un nuevo armónico, el cual se modifica de la siguiente forma:

para cada $i \in [1, N]$ **hacer**
  si $U(0,1) \leq HMCR$ **entonces** /*consideración de la memoria*/
    inicio
    $x^l_i = x^j_i$, donde $j \sim U(1,\ldots, HMS)$
    si $U(0,1) \leq PAR(t)$ **entonces** /*ajuste del tono*/
    inicio
      $x^l_i = x^{best}_k$, donde *best* es el índice de la mejora armonía en la HM y $k \sim U(1,N)$
    fin _si
  sino /*selección aleatoria*/
    $x^l_i = LB_i + r \times (UB_i - LB_i)$
  fin_si
continuar_para

6.4. LA BÚSQUEDA ARMÓNICA Y EL CLUSTERING DE DOCUMENTOS WEB

En el 2007 se comparan tres formas de hibridación entre Búsqueda Armónica (HS) [36] y K-means. Esta hibridación es llamada HSCLUST [41] y se extiende a través de tres métodos diferentes dependiendo de la etapa en la que se ejecute el algoritmo K-means: 1) Sequential Hybridization (Hibridación Secuencial), el cual ejecuta primero el método de la Búsqueda Armónica y después ejecuta el algoritmo K-means utilizando los resultados de esta búsqueda para refinarlos y obtener el resultado final. 2) Interleaved Hybridization (Hibridación Intercalada), el cual integra el método K-means

dentro del HSCLUST, es decir después de K iteraciones de la Búsqueda Armónica, el K-means utiliza el mejor vector de la memoria armónica (HM) como punto de comienzo. La memoria armónica se actualiza si los vectores optimizados localmente son mejores que los que están en la memoria. Este procedimiento se repite hasta una condición de parada. 3) Hybridizing K-means as one step of HSCLUST (Hibridación de K-means como un paso de HSCLUST), el cual introduce dentro del HSCLUST un paso del método K-means. Primero se genera una nueva solución aplicando las operaciones armónicas, luego se aplica el proceso de K-means sobre la nueva solución y si resulta ser mejor que la peor solución dentro de la HM entonces la memoria se actualiza y el HSCLUST continua iterando. Entre las ventajas de los métodos híbridos descritos se encuentra que todos mejoran el método K-means haciéndolo menos dependiente de los parámetros iniciales, tales como los centros iniciales de los clusters escogidos aleatoriamente, además estos métodos pueden encontrar clusters de mejor calidad y las soluciones que producen convergen rápidamente al mejor óptimo. Como resultado general se establece que la Hibridación de K-means como un paso de HSCLUST es la mejor solución [41].

Luego en el 2008 presentan en detalle los algoritmos HClust, HKClust e IHKClust [26] [42], los cuales se describen a continuación. El HClust está basado en el algoritmo HS (Búsqueda Armónica), el cual es un algoritmo meta-heurístico que imita el proceso de improvisación de los músicos de jazz, se desarrolló recientemente y ha sido muy exitoso en una amplia variedad de problemas de optimización [26], además es bueno encontrando áreas prometedoras de espacios de búsqueda pero no tan bueno como K-means refinando estas áreas. El HKClust es un algoritmo híbrido entre los algoritmos K-means y el clustering basado en búsqueda armónica (HS). Este método usa el algoritmo K-means para reemplazar la etapa de refinamiento en el algoritmo HClust. El algoritmo híbrido combina la fuerza del algoritmo HClust con la velocidad de un K-means y las etapas de búsqueda global y de refinación local son llevadas a cabo por estos dos módulos, respectivamente. El IHKClust es un algoritmo que integra el algoritmo HKClust y mejora los resultados de la solución. El IHKClust tiene un comportamiento iterativo de dos pasos. En el primer paso de cada iteración, el algoritmo se aproxima suavemente cerca de la solución óptima y en el segundo paso de cada iteración, K-means refina el resultado. Esta propuesta muestra buenos resultados pero también algunos inconvenientes, entre ellos: la necesidad de definir previamente el valor de k, la sensibilidad al ruido derivada de usar el algoritmo k-means y la falta de un reporte de investigación en donde se usen data sets reconocidos mundialmente, por ejemplo Reuters 21578 o TREC, esto último para poder comparar el algoritmo frente a otros en el área.

Como se observa en los algoritmos descritos anteriormente aún existen mejoras por realizar en el clustering de documentos, en este sentido está dirigido el presente proyecto, puesto que se propone un algoritmo que 1) calcula el valor de k (número de grupos) sin la intervención del usuario, 2) al usar la mejor búsqueda armónica global es menos sensible al ruido, 3) mejora el proceso de convergencia, 4) maneja esquemas de etiquetado para presentar los resultados al usuario y 5) aunque la versión más parecida de hibridación es “Hybridizing K-means as one step of HSCLUST” [41], la hibridación que se implementa no es igual porque en el algoritmo

propuesto se usa un paso de k-means (un paso de acercamiento al óptimo local) y solo a la mejor de todas las soluciones generadas se le aplica un k-means completo.

7. PREPROCESAMIENTO DE DOCUMENTOS

Con el objetivo de unificar la entrada de datos al algoritmo de clustering de documentos web, se estudiaron varias herramientas de preprocesamiento de documentos y se determinó cual era la mejor para el contexto y las necesidades del presente proyecto. A continuación se presenta el estudio realizado, la herramienta seleccionada y la utilización de la misma.

Para poder implementar mecanismos de recuperación sobre una colección de documentos de texto es necesario obtener una representación de los mismos. Dicha representación tiene en cuenta un conjunto de criterios mediante los cuales se obtienen los términos y las relaciones entre éstos. Toda implementación de un Sistema de Recuperación de Información (SRI) comienza con la tarea de preprocesamiento de la colección. Esto se debe a que no todos los términos que componen un documento son igualmente representativos de su contenido. Cuestiones como su posición, la cantidad de ocurrencias o su función lingüística, entre otras, definen el grado de importancia de cada uno de los términos.

El resultado es una representación de la colección, que es computacionalmente adecuada para los procesos siguientes y que, generalmente, se describe como Indexación de la Colección [43].

El proceso de Indexación se puede dividir principalmente en cinco operaciones de texto (o transformaciones) [18]:

1. *Análisis lexicográfico:* Se extraen las palabras y se normalizan, con el fin de tratar los dígitos, guiones, signos de puntuación, y el tipo de las letras.

2. *Eliminación de palabras vacías*: o de alta frecuencia en la colección, con el fin de filtrar palabras con valores de discriminación muy bajos para propósitos de recuperación.

3. *Stemming*: Se reducen las palabras morfológicamente parecidas a una base, con la finalidad de aumentar la eficiencia de un SRI, permitiendo la recuperación de documentos que contengan variaciones sintácticas de los términos de la consulta.

4. *Selección de términos a indexar*: Se extraen aquellas palabras simples o compuestas que mejor representan el contenido de los documentos.

5. *Construcción de estructuras de clasificación de términos*: tales como tesauros, o extracción de la estructura directamente representada en el texto, para permitir la expansión de la consulta original con los términos relacionados.

A continuación se presentan en detalle cada una de las fases.

7.1. ANÁLISIS LEXICOGRÀFICO DEL TEXTO

El análisis lexicográfico del texto es el proceso de convertir un flujo de caracteres (el texto de los documentos) en un flujo de palabras (las palabras candidatas a ser adoptadas como términos índice) o tokens (por lo que también recibe el nombre de tokenization). Esto implica detectar el comienzo y el fin de las palabras bajo diversas circunstancias (al inicio, en el medio o en el fin de una oración). Alternativamente, el analizador deber ser capaz de tratar con símbolos no alfabéticos como dígitos, caracteres especiales que componen las palabras, los cuales generalmente se reemplazan por el caracter base relacionado. Además, se debe normalizar y expandir siglas y tratar guiones como conectores de términos. Luego de realizar este tratamiento todo el texto es, generalmente, transformado a mayúsculas o minúsculas.

Si el archivo a procesar posee marcas de formato también son removidas en esta etapa. Este es el caso de los mensajes de correo electrónico, los archivos latex, las páginas HTML y demás. Sin embargo, se puede dar la situación que algún SRI requiera conservar información sobre algún atributo ligado a oraciones o términos para un uso posterior. Algunos atributos que comúnmente se conservan son los títulos de los documentos, autores, títulos y subtítulos de secciones, etc.

Otro aspecto importante a tener en cuenta es el tratamiento de los números. Su inclusión en el índice depende del tipo de colección y de la importancia que tengan como posibles términos de búsqueda. Por ejemplo, en una colección con normativas, patentes o histórica (donde hay fechas) resulta importante conservar los términos numéricos. De no ser necesaria esta información, los números son fácilmente eliminados y no aparecen en el índice.

Los guiones plantean otra decisión difícil para el analizador, debido a que hay palabras que incluyen guiones como una parte integral. El procedimiento más adecuado es adoptar una regla general y especificar excepciones caso por caso.

El caso de los tipos de letras normalmente no es importante para la identificación de los términos índice. Como resultado el analizador léxico comúnmente convierte todo el texto a letra minúscula o mayúscula. Sin embargo, una vez más pueden ocurrir escenarios particulares que requieren hacer una distinción.

7.2. ELIMINACIÓN DE PALABRAS VACÍAS

Las palabras que son demasiado frecuentes entre los documentos en la colección no son buenos discriminantes. De hecho, una palabra que ocurre en el 80% de los documentos en la colección es poco útil para propósitos de recuperación. A tales palabras se les llama stopwords y normalmente son filtradas de los potenciales términos índice.

La eliminación de palabras vacías reduce considerablemente el tamaño de la estructura de indexación. Es común obtener una compresión en el tamaño de la estructura de indexación del 40% o más solo con la eliminación de las stopwords.

Puesto que la eliminación de palabras vacías proporciona compresión en la estructura de indexación, la lista de stopwords puede ser extendida para incluir artículos, preposiciones y conjunciones. Además algunos verbos, adverbios y adjetivos pueden ser tratados como stopwords.

7.3. STEMMING

Es una técnica de reducción que permite detectar variantes morfológicas de un mismo término y reemplazarlas por el término raíz o lema. Frecuentemente, el usuario cuando realiza una consulta especifica una palabra, pero solo una variante de esta palabra está presente en un documento relevante. Los plurales, los gerundios y los sufijos del tiempo pasado son ejemplos de las variaciones sintácticas que impiden una correspondencia perfecta entre la palabra de una consulta y una palabra respectiva en el documento. Este problema puede ser cubierto con la aplicación del stemming.

El stemming es bastante útil porque mejora el desempeño de la recuperación, puesto que reduce las variantes de una misma palabra raíz a un concepto común. Además, el stemming reduce el tamaño de la estructura de indización porque el número de términos índice distintos se reduce.

Las reglas que forman los algoritmos clásicos de stemming dependen del idioma de las colecciones de los documentos a procesar. El algoritmo clásico es el de Porter [44] cuya versión original está para el idioma inglés, existen adaptaciones de las reglas para operar en otros idiomas.

7.4. SELECCIÓN DE TÉRMINOS A INDEXAR

En esta fase se tiene por objetivo analizar cada documento a indexar con el fin de extraer el conjunto de palabras (simples o compuestas) que representen de mejor forma su contenido. Por regla general, las palabras de alta frecuencia se eliminan con la asistencia de un diccionario de palabras vacías y las palabras de baja frecuencia también se eliminan. Los términos que superan el proceso de filtrado componen el vocabulario de la colección.

7.5. UTILIZACIÓN DE TESAUROS

El tesauro es una lista escogida de palabras importantes en un dominio del conocimiento, para cada palabra en esta lista, existe un conjunto de palabras relacionadas, comúnmente derivadas de relaciones de sinonimia y polisemia.

Los propósitos principales de un tesauro son básicamente: proporcionar un vocabulario estándar (o sistema de referencia) para la indexación y la búsqueda; ayudar a los usuarios con la ubicación de términos para formular una consulta apropiada; y proporcionar jerarquías clasificadas que permitan la ampliación o reducción de la consulta actual de acuerdo a las necesidades del usuario.

Para el preprocesamiento de documentos se han implementado varias herramientas que ayudan a realizar cada una de las etapas contempladas anteriormente, lo cual

permite agilizar el desarrollo de un proyecto como el presente en el que se construye un algoritmo para el clustering de documentos web, ya que la fase de preprocesamiento se puede adaptar de una de las herramientas que se presentan a continuación, esto hace posible la reutilización de código fuente ampliamente probado y usado por muchas aplicaciones garantizando un buen producto en esta fase, la cual precede la ejecución del algoritmo propuesto. A continuación se presenta el estudio que se realizó para la selección del algoritmo de preprocesamiento.

7.6. ESTUDIO DE TECNOLOGIAS PARA EL PROCESAMIENTO DE DOCUMENTOS

En el campo de la Recuperación de la información existen diferentes bibliotecas de código que ofrecen funciones importantes para los procesos de indexación y búsqueda de documentos, muchas de las cuales son Open Source, por lo tanto su código fuente está a disposición de la comunidad y puede ser reutilizado en cualquier aplicación. Es importante aclarar que las bibliotecas de funciones no se tratan de aplicaciones que se descargan, ejecutan e instalan sino de APIs (API - Application Programming Interface), a través de las cuales se añaden, con esfuerzos de programación, capacidades de indexación y/o búsqueda a cualquier sistema que se esté desarrollando. Entre las bibliotecas más destacadas se encuentran Lemur, Xapian, Terrier y Lucene. A continuación se presenta cada una de ellas.

7.6.1. Lemur.

Cómo sistema de recuperación de la información, Lemur permite todas las etapas desde la indexación a la búsqueda de documentos. Lemur aporta una API implementada en C++ y está diseñada para trabajar en todos los sistemas operativos, permite la indexación incremental e indexa atributos de los documentos [45]. Está licenciada bajo la licencia BSD (Berkeley Software Distribution) y puede ser obtenida desde [46]. Esta librería ha sido bien mantenida y muy utilizada en investigación. Proporciona indexadores capaces de leer archivos PDF, HTML y XML [47].

7.6.2. Xapian.

Es una biblioteca de funciones OpenSource de Recuperación de Información para crear motores de búsqueda, está escrita en C++, pero también se encuentra disponible en otros lenguajes como Perl, Python, PHP, Java, C#, y Ruby. Xapian contiene un API potente y adaptable, enfocado en la recuperación probabilística, que facilita los procesos de indexación y búsqueda [45]. Está disponible bajo la licencia GPL (General Publical License) desde [48]. Xapian proporciona paquetes software pre-compilados principalmente para las distribuciones de Linux [47].

7.6.3. Terrier.

Terrier está implementado en Java, permite indexar una colección de documentos de forma que se sabe cuántos documentos contienen un término determinado. También permite realizar búsquedas [45]. Terrier ha sido utilizada para búsquedas en la web y búsquedas en empresas, también en aplicaciones de escritorio, en la intranet, y en motores de búsqueda específicos, así como para el desarrollo y la evaluación de

nuevas técnicas de recuperación textual. Terrier está disponible bajo la licencia Mozilla Free License desde [49].

7.6.4. Lucene.

Lucene es una librería de funciones que permite tanto la indexación como la búsqueda de documentos. Esta creada bajo una metodología orientada a objetos e implementada completamente en java. Lucene tiene versiones para otros lenguajes como Perl, Phyton, C#, Ruby y C++. Lucene es multiplataforma, permite la indexación incremental, posee algoritmos de búsquedas confiables, realiza stemming, búsquedas por campos y permite la indexación de documentos con formato TXT, PDF, DOC, RTF, XML, PPT y HTML [47]. Lucene tiene muchas ventajas en cuanto a otras bibliotecas de funciones de Recuperación de la Información (RI). Está disponible bajo la licencia Apache Software License desde [50] y en muy pocos años ha sido la librería libre de RI escrita en java más popular del mundo.

7.6.5. Comparación de bibliotecas de funciones para la recuperación de la información.

A continuación (ver Tabla 1- Resumida de [45, 47]) se presenta una comparación de las principales bibliotecas de RI, en la cual se tienen en cuenta las características más relevantes y un test de rendimiento (ver Tabla 2- Tomada de [51]) realizado en el trabajo “A Comparison of Open Source Search Engines” [51], en el que se prueban las librerías con varios datasets.

Cabe mencionar que fue de mucha importancia investigar las tecnologías diferentes a Lucene, puesto que permitió conocer las ventajas o desventajas que una tecnología u otra pueden presentar. También fue importante saber que ambientes de desarrollo abarcan a Lucene además de Java.

	LUCENE	TERRIER	XAPIAN	LEMUR
Multiplataforma	Si	No	Si	Si
Lenguaje de implementación	Java	java	c++	c++
Soporte para otros lenguajes	perl, phyton, **c#**, ruby y c++	no	perl, python, php, tcl, c# y ruby	java y c#
Archivos que indexa	pdf, word, html, htm, txt, xml, rtf, entre otros	html , pdf, word, xls, ppt, txt	pdf, word, html, htm, txt, xml, rtf, entre otros	pdf, word, html, ppt, txt, xml, rtf, entre otros
Stemming para varios idiomas	Si	Si	Si	Si
Búsqueda mientras actualiza índice	Si	No	No	No
Indexación incremental	Si	No	no	Si
Modelo de representación	Espacio vectorial	Probabilístico	Probabilístico	Probabilístico
Búsquedas por cualquier campo	Si	No	No	No
Tecnologías OpenSource	Si	Si	Si	Si
Ultima actualización	Versión: Lucene.Net 2.3.2 Fecha: 24/07/2009	Versión: Terrier 2.2.1 Fecha: 29/01/2009	Versión: Xapian 1.0.16 Fecha: 10/09/2009	Versión: Lemur 4.10.1 Fecha: 28/07/2009

Tabla 1. Comparación de librerías de RI

	LUCENE	XAPIAN	TERRIER	LEMUR
Máximo en uso de CPU	**99.4%**	100.0%	99.5%	100.0%
Máximo en uso de RAM	20.0 %	26.8 %	58.1 %	**7.3 %**
Tamaño del índice (con respecto al tamaño de la colección)	**26%**	104%	52%	63%

Tabla 2. Comparación del rendimiento de las librerías de RI

Con base en el estudio realizado sobre las librerías más importantes que ofrecen capacidades de indexación y búsqueda, se tomó la decisión de seleccionar Lucene en su implementación para C# llamada Lucene.net, disponible bajo la licencia Apache Software License en [52]. Esta decisión estuvo motivada principalmente porque Lucene, en comparación con las otras librerías, posee mejores características, en especial: Lucene es la única librería que utiliza como modelo de representación de los documentos el modelo de espacio vectorial, el cual se considera el más apropiado para el desarrollo del algoritmo hibrido, tema central del proyecto; existe una versión para el lenguaje de programación C#, en el cual se pensó desde el inicio del proyecto para realizar la implementación; Lucene presenta mejores características en cuanto al uso de CPU y el tamaño del índice creado, lo cual es importante para reducir la dimensionalidad de los documentos y por lo tanto influye en el desempeño del algoritmo de clustering; además Lucene está siendo utilizado por una gran cantidad de aplicaciones tanto de escritorio como aplicaciones web, una lista de estas aplicaciones se puede consultar en [53].

A continuación se presenta en detalle la estructura de Lucene.net y se especifican las funciones que se reutilizaron para implementar la etapa de pre-procesamiento que antecede al algoritmo de clustering propuesto.

7.7. LUCENE.NET

Como se mencionó en la sección anterior Lucene.net es una traducción de Lucene al lenguaje C#, por lo tanto la funcionalidad ofrecida por las dos librerías es la misma, es decir la indexación y la búsqueda. Lucene permite a las aplicaciones tratar con las reglas de negocio específicas al dominio del problema mientras oculta la complejidad de la implementación de la indexación y la búsqueda, siendo un API sencilla de usar. Lucene puede verse como una capa sobre la que se construyen las aplicaciones, como se representa en la Figura 3 (figura adaptada de [54], pág. 8). Lucene no se preocupa sobre la fuente de los datos, su formato, ni incluso sobre el idioma, mientras que quien haga uso de Lucene pueda convertir esto a texto. Esto significa que Lucene se puede utilizar para indexar y buscar datos almacenados en: páginas web en servidores web remotos, documentos almacenados en sistemas locales de archivos o en bases de datos locales, archivos de texto plano, documentos de Microsoft Word, archivos HTML o pdf, o cualquier otro formato del que se pueda extraer información textual [54].

Las fases de pre-procesamiento de documentos que implementa Lucene son las típicas en esta área, las cuales se definieron previamente en la presentación del proceso de

indexación, entre ellas se encuentran: análisis lexicográfico, eliminación de palabras vacías, stemming y utilización de tesauros, de las cuales se puede utilizar independientemente cualquiera de ellas según se requiera.

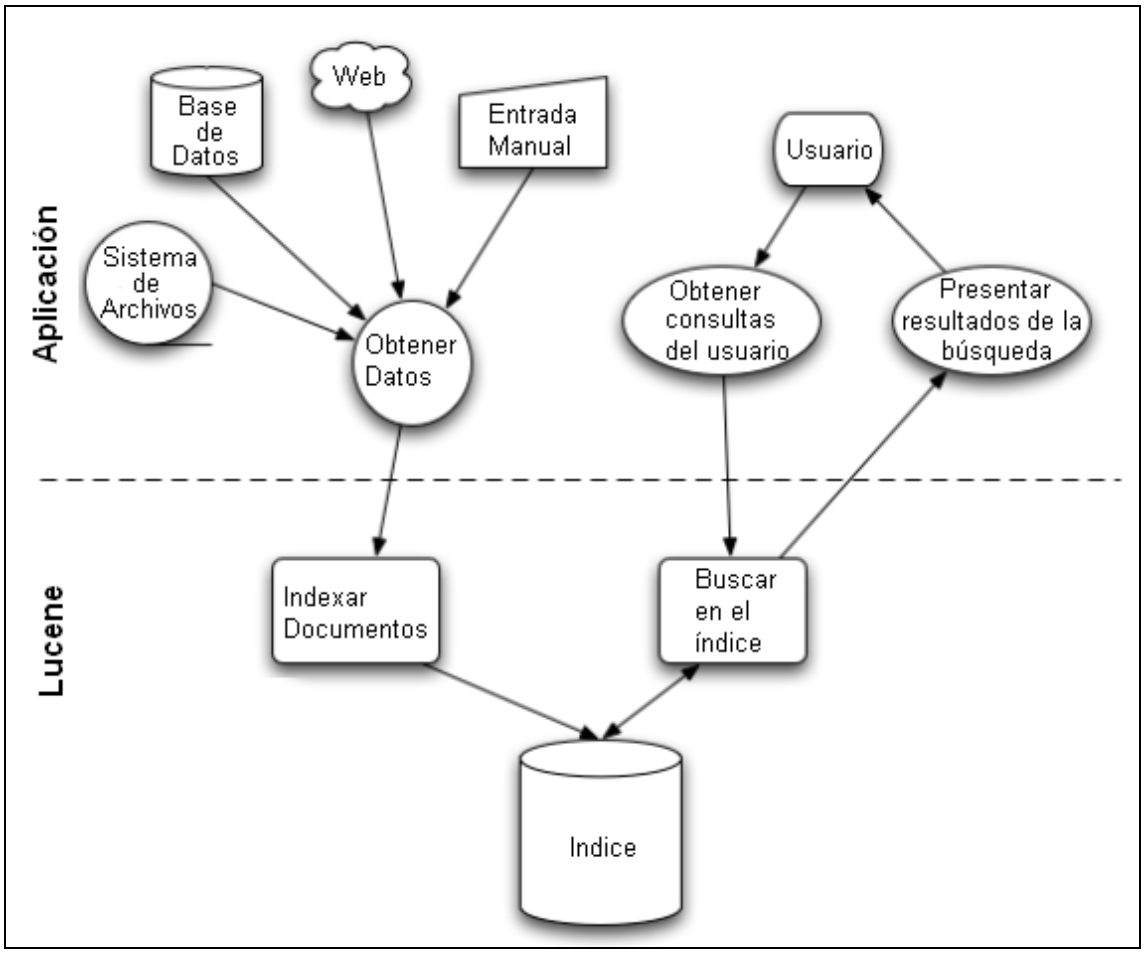

Figura 3. Integración típica de una aplicación con Lucene

Dado que Lucene es una librería de programación orientada a objetos, se compone de una serie de clases que tienen métodos y propiedades. A continuación se presentan las clases de las cuales se compone el API [54, 55].

7.7.1. Clases para el proceso de indexación.

1. **IndexWriter**: Es el componente central del proceso de indexación. Esta clase crea un nuevo índice y añade documentos al índice existente. Es un objeto que da acceso de escritura al índice pero no de lectura o de búsqueda.

2. **Directory**: Es la localización del índice de Lucene. En las aplicaciones es necesario almacenar el índice de Lucene, para lo cual se deben implementar las siguientes clases:

a. **FSDirectory**: Mantiene la lista real de archivos en un directorio.

b. **RAMDirectory**: Mantiene los datos en memoria. Para mayor rapidez de búsqueda, cuando un test finaliza, el índice es automáticamente destruido.

3. **Analyzer**: Se especifica en el constructor de la clase IndexWriter. Extrae los términos índice del texto que se va a indexar. Si el contenido a ser indexado no está en formato texto, debe convertirse primero.

4. **Document**: Es una colección de campos. Los campos de un documento presentan los metadatos asociados a ese documento, y son los típicos campos de autor, título, materia, fecha de modificación, etc. Estos campos son indexados y almacenados en forma separada como campos de un documento.

5. **Field**: Es una porción de los datos que es consultada y localizada en el índice durante la búsqueda. Lucene ofrece 4 tipos de campos a escoger:

a. **Keyword**: No es analizado pero sí es indexado y almacenado en el índice. Se utiliza para campos cuyo valor original debe ser preservado, como por ejemplo, URL, rutas del sistema de ficheros, nombres, fechas, números de seguridad social, etc.

b. **UnIndexed**: No es analizado ni indexado pero sí almacenado su valor en el índice. Se utiliza para campos que necesitan ser desplegados con resultados de búsqueda (como una URL o una llave primaria de una base de datos), pero cuyos valores no serán buscados directamente. No debe utilizarse para almacenar campos con valores muy largos si el tamaño del índice es importante.

c. **Unstore**: El opuesto al UnIndexed. Es analizado e indexado pero no almacenado en el índice. Se utiliza para indexar una cantidad larga de texto que no necesita ser recuperado en su forma original.

d. **Text**: Es analizado e indexado. Los campos pueden ser buscados, pero debe tenerse cuidado con el tamaño del campo.

Todo campo tiene un nombre y un valor, que son pasados como argumentos al definir el tipo de campo a crear.

7.7.2. Clases para el proceso de búsqueda.

1. **IndexReader**: Es una clase que accede al índice en modo de sólo lectura.

2. **IndexSearcher**: Se utiliza para buscar qué objetos IndexReader están indexados. El IndexSearcher es importante puesto que permite acceder al índice mediante muchos métodos de búsqueda. Es una clase que analiza e indexa la consulta en modo de sólo lectura.

3. **Term**: Es la unidad más básica para la búsqueda. Similar al objeto File, consiste en una dupla de cadena de caracteres: el nombre del campo y el valor del campo.

4. **Query**: Es la clase padre común y abstracta, la cual se crea en el proceso de búsqueda. Lucene cuenta con una serie de subclases del objeto Query. El Query más básico de Lucene es TermQuery, pero existen otros como el BooleanQuery, PhraseQuery, PrefixQuery, PhrasePrefixQuery, RangeQuery, FilteredQuery y SpanQuery.

5. **TermQuery**: Es el tipo de Query básico soportado por Lucene. Se utiliza para encontrar documentos que contienen campos con valores específicos.

6. **Hits**: Es una colección de punteros para el ranking de los resultados de búsqueda.

7. **QueryParser**: Esta clase realiza un parsing de las consultas de los usuarios. Esta clase es una de las más convenientes y rápidas en su implementación dentro de Lucene. Partiendo de la consulta del usuario, la analiza y crea internamente los objetos del tipo de la clase Query para satisfacer la búsqueda.

Debido a las necesidades específicas del presente proyecto, de la implementación de Lucenet.NET, solo se utilizó la funcionalidad correspondiente al proceso de indexación, en sus fases de análisis léxico, eliminación de palabras vacías y stemming; no se necesitó reutilizar más fases del indexado, ni la funcionalidad de búsqueda, puesto que esta última se implementa como parte del desarrollo del proyecto. Las clases que se utilizaron en la indexación fueron: *IndexWriter*, *Directory*, *Analyzer*, *Document*, *Field* e *IndexReader* (Para ver en detalle remitirse al Anexo D).

PARTE III – ALGORITMO PARA CLUSTERING BASADO EN GBHS Y K-MEANS

8. PREPROCESAMIENTO PARA EL CLUSTERING DE DOCUMENTOS

La fase de pre-procesamiento para el algoritmo de agrupamiento propuesto quedó conformada por tres pasos principales que a su vez se dividen en otros pasos. El primer paso "Cargar los data sets" varía dependiendo de si el agrupamiento se realiza sobre snippets retornados de consultas a la web o si se realiza sobre las noticias de Reuters 21578 o los snippets de DMOZ. El segundo paso es "Identificar el Lenguaje de los documentos obtenidos", para lo cual se emplea la Técnica de *stopwords*. Y el tercer paso "Obtener la matriz de términos por documentos (TXD) o la matriz de términos frecuentes por documentos (TFXD)" es el mismo paso para cualquier data set que se haya escogido, en este paso se reutiliza parte del código fuente de Lucene.net para la creación del índice. A continuación se presentan en detalle los pasos mencionados.

8.1. CARGAR DATASETS

- **Agrupamiento en snippets consultados de la web**

Cuando se realiza una consulta en la web, se utilizan las APIs de búsqueda proporcionadas por Google, Yahoo y/o MSN Live, las cuales reciben la consulta a realizar y retornan una serie de snippets con sus respectivos títulos y urls, los cuales están asociados con la consulta realizada. Las APIs están disponibles en internet, desde donde se descarga el código fuente y se integra en la aplicación. El snippet que se obtiene es utilizado como fuente de datos para el siguiente paso en la fase de pre-procesamiento.

- **Agrupamiento en data set de Reuter 21578 o DMOZ**

Cuando se realiza el agrupamiento en data sets se consulta en una base de datos, si son las noticias de la colección Reuter 21578 se consulta la base de datos BDReuter, si son los snippets de la colección DMOZ se consulta la base de datos BDDmoz. Para obtener los documentos a agrupar se realizan consultas a la base de datos, de las cuales se obtienen los data sets consultados y los temas que contienen los data sets.

8.2. IDENTIFICAR EL LENGUAJE DE LOS DOCUMENTOS OBTENIDOS

Una vez obtenidos los documentos que se van a procesar se realiza una serie de pasos para identificar automáticamente el lenguaje en el que está escrito cada documento recuperado. Para tal fin, se han propuesto dos técnicas para la identificación del lenguaje [15, 56]. La primera técnica, la Técnica Trigrama, tiene en cuenta el inicio y la terminación de cada palabra, puesto que estas están asociadas con más probabilidad a un idioma que otro. La segunda, la Técnica de Small (Stop) Word, tiene en cuenta las palabras vacías de significado o *stopwords* (por ejemplo, las preposiciones o conjunciones), para ello se debe utilizar la lista de *stopwords* de cada lenguaje y se debe obtener la cantidad de *stopwords* de cada lenguaje en el documento, el lenguaje asociado a la mayor cantidad de *stopwords* halladas se define como el lenguaje del documento. En textos que contengan más de 30 palabras ambas técnicas se ejecutarán de manera satisfactoria, el error de identificación es menos del 0.5%. En textos que contengan menos de 5 palabras, sin embargo, el algoritmo basado en Trigrama presenta mejores resultados que la técnica de

stopwords, puesto que el porcentaje de error promedio es menor del 7%. Esto muestra que la identificación del lenguaje es posible aún en los títulos o en los resúmenes de los documentos.

Debido a que los textos recuperados son los snippets (resúmenes) retornados por Google, Yahoo! y MSN Live, se consideró más apropiada la técnica de las *stopwords,* puesto que los resúmenes están compuestos por suficientes palabras. Por otra parte esta técnica es más rápida ya que las *stopwords* son menos que las palabras utilizadas por la Técnica Trigrama.

Los pasos realizados en el algoritmo para la identificación del lenguaje (adaptado de [15, 56]) se presentan en la Figura 4.

```
01  Para cada documento hacer
02      Para cada lista de stopwords hacer
03              Contar las ocurrencias de los términos de la lista de stopwords en el
                documento
04      Continuar
05      Escoger la lista de stopwords que registró el número más alto de ocurrencias
06      Si el idioma de la lista de stopwords es diferente a inglés
07              Eliminar el documento
08      Sino
09              Establecer el idioma del documento como inglés
10      Fin_Si
11  Continuar_Para
```

Figura 4. Algoritmo para la identificación del lenguaje

8.3. OBTENER MATRIZ TXD O TFXD

Después de tener los documentos pre-procesados se crea una matriz de términos por documentos (TXD) o una matriz de términos frecuentes por documentos (TFXD), según sea el caso, es decir si se va a utilizar un modelo de representación en el que se tienen en cuenta todos los términos de los documentos o si el modelo de representación tendrá en cuenta solo los itemsets (conjuntos de palabras) frecuentes de los documentos. Estas opciones se implementaron con el fin de comparar cual modelo ofrecía mejores resultados en cuanto a la representación de documentos. Los pasos realizados en ambos casos se presentan a continuación.

Matriz TXD

- ***Crear índice****:* Se utilizan las fuentes de datos definidas en la sección 8.1, según sea el caso (consulta a la web o agrupamiento de noticias o snippets). Luego se “Crea el índice” y se “Indexan los documentos” (para ver detalles remitirse al Anexo D).

- ***Obtener los términos de todos los documentos****:* Se “Consulta en el índice” y se obtiene los términos de los documentos (para ver detalles remitirse al Anexo D).

- ***Crear la Matriz de términos por documentos***: Primero se crea la matriz TF, la cual relaciona los documentos con los términos y sus frecuencias asociadas, luego se crea la matriz IDF que relaciona los documentos con los términos y sus frecuencias invertidas asociadas y por último se crea la matriz de Pesos, que multiplica la matrices TF e IDF. Estas matrices se implementan de la siguiente forma:
    - *Matriz TF*: Utilizando los documentos y los términos en ellos se crea una matriz de N documentos por D dimensiones o términos, en la que se relacionan las frecuencias de los términos en cada documento. Luego cada elemento de la matriz se divide por la frecuencia más alta de la fila a la que pertenece cada elemento.

    - *Matriz IDF*: Esta matriz se crea de la misma forma que la matriz TF, la diferencia es que en lugar de relacionar las frecuencias de los términos en cada documento, se relacionan es las frecuencias invertidas de los términos en cada documento, las cuales se calculan con la siguiente fórmula:

      $Log\ (N / n_i + 1)$

      Donde, N es el número total de documentos y n_i es el número de documentos donde aparece el término i.

    - *Matriz de Pesos*: Esta matriz se calcula multiplicando las matrices TF e IDF. Esta es la matriz que se utiliza en los cálculos posteriores.

Matriz de TFXD

- ***Crear índice***: Se utilizan las fuentes de datos definidas en la sección 8.1, según sea el caso (consulta a la web o agrupamiento de noticias o snippets). Luego se “Crea el índice” y se “Indexan los documentos” (para ver detalles remitirse al Anexo D).

- ***Obtener los términos de todos los documentos:*** Se “Consulta en el índice” y e obtienen los términos de los documentos (para ver detalles remitirse al Anexo D).

- ***Obtener lista de Itemsets (fpgrowth)***: Los términos de todos los documentos consultados en el paso anterior se utilizan para obtener los itemsets frecuentes, que se obtienen después de invocar la función que extrae patrones frecuentes presentes en la lista de términos, esto se lleva a cabo utilizando el algoritmo FPGrowth, cuya implementación se tomó del código fuente disponible en Internet en CodePlex Open Source Community en la sección de Frequent Pattern Miner [57]. Para ver en detalle el algoritmo FPGrowth referirse al Anexo C.

- ***Crear la Matriz de términos frecuentes por documentos***: Este paso se realiza de manera similar a como se realiza el último paso (crear la matriz de términos por documentos) en la creación de la Matriz TXD, la diferencia es que en lugar de utilizar sólo los términos de los documentos, se utilizan los itemsets o conjuntos de términos frecuentes en los documentos.

9. EL ALGORITMO DE CLUSTERING DE DOCUMENTOS

Hasta el momento se han presentado los pasos de la etapa de pre-procesamiento de documentos, ahora se explicará el algoritmo de clustering de documentos y finalmente se explicará la etapa de etiquetado, en la que se le asignan los nombres a los grupos de documentos.

Para la construcción del algoritmo de clustering se realizó una hibridación del algoritmo de la Mejor Búsqueda Armónica Global y el algoritmo K-means, por lo tanto primero se presentarán estos dos algoritmos.

9.1. ALGORITMO MEJOR BÚSQUEDA ARMÓNICA GLOBAL

La Mejor Búsqueda Armónica Global (GBHS) [11] es una nueva variante del algoritmo Búsqueda Armónica. GBHS está inspirado por el concepto de inteligencia de enjambre como se propuso en la Optimización de Enjambre de Partículas (PSO) [58]. Los pasos en el procedimiento de GBHS son los siguientes [11]:

1: **Inicializar los parámetros del problema y del algoritmo**: El problema de optimización está definido como optimizar (minimizar o maximizar) $f(x)$ sujeta a $x_i \in X,\ i = 1,2, \ldots, N$, donde $f(x)$ es la función objetivo, x es el conjunto de cada variable de decisión x_i, N es el número de variables de decisión, X_i es el conjunto del rango posible de valores de cada variable de decisión, que es ${}_Lx_i \leq X_i \leq {}_Ux_i$, donde ${}_Lx_i$ y ${}_Ux_i$ son límites inferior y superior para cada variable de decisión. Además, los parámetros de HS son especificados en este paso, para los cuales el algoritmo establece unos valores por defecto. Estos parámetros son el Tamaño de la Memoria Armónica (HMS, un valor típico es entre 4 y 10), la Tasa de Consideración de la Memoria Armónica (HMCR, un valor típico es 0.95), la Tasa de Ajuste del Tono (PAR, un valor típico está entre 0.3 y 0.99) y el número de improvisaciones (NI) o criterio de parada [11, 26, 36, 40].

Para escoger el criterio de parada (NI) en el algoritmo propuesto se tuvo en cuenta el tiempo que transcurría cuando se aumentaba el número de improvisaciones. En la Figura 5 se observa que a medida que aumenta el número de improvisaciones, el tiempo de procesamiento también aumenta. Por lo tanto, para definir el número de improvisaciones se tuvo en cuenta el tiempo que tardaba el algoritmo en el procesamiento y el tiempo que se debe esperar para obtener los resultados de los buscadores Google, Yahoo! y MSN, para no sobrepasar el tiempo tolerable por los usuarios en la web, el cual se ha determinado según algunos estudios realizados [59-61]:

- 0.1 segundos es el límite para que el usuario perciba que el sistema está reaccionando instantáneamente.
- 1 segundo es el valor considerado como el límite para que el flujo del pensamiento se perciba como ininterrumpido.
- 10 segundos es el tiempo que tarda un usuario en quitar la atención en un diálogo si éste no reacciona.

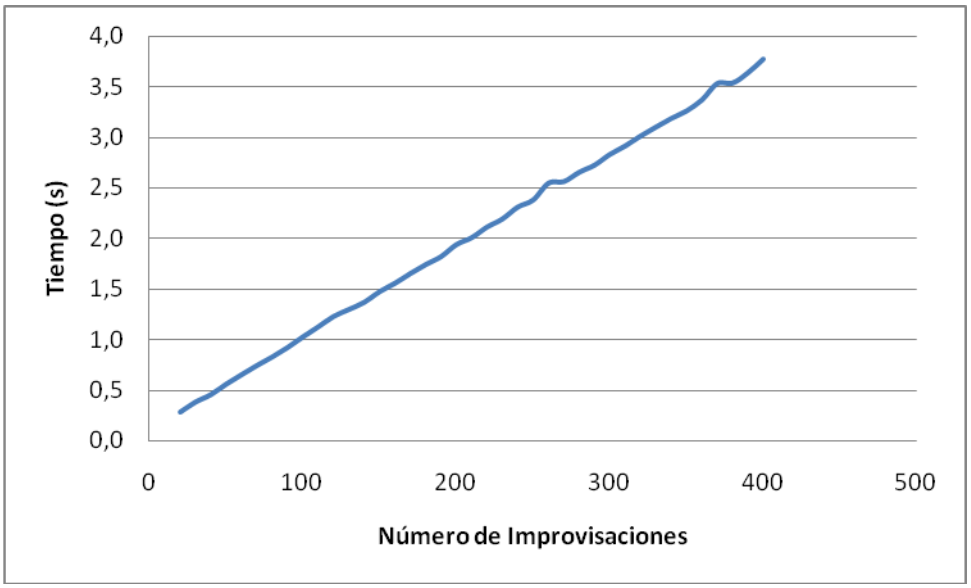

Figura 5. Tiempo de respuesta vs. el número de improvisaciones del IGBHSK

El tiempo de espera del algoritmo propuesto es en promedio 0.3 segundos cuando se establecen el Número de Improvisaciones a 30, y el tiempo de obtención de los documentos desde los buscadores Google, Yahoo! y MSN, es 2.2 segundos en promedio, para un total 2.5 segundos en el tiempo de respuesta.

2: **Inicializar la Memoria Armónica**: La Memoria Armónica (HM) es una dirección de memoria donde todos los vectores solución (conjunto de variables de decisión) son almacenadas (Ver la Figura 6). La HS inicial es generada desde una distribución uniforme en los rangos ${}_Lx_i$ y ${}_Ux_i$, donde $1 \leq i \leq N$. Este paso es llevado a cabo como sigue: $x_i^j = {}_Lx_i + \text{Rand}^*({}_Ux_i - {}_Lx_i)$, donde j=1,2,…,HMS; y Rand es un número aleatorio uniformemente distribuido entre 0 y 1 (Rand ~U(0,1)).

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix}$$

Figura 6. Memoria Armónica

3: **Improvisar una nueva armonía:** La generación de una nueva armonía es llamada improvisación. Un nuevo vector armónico, $x^T = (x_1^T, x_2^T, \ldots, x_N^T)$, es generado con base en tres reglas: la consideración de la memoria, el ajuste del tono y la selección aleatoria. En la consideración de la memoria, el valor de la primera variable de decisión x_1^T para el nuevo vector es escogido de cualquiera de los valores en el rango específico de HM $x_1^1 \ldots x_1^{HMS}$. El mismo procedimiento se sigue para las otras variables de decisión $(x_2^T,$

$x_3^T, \ldots, x_N^T$). HMCR es la tasa de elegir un valor de los valores históricos almacenados en HM, mientras que 1 - HMCR es la tasa de seleccionar aleatoriamente un valor del rango de valores posibles. Por ejemplo, HMCR de 0.95 indica que el algoritmo HS escogerá el valor de la variable de decisión desde los valores almacenados históricamente en HM con 95% de probabilidad o desde el rango completo con una probabilidad del 5% (100 - 95). PAR es la tasa de ajuste del tono, mientras 1 – PAR es la tasa sin el ajuste del tono. Si la decisión del ajuste del tono para x_i^T es positiva, entonces x_i^T es reemplazado como sigue: $x_i^T = x_I^{best}$ donde best es el índice de la mejor armonía en HM y k~U(0,1)(PSO). En este paso, la consideración de HM, el ajuste del tono o la selección aleatoria son aplicados sucesivamente a cada variable del nuevo vector armónico.

4: **Actualizar la memoria armónica:** El nuevo vector armónico, $x^T = (x_1^T, x_2^T, \ldots, x_N^T)$ reemplaza el peor vector armónico en la HM, si su *fitness* (juzgado en términos del valor de la función objetivo) es mejor que el segundo. El nuevo vector armónico es incluido en HM y el peor vector armónico existente es excluido de HM.

5: **Verificar el criterio de parada**: Si el criterio de parada (por ejemplo, el número máximo de improvisaciones, NI) se satisface, los cálculos terminan. De otro modo, los pasos 3 y 4 se repiten.

Los parámetros HMCR y PAR del GBHS ayudan al método en la búsqueda de soluciones mejoradas global y localmente, respectivamente. PAR tiene un profundo efecto en el desempeño del algoritmo GBHS. Además, el ajuste de este parámetro es muy importante.

9.2. EL ALGORITMO K-MEANS

El algoritmo K-means es un algoritmo de clustering particional. El algoritmo K-means es el más simple y más común que emplea el criterio de la suma del error cuadrado (SSE). Este algoritmo es popular porque encuentra un mínimo (o máximo) local en un espacio de búsqueda, es fácil de implementar y su complejidad temporal es *O(n),* donde *n* es el número de objetos (registros o patrones). Desafortunadamente, la calidad del resultado depende de los puntos iniciales y puede converger a un mínimo local del valor de la función de criterio si la partición inicial no se escoge apropiadamente [12, 23, 25, 62]. Las entradas de K-means son: el número de grupos (valor de K) y un conjunto (tabla, arreglo o colección) que contenga n objetos (o registros) en un espacio de características de dimensionalidad D, definido formalmente por $X = \{x_1, x_2, \ldots, x_n\}$ (En nuestro caso, x_i es un vector fila, por razones de implementación). Las salidas de K-means son un conjunto que contiene K centros. Los pasos en el procedimiento de K-means se pueden resumir como se muestran en la Figura 7.

```
001  Seleccionar una partición inicial
     Repetir
002      Re-calcular la membrecía
003      Actualizar los centros
004  Hasta (Criterio de parada)
005  Retornar la solución
```

Figura 7. Algoritmo K-means

1: **Seleccionar una Partición Inicial:** Escoger arbitrariamente K centros como una solución inicial. Estos K centros están definidos como $C=\{c_1,c_2,\ldots,c_k\}$, y cada c_j es un vector fila de D-dimensional. Existen varios enfoques para seleccionar estos K centros iniciales [38], por ejemplo Forgy sugirió seleccionar estas K instancias aleatoriamente desde el conjunto de datos. McQueen sugirió seleccionar los primeros K puntos en el conjunto de datos como las semillas preliminares y luego usar una estrategia incremental para actualizar y seleccionar los K centros reales de la solución inicial [38]. Linde propuso un método de División Binaria (BS) [38]. Huang y Harris propusieron el método de División Binaria de Búsqueda Directa (DSBS). Este método es similar a BS pero el paso de la división es mejorado a través del uso del Análisis de Componentes Principales (PCA) [38]. Entre otros. En el algoritmo propuesto, se usa la estrategia de Forgy para seleccionar los k valores iniciales.

2: **Re-calcular la Membrecía:** Para todos los objetos en el conjunto de datos es necesario re-calcular la membrecía de acuerdo a la solución actual. Varias medidas de distancia o similitud pueden ser utilizadas, por ejemplo: Distancia Euclidiana, Distancia de Manhattan, Métrica de Minkowski, Similitud de Coseno y Distancia de Mahalanobis. En el algoritmo propuesto, se usa la Similitud de Coseno definida formalmente como (1).

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

Donde d representa un documento, q representa una consulta, D es el número total de dimensiones, $W_{i,d}$ representa el peso asociado a un documento con respecto a cada grupo y $W_{i,q}$ representa el peso asociado a una consulta con respecto a cada grupo.

Cada objeto es asignado a un grupo específico. Esta asignación es fuerte o débil. En esta investigación, la asignación es fuerte y está definida por P_{ij} igual a 1 si $x_i \in c_j$, de otro modo es igual a 0.

3: **Actualizar los centros:** Para todos los grupos en la solución actual es necesario actualizar los centros de acuerdo a las nuevas membrecías de los objetos. Normalmente, el centro del grupo es el punto medio (2) de todos los objetos en el grupo, donde n_j es el número de objetos en el grupo j.

$$c_j = \frac{1}{n_j} \sum_{i=1}^{n} \left(x_i \times P_{i,j} \right) \quad where \quad n_j = \sum_{i=1}^{n} P_{i,j} \qquad (2)$$

4: **Hasta (Criterio de Parada):** Terminar la ejecución, si por ejemplo, no hay una reasignación (o una mínima reasignación) de patrones para nuevos centros de grupos, o hay una disminución mínima en un SSE. El criterio más comúnmente utilizado para distinguir la convergencia está basado en (3).

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n} P_{i,j} \left\| x_i - c_j \right\|^2 \quad (3)$$

5: **Retornar la Solución:** Retornar los K centros actuales $C=\{c_1,c_2,\ldots,c_k\}$.

En la literatura, se han reportado varios criterios para comparar dos o más soluciones de clustering y decidir cuál es mejor [27, 63, 64]. Los criterios más populares están basados en las matrices de dispersión dentro del grupo y entre grupos. En esta investigación, dos criterios fueron utilizados para encontrar automáticamente el número de grupos. El Criterio de Información Bayesiano (BIC) [65] expresado por (4) y el índice de Davies-Bouldin (DB) [14] expresado por (5).

$$BIC = n \times Ln\left(\frac{SSE}{n}\right) + k \times Ln(n) \quad (4)$$

$$DB = \frac{1}{k}\sum_{i=1}^{k} R_i$$
$$R_i = \max_{j, j\neq i}\left\{\frac{S_i + S_j}{d_{i,j}}\right\}$$
$$d_{i,j} = 1 - Sim_{\cos}(z_i, z_j) \quad (5)$$
$$S_i = \frac{1}{|C_i|}\sum_{x\in C_i}(1 - Sim_{\cos}(x, z_i))$$

Donde $|C_i|$ es el número de documentos en el grupo C_i y z_i es el centro de este grupo. S_i es el promedio de la similitud de coseno entre los documentos en el grupo C_i y su centroide z_i, $d_{i,j}$ representa la distancia entre el grupo C_i y el grupo C_j.

9.3. EL ALGORITMO ITERATIVO DE LA MEJOR BÚSQUEDA ARMÓNICA GLOBAL Y K-MEANS

El algoritmo propuesto, llamado Algoritmo Iterativo de la Mejor Búsqueda Armónica Global y K-means (IGBHSK) utiliza el algoritmo GBHS como una estrategia global de búsqueda en el espacio de la solución completa, el algoritmo K-means como una estrategia local para improvisar soluciones; y el Criterio de Información Bayesiano (BIC) o el índice de Davies-Bouldin (DB) para encontrar el número de grupos automáticamente. En cuanto a estos índices se puede utilizar uno u otro, debido a que se pretendía determinar con cual criterio el algoritmo propuesto obtenía mejores resultados. En IGBHSK, cada vector solución utilizado en el algoritmo GBHS tiene diferente número de grupos (centroides), y la función objetivo (basada en BIC o DB) del algoritmo GBHS depende de la ubicación de los centroides en cada vector solución y del número de centroides (valor de K).

IGBHSK tiene una rutina principal que ejecuta tres pasos básicos: Inicializar los parámetros del algoritmo, inicializar los mejores resultados de la memoria y llamar la rutina GBHSK en varias iteraciones y finalmente, retornar el mejor resultado. A continuación se presentan estos pasos en detalle.

1: **Inicializa los parámetros del algoritmo:** En esta investigación, el problema de optimización consiste en minimizar el criterio BIC o DB, llamado Función de *Fitness*. IGBHSK necesita unos parámetros específicos, el tamaño de los mejores resultados de la memoria (BMRS) y otros parámetros del algoritmo (HMS, HMCR, PAR y NI).

2: **Inicializar los mejores resultados de la memoria y llamar la rutina GBHSK:** La memoria de los mejores resultados (BMR) es una dirección de memoria donde los mejores vectores solución son almacenados (ver Figura 8). Cada fila en BMR almacena el resultado de un llamado a la rutina GBHSK, en un ciclo básico. Cada vector fila en BMR tiene dos partes: los centroides y el valor de fitness del vector.

$$BMR = \begin{bmatrix} Centroids_1 & Fitness_1 \\ Centroids_2 & Fitness_2 \\ \vdots & \vdots \\ Centroids_{BMRS-1} & Fitness_{BMRS-1} \\ Centroids_{BMRS} & Fitness_{BMRS} \end{bmatrix}$$

Figura 8. Mejores resultados de la memoria

Como se mencionó anteriormente, Lucene.NET es utilizado inicialmente en la etapa de pre-procesamiento. Esta etapa incluye: la eliminación de palabras vacías, algoritmo de lematización de Porter y la construcción de la matriz de términos por documentos (TDM con N documentos por D dimensiones o términos). Finalmente, el algoritmo FP-Growth [20, 24, 66] es utilizado para construir una matriz de términos frecuentes por documentos (FTDM). FTDM es utilizado para reducir la alta dimensionalidad de las colecciones de documentos y para definir las etiquetas de grupo. Además, las dimensiones con un rango igual a cero (0) son eliminadas.

3: **Seleccionar los mejores resultados:** Encontrar y seleccionar los mejores resultados de la memoria de los mejores resultados (BMR). El mejor resultado es la fila con el valor de fitness más alto (minimizar *f(x)*). Esta fila es la mejor solución de agrupamiento (centroides y fitness).

4: **Asignar etiquetas a los grupos**: el algoritmo IGBHSK contempla dos métodos de asignación de etiquetas a cada grupo. El primero es un conjunto de términos estadísticamente más representativos (SRT) basado en el concepto probabilístico introducido por Smith y Medin [67] y el segundo es similar a Lingo [15], basado en frases frecuentes (FPH). Más adelante se presentan en detalle estos métodos.

5: **Solapar los grupos**: Finalmente, cada grupo incluye documentos que quedan también en otros grupos, si estos documentos están a una distancia menor o igual que la distancia promedio del grupo.

Estos pasos se resumen en el algoritmo de la Figura 9.

```
01  Eliminación de palabras vacías
02  Algoritmo de Lematización de Porter
03  Construcción de la matriz de Términos por Documentos o la
    matriz de Términos Frecuentes por Documentos
04   Eliminar las dimensiones con un rango igual a cero
05  Para cada i Є [1, BMRS] hacer
06      BMR[i] = GBHSK (TDM o FTDM)
07  Continuar_Para
08  Seleccionar los mejores resultados
09  Asignar etiquetas a los grupos
10  Solapar los grupos
```

Figura 9. Inicializar los mejores resultados de la memoria y llamar la rutina GBHSK

IGBHSK puede ser ejecutado en diferentes formas (ver Figura 10). La primera opción que se puede escoger está relacionada con el modelo de representación de documentos (TDM o FTDM). La segunda opción consiste en seleccionar la función de fitness (BIC o DB), y finalmente, es necesario escoger el método de etiquetado (términos estadísticamente más representativos o frases frecuentes).

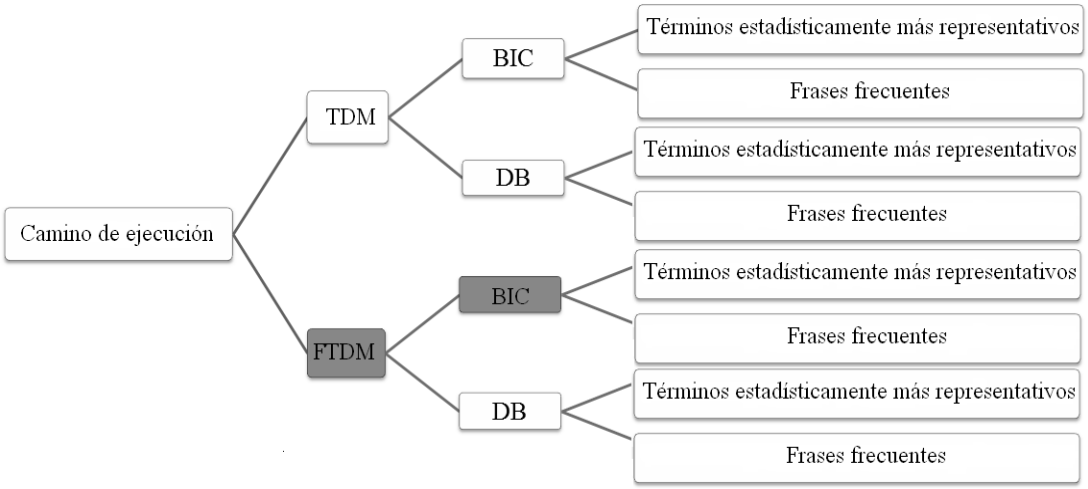

Figura 10. Caminos de ejecución del algoritmo IGBHSK

La rutina GBHSK es el algoritmo GBHS con algunos cambios, los cuales trabajan como se muestra a continuación:

1: **Inicializar la memoria armónica:** La HM es una dirección de memoria donde todos los vectores solución son almacenados. Cada vector solución es creado con un número aleatorio de centroides (k centroides con la estrategia de Forgy y valores en todas las

dimensiones) y un Fitness para esta solución. Los centroides iniciales son seleccionados aleatoriamente del conjunto de datos original (a diferencia del algoritmo original GBHS). La estructura general de HM es similar a BMR. En este paso, HMS vectores solución (centroides) son generados y luego el valor de fitness para cada vector es calculado.

2: **Improvisar una nueva armonía:** un nuevo vector armónico (centroides) es generado. Una variación del paso 3 en el algoritmo original GBHS se usa para crear los centroides en la solución actual. El proceso de selección aleatoria se ejecuta desde el conjunto de datos original (estrategia de Forgy) (ver Figura 11). Después, **un ciclo del algoritmo k-means** (Figura 7 pasos 02 y 03) se ejecuta y luego el valor de fitness para esta solución se calcula.

3: **Actualizar la memoria armónica**: el nuevo vector armónico reemplaza el peor vector armónico en la HM, si su valor de fitness es mejor que el segundo.

4: **Verificar el criterio de parada:** si el número máximo de improvisaciones (NI) se satisface, la iteración termina. De otro modo, los pasos 2 y 3 en GBHSK se repiten.

5: **Seleccionar la mejor armonía en HM**: la mejor armonía, que tiene el mínimo valor de fitness, es encontrada y seleccionada. Luego, el algoritmo k-means (Figura 7 sin el paso 01, porque tiene información sobre los centroides iniciales) se ejecuta y después se calcula un nuevo valor de fitness con la ubicación final de los centroides.

6: **Retornar el mejor resultado en la memoria armónica:** Retornar la mejor armonía (centroides y fitness) para IGBHSK.

```
01  Para i=1 hasta D (número de dimensiones) hacer
02    Si U (0, 1) ≤ HMCR entonces
03    Inicio  /*consideración de la memoria armónica*/
04      j ~ U (1… HMS); c ~ U (1… HM[j].k)
05      NuevoCentroide [i] = HM [j].Centroide[c][i]
06      Si U(0,1) ≤ PAR entonces
07      Inicio /*ajuste del tono PSO*/
08        c ~ U (1… HM[mejor].k)
09        NuevoCentroide [i] = HM[mejor].Centroide[c][i]
10      Fin-si
11    Sino /*Selección aleatoria con estrategia de Forgy*/
12      Rand ~ U (1… N)
13      NuevoCentroide [i] = TDM[Rand][i] o FTDM[Rand][i]
14    Fin-si
15  Continuar_Para
```

Donde mejor, es el índice de la mejor armonía en HM (vector solución con los valores más bajos de fitness); y c es un número aleatorio entre 1 y el número de centroides en el vector solución actual.

Figura 11. Improvisación de una nueva armonía

El algoritmo completo se puede resumir como se muestra en la Figura 12.

9.4. TÉRMINOS ESTADÍSTICAMENTE MÁS REPRESENTATIVOS PARA EL ETIQUETADO

Un grupo está representado por un concepto probabilístico, el cual es un conjunto de términos estadísticamente más representativos (términos y sus frecuencias asociadas en el grupo) [67]. Este método trabaja como sigue:

1: **Inicializar los parámetros del algoritmo:** Se define el *umbral máximo de términos* y el *umbral de frecuencia mínima de términos*. El umbral máximo de términos representa el número máximo de términos que puede comprender la etiqueta de cada grupo. El umbral de frecuencia mínima de términos representa el porcentaje de la suma total de los términos que se consideran como la etiqueta de cada grupo.

2: **Construcción de la etiqueta y el grupo "Otros":** Al final, si algunos documentos no corresponden con la etiqueta de algún grupo, entonces se asignan al grupo "Otros". Esto solo ocurre cuando la fuente es una matriz FTDM.

3: **Inducción de las etiquetas candidatas:** Para cada grupo, una nueva matriz de términos por documentos se crea solo con las frecuencias de los documentos del grupo. Después, se calcula la frecuencia total para los términos y luego los términos que alcanzan el *umbral de frecuencia mínima de términos* son seleccionados. Posteriormente, los términos con las frecuencias más altas son seleccionados, siempre y cuando no excedan el *umbral máximo de términos*.

4: **Eliminar los términos repetidos**: Para definir las etiquetas finales, los términos repetidos se eliminan de las etiquetas candidatas.

5: **Mejora visual de las etiquetas:** cada término en la etiqueta es reemplazado con la representación más larga de este, pero si el término es un verbo, este término es reemplazado por el infinitivo del verbo.

9.5. FRASES FRECUENTES PARA EL ETIQUETADO

Este paso corresponde con el paso 2 llamado "Extracción de frases frecuentes" en Lingo [33], pero en IGBHSK este método es usado para cada grupo generado en los pasos previos. Para el método citado anteriormente, se hicieron algunos cambios en el algoritmo original, de modo que trabaja de la siguiente manera:

1: **Conversión de la representación**: Cada documento en el grupo actual es convertido de una representación basada en caracteres a una representación basada en palabras.

2: **Concatenación de documentos**: Todos los documentos en el grupo actual son concatenados y se crea un nuevo documento con la versión invertida del documento concatenado.

3: **Completar el descubrimiento de frases:** Las frases completas a la derecha y las frases completas a la izquierda son descubiertas en el documento actual, luego las frases

completas a la derecha y las frases completas a la izquierda son ordenadas alfabéticamente, y después las frases completas a la derecha y a la izquierda se combinan en un conjunto de frases completas.

4: **Selección final:** Los términos y las frases cuya frecuencia excede el *umbral de frecuencia de términos* son seleccionados para el grupo actual.

5: **Construcción de la etiqueta y el grupo "Otros":** Si algunos documentos no alcanzan el *umbral de frecuencia de términos,* entonces se envían al grupo "Otros".

6: **Inducción de las etiquetas de grupo**: En el grupo actual, se construye una matriz de términos por documentos. Luego, usando la similitud de coseno, se seleccionan los mejores términos o frases candidatas para el grupo (que optimiza SSE).

9.6. SOLAPAMIENTO Y ORDENAMIENTO DE GRUPOS Y DOCUMENTOS

El solapamiento de grupos se refiere a que puede haber documentos que pertenecen a más de un grupo, esto es importante desde el punto de vista del usuario en Recuperación de Información.

En cuanto al ordenamiento de los grupos, estos se encuentran ordenados desde el más compacto (contiene documentos estrechamente relacionados) al menos compacto, esto es importante para el usuario puesto que le permite determinar con más rapidez si el grupo es relevante o no para su consulta.

En cuanto al ordenamiento de los documentos dentro de cada grupo, estos se encuentran ordenados teniendo en cuenta el documento que es más representativo del grupo (el centroide), por lo tanto primero aparecen los documentos más relacionados con el documento representativo y luego los más alejados a este documento. Después de estos documentos aparecen los documentos adicionados por solapamiento, estos documentos se ordenan de la misma forma que se ordenan los documentos anteriores (es decir, los originales o no solapados).

9.7. COMPLEJIDAD DEL ALGORITMO

IGBHSK repite la rutina GBHSK BMRS veces y luego hace el ordenamiento de un vector con BMRS filas. La mayor carga computacional ocurre en cada paso de la rutina GBHSK. La rutina GBHSK genera HMS vectores solución y luego NI improvisaciones. Para cada vector generado en la rutina GBHSK, se ejecuta un paso del algoritmo K-means y se calcula el valor de fitness (índice BIC o DB). Finalmente, la rutina GBHSK encuentra y selecciona la mejor solución, ejecuta el algoritmo K-means para esta solución y re-calcula el valor de fitness. Un paso del algoritmo K-means y el cálculo de la función de fitness de una solución dada toma $O(n*k*D)$ y $O(n*D)$ veces, respectivamente. El algoritmo total K-means y re-calcular el valor de fitness toma $O(n*k*D*L)$ veces (donde L es el número de iteraciones tomadas por el algoritmo K-means para converger). Por otra parte el etiquetado de los grupos toma $O(n*D)$ veces. Realizar el solapamiento de los grupos toma $O(k*n*D)$ veces. Por lo tanto, la complejidad global del algoritmo propuesto es $O((n*k*D*(L+HMS+NI)*BMRS))$.

```
IGBHSK
---------------------------------------------------------------------------------------
ENTRADA:   TDM, BMRS, HMS, HMCR, PAR, BW, NI
SALIDA:   K-centroides, Fitness
---------------------------------------------------------------------------------------
PASO 1:   Inicializar parámetros del algoritmo: Verificar parámetros.
PASO 2:   Inicializar la memoria de los mejores resultados y llamar  a  la rutina GBHSK.
          Eliminar las palabras vacías
          Algoritmo de lematización de Porter
          Construir la Matriz de Términos por Documentos o la Matriz de Términos Frecuentes por Documento
          Eliminar las dimensiones con rango igual a cero
          Para  cada i Є [1, BMRS] hacer
             BMR[i] = GBHSK (TDM o FTDM)
          continuar-Para
PASO 3:   Seleccionar los mejores resultados de BMR
PASO 4:   Asignar etiquetas a los grupos.
PASO 5:   Solapar los grupos.
```

```
GBHSK
---------------------------------------------------------------------------------------
ENTRADA:   TDM
SALIDA:   K-centroides, Fitness
---------------------------------------------------------------------------------------
PASO 1:   Inicializar la Memoria Armónica (HM): definir los centroides
          (estrategia de forgy), ejecutar 1-means y Calcular fitness
          (BIC) para cada vector solución generado en HM.
PASO 2:   Improvisar un nuevo vector armónico: definir centroides para esta solución.
          Para i=1 hasta D (número de dimensiones) hacer
             Si U (0, 1) ≤ HMCR entonces
             Inicio  /*consideración de la memoria   armónica*/
                j ~ U (1… HMS); c ~ U (1… HM[j].k)
                NuevoCentroide [i] = HM [j].Centroide[c][i]
                If U(0,1) ≤ PAR entonces
                Inicio /*ajuste del tono*/
                   c ~ U (1… HM[mejor].k)
                   NuevoCentroide [i] = HM[best] .Centroid[c][i]
                Fin-if
             Sino /*selección aleatoria con estrategia de forgy*/
                Rand ~ U (1… N)
                NuevoCentroide [i] = TDM[Rand][i] o FTDM[Rand][i]
             End-if
          Continuar_Para
          Ejecutar 1-means y Calcular fitness (BIC o DB) para nueva armonía
PASO 3:   Actualizar la memoria armónica: el Nuevo vector armónico  reemplaza el peor vector  armónico en la HM.
PASO 4:   Verificar el criterio de parada: si el número de improvisaciones (NI) se satisface, la iteración termina.
          De otro modo, los pasos 2 y 3 se repiten.
PASO 5:   Seleccionar la mejor armonía en HM: encontrar la mejor armonía,
          ejecutar K-means y Calcular fitness (BIC o DB) para la mejor armonía.
PASO 6:   Retornar la mejor armonía en la memoria armónica.
```

```
1-means (un paso del algoritmo K-means)
---------------------------------------------------------------------------------------
ENTRADA:   K-centroides
SALIDA:   K-centroides
---------------------------------------------------------------------------------------
PASO 1:   Re-calcular la membrecía con los centroides actuales
           (usando Similitud de Coseno)
PASO 2:   Actualizar los centros: para todos los centros en la solución actual, actualizar
          los centros de acuerdo a la nueva membrecía de los objetos.
PASO 3:   Retornar los centroides.
```

Figura 12. Pasos en el algoritmo Iterativo Búsqueda Armónica Global y K-means (IGBHSK)

10. METODOLOGÍA DE DESARROLLO DEL META-BUSCADOR

10.1. DESCRIPCION GENERAL DE LA METODOLOGIA

Para el desarrollo del proyecto se utilizó una instanciación del Proceso Unificado, específicamente las siguientes fases:

10.1.1. Planeación y elaboración.

En esta fase se definieron los requerimientos necesarios para el desarrollo del sistema, así como el estudio de las diversas alternativas para conseguir su desarrollo. En esta fase se obtuvieron los siguientes artefactos de manera preliminar. Casos de uso de alto nivel, Diagrama de Clases, Diagrama de la Base de Datos, Diseño Preliminar del Algoritmo y la Arquitectura.

10.1.2. Construcción.

Con los resultados obtenidos en la fase de Planeación y Elaboración se dio inicio a esta fase, con ella se logró obtener una versión operativa del sistema en las siguientes etapas.

- **Análisis**: En esta etapa se obtuvo una concepción clara de los requisitos a desarrollar en cada ciclo, se desarrolló la descripción de los casos de uso de alto nivel y el modelo conceptual específico para cada ciclo.

- **Diseño**: Teniendo en cuenta el análisis del sistema se procedió a diseñar un algoritmo preliminar para el clustering de documentos y luego se generó una solución lógica del prototipo software que finalmente fue implementado, para lo cual se diseñaron los casos de uso reales y los diagramas de clases de diseño.

- **Implementación**: Se implementaron los componentes lógicos obtenidos en la etapa de diseño.

- **Pruebas**: Se realizaron las validaciones pertinentes para verificar el resultado de la implementación generada en esta fase.

10.1.2.1. Ciclos de desarrollo

Los ciclos de desarrollo permitieron dividir la funcionalidad completa del sistema en tareas más pequeñas que facilitaron la labor de construcción del sistema cumpliendo con cada una de las etapas mencionadas anteriormente.

- Ciclo 1 – Algoritmo de Clustering Inicial: En este ciclo se modeló e implementó el algoritmo inicial utilizando Microsoft Visual Studio 2005 y C# como lenguaje de programación.

- Ciclo 2 – Algoritmo de Clustering final con manejo de stopwords (palabras no relevantes en las búsquedas): En este ciclo se modeló e implementó el algoritmo final

teniendo en cuenta las stopword en la etapa de pre-procesamiento, lo cual ayuda a disminuir la dimensionalidad de los documentos procesados.

- Ciclo 3 – Mecanismo de Autenticación de usuarios: Se agregó la funcionalidad de autenticación utilizando los servicios de Windows Live ID.

- Ciclo 4 – Funcionalidad oculta relacionada con stemming (raíces gramaticales de las palabras): Se agregó la funcionalidad de stemming, para ello se tuvo en cuenta el stemming que se realiza en Lucene (Api que ofrece esta funcionalidad).

- Ciclo 5 – Funcionalidad oculta relacionada con la evaluación (relevancia). En este ciclo se diseñó e implementó el prototipo Windows para calcular las medidas de Precisión, Exhaustividad y Medida F.

- Ciclo 6 – Funcionalidad relacionada con los servicios de autocompletar y traducción: En este ciclo se utilizaron los servicios ofrecidos por Google en cuanto a autocompletar y traducir, los cuales se emplearon en la búsqueda y traducción de documentos (y etiquetas) respectivamente.

- Ciclo 7 – Encuesta de satisfacción por parte del usuario: En este ciclo se diseñó e implementó el prototipo final (aplicación web) para realizar las pruebas de funcionalidad y usabilidad de la aplicación para medir la satisfacción del usuario.

10.1.3. Transición.

Luego de culminar la fase de construcción del sistema, se realizaron las respectivas comparaciones entre los resultados que retornan los meta-buscadores que realizan una funcionalidad similar y los resultados presentados por el meta-buscador propuesto, con el fin de realizar su adecuación final e implantación. Posteriormente se llevaron a cabo pruebas con estudiantes con el objetivo de comparar el desempeño del meta-buscador, las cuales fueron descritas en la sección 9.5.

10.1.4. Documentación y divulgación de resultados.

A lo largo de cada fase se desarrolló la documentación respectiva. Finalmente la divulgación se termina con la realización de la monografía y la sustentación de la tesis ante los jurados definidos por la Facultad de Ingeniería Electrónica y Telecomunicaciones, FIET. La documentación presentó los resultados obtenidos del uso del meta-buscador y las recomendaciones sugeridas. Por otra parte se realizó la presentación del meta-buscador a cuatro grupos del programa de Ingeniería de Sistemas de la Universidad del Cauca, quienes participaron de la evaluación del mismo.

10.2. ARQUITECTURA DEL SISTEMA

Para el sistema se definió una arquitectura multicapa que consta de tres capas, lógica de presentación, lógica de negocio y lógica de servicios. De entre las ventajas, ampliamente conocidas, de utilizar este tipo de arquitectura se encontró principalmente: la flexibilidad, permite alta escalabilidad, y facilita la construcción y el mantenimiento del sistema. En la Figura 13 se muestra la arquitectura y sus componentes.

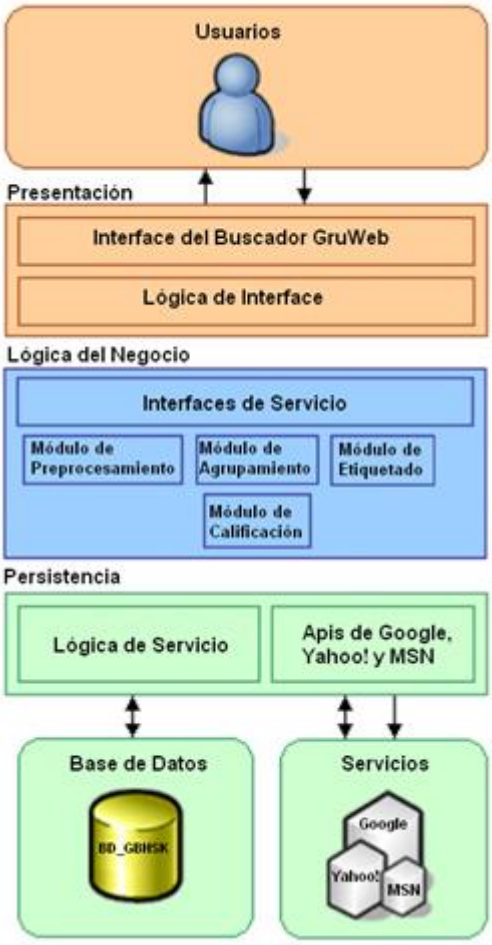

Figura 13. Arquitectura del Sistema

A continuación se hace una breve descripción de las funciones que se realizan en cada uno de las capas de la arquitectura.

- **Capa de Presentación**: incluye los componentes de interfaz de usuario, páginas web, las cuales permiten procesar y dar formato a los datos de usuario, así como adquirir y validar los datos entrantes procedentes de éstos. Esta capa se comunica únicamente con la capa de negocio por medio de las Interfaces del Servicio.

- **Capa de Lógica de Negocio**: En esta capa es donde se implementan, por medio de servicios, las reglas de negocio y procesos relacionados con las funcionalidades que ofrece el sistema. A su vez éste se divide en las interfaces del servicio y en 4 módulos:

  (1) **Interfaces del Servicio:** dan soporte a la implementación de servicios para que los clientes puedan acceder a los métodos suministrados por los módulos en la capa de negocio.

  (2) **Módulo de pre-procesamiento**: encargado de realizar los pasos en la fase de pre-procesamiento de documentos. En este módulo se gestiona la carga de los data sets, la identificación del lenguaje, la eliminación de stopwords y el stemming.

Este módulo se comunica con el módulo de agrupamiento a quien le envía los documentos pre-procesados y algunos parámetros que utiliza el algoritmo de agrupamiento.

(3) **Módulo de Agrupamiento**: encargado se inicializar los parámetros propios del algoritmo y realizar el agrupamiento de los documentos utilizando el algoritmo propuesto IGBHSK. Este módulo se comunica con el módulo de etiquetado a quien le envía los grupos de documentos formados para realizar el etiquetado.

(4) **Módulo de Etiquetado**: encargado de realizar la asignación de nombres a cada uno de los grupos de documentos generados. Este módulo termina los resultados del agrupamiento ya que complementa los grupos asignándoles nombre a cada uno.

(5) **Módulo de Calificación**: encargado de gestionar la calificación de los resultados del agrupamiento. Este módulo actúa después de que se han ejecutado los módulos de pre-procesamiento, agrupamiento y etiquetado, es decir luego de haberse producido la búsqueda y el agrupamiento de sus resultados.

- **Capa de Persistencia**: En esta capa se encuentra la lógica que permite dar persistencia a los objetos de la lógica del negocio, recuperación de información desde la base de datos, comunicarse con las APIs de Google, Yahoo! y MSN Live y recuperar instancias de la base de datos donde se encuentra almacenada. El principal objetivo de esta capa es ocultar a las capas superiores los detalles técnicos y de lenguaje necesarios para acceder a los repositorios donde están almacenados los datos. Esta capa está constituida por las siguientes subcapas:

  (1) **Lógica de servicios**: implementa la persistencia y el acceso a los datos ocultando los detalles de los repositorios de datos a las capas superiores. El acceso a datos implementa componentes software que se encargan de acceder a la base de datos para leer y escribir el estado de los objetos de negocio, independizando la aplicación del acceso al motor de la base de datos. La subcapa comunica la capa de negocio con Microsoft SQL Server 2005 que fue el motor escogido para el almacenamiento de la información.

  (2) **APIs de Google, Yahoo! y MSN Live**: Este subnivel por medio de las APIs de Google, Yahoo! y MSN Live permite acceder a los servicios ofrecidos por estos buscadores en internet. Entre los servicios accedidos se encuentran: servicios de búsqueda en los tres buscadores y los servicios de autocompletar y traducción proporcionados por Google.

10.3. ANÁLISIS Y DISEÑO

A continuación se muestran algunos resultados generales del sistema, resultado del análisis, diseño, implementación y pruebas en los diferentes ciclos, llevados a cabo para el desarrollo del meta-buscador.

10.3.1. Casos de uso de alto nivel.

En la Figura 14 se muestran las operaciones que el usuario del sistema (el actor Cliente) pueden realizar, a saber: Iniciar sesión, realizar búsqueda, escoger parámetros y calificar grupos.

10.3.2. Casos de uso reales.

A continuación se muestra el caso de uso real más importante del sistema (ver Tabla 3), los demás casos de uso se encuentran en el Anexo G.

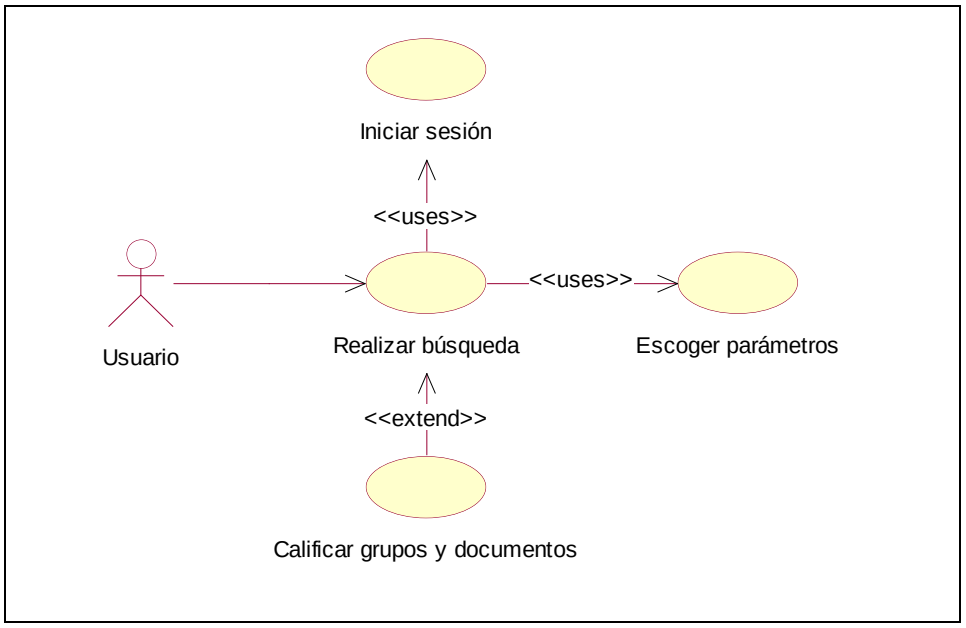

Figura 14. Casos de uso para los usuarios

CASO DE USO REAL: REALIZAR BÚSQUEDA
Actores: Cliente.
Propósito: Realizar el agrupamiento por temas según la consulta realizada.
Resumen: El cliente ingresa la consulta deseada, seleccionado el grupo deseado y revisa los documentos dentro de este.
Tipo: Primario.

CURSO NORMAL DE LOS EVENTOS	
Acción del actor	**Respuesta del sistema**
1. El usuario (Cliente), digita la consulta deseada en el cuadro de texto [A].	
2. El usuario da click en el botón Buscar [B].	3. El sistema obtiene los documentos de la Web por medio de Google, Yahoo! y MSN Live.
	4. El sistema realiza el pre-procesamiento de los documentos.
	5. El sistema crea los grupos.
	6. El sistema muestra los grupos creados.
7. El usuario selecciona el grupo deseado [C].	
8. El usuario selecciona y visualiza el documento deseado [D].	
CURSO ALTERNO 1	
Acción del actor	**Respuesta del sistema**
2. El usuario da click en el botón Buscar [B] sin digitar la consulta en el cuadro de texto [A].	3. El sistema muestra un mensaje informándole que debe digitar el texto de la consulta.
CURSO ALTERNO 2	
Acción del actor	**Respuesta del sistema**
1. El usuario, digita la consulta en el cuadro de texto [A] en un lenguaje diferente al inglés.	
2. El usuario da click en el botón Buscar [B].	3. El sistema muestra un mensaje informándole que debe digitar el texto de la consulta en inglés.

Tabla 3. Caso de Uso Real Realizar Búsqueda

10.3.3. Diagrama de clases.

En la Figura 15 se muestra de manera general el diagrama de clases del sistema, en el Anexo H se ilustran con más detalle cada una de las Clases. Más adelante, en la Tabla 4, se describe la funcionalidad de cada Clase.

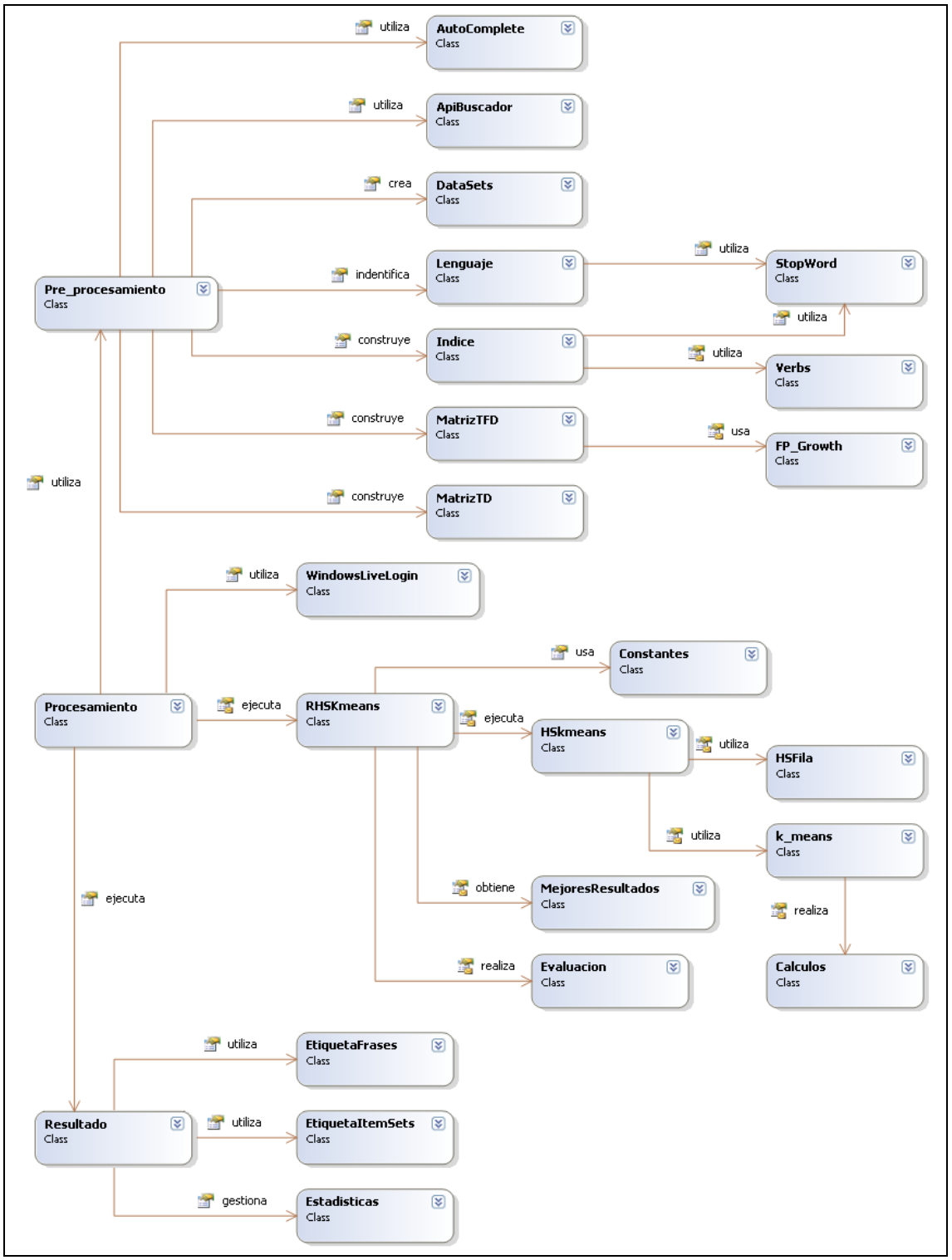

Figura 15. Diagrama General de Clases del Sistema

CLASE	FUNCION
AutoComplete	Funcionalidad provista por Google que autocompleta la consulta en la caja de texto donde se ingresa la consulta, es decir a medida que el usuario escribe la consulta, le aparecen sugerencias de temas asociados.
ApiBuscador	Obtiene y estandariza los documentos provenientes de los buscadores Google, Yahoo! y MSN Live. También provee la funcionalidad provista por Google de traducción, para traducir las etiquetas, los títulos y el resumen del documento.
DataSets	Provee la funcionalidad para crear el Data Set con los documentos.
StopWord	Provee la lista de palabras vacías (de significado) de los diferentes idiomas.
Verbs	Provee la lista de verbos del idioma Inglés.
Lenguaje	Provee la funcionalidad correspondiente a la identificación del lenguaje de los documentos.
Indice	Provee la funcionalidad correspondiente a la indexación de los documentos tales como: análisis léxico, eliminación de palabras vacías y stemming. Igualmente provee la funcionalidad correspondiente a la obtención de las raíces de los documentos para el proceso de etiquetado.
FP-Growth	Obtiene la lista de Itemsets Frecuentes de los documentos.
MatrizTFD	Provee la funcionalidad para crear la Matriz de Términos Frecuentes por Documentos.
MatrizTD	Provee la funcionalidad para crear la Matriz de Términos por Documentos.
Pre-procesamiento	Provee toda la funcionalidad correspondiente al pre-procesamineto de los documentos.
WindowsLiveLogin	Funcionalidad provista por Windows Live ID (servicio Passport), para la autenticación de los usuarios en la aplicación.
Constantes	Mantiene los parámetros para el algoritmo.
Calculos	Provee la funcionalidad para realizar el cálculo de funciones como el Índice BIC, Índice DB, Similitud de Cosenos, entre otras.
k-means	Provee la funcionalidad del algoritmo k-means.
HSFila	Provee la funcionalidad correspondiente a la Creación de Centroides Aleatorios y a la Creación de un Improviso.
HSkmeans	Provee la funcionalidad correspondiente a la creación de una solución del algoritmo IGBHSK.
MejoresResultados	Se encarga de mantener las mejores soluciones del algoritmo IGBHSK.
Evaluacion	Provee la funcionalidad para el cálculo de las medidas de relevancia (Precisión, Exhaustividad y Medida F). Esta clase se utiliza solamente para la evaluación con los DataSets de Reuters y Dmoz.
RHSKmeans	Provee la funcionalidad principal para la ejecución del algoritmo IGBHSK.
EtiquetaFrases	Obtiene las etiquetas de los grupos, para el modelo de Frases Frecuentes.
EtiquetaItemSets	Obtiene las etiquetas de los grupos, para el modelo de Términos Estadísticamente Representativos.
Estadisticas	Provee la funcionalidad para la calificación tanto de los grupos como de

	los documentos.
Resultado	Provee la funcionalidad correspondiente a la asignación de etiquetas, calificación de grupos y documentos, ordenamiento de grupos y documentos.
Procesamiento	Es la clase principal, que procesa la consulta del usuario para obtener los documentos, realizar el pre-procesamiento de los mismos, ejecutar el algoritmo, crear los resultados y procesar la calificación de grupos y documentos.

Tabla 4. Descripción de las Clases

10.3.4. Modelo de la base de datos.

A continuación en la Figura 16 se muestra el modelo de la base de datos BDReuter y en la Tabla 5 se explica la finalidad de cada tabla de esta base de datos. Luego en la Figura 17 se muestra el modelo de la base de datos BDDmoz y en la Tabla 6 se explica la finalidad de cada tabla de esta base de datos. Finalmente, en la Figura 18 se muestra el modelo de la base de datos BG_GBHSK (maneja las calificaciones) y en la Tabla 7 se explica la finalidad de cada tabla de esta base de datos.

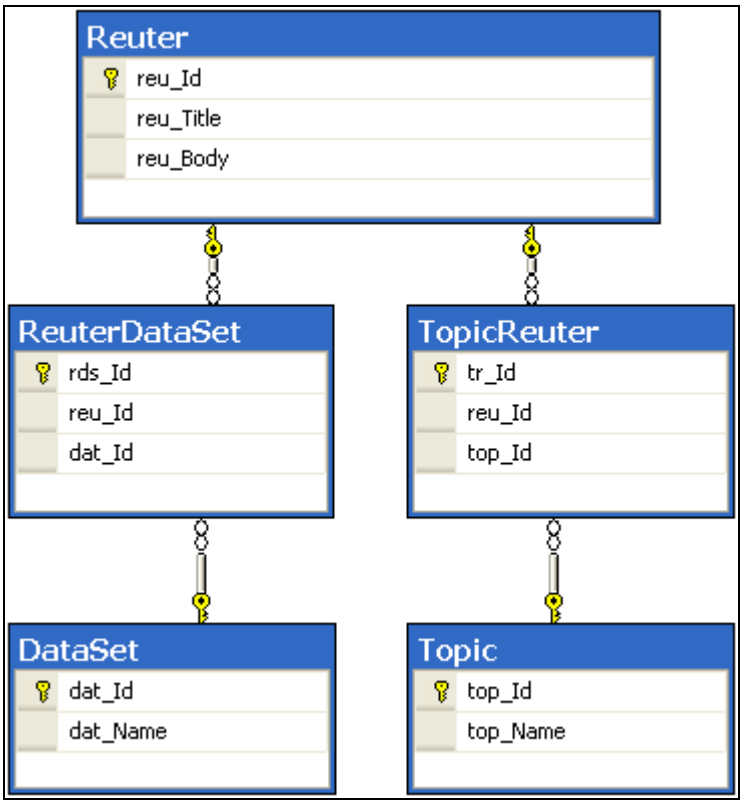

Figura 16. Modelo de la Base de Datos de BDReuter

TABLA	FUNCION
Topic	Mantiene la información correspondiente a los temas a los que pertenecen los documentos de la colección Reuters (colección de noticias).
TopicReuter	Mantiene la información correspondiente al tema al que pertenece cada uno de los documentos.
DataSet	Mantiene la información de los cuatro (4) Data Sets creados para realizar las evaluaciones.
ReuterDataSet	Mantiene la información correspondiente a los Data Sets a los que pertenece cada uno de los documentos.
Reuter	Mantiene la información de cada uno de los documentos de la colección Reuters.

Tabla 5. Descripción de las tablas de la base de datos BDReuter

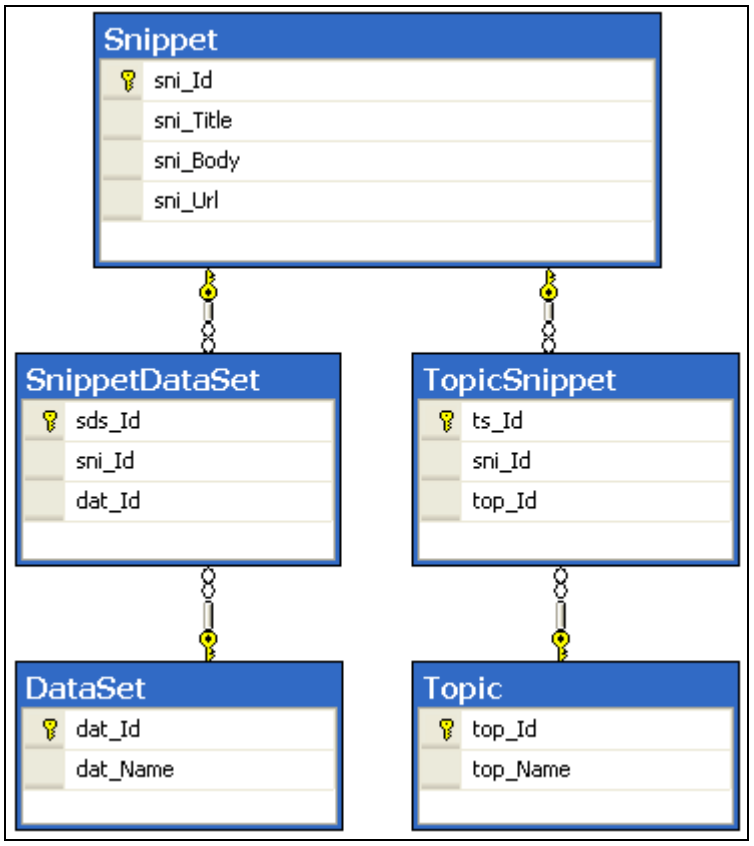

Figura 17. Modelo de la Base de Datos de BDDmoz

TABLA	FUNCION
Topic	Mantiene la información correspondiente a los temas a los que pertenecen los documentos de la colección Dmoz (colección de descripciones de sitios web).
TopicSnippet	Mantiene la información correspondiente al tema al que pertenece cada

	uno de los documentos.
DataSet	Mantiene la información de los cuatro (4) Data Sets creados para realizar las evaluaciones.
SnippetDataSet	Mantiene la información correspondiente a los Data Sets a los que pertenece cada uno de los documentos.
Snippet	Mantiene la información de cada uno de los documentos de la colección Dmoz.

Tabla 6. Descripción de las tablas de la base de datos BDDmoz

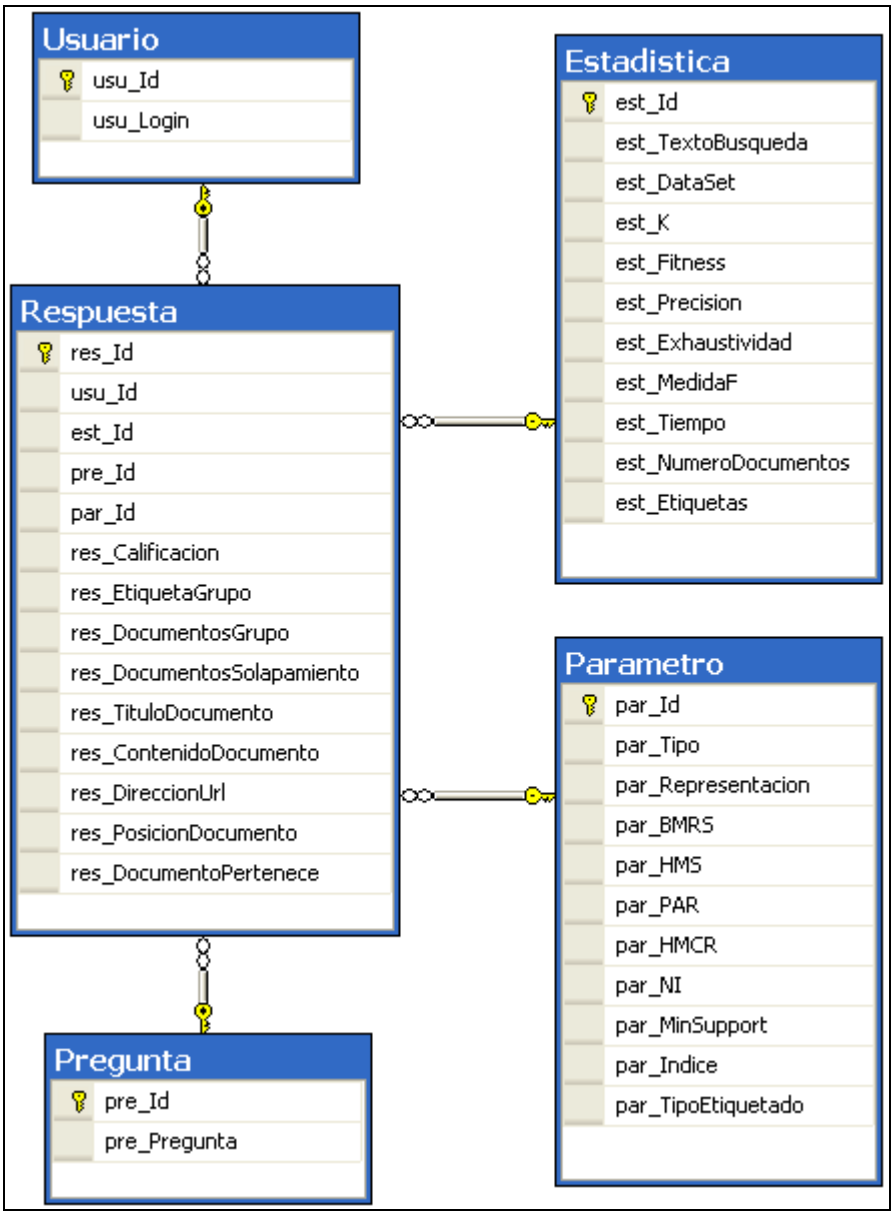

Figura 18. Modelo de la Base de Datos de BG_GBHSK

TABLA	FUNCION
Usuario	Mantiene la información de los usuarios del sistema.
Estadistica	Mantiene la información correspondiente a los resultados arrojados por el algoritmo. El registro de *est_TextoBusqueda* (texto de la consulta) se realiza solo para el caso de la calificación con los documentos obtenidos de la Web y el registro de las medidas de relevancia *est_Precision*, *est_Exhaustividad* y *est_MedidaF* para el caso de la calificación con los Data Sets de Reuters y Dmoz.
Parametro	Mantiene la información correspondiente a los parámetros escogidos para la ejecución del algoritmo IGBHSK.
Pregunta	Mantiene las preguntas correspondientes a la calificación de los grupos y documentos.
Respuesta	Mantiene el registro de la calificación de los grupos y los documentos. El registro de *res_TítuloDocumento*, *res_ContenidoDocumento*, *res_DireccionUrl*. *res_PosicionDocumento*, y *res_DocumentoPertenece* solo se realiza en el caso de calificar un documento.

Tabla 7. Descripción de las tablas de la base de datos BG_GBHSK

10.4. IMPLEMENTACION

Inicialmente se implementó un prototipo para realizar pruebas de laboratorio (aplicación Windows) y finalmente se implementó el meta-buscador (aplicación Web) para el trabajo directo con los usuarios. Las dos aplicaciones presentan la misma funcionalidad, excepto que en la aplicación web el servicio de autenticación se basa en Windows Live ID (Passport) y en la aplicación Windows se usa un servicio interno de autenticación de usuarios. Para ver los detalles de las aplicaciones remitirse al Anexo I. A continuación se presentan de manera general los componentes de la interfaz de la aplicación final (el meta-buscador).

10.4.1. Componente de autenticación.

En esta parte de la interfaz se realiza la funcionalidad de autenticación de usuarios, para lo cual se utilizan los servicios ofrecidos por Windows Live ID (que utiliza internamente el servicio de Passport), mediante ellos se inicia sesión con una cuenta de Windows Live ID, por ejemplo la del Hotmail, y el meta-buscador internamente gestiona los usuarios para conocer quiénes están accediendo al meta-buscador y de esta forma poder llevar un registro adecuado de los resultados de la agrupación de los documentos, permitiendo así realizar estadísticas de los resultados que se obtienen en las mismas.

10.4.2. Componente de búsqueda.

En esta parte de la interfaz se ingresa el tema a consultar, se ejecuta la búsqueda y el agrupamiento de los resultados y se muestra los grupos generados con sus respectivos documentos asociados.

10.4.3. Componente para opciones de búsqueda.

En parte de la interfaz se permite la modificación de los parámetros del algoritmo tanto las opciones generales como las opciones avanzadas, si no se modifican estos parámetros el algoritmo internamente utiliza los valores establecidos por defecto para todos los parámetros que utiliza.

10.4.4. Componente de calificación.

En esta parte de la interfaz se presentan las preguntas asociadas tanto a cada grupo generado como a cada documento. A medida que se selecciona un grupo o un documento se habilita la calificación del mismo, para ello se presentan las preguntas con sus respectivas opciones de respuesta.

11. EVALUACIÓN

11.1. DATA SETS

Para la realización de pruebas se utilizaron dos colecciones ampliamente reconocidas por la comunidad académica y científica de RI, estas colecciones son Reuters 21578 y DMOZ.

Para la colección de Reuter 21578, se instanciaron cuatro conjuntos de datos (data sets), los cuales se describen en la Tabla 8.

	DS1	DS2	DS3	DS4
Temas	money-supply 55	money-supply 80	money-supply 50	money-supply 80
	coffee 60	coffee 90	coffee 50	coffee 80
	sugar 45	sugar 70	sugar 50	sugar 80
	interest 40	interest 60	interest 50	interest 80
		ship 100		ship 80
Cantidad de documentos	40 - 60	60 - 100	50	80
Términos	2613	3991	2697	3977
Términos Frecuentes	981	189	419	203

Tabla 8. Descripción de la Colección Reuters 21578

DMOZ (Open Directory Project - ODP ó Directory Mozilla), es un índice temático o directorio de recursos web. De este directorio se instanciaron cuatro data sets, los cuales se describen en la Tabla 9.

11.2. PARÁMETROS Y MEDIDAS DE IGBHSK

La mayoría de valores de los parámetros del algoritmo fueron iguales para todos los data sets. BRMS es igual a 5, HMS es igual a 5, HMCR igual a 0.95, PAR igual a 0.35, y NI

igual a 30. El valor de Kmax fue igual a $\sqrt{N/2}+1$(tomado de [68]) donde N es el número de los documentos. Para el algoritmo FP-Growth se utilizó un valor de soporte mínimo de 10% en los data sets de Reuter y un valor de soporte de 5% para los data sets de DMOZ.

Estos valores se escogieron teniendo en cuenta varios aspectos: 1) a medida que aumenta el soporte mínimo, la cantidad de itemsets en los documentos que cumplen con el porcentaje de soporte será menor, 2) el tamaño de los documentos a los que se les aplicará el algoritmo FP-Growth es importante, debido a que si los documentos son grandes y el soporte es pequeño el tiempo de procesamiento aumenta, y 3) si los documentos son pequeños y el soporte es grande se obtienen pocos itemsets. Por lo anterior y dadas las características de los data sets de prueba, se escogió el soporte de 10% para la colección de Reuters (documentos grandes) y 5% para la colección de DMOZ (documentos pequeños).

	DS5	**DS6**	**DS7**	**DS8**
Temas	Coins 42	Diamonds 43	Coins 42	Algorithms 27
	Diamonds 43	Food 25	Diamonds 43	Number Theory 34
	Motorcycles 27	Greek 31	Travel and Tourism 10	Volleyball 11
		Virtual Reality 13	Weather 40	Adventure 19
				Dogs 42
Cantidad de documentos	27 - 43	13 - 43	10 - 43	11 - 42
Términos	350	457	425	589
Términos Frecuentes	63	39	63	24

Tabla 9. Descripción de la Colección DMOZ

Dos preguntas importantes se formularon: ¿el número de grupos está correctamente identificado? y, ¿los documentos están agrupados en un modo apropiado? Para responder estas preguntas, se calculó la Precisión (P), la Exhaustividad (R) y la Medida F (F), puesto que los grupos “verdaderos” de cada data set eran conocidos [27]. En este sentido, se busca obtener valores altos de P, R y F.

11.3. RESULTADOS

El algoritmo fue ejecutado 10 veces sobre la matriz de Términos por Documentos (TDM) con el criterio BIC y se calcularon valores promedios. Después, el mismo algoritmo fue ejecutado 10 veces usando el índice de DB sobre la matriz (TDM). Estos resultados se muestran en la Tabla 10.

	K			P		R		F	
	Ideal	**BIC**	**DB**	**BIC**	**DB**	**BIC**	**DB**	**BIC**	**DB**
DS 1	4	**9,3**	9,5	**84,0%**	79,4%	51,6%	**56,0%**	63,8%	**65,5%**
DS 2	5	13,1	**12,2**	**82,6%**	77,0%	48,3%	**56,9%**	60,5%	**65,3%**
DS 3	4	8,7	**8,5**	**83,4%**	75,5%	50,0%	**56,1%**	62,0%	**64,0%**
DS 4	5	13,3	**12,3**	**86,3%**	76,4%	48,5%	**54,3%**	61,8%	**63,1%**
DS 5	3	6,6	**4,1**	**97,4%**	90,8%	65,7%	**84,6%**	78,3%	**87,4%**
DS 6	4	7	**3,1**	**85,6%**	49,8%	**62,6%**	54,0%	**72,0%**	51,3%
DS 7	4	7,4	**2,6**	**91,3%**	49,9%	**71,2%**	59,3%	**79,7%**	53,8%
DS 8	5	7,6	**6,9**	**66,9%**	59,8%	**50,5%**	49,6%	**57,3%**	53,9%
Prom.	4,3	9,1	**7,4**	**84,7%**	69,8%	56,0%	**58,8%**	**66,9%**	63,1%

Tabla 10. Precisión (P), Exhaustividad (R) y Medida F (F) para dos criterios (BIC y DB) en la Matriz de Términos por Documentos (TDM)

Después, el algoritmo se corrió 10 veces sobre la Matriz de Términos Frecuentes por Documentos (FTDM) con el criterio BIC, de la misma forma se calcularon los promedios de las ejecuciones. Luego, el mismo algoritmo se ejecutó 10 veces usando el índice DB sobre la misma matriz (FTDM). Estos resultados promisorios se muestran en la Tabla 11.

	K			P		R		F	
	Ideal	**BIC**	**DB**	**BIC**	**DB**	**BIC**	**DB**	**BIC**	**DB**
DS 1	4	10	**2,3**	**86,1%**	37,6%	**48,9%**	46,5%	**62,3%**	41,3%
DS 2	5	14,2	**12,3**	**88,3%**	85,5%	42,9%	**48,6%**	57,7%	**61,8%**
DS 3	4	10,2	**8,1**	**85,7%**	78,3%	49,7%	**53,2%**	**62,7%**	62,7%
DS 4	5	14,2	**8,7**	**89,0%**	72,4%	45,0%	**51,3%**	**59,7%**	58,4%
DS 5	3	7,5	**4,5**	**90,8%**	68,6%	47,9%	**56,8%**	**62,6%**	61,2%
DS 6	4	7,6	**4,2**	**80,4%**	51,0%	**50,6%**	47,9%	**62,0%**	48,8%
DS 7	4	8,8	**4,8**	**87,9%**	55,6%	**54,7%**	49,2%	**67,4%**	50,9%
DS 8	5	8,8	**4,6**	**73,1%**	40,4%	**51,7%**	35,7%	**60,4%**	36,6%
Prom.	4,3	10,2	**6,2**	**85,2%**	61,2%	**48,9%**	48,6%	**61,9%**	52,7%

Tabla 11. Precisión (P), Exhaustividad (R) y Medida F (F) para dos criterios (BIC y DB) en la Matriz de Términos Frecuentes por Documentos (FTDM)

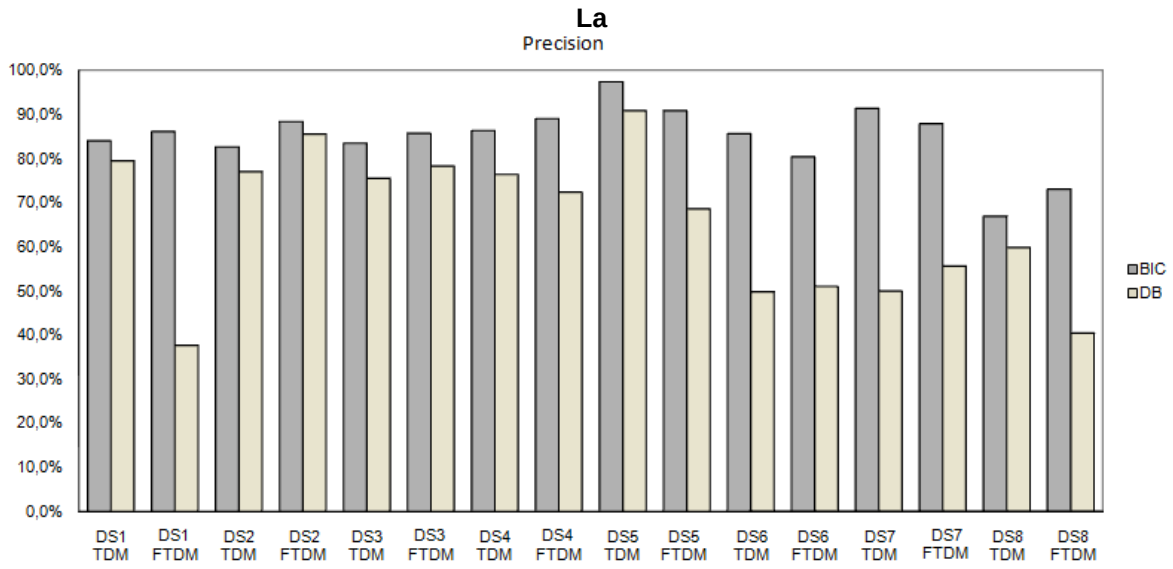

Figura 19 presenta los resultados de Precisión para el criterio BIC y DB en la Matriz TDM o la Matriz FTDM. En general, el algoritmo obtuvo mejores resultados de precisión usando el criterio BIC sobre la Matriz de Términos Frecuentes por Documentos, aunque el índice DB reportó valores más apropiados de k (número de grupos) en FTDM. Con relación a la Medida F, los resultados son mejores para la Matriz de Términos por Documentos (5% más), aunque en este camino de ejecución de IGBHSK, no hay reducción de la dimensionalidad.

11.4. COMPARACION

Carrot2 es un buscador web que realiza agrupamiento de documentos web para mostrar los resultados de las consultas. Carrot2 implementa el modelo de Lingo [15]. Este modelo está basado en la Descomposición de Valores Singulares y Frases Frecuentes. Para comparar IGBHSK con Carrot2, se utilizó la última versión de Carrot2, llamada Workbench.

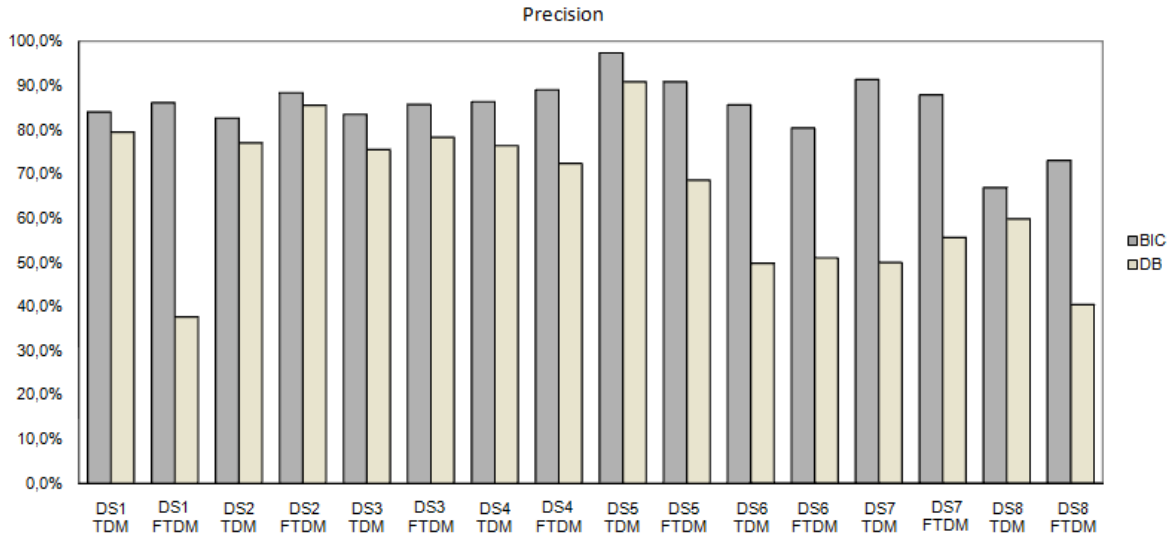

Figura 19. Precisión para los criterios BIC y DB

Esta versión de Carrot2 es una aplicación de escritorio y puede ser ejecutada con diferentes fuentes (no solo con los resultados de búsquedas web). Se utilizaron los data sets basados en DMOZ (data set 5 a 8 descritos en la sección 9.1) para comparar IGBHSK con Carrot2. En el algoritmo IGBHSK se utilizaron los resultados de una sola ejecución del algoritmo para compararlos con los resultados de Carrot2.

La Tabla 12 muestra los resultados de Precisión, Exhaustividad, Medida F, el Número de Etiquetas que fueron realmente Representativas de cada grupo (NRL) y el Número de Documentos en el grupo con etiqueta “Otros temas” (OTC). En general, IGBHSK tiene mejores resultados (precisión, exhaustividad y medida F) que Carrot2. IGBHSK definió un valor mejor de k (número de grupos) que Carrot2. IGBHSK tiene una proporción más alta de etiquetas representativas que Carrot2, finalmente, Carrot2 reportó una gran cantidad de documentos en el grupo “Otros temas” (malo para el usuario final).

La Figura 20 muestra 9 grupos generados por IGBHSK y sus respectivas etiquetas para el Data set 8. En estos grupos generados, 5 de ellos coinciden con los temas originales: Algorithms, Number Theory, Volleyball, Adventure, y Dogs. En la Figura 21 se muestran 11 grupos generados por Carrot2, en los cuales ninguna de las etiquetas de grupo coincide con los temas originales del dataset 8, además el grupo “Otros temas” tiene 88 documentos, lo cual no es apropiado para las necesidades de los usuarios, a menos que este grupo estuviera conformado solo por documentos irrelevantes, pero este no es el caso.

11.5. EVALUACIÓN DE USUARIO

La evaluación de la aplicación construida se realizo en dos etapas, en la primera etapa se realizaron pruebas en la aplicación Windows y en la segunda etapa se realizaron pruebas en la aplicación Web, como se presenta en las siguientes secciones (para ver los detalles de la planeación y la ejecución de las pruebas remitirse al Anexo E).

		DS 5	DS 6	DS 7	DS 8
K	Real	3	4	4	5
	Carrot2	9	9	11	11
	IGBHSK	**8**	**8**	**8**	**9**
P	Carrot2	78.2%	68.1%	59.7%	58.5%
	IGBHSK	**93.7%**	**84.6%**	**88.6%**	**79.6%**
R	Carrot2	33.5%	42.8%	26.0%	29.4%
	IGBHSK	**46.8%**	**54.2%**	**54.7%**	**59.7%**
F	Carrot2	46.9%	52.6%	36.2%	39.2%
	IGBHSK	**62.5%**	**66.1%**	**67.7%**	**68.2%**
NRL	Carrot2	6 (67%)	6 (67%)	6 (55%)	**9 (82%)**
	IGBHSK	**7 (88%)**	**6 (75%)**	**5 (63%)**	7 (78%)
OTC	Carrot2	72	74	84	88
	IGBHSK	**0**	**1**	**0**	**0**

Tabla 12. Precisión (P), Exhaustividad (R) y Medida F (F), Número de Etiquetas Representativas (NRL) y el Número de Documentos en el grupo "Otros temas" (OTC) para IGBHSK y Carrot2

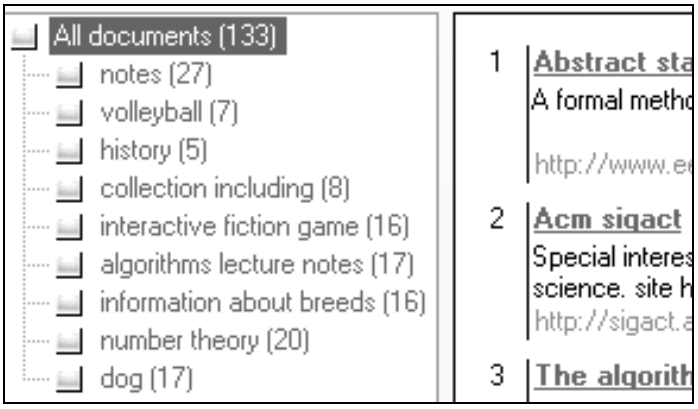

Figura 20. Resultados del Data set 8 en IGBHSK

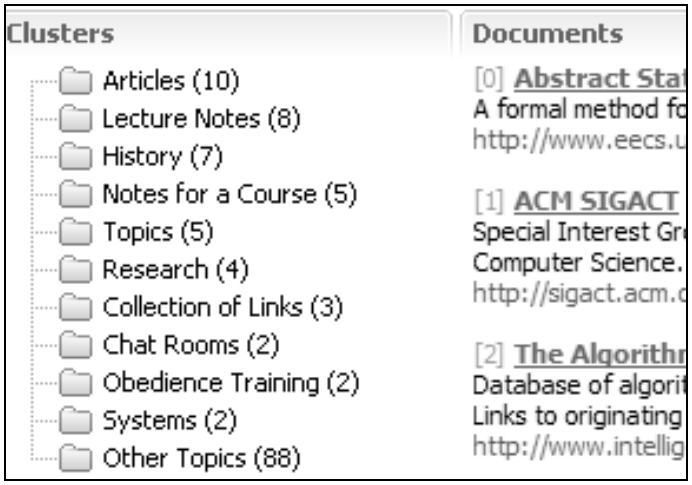

Figura 21. Resultados del Data set 8 en Carrot2

11.5.1. Usuarios participantes.

Para la realización de las pruebas a la aplicación se solicitó la colaboración a 62 estudiantes del programa de Ingeniería de Sistemas de la Universidad del Cauca. La descripción de los grupos de estudiantes que participaron se presenta en la Tabla 13. La duración de cada prueba fue en promedio 1 hora.

Grupo	N° Estudiantes	Asignatura	Profesor	Salón	N° de computadores	Fecha
1	6	Diferentes asignaturas	Ninguno	Oficina 105 - IPET	2	01/10/2009 02/10/2009
2	21	Gestión de Proyecto Informáticos (GPI)	Ing. Luz Marina Sierra	Sala 1 - Sistemas	21	03/11/2009
3	21	Estructuras de Lenguaje	Ing. Jimena Timaná	Sala 4 – Sistemas	21	05/11/2009
4	14	Conceptos Avanzados de Bases de Datos (CABD)	Ing. Martha Mendoza	Sala 4 – Sistemas	21	30/11/2009

Tabla 13. Grupos participantes en las pruebas del meta-buscador GruWeb

Inicialmente se realizaron las pruebas de la aplicación de escritorio, en ellas participaron los estudiantes del grupo 1. Finalmente se realizaron las pruebas a la aplicación web, en ellas participaron los estudiantes de los grupos 2, 3 y 4. En las siguientes secciones se detallarán las pruebas realizadas.

11.5.2. Evaluación de la aplicación Windows.

Un método preliminar de evaluación basado en el usuario fue utilizado para evaluar los resultados del agrupamiento producidos por IGBHSK cuando las fuentes de datos son Google, Yahoo! y MSN Live. Para realizar esta evaluación se utilizó la aplicación inicial en la que se implementó el algoritmo, la cual fue desarrollada como una aplicación Windows.

Como se mencionó anteriormente, en las pruebas de la aplicación Windows participaron los estudiantes del grupo 1 (descritos en la Tabla 13), quienes respondieron las siguientes preguntas.

Con respecto a cada grupo:

- P1: ¿El nombre del grupo representa los documentos que contiene? (mucho - R3, poco – R2, o nada – R1).
- P2: ¿El grupo es útil para su consulta? (útil - R3, moderadamente útil – R2, o no útil – R1).

Luego, para cada documento en cada grupo, los usuarios respondieron:

- P3: ¿El documento se ajusta a la descripción del grupo? (muy bien - R3, moderadamente – R2, o nada – R1).
- P4: ¿La ubicación del documento en el grupo, según su importancia es:? (correcta - R3, moderadamente correcta – R2, o incorrecta – R1).

Resultados obtenidos por el Grupo 1 de Usuarios

En la Tabla 14 se presentan las consultas y opciones que los usuarios del grupo 1 utilizaron en el proceso de evaluación. Para elegir los temas de las consultas se tuvo en cuenta las asignaturas que los estudiantes cursaron en el Programa de Ingeniería de Sistemas.

Los resultados generales de IGBHSK obtenidos por el grupo 1 se muestran en la Figura 22. La mayoría de las respuestas de los usuarios son R3 o R2. Por lo tanto, los resultados preliminares fueron muy prometedores y se dio paso a la realización de un conjunto de experimentos más controlados con otros usuarios, buscando generalizar los resultados.

Consulta	Representación de los documentos	Función de Fitness	Etiquetado
data structure	TDM	DB	SRT
database	TDM	DB	FPH
operating system	FTDM	BIC	SRT
software quality	FTDM	BIC	FPH

Tabla 14. Consultas y Opciones en el Proceso de Evaluación Inicial realizada por el grupo 1

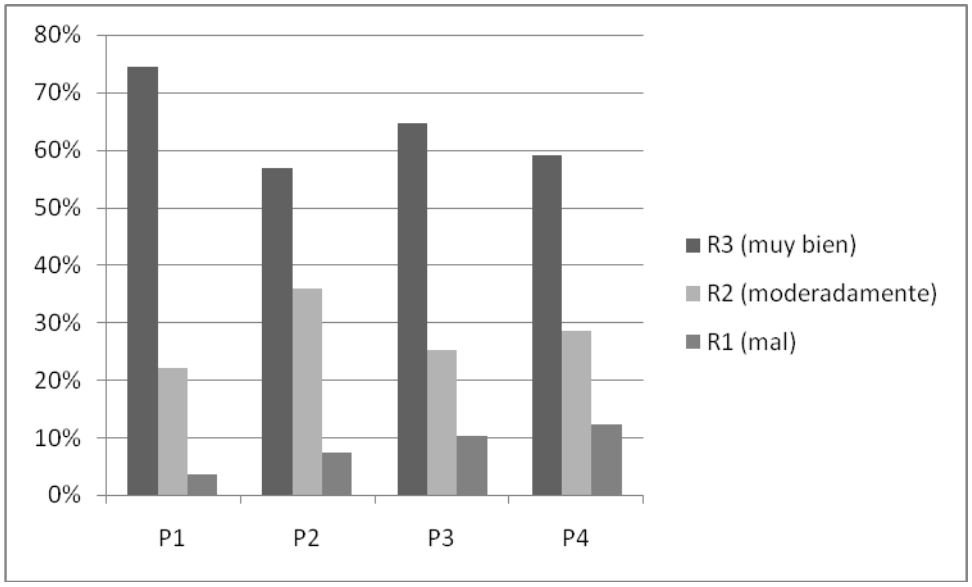

Figura 22. Resultados generales obtenidos por el grupo 1 para las cuatro preguntas

Para la pregunta 1 (¿El nombre del grupo representa los documentos que contiene?), los resultados generales son buenos (R3) y los mejores resultados se obtuvieron cuando el

algoritmo usa la matriz de términos por documentos y términos estadísticamente más representativos para definir la etiqueta de grupo. En este caso la función de fitness no fue muy importante (Ver la Figura 23).

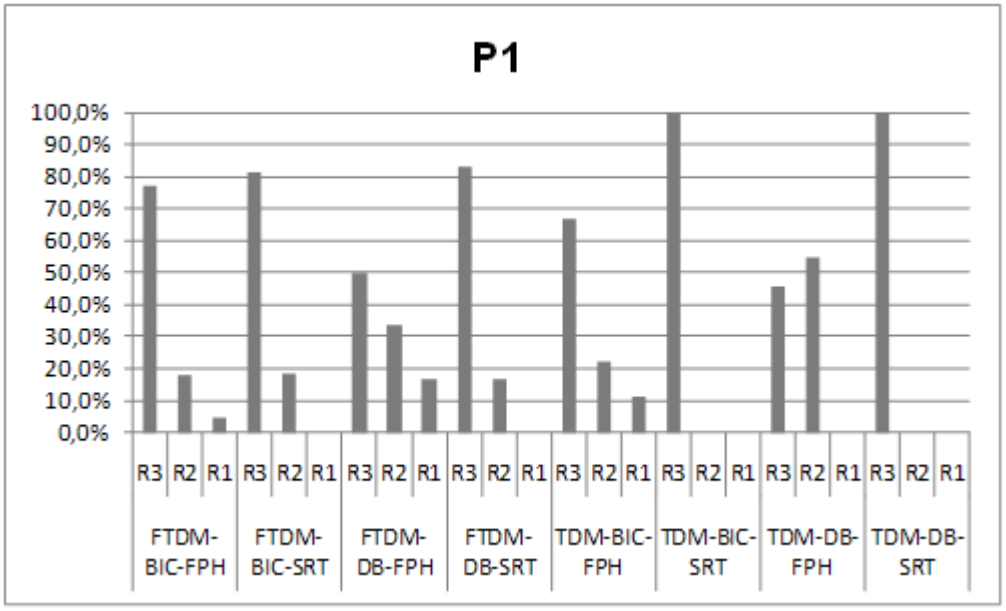

Figura 23. Resultados específicos para P1 en diferentes escenarios en la Evaluación Inicial realizada por el grupo 1

Para la pregunta 2 (¿El grupo es útil para su consulta?), los resultados generales son buenos (entre R3 y R2) y de manera similar a P1, los mejores resultados se obtuvieron cuando el algoritmo utiliza la matriz de términos por documentos y términos estadísticamente más representativos para definir la etiqueta de los grupos. En este caso la función de fitness no fue determinante (Ver Figura 24).

Para la pregunta 3 (¿El documento se ajusta a la descripción del grupo?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos frecuentes por documentos y el Criterio de Información Bayesiano. En este caso el método de etiquetado no fue determinante (Ver Figura 25).

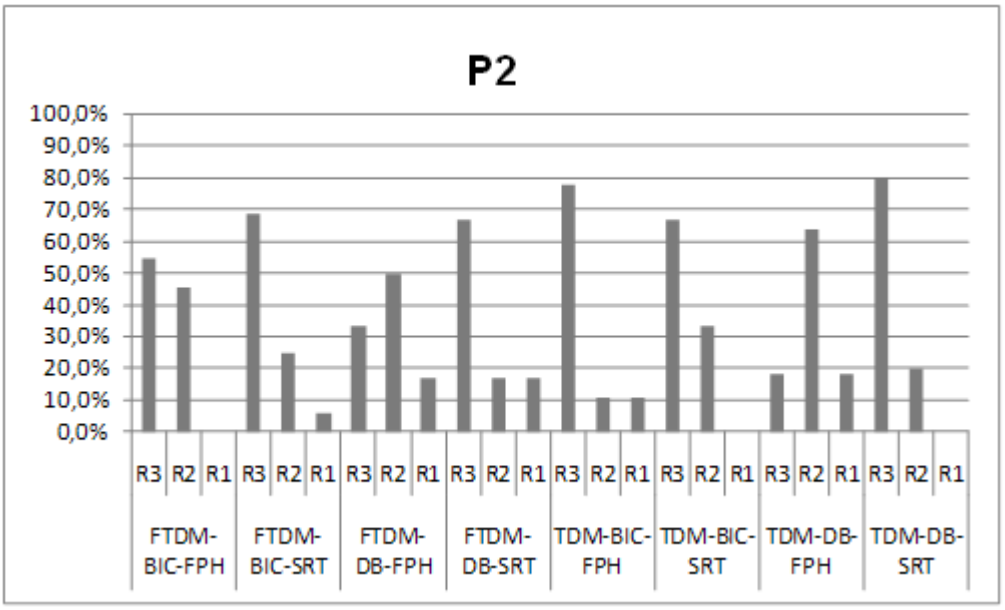

Figura 24. Resultados específicos para P2 en diferentes escenarios en la Evaluación Inicial realizada por el grupo 1

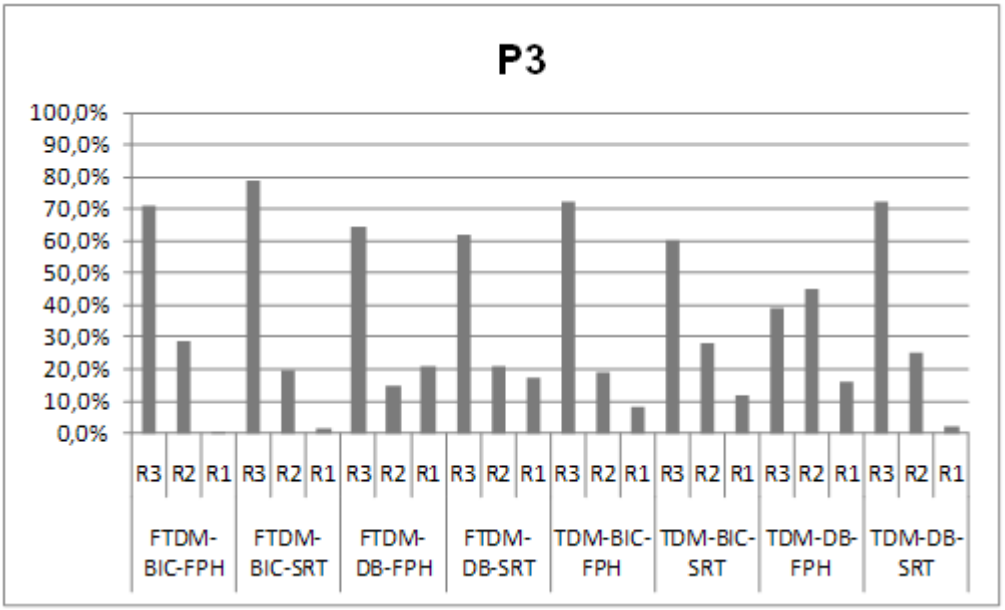

Figura 25. Resultados específicos para P3 en diferentes escenarios en la Evaluación Inicial realizada por el grupo 1

Para la pregunta 4 (¿La ubicación del documento en el grupo, según su importancia es:?), los resultados generales son buenos (entre R3 y R2) y de manera similar a P3, los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos frecuentes por documentos y el Criterio de Información Bayesiano. En este caso el método de etiquetado no fue determinante (Ver Figura 26).

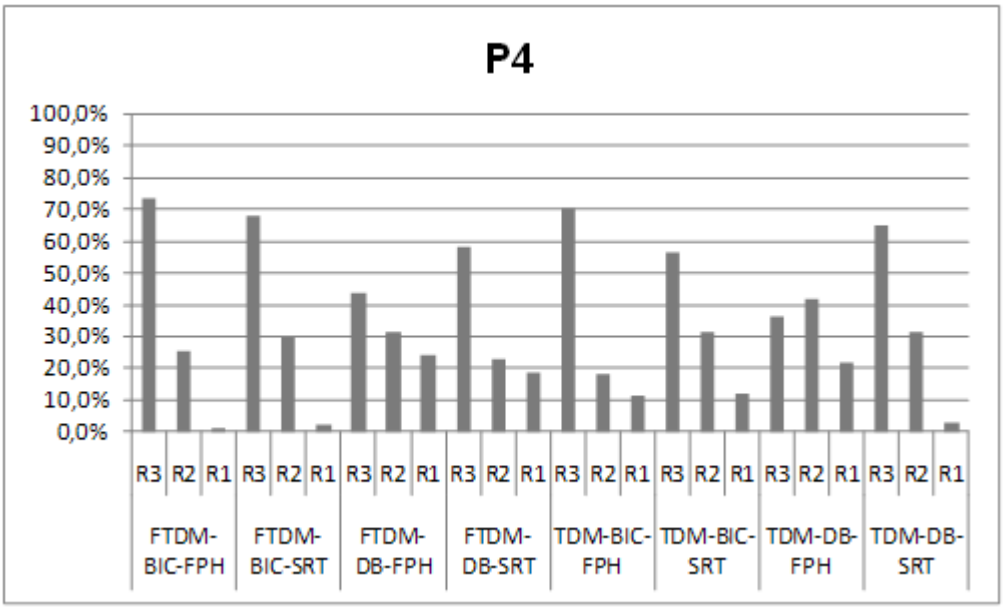

Figura 26. Resultados específicos para P4 en diferentes escenarios en la Evaluación Inicial realizada por el grupo 1

11.5.3. Evaluación de la aplicación web.

Un método de evaluación basado en el usuario fue utilizado finalmente para evaluar los resultados del agrupamiento producidos por IGBHSK cuando las fuentes de datos son Google, Yahoo! y MSN Live. Para realizar esta evaluación se utilizó la aplicación final en la que se implementó el algoritmo, la cual fue desarrollada como una aplicación Web.

Por otra parte en las opciones de parámetros para ejecutar el algoritmo solo se tuvo en cuenta, en estas pruebas, el Criterio de Información Bayesiano puesto que en las pruebas iniciales (a la aplicación de escritorio) las combinaciones de parámetros que incluían este criterio presentaron mejores resultados en comparación con el índice de Davies_Bouldin. Igualmente en las pruebas para medir la Relevancia (Precisión, Exhaustividad y Medidad F) los resultados con el BIC fueron mejores.

Como se mencionó anteriormente, en las pruebas de la aplicación Web participaron los estudiantes de los grupos 2, 3 y 4 (descritos en la Tabla 13), quienes respondieron las siguientes preguntas.

Con respecto a cada grupo:

- P1: ¿El nombre del grupo representa los documentos que contiene? (mucho - R3, poco – R2, o nada – R1).
- P2: ¿El grupo es útil para su consulta? (útil - R3, moderadamente útil – R2, o no útil – R1).

Luego, para cada documento en cada grupo de documentos, los usuarios respondieron las siguientes preguntas:

- P3: ¿El documento se ajusta a la descripción del grupo? (muy bien - R3, moderadamente – R2, o nada – R1).
- P4: ¿La ubicación del documento en el grupo, según su importante es:? (correcta - R3, moderadamente correcta – R2, o incorrecta – R1).

Resultados obtenidos por el Grupo 2 de Usuarios

A continuación se presentan las consultas y los resultados obtenidos por el grupo 2 (estudiantes de GPI) en la evaluación de la aplicación Web.

La Tabla 15 presenta los temas de las consultas y las opciones que los usuarios utilizaron en el proceso de evaluación final. Para elegir los temas de las consultas se tuvo en cuenta la asignatura del Programa de Ingeniería de Sistemas, en la que los profesores permitieron realizar las pruebas del meta-buscador en horas de clase, en este caso el grupo 2 de estudiantes se encontraba en horas de clase de la asignatura Gestión de Proyectos Informáticos, por lo tanto los temas los habían estudiado en los últimos días de esa clase.

Consulta	Representación de los Documentos	Función de Fitness	Etiquetado
Quality Assurance	TDM	BIC	SRT
Performance Report	TDM	BIC	SRT
Quality Assurance	TDM	BIC	FPH
Performance Report	TDM	BIC	FPH

Tabla 15. Consultas y Opciones en el Proceso de Evaluación Final en el Grupo 2

Los resultados de IGBHSK en el grupo 2 se muestran en la Figura 27. La mayoría de las respuestas de los usuarios son R3 o R2. Por lo tanto, los resultados finales son muy satisfactorios y confirman la eficiencia del meta-buscador que implementa el algoritmo IGBHSK.

Para la pregunta 1 (¿El nombre del grupo representa los documentos que contiene?), los resultados generales son buenos (R3) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo. (Ver Figura 28).

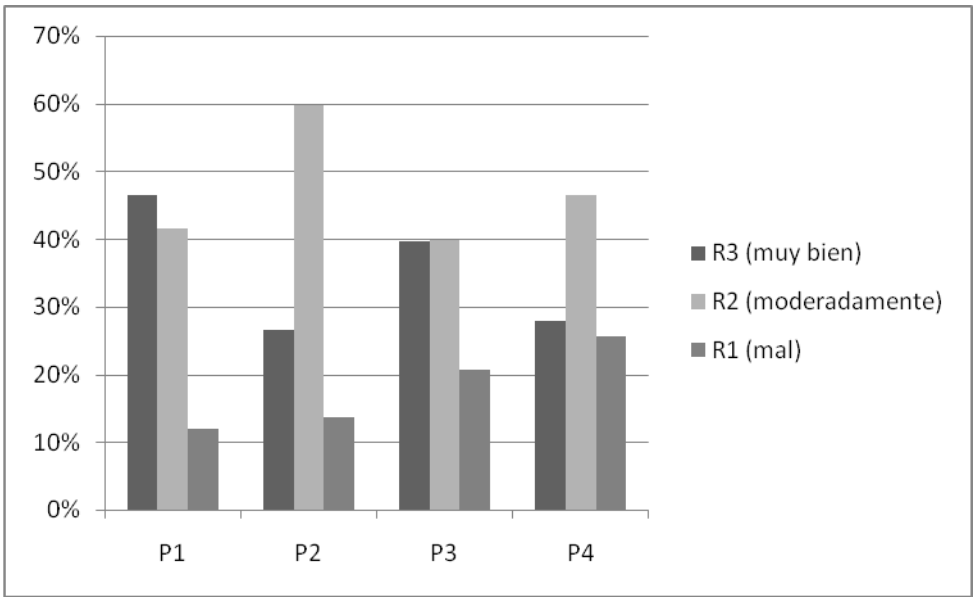

Figura 27. Resultados para las cuatro preguntas en la Evaluación Final realizada por el grupo 2

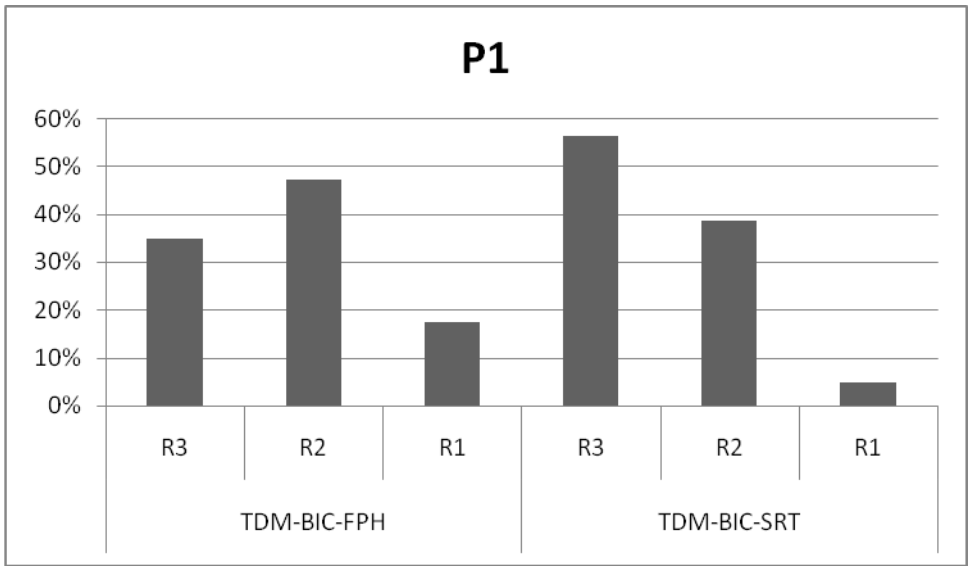

Figura 28. Resultados específicos para P1 en diferentes escenarios de la evaluación final realizada por el grupo 2

Para la pregunta 2 (¿El grupo es útil para su consulta?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 29).

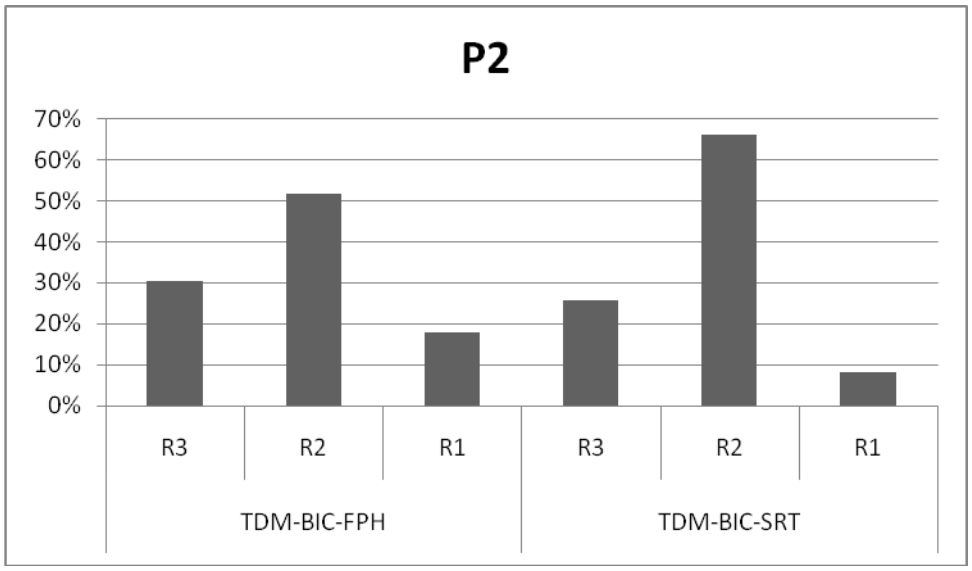

Figura 29. Resultados específicos para P2 en diferentes escenarios de la evaluación final realizada por el grupo 2

Para la pregunta 3 (¿El documento se ajusta a la descripción del grupo?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 30).

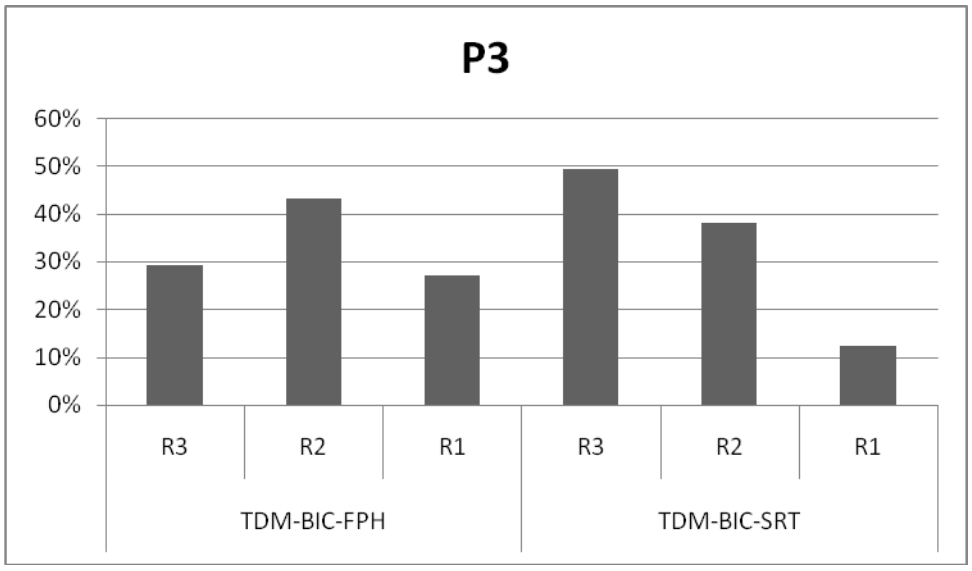

Figura 30. Resultados específicos para P3 en diferentes escenarios de la evaluación final realizada por el grupo 2

Para la pregunta 4 (¿La ubicación del documento en el grupo, según su importancia es:?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 31).

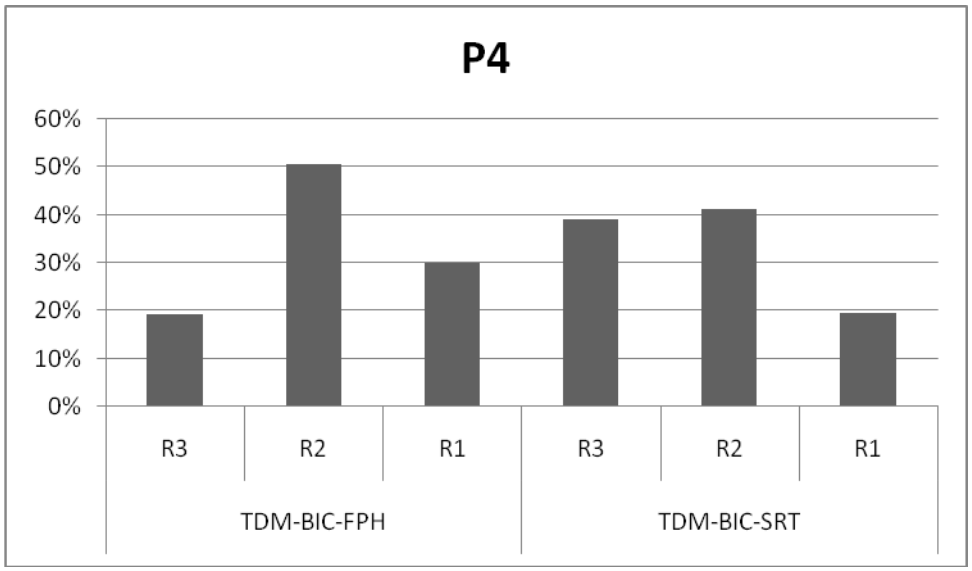

Figura 31. Resultados específicos para P4 en diferentes escenarios de la evaluación final realizada por el grupo 2

Resultados obtenidos por el Grupo 3 de Usuarios

A continuación se presentan las consultas y los resultados obtenidos por el grupo 3 (estudiantes de Estructuras de Lenguaje) en la evaluación de la aplicación Web.

En la Tabla 16 se presentan ejemplos de las consultas y las opciones que los usuarios utilizaron en el proceso de evaluación final. Para elegir los temas de las consultas se tuvo en cuenta la asignatura del Programa de Ingeniería de Sistemas, en la que los profesores permitieron realizar las pruebas del meta-buscador en horas de clase, en este caso el grupo 3 de estudiantes se encontraba en horas de clase de la asignatura Estructuras de Lenguaje, por lo tanto los temas los habían estudiado en los últimos días de esa clase.

Consulta	Representación de los Documentos	Función de Fitness	Etiquetado
Java	FTDM	BIC	SRT
Functional Programming	FTDM	BIC	SRT
Java	FTDM	BIC	FPH
Functional Programming	FTDM	BIC	FPH

Tabla 16. Consultas y Opciones en el Proceso de Evaluación Final en el Grupo 3

Los resultados de IGBHSK en el grupo 3 se muestran en la Figura 32. La mayoría de las respuestas de los usuarios son R3 o R2. Por lo tanto, los resultados finales son muy satisfactorios y confirman la eficiencia del meta-buscador que implementa el algoritmo IGBHSK.

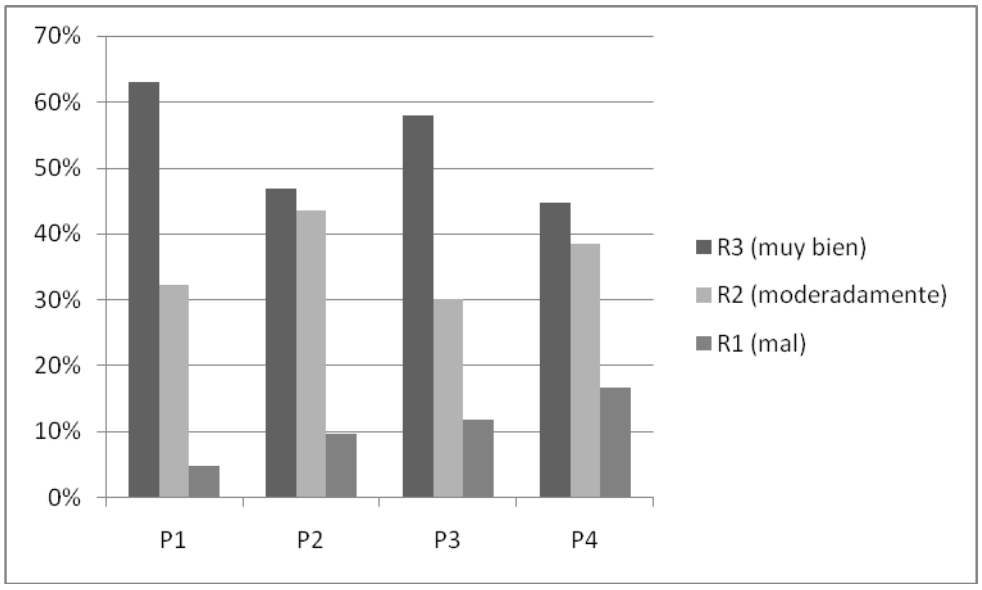

Figura 32. Resultados para las cuatro preguntas en la Evaluación Final realizada por el grupo 3

Para la pregunta 1 (¿El nombre del grupo representa los documentos que contiene?), los resultados generales son buenos (R3) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo. (Ver Figura 33).

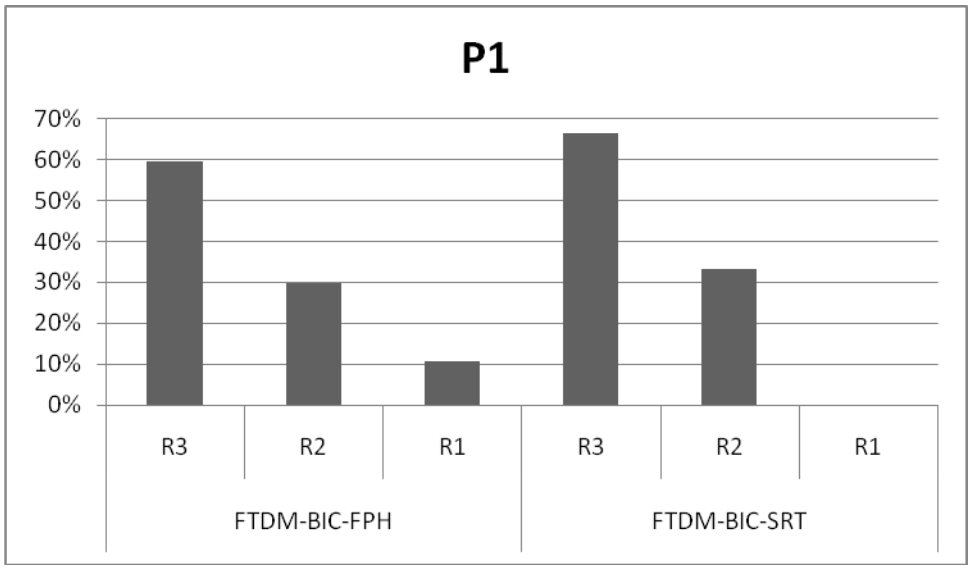

Figura 33. Resultados específicos para P1 en diferentes escenarios de la evaluación final realizada por el grupo 3

Para la pregunta 2 (¿El grupo es útil para su consulta?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 34).

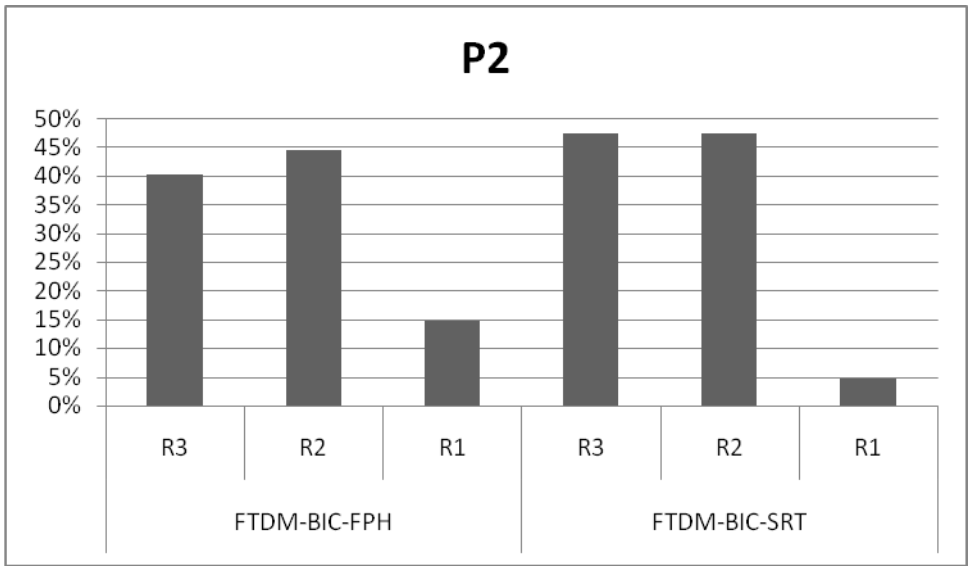

Figura 34. Resultados específicos para P2 en diferentes escenarios de la evaluación final realizada por el grupo 3

Para la pregunta 3 (¿El documento se ajusta a la descripción del grupo?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 35).

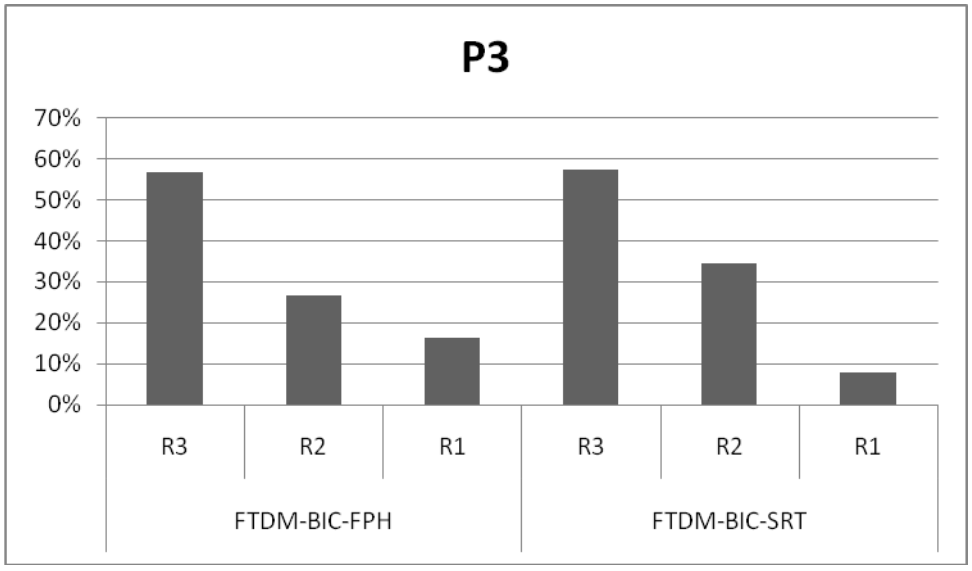

Figura 35. Resultados específicos para P3 en diferentes escenarios de la evaluación final realizada por el grupo 3

Para la pregunta 4 (¿La ubicación del documento en el grupo, según su importancia es:?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos frecuentes por documentos, el

Criterio de Información Bayesiano y frases frecuentes para definir la etiqueta de grupo (Ver Figura 36).

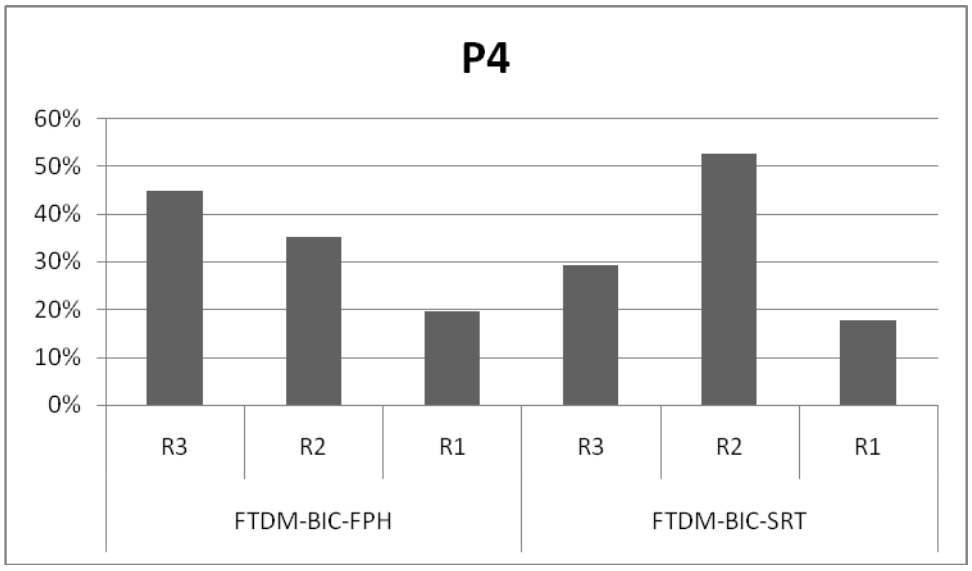

Figura 36. Resultados específicos para P4 en diferentes escenarios de la evaluación final realizada por el grupo 2

Resultados obtenidos por el Grupo 4 de Usuarios

A continuación se presentan las consultas y los resultados obtenidos por el grupo 4 (estudiantes de CABD) en la evaluación de la aplicación Web.

En la Tabla 17 se presentan ejemplos de las consultas y las opciones que los usuarios utilizaron en el proceso de evaluación final. Para elegir los temas de las consultas se tuvo en cuenta la asignatura del Programa de Ingeniería de Sistemas, en la que los profesores permitieron realizar las pruebas del meta-buscador en horas de clase, en este caso el grupo 3 de estudiantes se encontraba en horas de clase de la asignatura Conceptos Avanzados de Bases de Datos, por lo tanto los temas los habían estudiado en los últimos días de esa clase.

Por otra parte para escoger los parámetros utilizados en la consultas del grupo 4 de estudiantes, se tuvo en cuenta los resultados obtenidos en las dos pruebas anteriores, los cuales presentaron mejores resultados cuando se utiliza la Matriz de Términos Frecuentes por Documentos en la Representación de los Documentos, por lo tanto este parámetro se dejó fijo en FTDM.

Los resultados de IGBHSK en el grupo 4 se muestran en la Figura 37. La mayoría de las respuestas de los usuarios son R3 o R2. Por lo tanto, los resultados finales son muy satisfactorios y confirman la eficiencia del meta-buscador que implementa el algoritmo IGBHSK.

Consulta	Representación de los Documentos	Función de Fitness	Etiquetado
Index Bitmap	FTDM	BIC	SRT
Index b-tree	FTDM	BIC	FPH
Index Bitmap	FTDM	BIC	FPH
Index b-tree	FTDM	BIC	SRT

Tabla 17. Consultas y Opciones en el Proceso de Evaluación Final en el Grupo 4

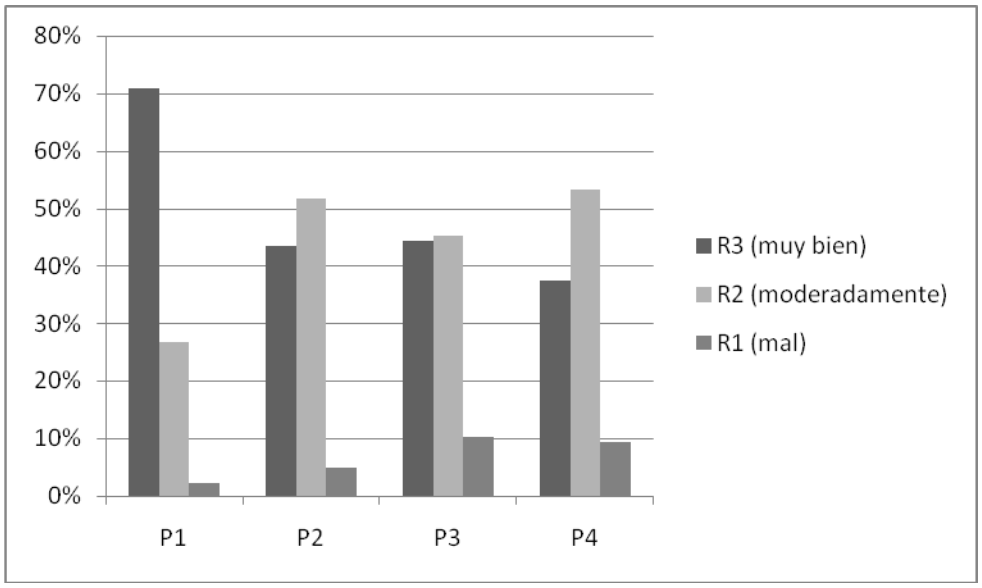

Figura 37. Resultados para las cuatro preguntas en la Evaluación Final realizada por el grupo 4

Para la pregunta 1 (¿El nombre del grupo representa los documentos que contiene?), los resultados generales son buenos (R3) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo. (Ver Figura 38).

Para la pregunta 2 (¿El grupo es útil para su consulta?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 39).

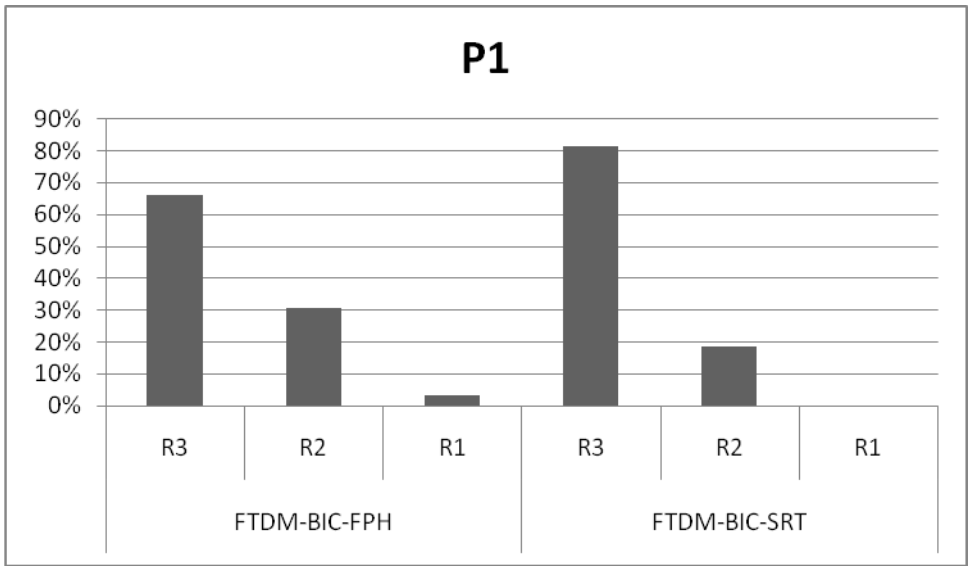

Figura 38. Resultados específicos para P1 en diferentes escenarios de la evaluación final realizada por el grupo 4

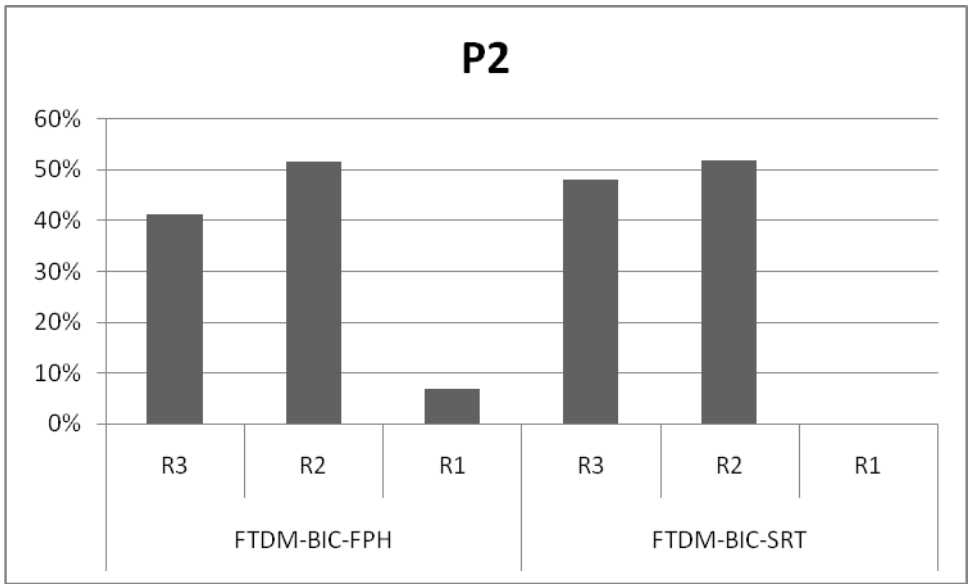

Figura 39. Resultados específicos para P2 en diferentes escenarios de la evaluación final realizada por el grupo 4

Para la pregunta 3 (¿El documento se ajusta a la descripción del grupo?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados son obtenidos cuando el algoritmo utiliza la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 40).

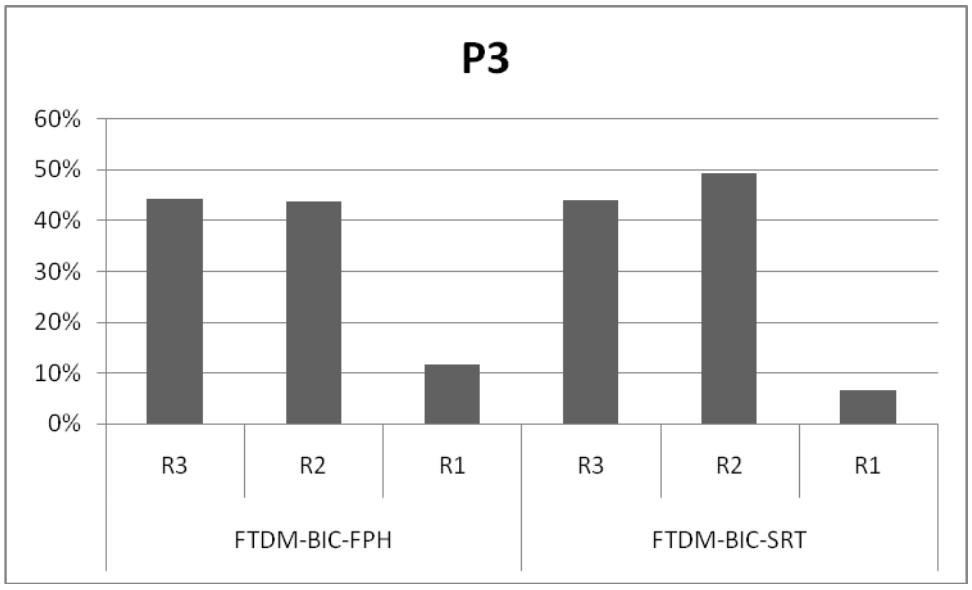

Figura 40. Resultados específicos para P3 en diferentes escenarios de la evaluación final realizada por el grupo 4

Para la pregunta 4 (¿La ubicación del documento en el grupo, según su importancia es:?), los resultados generales son buenos (entre R3 y R2) y los mejores resultados se obtuvieron cuando el algoritmo usa la matriz de términos frecuentes por documentos, el Criterio de Información Bayesiano y términos estadísticamente más representativos para definir la etiqueta de grupo (Ver Figura 41).

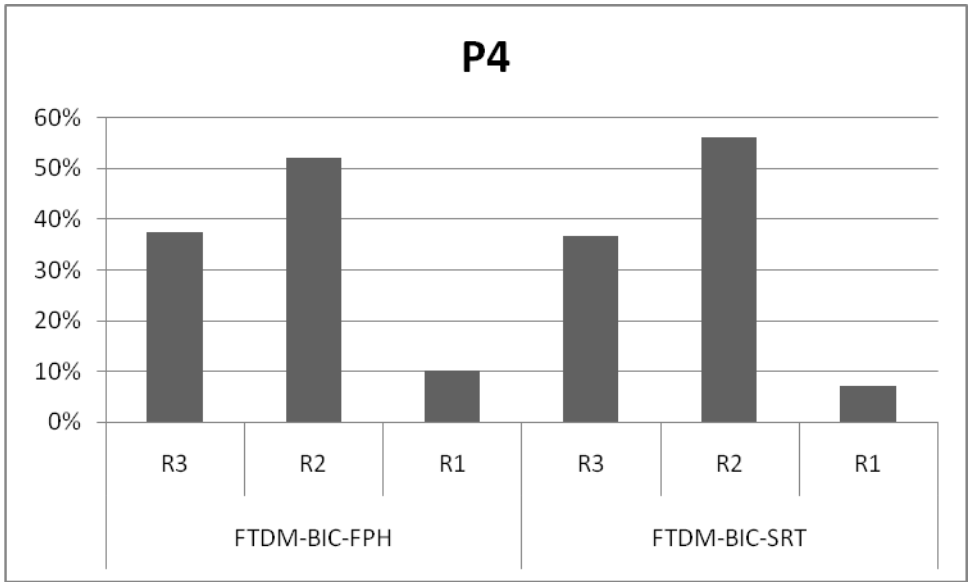

Figura 41. Resultados específicos para P4 en diferentes escenarios de la evaluación final realizada por el grupo 2

11.5.4. Evaluación de usabilidad.

Para realizar la prueba de usabilidad a la aplicación web del meta-buscador se utilizó un test de usabilidad, para el cual se tuvo en cuenta las pautas de evaluación heurística planteada por el Dr. Jaime Sánchez I. de la Universidad de Chile. Además, la encuesta fue revisada y corregida finalmente por el ingeniero Carlos Alberto Árdila Albarracín, Ingeniero de Sistemas del Departamento de Sistemas de la Universidad del Cauca, quien tiene amplios conocimientos sobre el tema (una de las asignaturas que actualmente orienta es Calidad del Software).

En la Tabla 18 se presenta una breve descripción de cada una de las secciones que se incluyeron en la evaluación y las preguntas que se hicieron dentro de cada una de ellas.

Las siguientes preguntas tienen los siguientes posibles valores de respuesta:
Pésimo___ Deficiente____ Aceptable____ Bueno____ Excelente___

SECCIÓN: VISIBILIDAD DEL ESTADO DEL SISTEMA	
Descripción	El usuario siempre debe saber que se está haciendo. El usuario debe conocer por ejemplo, cuando se encuentra apuntando un hipervínculo, descargando imágenes o la misma página. Los efectos para dar a conocer son variados, como resaltar texto, tipos de apuntadores, barras de estado, etc.
Preguntas:	1. El sitio muestra claramente dónde se encuentra el usuario. 2. Los enlaces posibles de explorar están claramente señalados.
SECCIÓN: RELACIÓN ENTRE SISTEMA Y MUNDO REAL	
Descripción	El sitio web, debe manejar el mismo lenguaje que la audiencia objetivo. El vocabulario, conceptos y frases deben ser de uso común de los usuarios.
Preguntas:	3. El lenguaje es claro. 4. Las palabras son de significado conocido. 5. Los iconos generan significado consistente con el dominio de la aplicación.
SECCIÓN: CONTROL DEL USUARIO Y LIBERTAD	
Descripción	Los usuarios frecuentemente realizan acciones que no desean, es decir: los humanos se equivocan. Se debe proveer de “salidas de emergencia”' ante todo tipo de acción que el usuario pudiera realizar. También deben existir funciones para deshacer y rehacer las acciones.
Preguntas:	6. Es fácil volver a la página principal desde cualquier ubicación 7. Provee botones propios para volver o dar paso a otra página 8. El sitio es soportado por distintos visores sin dificultad (Mozilla

	Firefox, Chrome, Internet Explorer).
SECCIÓN: CONSISTENCIA Y ESTÁNDARES	
Descripción	Todas las páginas de un sitio web deben guardar similitudes y convenciones, tales que las hagan consistentes ante la vista de los usuarios. Los usuarios no deben preocuparse por cambios de vocabulario o estilo en las diferentes páginas de un sitio, con la consecuente necesidad del usuario de revisar completamente el funcionamiento y opciones del sitio.
Preguntas:	9. Existe coherencia entre el nombre de un enlace y el sitio al que apunta. 10. El título de la página es coherente con el contenido.
SECCIÓN: ESTÉTICA Y DISEÑO MINIMALISTA	
Descripción	Los diálogos deben ser concretos y sin información irrelevante. La información extra puede disminuir la visibilidad relativa de la información importante. La información más importante debe estar a primera vista del usuario.
Preguntas:	11. El contenido está bien clasificado. 12. El contenido está bien distribuido en el sitio.
SECCIÓN: VELOCIDAD Y MEDIOS	
Descripción	Los sitios deben ser rápidos y ser eficientes en su uso tanto para usuarios novatos como expertos. Los usuarios novatos deben tener accesos rápidos a las opciones que desean, así como los novatos pasos fáciles e inequívocos hacia su objetivo. La rapidez puede alcanzarse mediante el uso de comunicación tipo texto, o de marcado, más que de imágenes o flash.
Preguntas:	13. Los medios utilizados (imágenes, video, sonido) demoran en exceso la carga del sitio. 14. La calidad técnica de videos, imágenes y sonido es aceptable.

Tabla 18. Preguntas Encuesta de Usabilidad

La encuesta se entregó impresa a los 56 usuarios (divididos en tres grupos) quienes evaluaron la aplicación final (aplicación web), diligenciando la encuesta una vez terminaron de usar el sistema y respondiendo las cuatro preguntas mencionadas en las secciones anteriores (relacionadas con los resultados del agrupamiento). Los resultados generales de la encuesta realizada en los tres grupos de usuarios se presentan a continuación, y concuerdan con que el meta-buscador cumple positivamente con las secciones evaluadas por la encuesta.

Por otra parte, a medida que se iban realizando las pruebas con cada grupo se iban implementando las mejoras sugeridas por los usuarios y algunos cambios para mejorar los resultados del test de usabilidad.

Para ver los detalles de las preguntas evaluadas en cada grupo remitirse al Anexo F.

Resultados del test de usabilidad realizado por el Grupo 2 de usuarios (Estudiantes de Gestión de Proyectos Informáticos)

En la Figura 42 se presentan los resultados generales obtenidos en las 14 preguntas del test de usabilidad.

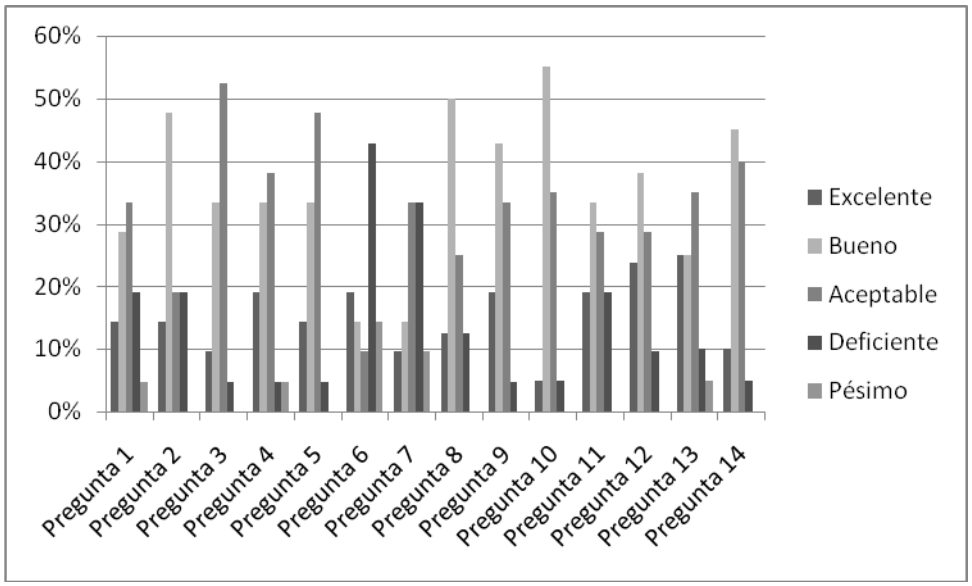

Figura 42. Resultados obtenidos en la Evaluación de Usabilidad en el Grupo 2 de Usuarios

Con base en los resultados obtenidos en este primer test de usabilidad se implementaron algunas mejoras tales como: una “Miga de Pan”, la cual indica la ubicación del usuario en la página donde se encuentra ubicado en cada momento; se implementaron controles AJAX para evitar el refresco continuo (o actualización) de las páginas.

Resultados del test de usabilidad realizado por el Grupo 3 de usuarios (Estudiantes de Estructuras de Lenguaje)

En la Figura 43 se presentan los resultados generales obtenidos en las 14 preguntas del test de usabilidad.

Con base en los resultados obtenidos en este segundo test de usabilidad se implementaron algunas mejoras tales como: cambios en el manejo de las variables de sesión, lo cual ayudó aumentar el tiempo que duraban las sesiones permitiendo que los usuarios continuaran con las búsquedas sin interrupciones.

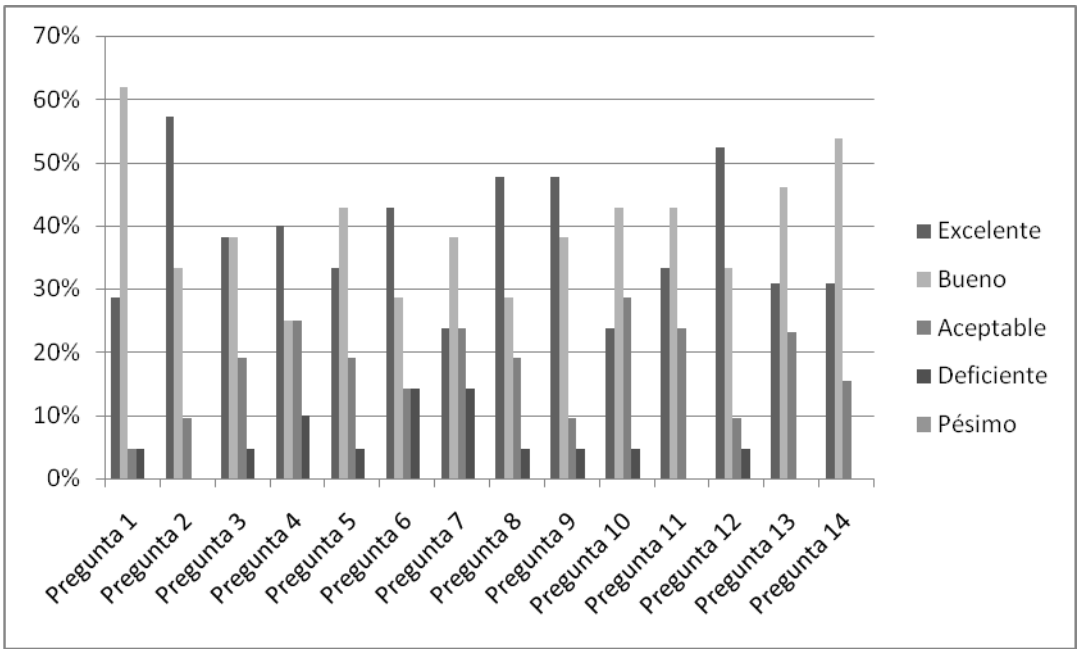

Figura 43. Resultados obtenidos en la Evaluación de Usabilidad en el Grupo 3 de Usuarios

Resultados del test de usabilidad realizado por el Grupo 4 de usuarios (Estudiantes de Conceptos Avanzados de Bases de Datos)

En la Figura 44 se presentan los resultados generales obtenidos en las 14 preguntas del test de usabilidad.

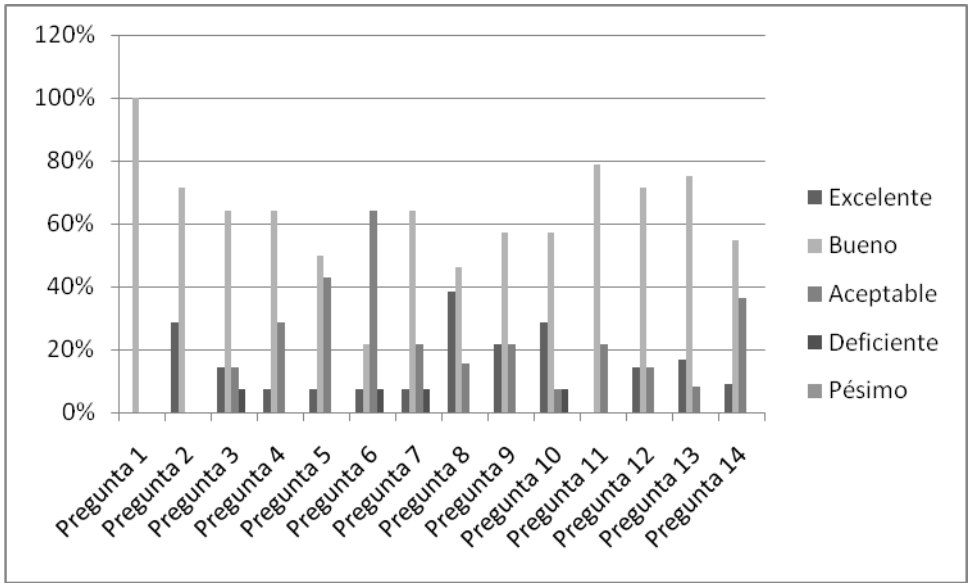

Figura 44. Resultados obtenidos en la Evaluación de Usabilidad en el Grupo 4 de Usuarios

PARTE IV – CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

12. CONCLUSIONES

Se diseñó, implementó y evaluó el algoritmo IGBHSK que combina la meta-heurística de la Mejor Búsqueda Armónica Global y el algoritmo K-means. Este algoritmo encuentra automáticamente el número de grupos, asigna nombres adecuados a los grupos de documentos y como se mostró en las pruebas, además de ser viable en el clustering de documentos web mostró mejores resultados que LINGO, uno de los algoritmos más recientes y citados en la literatura de clustering de documentos web.

El modelo de representación de documentos "Términos Frecuentes por documentos" es más adecuado que el modelo estándar de "Términos por documentos", debido a que el primero, por una parte reduce la dimensionalidad de los documentos y por otra parte arrojó mejores resultados (mayores promedios de precisión) en las pruebas realizadas por los usuarios tanto en la aplicación de escritorio como en la aplicación web (meta-buscador).

Se comparó el Criterio de Información Bayesiano (BIC) y el índice de Davies-Bouldin (DB) para encontrar la forma más apropiada de definir la mejor función de fitness (que permite hallar el valor de k). En este sentido, los resultados favorecieron a BIC, específicamente en el clustering de documentos web.

El algoritmo IGBHSK presentó mejores resultados en la evaluación cuando se ejecutó la Matriz de Términos Frecuentes por Documentos y el Criterio de Información Bayesiano. Para esto, los valores por defecto que se deben usar para futuras pruebas o la liberación del meta-buscador para el uso masivo de los usuarios deben ser estos dos.

Los resultados obtenidos en el laboratorio con los data sets de Reuters y DMoz, en relación con la medida de precisión superaron los resultados obtenidos por la mayoría de los algoritmos propuestos anteriormente para el clustering de documentos, los cuales se encuentran entre el 60% y 80%, mientras que IGBHSK alcanzó en promedio 85.2% con un promedio aproximado de 0,3 segundos en tiempo de ejecución[12].

La arquitectura del meta-buscador se puede modificar fácilmente para agregar nuevos componentes relacionados con el pre-procesamiento, el algoritmo de clustering o el tipo de etiquetado, lo que la convierte en una arquitectura adaptable que puede servir como base para futuras investigaciones sobre el clustering de documentos web.

El meta-buscador web (GruWeb) se desarrolló utilizando la metodología seleccionada sin presentar ningún problema que requiera ser mencionado. Para su uso en las pruebas se utiliza una sección de preguntas, la cual a futuro se eliminará para permitir que las personas lo puedan usar masivamente.

[12] La ejecución del algoritmo propuesto se realizó en un computador con las siguientes características: Intel Pentium 4 con 3,2 GHz, RAM de 2 GB.

13. RECOMENDACIONES Y TRABAJO FUTURO

Tener en cuenta los títulos de los documentos que se recuperan como resultados de las consultas a Google, Yahoo! y MSN Live, para realizar la representación de los documentos, el procesamiento y el etiquetado de los grupos generados. Se podría tener en cuenta asignándole un peso importante con respecto al snippet del documento. Esta modificación podría dar mejores resultados en el agrupamiento, puesto que los títulos son por naturaleza resúmenes adecuados de los documentos y en algunas ocasiones son más representativos que los snippets.

Comparar la precisión del sistema cuando se usan los snippets o el texto completo del documento, frente al tiempo de respuesta. Inclusive, estudiar la incorporación del análisis de los metadatos semánticos que actualmente tienen algunas páginas y recursos en Internet.

Utilizar una ontología como WordNet o un tesauro para mejorar tanto el proceso de creación de los grupos como el proceso de etiquetado de los mismos. Además, experimentar con variaciones de tiempo (0,4 a 0,8 segundos) en el procesamiento del algoritmo, con el fin de medir los cambios en la precisión del algoritmo.

Implementar otros algoritmos de stemming para permitir la búsqueda en idiomas diferentes al inglés (Recuperación de Información Multi Lenguaje). Y analizar el impacto de manejar múltiples idiomas en el algoritmo de clustering, o proponer una mejora al IGBHSK para que pueda mantener o mejorar los resultados obtenidos en esta investigación con el manejo de nuevos idiomas.

Implementar el algoritmo “Hybridizing K-means as one step of HSCLUST” [41], incluyendo el uso de BIC y DB para definir automáticamente el número de grupos, y comparar los resultados con IGBHSK. Teniendo en cuenta no sólo los valores de precisión, exhaustividad y medida F, sino también el tiempo de cómputo.

PARTE V – GLOSARIO Y REFERENCIAS BIBLIOGRÁFICAS

14. GLOSARIO

1. **SVD**: Acrónimo de Singular Value Decomposition. Técnica del álgebra lineal para factorizar matrices.

2. **MINERÍA DE DATOS**: DM o Data Mining, es el proceso de extracción de información y patrones de comportamiento que permanecen ocultos en grandes cantidades de información.

3. **ONTOLOGÍA**: Es la formulación de un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con la finalidad de facilitar la comunicación y compartir la información entre diferentes sistemas. Para que la ontología sea aceptada debe ser formal y aceptada por una comunidad de expertos.

4. **MOTOR DE BÚSQUEDA**. Sistemas que recorren la red recolectando e indexando la mayor cantidad de información posible, gracias a programas automáticos conocidos como robots, también llamados spider.

5. **META-BUSCADOR**: Sistemas que no disponen de bases de datos propias, puesto que buscan en otros buscadores. Recogen la petición del usuario y la envían a los buscadores, éstos la devuelven y los meta-buscadores la clasifican antes de presentarla al usuario.

6. **ÍNDICES TEMÁTICOS:** Son listas de recursos organizadas en jerarquías desde lo más general a lo más específico. El proceso de clasificación se hace de forma manual.

7. **K-MEANS:** Algoritmo que tiene como objetivo minimizar las diferencias de los elementos en cada clúster al mismo tiempo que maximiza la diferencia de los elementos que caen en diferentes clústeres.

8. **CLÚSTER:** Es un punto usado para representar un conjunto de valores que tienen algo en común y se pueden agrupar en función de una característica determinada.

15. REFERENCIAS BIBLIOGRÁFICAS

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

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

[3] K. Hammouda, "Web Mining: Clustering Web Documents A Preliminary Review," 2001.

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

[5] N. G. Venkat, V. R. Vijay, I. G. William, and K. Rajesh, "Information Retrieval on the World Wide Web," *IEEE Internet Computing,* vol. 1, pp. 58-68, 1997.

[6] Netcraft Ltda., "December 2008 Web Server Survey," 2008.

[7] F. Can, R. Nuray, and A. B. Sevdik, "Automatic performance evaluation of Web search engines," *Information Processing & Management,* vol. 40, pp. 495-514, 2004.

[8] 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.

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

[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] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation,* vol. 198, pp. 643-656, 2008.

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

[13] P. Ma, W. Zhong, Y. Feng, and J. S. Liu, "Bayesian Functional Data Clustering for Temporal Microarray Data," in *International Journal of Plant Genomics*. vol. 2008, 2008, p. 4 pages.

[14] 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.

[15] S. Osiński, "An Algorithm for clustering of web search results." vol. Master Poland: Poznań University of Technology,, 2003, p. 91.

[16] G. Bolaños and E. Perez, "Buscador Inteligente Basado en Minería de Datos," in *Sistemas* Popayán: Universidad del Cauca, 2009.

[17] V. D. Fresno Fernández, "Representación Autocontenida de Documentos HTML: una propuesta basada en Combinaciones Heurísticas de Criterios," in *Departamento de Ingeniería Telemática y Tecnología Electrónica*. vol. Doctoral Degree Madrid, España: Universidad Rey Juan Carlos, 2006, p. 245.

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

[19] L. Huang, H. Chen, X. Wang, and G. Chen, "A fast algorithm for mining association rules," *J. Comput. Sci. Technol.,* vol. 15, pp. 619-624, 2000.

[20] 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.

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

[22] O. E. O. M. R. M. K. Oren Zamir, "Fast and Intuitive Clustering of Web Documents," AAAI, 1997.

[23] P. Berkhin, "Survey Of Clustering Data Mining Techniques," 2002.

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

[25] J. Han, M. Kamber, and A. K. H. Tung, "Spatial Clustering Methods in Data Mining: A Survey," in *Geographic Data Mining and Knowledge Discovery*: Taylor and Francis, 2001.

[26] 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.

[27] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000.

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

[29] Z. Xueping, W. Jiayao, W. Fang, F. Zhongshan, and L. Xiaoqing, "A Novel Spatial Clustering with Obstacles Constraints Based on Genetic Algorithms and K-

Medoids," in *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications - Volume 01*: IEEE Computer Society, 2006.

[30] D. Zhang and Y. Dong, "Semantic, Hierarchical, Online Clustering of Web Search Results," in *Advanced Web Technologies and Applications*, 2004, pp. 69-78.

[31] S. Osiński, "Dimensionality Reduction Techniques for Search Results Clustering," 2004.

[32] S. Osiński and D. Weiss, "Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data," 2004.

[33] S. Osiński, J. Stefanowski, and D. Weiss, "Lingo: Search results clustering algorithm based on Singular Value Decomposition," 2004.

[34] N. Chi Lang and N. Hung Son, "A tolerance rough set approach to clustering web search results," in *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases* Pisa, Italy: Springer-Verlag New York, Inc., 2004.

[35] S. Wei and P. Soon Cheol, "Genetic Algorithm-based Text Clustering Technique: Automatic Evolution of Clusters with High Efficiency," in *Proceedings of the Seventh International Conference on Web-Age Information Management Workshops*: IEEE Computer Society, 2006.

[36] Z. Geem, J. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *SIMULATION,* vol. 76, pp. 60-68, 2001.

[37] A. Cao, Q. Song, and X. Yang, "Robust information clustering incorporating spatial information for breast mass detection in digitized mammograms," *Computer Vision and Image Understanding,* vol. 109, pp. 86-96, 2008.

[38] S. J. Redmond and C. Heneghan, "A method for initialising the K-means clustering algorithm using kd-trees," *Pattern Recognition Letters,* vol. 28, pp. 965-973, 2007.

[39] L. d. Santos Coelho and D. L. de Andrade Bernert, "An improved harmony search algorithm for synchronization of discrete-time chaotic systems," *Chaos, Solitons & Fractals,* vol. In Press, Corrected Proof, 2008.

[40] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation,* vol. 188, pp. 1567-1579, 2007.

[41] 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.

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

[43] G. H. Tolosa and F. R. A. Bordignon, "Introducción a la Recuperación de Información," 2007.

[44] M. Porter, "An algorithm for suffix stripping," *Program,* vol. 14, pp. 130-137, 1980.

[45] J. P. Ramos Hernández and G. A. Hernández, "Indización y Búsqueda a través de Lucene," 2008.

[46] Lemur, *Sitio web del proyecto Lemur*: Disponible de: http://www.lemurproject.org/.

[47] E. Eckard and J.-C. e. Chappelier, "Free Software for research in Information Retrieval and Textual Clustering," 2007.

[48] Xapian, *Sitio web de Xapian*: Disponible de: http://www.xapian.org.

[49] Terrier, *Sitio web de Terrier*: Disponible desde: http://ir.dcs.gla.ac.uk/terrier/.

[50] Lucene, *Sitio web de Lucene*: Disponible desde: http://lucene.apache.org/.

[51] C. Middleton and R. Baeza-Yates, "A Comparison of Open Source Search Engines," 2008.

[52] Lucene.net, *Sitio web de Lucene.net*: Disponible en: http://incubator.apache.org/lucene.net/.

[53] Lucene-java Wiki, *Applications and web applications using Lucene*: Disponible en: http://wiki.apache.org/lucene-java/PoweredBy.

[54] O. Gospodnetic, E. Hatcher, and D. Cutting, *Lucene in action*: Mannaging, 2005.

[55] V. P. Madrid Gorelov, A. F. Zazo, C. G. Figueroa, and J. L. Alonso Berrocal, "Librerías Lucene y dotLucene para Recuperación de Información. Estudio y desarrollo de casos prácticos," 2007.

[56] G. Grefenstette, "Comparing two language identification schemes," 1995.

[57] O. S. C. Codeplex, "Frequent Pattern Miner ": Sitio web disponible en: http://www.codeplex.com/fpminer/SourceControl/ListDownloadableCommits.aspx.

[58] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in *IEEE Int'l. Conf. on Neural Networks.*, Perth, Australia, 1995, pp. 1942–1948.

[59] R. B. Miller, "Response time in mancomputer conversational transactions," *Proc. AFIPS Fall Joint Computer Conference,* vol. 33, pp. 267-277, 1968.

[60] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer: An information workspace," *ACM CHI'91 Conference,* pp. 181-188, 1991.

[61] J. Nielsen, "Designing Web Usability: the Practice of Simplicity," *New Riders,* 1999.

[62] G. H. O. Mahamed, P. E. Andries, and S. Ayed, "An overview of clustering methods," *Intell. Data Anal.,* vol. 11, pp. 583-605, 2007.

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

[64] A. Webb, *Statistical Pattern Recognition, 2nd Edition*: {John Wiley & Sons}, 2002.

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

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

[67] H. Ralambondrainy, "A conceptual version of the K-means algorithm," *Pattern Recognition Letters,* vol. 16, pp. 1147-1157, 1995.

[68] S. Weiguo, L. Xiaohui, and M. Fairhurst, "A Niching Memetic Algorithm for Simultaneous Clustering and Feature Selection," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 20, pp. 868-879, 2008.