Universidad del Biobío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información Profesor Guía: Claudio Gutiérrez S.

Obtención de datos empíricos para la nueva medida de similitud QDSM usando algoritmos de clustering en recuperación de la información.

"Informe de Proyecto de Título"

Fecha: 20 de Marzo de 2017

Alumno:
Daniel A. Ibacache Soto.

Ingeniería Civil en Informática

AGRADECIMIENTOS

Primeramente, quiero expresar mi gratitud a mi Profesor Guía Claudio Gutiérrez Soto, por su constante apoyo y motivación para realizar este proyecto de título, además de guiarme paso a paso en la investigación y en el desarrollo de esta tesis.

También quiero agradecer a mis amigos Frederick Lara y Delia Moncada, ya que siempre estuvimos juntos en la realización de nuestras tesis, motivandonos y ayundandonos entre nosotros, además de las divertidas situaciones que surgieron.

Por último, pero no menos importante, quiero agradecer a mi familia, compuesta por mi hermano Marco y mi hermana Gisel, mi madre Patricia y mi padre Marco, por la motivación y la confianza que tienen hacia mi persona. Especialmente a mis padres les agradezco su total y desinteresado apoyo durante mis años de estudios, ya que con su gran esfuerzo y dedicación, soy la persona que soy ahora, que está terminando sus estudios universitarios para llegar a ser un profesional.

RESUMEN

La recuperación de la información (RI) obtiene material (usualmente documentos) de una naturaleza no estructurada (frecuentemente textos) la cual satisface la necesidad de información de los usuarios en grandes colecciones (normalmente almacenados en los computadores). Un sistema de recuperación de la información (SRI) propone representar y almacenar grandes cantidades de información, con el objetivo de facilitar y acelerar la búsqueda de información relevante para una consulta del usuario. Se pueden encontrar dos procesos principales en un SRI: indexación y matching. El proceso de indexación corresponde a la representación y almacenamiento de los documentos. En cambio el matching pretende estimar si un documento es relevante de acuerdo a la consulta enviada por un usuario. Generalmente el matching es representado por una puntuación. Existe una amplia gama de propuestas en RI que tratan la indexación, funciones de matching, modelos formales, entre otros. Sin embargo, existe una cantidad mínima de propuestas que toman ventaja de las búsquedas realizadas anteriormente por los usuarios. Las búsquedas pasadas proveen una fuente útil de información para nuevos usuarios (nuevas consultas). Por ejemplo, un usuario que busca un nuevo tema puede beneficiarse de las búsquedas realizadas anteriormente por otros usuarios que buscaron el mismo tema.

Los documentos relevantes de las consultas pasadas podrían ser utilizados para responder una nueva consulta. Este principio podría beneficiarse del clustering de las búsquedas pasadas de acuerdo a su similitud. El clustering consiste en construir grupos de documentos similares. Se pueden identificar dos categorías de clustering: clustering estático y clustering de post-recuperación. El clustering estático es la aplicación tradicional del método clúster en una colección de documentos. En cambio, el clustering de post-recuperación incluye la información de la consulta al clustering de documentos. Típicamente, las funciones de similitud (entre documentos) como la distancia del coseno, se utilizan en el clustering estático. Sin embargo, esas funciones no consideran el contexto específico bajo el cual se juzga la similitud de dos objetos.

Este proyecto de título se centra en obtener resultados empíricos de la medida de distancia del coseno, comparándolo con una nueva medida de similitud denominada QDSM (Query-Document Similarity Measure), usando algunos algoritmos de clustering, utilizando las consultas pasadas junto con los documentos asociados a ellas.

ABSTRACT

Information retrieval (IR) obtains material (usually documents) of an unstructured nature (usually texts) that satisfies the information need of users on large collections (usually stored in computers). Using an information retrieval system (IRS) is posible to represent and store a huge ampunt of information, at the same time that facilitate and accelerate the searches of relevant information for a user's query. Two main processes can be found in an IRS: indexing and matching. Indexing process corresponds to the representation and storing of the documents. In contrast, the matching tries to estimate if a document is relevant according to the query issued by a user. Usually, Matching is represented via a score.

A wide range of approaches in IR exist dealing with indexing, matching functions, formal models, among others. Nevertheless, few proposals take advantage from searches performed previously by users. Past searches provide a useful source of information for new users (new queries). For example, a user searching for a new topic could benefit from past searches made previously by other users who searched the same topic.

Relevant documents from past queries could be used to answer the new query. This principle could benefit from clustering of past searches according to their similarities. Clustering consists of building groups of similar documents. Two categories of clustering can be identified: static clustering and post-retrieval clustering. Static clustering is the traditional application of the cluster method on a document collection. In contrast, post-retrieval clustering includes information from the query into document clustering. Typically, similarity functions (between documents) such as cosine distance, are used in static clustering. Nevertheless, these functions do not consider the specific context under which the similarity of two objects is judged.

This thesis focuses on obtaining empirical results of the cosine distance measure, comparing it with a new similarity measure called QDSM (Query-Document Similarity Measure), using some clustering algorithms, and the past queries along with the associated documents to them.

Lista de Acrónimos

AOL	America Online
Clustering	Agrupamiento
idf	Frecuencia inversa de documentos
IFS (Inverted File Search)	Búsqueda de archivo invertido
LCS (Longest Common Subsequence)	Subsecuencia común más larga
MBW	Motores de Búsqueda Web
N-Nearest Neighbour test	Prueba del N-vecino más cercano
QDSM (Query-Document Similarity Measure)	Medida de similitud de consulta-documento
QSSM (Query-Sensitive Similarity Measures)	Medida de similitud sensible a la consulta
RI:	Recuperación de la Información
SRI	Sistema de Recuperación de la Información
tf	Frecuencia de términos
TREC (Text Retrieval Conferences)	Conferencias de Recuperación de Texto

Índice General

Índice Figuras

Índice Tablas

Capítulo 1: Introducción

La recuperación de la información (RI) implica varias tareas como la búsqueda de información, organización y almacenamiento. Estas tareas son llevadas a cabo con el propósito de proporcionar información relevante para los usuarios. Los requerimientos de los usuarios son realizados a través de consultas, para poder encontrar la información que necesitan. Es importante mencionar que hoy en día, es posible encontrar diferentes fuentes de información, tales como vídeo, audio, archivos XML, por mencionar algunos.

Usualmente, la información se almacena en un documento o en un conjunto de documentos (colección de documentos). Por otro lado, la relevancia sobre un documento o un conjunto de documentos es proporcionada por los usuarios. Esto último se conoce como juicios de los usuarios, así, dada una consulta y un conjunto de documentos relacionados con esa consulta, cada documento puede ser clasificado por los usuarios como “relevante” o “no relevante”, según la pertinencia de éste documento con respecto a la información que se necesita. El conjunto de documentos, el conjunto de consultas y sus juicios de usuarios son denominados por la comunidad de RI como colección de pruebas (test collection) (Sparck Jones & Van Rijsbergen, 1976).

Actualmente, una parte de la información se encuentra dispersa en miles y millones de documentos, por lo que resulta necesario la automatización de las tareas previamente mencionadas. Así, un Sistema de Recuperación de la Información (SRI) proporciona el apoyo necesario al usuario en la búsqueda de la información en una colección de documentos. Un SRI proporciona un conjunto de documentos clasificados que están relacionados con la consulta que ha sido enviada por un usuario. Comúnmente, las listas de los documentos proporcionados por un SRI es clasificado por una puntuación decreciente. Esta puntuación representa la similitud entre documentos y la consulta calculada por el SRI (la similitud también puede ser aplicada entre consultas, así como entre los documentos). Sin embargo, todos los documentos recuperados no satisfacen las necesidades del usuario, es decir, no todos los documentos que aparecen en la lista son relevantes para el usuario. Además, la posición de los documentos relevantes en la lista es muy importante para el usuario. Por lo tanto, una buena situación es cuando los documentos relevantes aparecen juntos y en la parte superior de la lista (es decir, los documentos relevantes no deben estar ampliamente dispersos). Esta propiedad es denominada

precisión[1] en RI. Para ilustrar esto, podemos suponer el siguiente escenario: para una consulta *q*, hay un subconjunto de documentos relevantes *sdr* para una colección de documentos. Un SRI S_1 suministra una lista de documentos donde *sdr* aparece en la parte superior de la lista. De forma similar, otra SRI S_2 produce una lista de documentos donde *sdr* aparece en la parte inferior de la lista. Estas dos listas de documentos entregadas por su respectivo SRI, se encuentra representado en la Figura 1.1, además de representar el subconjunto de documentos relevantes *sdr* en cada lista de los documentos. Cuando comparamos ambos sistemas, S_1 proporciona una mejor precisión que S_2 porque el mismo *sdr* aparece en la parte superior. Un importante y difícil desafío para un SRI es proporcionar la mejor precisión posible. Hoy en día, los SRI más utilizados son los Motores de Búsqueda Web (MBW). Los MBW responden a millones de consultas por día en colecciones que implican billones de documentos. Además, con el crecimiento sostenido en el número de documentos, la tarea de encontrar los documentos relevantes para una consulta enviada por un usuario puede no resultar sencilla. En resumen, el desafío más relevante para un SRI es proporcionar la mejor precisión posible.

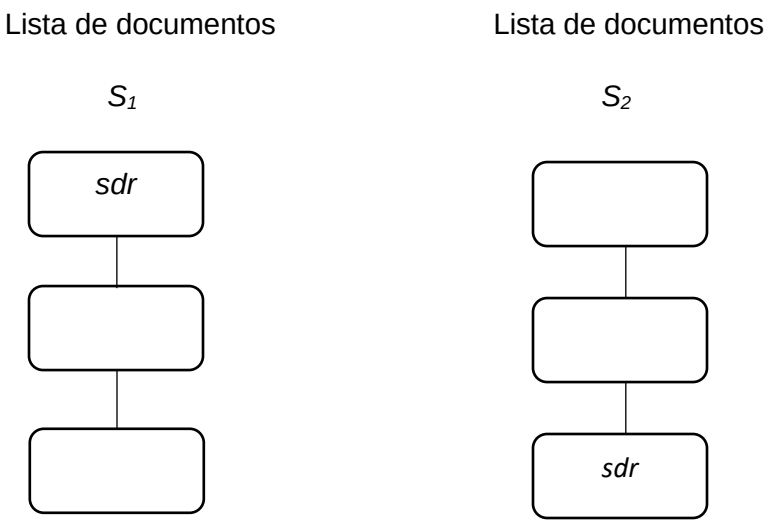


Figura 1.1: Representación de dos listas de documentos entregados por los SRI S_1 y S_2, en respuesta a una consulta *q*.

[1] La precisión es la fracción de documentos recuperados que son relevantes para la necesidad de información del usuario.

Existe una gran variedad de contribuciones en RI, que abordan diferentes perspectivas como funciones matemáticas, indexación, modelos formales y retroalimentación relevante que pueden ser encontradas en la literatura (Baeza-Yates y Ribeiro-Neto, 1999; Rijsbergen, 1979; Sparck Jones y Willett, 1997a). Sin embargo, pocas de estas contribuciones toman ventaja de las búsquedas realizadas anteriormente.

El clustering (agrupamiento) y los archivos invertidos son las estructuras de datos más utilizadas para construir los índices y el acceso a los documentos. Los archivos invertidos ayudan a responder eficientemente a las consultas de palabras claves mientras que el clustering permite construir grupos de documentos similares. El clustering en RI ha sido usado para mejorar la eficiencia[2] (en tiempo) y la efectividad[3] (en precisión) de los SRI. En general, dos mayores categorías de clustering son fácilmente identificables: el clustering estático y el clustering de post-recuperación. Por un lado, el clustering estático es la aplicación tradicional del método clustering en una colección de documentos. En cambio, el clustering de post-recuperación incluye la información desde la consulta hacia el clustering de documentos. El clustering de una consulta es un tipo de clustering de post-recuperación, el cual está dedicado a encontrar consultas similares en un clúster. La similitud entre las consultas está relacionada con la intersección entre los términos de una consulta. Típicamente, las funciones de similitud como Jaccard o coseno son usadas para medir la similitud entre las consultas (Salton and McGill, 1983). En la literatura, existen dos tipos de clustering de una consulta que son identificables basado en la medida de la similitud. El primero, reúne aproximaciones que no tienen en cuenta la medida de similitud en el contexto en el que estos son usados. El segundo tipo está formado por las propuestas que tengan en cuenta el contexto en el que se utiliza la similitud. En cuanto al segundo tipo, el primer enfoque fue propuesto por Tombros y Rijsbergen (2004), donde indica que la similitud entre los documentos no son estáticos y cambian explícitamente acorde a la consulta. Esas medidas son denominadas medidas de similitud sensibles a la consulta (Query-Sensitive Similarity Measures) (QSSM). Resultados empíricos utilizando estas medidas muestran un aumento significativo en el número de documentos relevantes usando la prueba del N-vecino más cercano (N-Nearest Neighbour test). Hearst y Pedersen (1996), llevaron a cabo varios experimentos con el objetivo de analizar el rendimiento de este tipo

[2] Capacidad para realizar o cumplir adecuadamente una función con el menor número de recursos posible.
[3] Capacidad de cumplir con los objetivos propuestos de forma satisfactoria independientemente de los recursos utilizados.

de enfoque. Los resultados finales de los autores confirman que los documentos recuperados por una consulta, tienden a aparecer en los mismos clústeres.

Inspirado en estas investigaciones, las cuales utilizan QSSM, se pretende utilizar una nueva medida de similitud denominada medida de similitud de consulta-documento (Query-Document Similarity Measure) (QDSM), propuesta por Gutiérrez-Soto (2016). Básicamente la idea es almacenar las consultas pasadas (las cuales ya han sido procesadas por el SRI) junto con la lista de los documentos recuperados (de ellos podemos saber cuáles son relevantes y no relevantes una vez efectuada la recuperación). Usando las consultas pasadas y la posición de los documentos en la lista recuperada podemos utilizar QDSM, la cual toma en consideración la distancia del coseno entre ambas consultas pasadas, más la distancia proveída por el algoritmo LCS (Longest Common Subsequence) entre los documentos relevantes de ambas listas (para cada par de consultas). El LCS es un algoritmo que calcula la subsecuencia común más larga de dos arreglos (pueden ser cadenas), es decir, es la subsecuencia que se encuentra tanto en el primer arreglo, como en el segundo, y es de longitud máxima, además las subsecuencias no necesitan tener posiciones consecutivas en la secuencia original.

Así, al ejecutar una nueva consulta (la cual ha sido procesada por el SRI), esta será comparada con los clústeres creados usando QDSM, los cuales debería de proveer una mejor efectividad al momento de recuperar los documentos (una mejor precisión). Por consiguiente, con este proyecto de título lo que se busca obtener son resultados empíricos que contrastan la medida QDSM con la medida del coseno, usando al menos un algoritmo de clustering ampliamente usados en RI.

1.1 Objetivos del proyecto de título

El objetivo principal de este proyecto de título es comparar a través de resultados empíricos, la medida de distancia QDSM con la distancia del coseno sobre algoritmos de clustering, con el propósito de mejorar la efectividad en la recuperación de los documentos, los documentos relevantes tienden a ser más similares los unos a los otros en comparación a los no relevantes, lo cual se reduce en mejores clústeres.

Para lograr este objetivo, se realiza un estudio del estado del arte y de los algoritmos más utilizados en el contexto de RI, además de encontrar un dataset adecuado para poder implementar un algoritmo de clustering, para luego obtener los resultados empíricos que contrastan la medida QDSM con la medida del coseno.

1.2 Descripción de capítulos

Este proyecto de título está organizado en 4 capítulos (incluyendo éste capítulo). A continuación se presenta un resumen de los contenidos de los siguientes capítulos.

Capítulo 2: En este capítulo se da a conocer los conceptos principales de la recuperación de la información con el propósito de tener una terminología básica para desarrollarlo en los siguientes capítulos.

Capítulo 3: Este capítulo presenta el concepto de clustering en RI, además se presentan los trabajos relacionados a ella, junto con los métodos de clustering que se van a implementar. Además en este capítulo se describe el entorno experimental que se llevó a cabo, también se muestran los resultados experimentales junto con sus respectivas conclusiones.

Capítulo 4: Este capítulo se presenta las principales contribuciones de este trabajo tomando en cuenta los objetivos, además de las contribuciones que se pueden llevar a cabo en un futuro.

Capítulo 2: Recuperación de la información

La recuperación de la información (RI), es una rama de las ciencias de la computación, que se encarga de la organización, estructuración, análisis, almacenamiento y búsqueda de información. Varias definiciones de RI pueden ser encontradas en la literatura, entre ellas Baeza-Yates y Ribeiro-Neto (1999) aportan lo siguiente:

" … *El sistema de RI de alguna manera debe interpretar los contenidos de los elementos de la información (documentos) en una colección y clasificarlo de acuerdo a un grado de relevancia para la consulta del usuario. Esta interpretación del contenido de un documento implica la extracción de información sintáctica y semántica del texto del documento*… (Baeza-Yates y Ribeiro-Neto, 1999) ".

Del párrafo anterior, tres aspectos importantes deberían ser considerados en esa definición. Primero, hay un usuario con una necesidad de información (consulta). Segundo, una colección de documentos con la que se compara esta consulta. Tercero, la lista de los documentos proporcionados por un SRI en respuesta a la consulta. Así, un SRI tiene como objetivo suministrar un conjunto de documentos el cual satisface la necesidad de información de un usuario. Un usuario que quiere buscar información, expresa su requerimiento a través de una consulta, la cual es entregada al sistema. Con el objetivo de procesar esta consulta, el SRI tiene una representación interna de sí misma. Una vez formada esta representación de la consulta, es contrastada con la colección de documentos. Como resultado, una lista clasificada de documentos es proporcionada al usuario. Así, el usuario puede verificar la lista final de los documentos y si los documentos no son suficientemente relevantes para el usuario, puede enviar una nueva consulta o reformular la consulta.

Varios modelos de RI han sido desarrollados, donde es posible encontrar no sólo una descripción de cuántos documentos y consultas son representadas sino también la manera en cómo se lleva a cabo el proceso de matching[4] entre un documento y una consulta. Los modelos más generalizados en la RI son el Booleano, el Espacio Vectorial (Salton, 1971;

[4] El matching estima si un documento es relevante de acuerdo a la consulta enviada por un usuario y es representado por una puntuación.

Salton et al., 1975), el Probabilístico (Robertson et al., 1981), y el Lógico (Van Rijsbergen, 1986).

El modelo espacio vectorial es el más utilizado en los SRI, por lo cual se profundizará más en este modelo que en los otros. Este modelo representa documentos y consultas a través de vectores, basados en las frecuencias de términos en los documentos. La medida de similitud entre una consulta y un documento corresponde al coseno entre los términos de una consulta y los términos de un documento.

Los primeros SRI aparecen con el propósito de apoyar la búsqueda automatizada de material de bibliografías por los usuarios. Como respuesta al crecimiento exponencial de la información disponible en formato electrónico, los SRI se ampliaron en otros ámbitos. Así, es posible encontrar una amplia gama de investigaciones basadas en los SRI.

Algunas investigaciones se refieren a la recuperación de información basada en texto (Salton et al., 1993; Liu y Croft, 2002). Otras investigaciones basadas en XML, con técnicas de estudio que permiten la recuperación de segmentos de información a partir de datos estructurados (Fuhr y Lalmas, 2007). Así, la World Wide Web se ha convertido en el medio más popular utilizado por la gente para buscar información. Los SRI se han desarrollado en Internet como Motores de Búsqueda Web (MBW), que indexan una gran cantidad de información y promueven una forma sencilla de proporcionar acceso a la información.

Hoy en día, se han realizado muchos trabajos con el objetivo de explotar las características que caracterizan las páginas web, como la estructura HTML, la popularidad de las páginas web y las estructuras de hipervínculos, entre otros (Bharat y Henzinger, 1998; Kleinberg, 1999). En Broder (2002), se presenta una taxonomía que clasifica las consultas web en tres categorías principales: navegacional, informativa y transaccional. A grandes rasgos, las consultas de navegación se refieren a cómo los usuarios pueden acceder a una página o documento en particular. En las consultas informativas, la información de las consultas proporciona al usuario información relacionada con un tema en particular. En este tipo de consulta, el usuario puede estar interesado en más de un solo documento. Por último, las consultas transaccionales abordan los servicios de localización con los que el usuario debe interactuar.

2.1 Representando documentos y consultas

Generalmente, los documentos son transformados por un SRI desde la forma original a una representación interna, este proceso se denomina indexación. El propósito de este proceso es proporcionar una representación de la información lo más exacta posible. Para lograr este objetivo, se asigna un conjunto de características de indexación para cada documento. Las características más relevantes de un documento corresponden a una lista de palabras, que permiten discriminar entre ellas, estos son conocidos como *términos*. De hecho, un documento no sólo está representado por esta lista de *términos*, sino que también se accede por los *términos* que pertenecen a su lista. Con el fin de obtener un alto nivel de representación, los *términos* deben estar involucrados en el proceso de indexación (Lewis y Jones, 1996), el cual permite recuperar las unidades frasales, o el uso de métodos lingüísticos, semánticos y basados en el conocimiento.

Antes de la indexación se produce un proceso de normalización. El objetivo de este proceso es proporcionar sólo términos relevantes. Por ejemplo, las palabras con altas frecuencias en el documento (Stop-words) no serán consideradas en la indexación (Rijsbergen, 1979). Las Stop-words son palabras conocidas como artículos (por ejemplo, el, la, los, las) y preposiciones (por ejemplo, en, por, de, para). La principal ventaja de este proceso es reducir el volumen de texto hasta un 50 por ciento. Otro proceso antes de la indexación consiste en eliminar los sufijos de las palabras restantes del texto de entrada. Para ello, se aplica un algoritmo de derivación para reducir las palabras a una forma de raíz común (conocida como raíz). Por ejemplo, el algoritmo de derivación reduce las palabras 'cardiovascular' y 'cardiología' a la palabra cardio, que estará en el vocabulario de los términos indexados.

Cada palabra que se selecciona como una característica de indexación para un documento en particular puede considerarse discriminatorio entre ese documento y todos los demás documentos en la colección. Para cuantificar el poder discriminante de los términos, se han utilizado métodos que asignan pesos numéricos a los términos. Luhn (1958), asoció el poder discriminatorio de un término índice con su frecuencia de ocurrencia dentro de un documento denominado frecuencia de término (*tf*).

Con el fin de obtener términos que sean representativos de un documento contra otros documentos que pertenecen a la colección (al vocabulario de los términos indexados), es

necesario esforzarse en la presencia de términos no frecuentes en contraste con los términos que aparecen en todos los documentos. Para lograr este objetivo, se introdujo el concepto de frecuencia inversa de documentos (*idf*) (Jones, 1972). Así, el peso de un término en un documento se incrementa si aparece más a menudo en ese documento. Por el contrario, el peso de un término en un documento *idf* disminuye si aparece con frecuencia en otros documentos. Por lo tanto, para una colección de documentos que contiene *N* documentos (podemos suponer una colección con *N* documentos), si el término *i* se produce en n_i documentos, entonces el peso *idf* de un término es dado por log $\left(\frac{N}{n_i}\right)$.

Una función de ponderación del término corresponde a la combinación de los pesos *tf* e *idf*, que comúnmente se conoce como peso *tf-idf* (Salton (1971)):

$w_{ij} = \frac{log(freq_{ij}+1)}{log(lenght_j)} \log(\frac{N}{n_i})$

w_{ij} = peso *tf-idf* del termino i en documento j

$freq_{ij}$ = frecuencia del término i en documento j

$length_j$ = largo (en términos) de documento j

N = número de documentos en la colección

n_i = número de documentos que el término i está asignado

Salton y Buckley (1987) ofrecen una visión general de varios esquemas de ponderación así como medidas de evaluación en el dominio de RI.

Los SRI utilizan tres categorías principales de estructuras de datos para almacenar los términos, índices lexicográficos (índices que se ordenan), estructuras de archivos en clústeres e índices basados en el hashing. Sin embargo, las estructuras de datos más utilizadas por las MBW corresponden al archivo invertido (Ouksel, 2002). Esta estructura corresponde a una lista, que contiene los términos representativos de la colección de documentos, por lo que los términos que pertenecen en la consulta coinciden con palabras clave (términos) que están en la lista. Por lo tanto, es factible localizar inmediatamente todos los documentos en la colección que contienen esta palabra clave (Rijsbergen, 1979).

2.2 Operaciones de una consulta

Un SRI tiene como objetivo principal apoyar al usuario en la búsqueda de aquellos documentos que puedan satisfacer su necesidad de información. Los requisitos de información de un usuario se expresan de una manera que pueda ser entendida por el SRI. Esta manera de expresar los requisitos del usuario se denomina "consulta".

La formulación de una consulta puede realizarse para algunos SRI mediante el uso de operadores booleanos. Un ejemplo de este tipo de consulta podría ser:

((Simulación Y algoritmos) NO aleatorios)

La principal desventaja de los SRI booleanos es que sus resultados son poco intuitivos para los usuarios no experimentados, por lo que la formulación de una consulta ad-hoc puede ser ineficaz (Sparck Jones y Willett, 1997a). Como consecuencia, estos sistemas han sido reemplazados por sistemas que proporcionan la formulación de una consulta a través del lenguaje natural sin el uso de operadores específicos. Estos sistemas se basan en el mejor match o búsqueda de similitud, que se calcula para cada documento, así como para cada consulta. El matching entre documentos y consultas se presenta en la Sección 2.3.

De la misma manera que los documentos, las consultas se procesan antes de ser indexadas, es decir, el procesamiento léxico y la ponderación de términos son ejecutados por los SRI.

2.3 Matching entre documentos y consultas

A través del modelo booleano, un SRI encuentra el subconjunto de documentos de la colección completa, en la que cada documento que pertenece al subconjunto tiene al menos un término de la consulta enviada al SRI, el cual se le presenta al usuario de una manera no clasificada. Por otra parte, es factible encontrar sistemas que tengan mejores métodos de comparación y proporcionar el resultado de acuerdo a una puntuación de relevancia sobre la pertinencia del documento con respecto a la consulta. Así, un SRI suministra una lista clasificada de documentos, ordenados de manera decreciente (considerando la del documento más similar a la consulta hasta el menos similar) para el usuario.

En el modelo espacio vectorial (Salton y McGill, 1986), tanto los documentos como las consultas se representan como vectores en un espacio multidimensional. La representación espacial corresponde a los términos indexados de la colección de documentos.

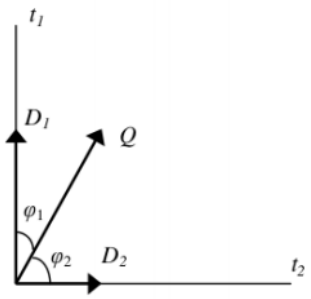


Figura 2.1: Representaciones de documentos y consultas en el modelo vectorial.

Un ejemplo de modelo vectorial se presenta en la Figura 2.1. En este modelo se representa un documento y una consulta. En este ejemplo, el espacio vectorial se muestra en dos dimensiones. Así, la dimensión corresponde a los términos t_1 y t_2. Los documentos D_1 y D_2 se representan en el espacio utilizando cada uno de los pesos de términos del documento como coordenadas. Los pesos corresponden a pesos de frecuencia de pesos de términos: $D_1 = t_1$, y $D_2 = t_2$. Por otra parte, la consulta Q se representa como $Q = t_1, t_2$. Además, los ángulos entre los vectores de D_1 - Q y D_2 - Q se presentan en la Figura 2.1.

En este espacio se pueden definir medidas para cuantificar la similitud entre las consultas y los documentos. El mejor match entre una consulta particular y un conjunto de documentos corresponde al documento más cercano con respecto a la consulta acorde con la medida de similitud. Una amplia gama de fórmulas de distancia se pueden encontrar en la literatura de RI. Un detalle de la mayoría de ellos se puede encontrar en libros tales como Rijsbergen (1979) y Salton y McGill (1986).

Una manera simple de comparar un documento con respecto a una consulta es contando el número de términos comunes entre ellos. De esta manera, podemos asumir que ambos están representados como vectores de longitud n (donde n es el número de términos en la colección). Por lo tanto, tenemos la siguiente medida:

$$\mathrm{sim}(\mathrm{D},\mathrm{Q}) = \sum_{i=1}^{n} D_i Q_i \qquad (2.1)$$

Esta medida se conoce como la función de matching de nivel de coordinación. En contraste, otras medidas de similitud pueden ser normalizadas de acuerdo con la longitud del documento y la consulta.

La medida más utilizada por los SRI corresponde a la distancia del coseno. La razón de su popularidad se debe a la interpretación geométrica del modelo vectorial.

$$\text{sim (D,Q)} = \frac{\sum_{i=1}^{n} D_i Q_i}{\sqrt{\sum_{i=1}^{n} (D_i)^2 \sum_{i=1}^{n} (Q_i)^2}} \qquad (2.2)$$

La medida de coseno es la función que proporciona los ángulos entre los vectores del documento y la consulta (ver la ecuación 2.2), y cuyos rangos de valores están comprendidos entre:

- 0 (los vectores de los documentos y la consulta forman un ángulo de 90°, es decir, son diferentes).
- 1 (los vectores del documento y la consulta forman un ángulo de 0°, es decir, son iguales).

En la Figura 2.1, la similitud entre los documentos D_1 y D_2 es 0, porque el ángulo entre los dos vectores es de 90°. Esto significa que no tienen términos en común.

2.4 Evaluación de los SRI

Una gran variedad de metodologías se ocupa de la evaluación de los SRI. La tarea de la evaluación implica una complejidad intrínseca porque implica temas de diferentes áreas de investigación como cognición, estadística, diseño experimental, diseño de sistemas, interacción humana-computador, entre otros. En esta sección se presenta un resumen de los aspectos envueltos en la evaluación con énfasis en los aspectos centrales.

Se pueden evaluar varios aspectos del proceso de RI. Estos aspectos deben estar relacionados, entre otros, con la velocidad de un SRI, las interfaces, el nivel de interacción del usuario final y el formato de la información presentada a los usuarios. Las medidas de efectividad más utilizadas corresponden a la precisión y la recuperación. La precisión representa la fracción de los documentos recuperados que son relevantes, mientras que la recuperación es la proporción de los documentos relevantes que se han recuperado. De

acuerdo con la Figura 2.2, la precisión y la recuperación en la ecuación 2.3 y 2.4, pueden definirse como:

- *A*: Número de documentos relevantes.
- *B*: Número de documentos recuperados.

$$\text{Precisión} = \frac{|A \cap B|}{|\mathrm{B}|} \tag{2.3}$$

$$\text{Recuperación} = \frac{|A \cap B|}{|\mathrm{A}|} \tag{2.4}$$

Comúnmente, estas medidas son expresadas en un rango entre 0 y 1, aunque también se puede expresar en porcentajes como en Chowdhury (2010).

A partir de las definiciones anteriores, es necesario conocer el número total de documentos relevantes en una colección para calcular la recuperación. Sin embargo, no siempre es posible calcular esta medida, ya que implica no sólo una cantidad considerable de esfuerzo, sino también de tiempo. Por otro lado, las colecciones que permiten calcular estas medidas pueden ser tanto comerciales como de libre disponibilidad. Este tipo de colección de documentos se puede encontrar junto con un conjunto de consultas y juicios de los usuarios (sobre la relevancia de un documento con respecto a una consulta). Mediante el uso de estas colecciones, los investigadores de RI tienen la oportunidad de evaluar sus propuestas con resultados empíricos y comparar sus resultados con otros sistemas y propuestas.

En la Figura 2.2, se muestra un equilibrio entre el gráfico de recuperación-precisión (R-P). Esto significa que la precisión y la recuperación están relacionados de manera inversa. Cada vez que se incrementa esa precisión, la recuperación se reduce y viceversa. Esto está representado en casi todos los libros de texto y manuales de RI.

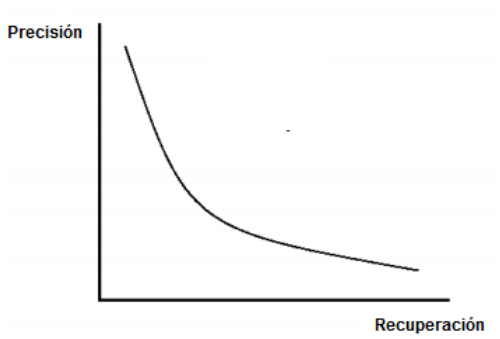


Figura 2.2: Un gráfico de recuperación-precisión.

Con el constante aumento de grandes colecciones, especialmente con la aparición de las Conferencias de Recuperación de Texto (TREC) (Harman, 1993), las cuales contienen miles de millones de documentos (implica el uso de muchos gigabytes de espacio en disco), se ha vuelto casi imposible proporcionar juicios exhaustivos sobre la pertinencia de los documentos (Tombros et al., 2002).

Con el objetivo de abordar este problema, una técnica ampliamente utilizada y bien conocida en estos casos se denomina pooling (Harman, 1993). El principio de esta técnica se basa en la combinación de los documentos de mayor clasificación de varios SRI, donde una determinada consulta se presenta en cada uno de ellos. De este modo, los juicios de los usuarios corresponden a un conjunto combinado de los SRI. Podemos deducir que esta técnica es efectiva, cuando los documentos relevantes recuperados son representativos de todos los documentos relevantes disponibles en los SRI (Harman, 1993).

El tipo de evaluación establecido anteriormente se basa en juicios proporcionados por algunos jueces expertos, basados en la relevancia actual (o relevancia algorítmica). En Schamber et al. (1990) y Barry (1994), la relevancia corresponde a un concepto multidimensional, que implica una sola dimensión. Inspiradas en este trabajo, se han presentado otras dimensiones sobre el concepto de relevancia a través de las metodologías de evaluación con respecto a la utilidad proporcionada por los SRI (Borlund e Ingwersen, 1997; Reid, 2000).

Es importante destacar que las conclusiones extraídas de los resultados empíricos de las investigaciones de RI son un tema que involucra un gran número de factores como la elección de medidas adecuadas de rendimiento, la validación estadística de resultados

empíricos, entre otros. Para abordar estos problemas, se ha propuesto una metodología para obtener inferencia científica a partir de experimentos en RI (Keen, 1992).

2.5 Resultados de búsquedas pasadas

El uso de los resultados de búsqueda pasadas para responder a una nueva búsqueda no es reciente. En la literatura de RI (Fitzpatrick y Dent, 1997; Hust, 2004; Cetintas et al., 2011), se pueden encontrar varias propuestas que tratan del uso de consultas históricas para mejorar los resultados de la búsqueda. La repetición (es decir, una consulta repetida, es una consulta que se ha presentado previamente en un SRI) y la reformulación de una consulta (es decir, una consulta modificada, es una consulta en que se añade o eliminan términos de una consulta presentada previamente en un SRI) tienen un considerable éxito en la explotación de la información disponible dentro de los archivos web. Los archivos Logs (es decir, archivos que guardan y registran el momento exacto en que se realiza una consulta por parte de un usuario) se han estudiado ampliamente considerando las consultas repetidas y la reformulación de consultas (es decir, agregando o eliminando términos). Además, las consultas históricas (es decir, las consultas históricas son consultas que se han presentado previamente en el SRI) se han utilizado en comunidades similares en el dominio de los sistemas de recomendación. Por otra parte, las propuestas en RI, que no se basan en archivos Log, también han estudiado el uso de consultas similares con el objetivo de mejorar la nueva búsqueda. Sin embargo, ambas propuestas se dedican principalmente a explotar las consultas repetidas más que las consultas similares.

Capítulo 3: Clustering

3.1 Clustering de documentos para la Recuperación de la Información (RI)

La tarea del clustering (agrupamiento) ha sido llevada a cabo durante mucho tiempo por los seres humanos (Willett, 1988). El clustering es una técnica estadística multivariable que busca reunir grupos de objetos de tal manera que los objetos en el mismo grupo son similares en un espacio, el cual suele ser multidimensional. Los grupos de los objetos están compuestos de tal manera que los objetos en el mismo clúster son similares entre sí y distintos a los objetos de otros clústeres (Gordon, 1987).

Las técnicas de análisis de un clúster han sido ampliamente aplicadas en diferentes campos de investigación como las ciencias médicas, ciencias sociales y las ciencias de la ingeniería, entre otras (Anderberg, 1973). De hecho, varias aplicaciones del análisis de clústeres se pueden encontrar en muchas áreas de investigación. Hoy en día, esta tarea ha sido totalmente automatizada gracias a los avances en la tecnología (Willett, 1988). Además, la aplicación de análisis de clústeres se ha llevado a cabo en RI no sólo para el clustering de términos, sino también para el clustering de documentos. El clustering de documentos se aplica sobre la base de términos compartidos entre documentos. El clustering de términos proporciona una representación grupal de términos que pertenecen a documentos o consultas.

El clustering de documentos opera normalmente sobre la base de la noción de similitud entre documentos. El conjunto de términos compartidos entre un par de documentos se utiliza típicamente como una indicación de la similitud del par. Según Rijsbergen (1979), uno de los primeros investigadores en sugerir el uso del clustering automático para RI fue Good (1958). El clustering de documentos se ha aplicado tradicionalmente de forma estática a una colección completa de documentos antes de aplicar la consulta, esta forma de aplicar el clustering se denomina clustering estático. Una aplicación alternativa de clustering es para agrupar documentos que han sido recuperados por un Sistema de Recuperación de la Información (SRI) en respuesta a una consulta. Esta forma de aplicar el clustering se denomina clustering de post-recuperación (Preece, 1973). Bajo el clustering de post-recuperación, es probable que los grupos de documentos resultantes sean

diferentes para diferentes consultas. Dos grandes tipos de clustering de documentos se han utilizado principalmente en RI, y estos son el aglomerativo y el jerárquico.

En el clustering aglomerativo, los documentos están representados por un vector en un espacio de *n* dimensiones, donde *n* corresponde al número de términos que componen el vocabulario de indexación de la base de datos. Así, dado un conjunto de *N* documentos, el clúster aglomerativo genera una organización simple de *k* clústeres mutuamente exclusivos, donde *k* se proporciona previamente, o se determina como parte del método de clustering. Usualmente, los requisitos computacionales de este método son bajos, los cuales están comprendidos entre $O(N)$ a $O(N\log N)$ (en tiempo) para el clustering de *N* documentos (Willett, 1988). Como resultado, los métodos aglomerativos fueron muy populares en un comienzo en RI (Salton, 1971). La idea básica detrás de este método es elegir alguna partición inicial de los documentos, después agregar los objetos de forma incremental a los clústeres para obtener una mejor partición (Anderberg, 1973).

Existen muchas aplicaciones de clustering jerárquico en la literatura de RI, que se han llevado a cabo considerando el uso de términos simples. En cambio, propuestas recientes con representaciones de documentos a través de unidades frasales tratan con diferentes niveles de análisis lingüístico. Este tipo de clustering ha sido mayoritariamente aceptado en la comunidad de RI (Willett, 1988). Comúnmente, cada documento *D* se representa como un vector $D = \{d_1, d_2,..., d_n\}$, donde *d* es el término y *n* es el número de términos que componen el vocabulario de indexación de la colección de documentos. Todos los términos que componen el vocabulario de indexación se utilizan en la representación de indexación (Rijsbergen, 1979). Posteriormente, se obtiene un peso relativo sobre la importancia de cada término de acuerdo a la variable w_{dn}, considerando la colección completa de documentos para aumentar la eficacia (Salton y Buckley, 1987). Una vez que se obtiene una representación apropiada para el conjunto de documentos para el clúster, es necesario tener una medida según el grado de similitud para todos los posibles pares de documentos que pertenecen a este conjunto. Para lograr este objetivo, se puede aplicar un gran número de medidas que cuantifican la semejanza entre objetos para proporcionar una categorización. Se distinguen cuatro clases principales de medidas: asociación, disimilitud, probabilística y coeficientes de correlación (Sneath y Sokal, 1973). La mayor parte de la literatura trata de la asociación y la disimilitud, mientras que el uso de coeficientes de correlación y probabilidades en el clustering de documentos es limitado.

Es importante mencionar que para este proyecto, los experimentos que se van a realizar, implementarán el clustering jerárquico. Este tipo de clustering es apropiado para implementarlo en este proyecto, ya que existe una amplia información en la literatura y es bien aceptado en la comunidad de RI (Tombros, 2002).

3.2 Trabajos relacionados

Esta sección presenta las propuestas de la literatura que tratan sobre clustering de consultas, los cuales toman en cuenta el contexto en el que se utiliza la similitud.

Una propuesta pionera fue presentada por Tombros et al. (2002) mediante la utilización del clustering jerárquico de una consulta específica con el objetivo de mejorar la eficacia en el proceso de recuperación. Para esto los autores han estudiado un conjunto de experimentos. La suposición de los autores, es que la jerarquía debe ajustarse a una consulta específica aumentando la probabilidad de poner los documentos relevantes a la consulta en clústeres cercanos. Específicamente, los autores han considerado dos aspectos principales de su investigación.

1) Estudiar progresivamente la variación de la eficacia óptima del clúster obtenida de la aplicación del clustering jerárquico a un mayor número de documentos de alta clasificación devueltos por una búsqueda de archivo invertido (IFS).
2) Comparar esta eficacia con la eficacia de un IFS.

En el entorno experimental se han considerado cinco colecciones de documentos CACM, CISI, LISA, Medline y TREC (WSJ). Además, se compararon cuatro métodos jerárquicos: Average Link, Ward, Complete Link y Single Link, mientras que en los experimentos se emplearon siete números diferentes de documentos de alta clasificación. Los resultados finales indicaron que no existe una variación estadísticamente significativa en la eficacia del clustering de una consulta específica para diferentes valores de los documentos de clasificación alta y que el clustering de una consulta específica supera significativamente al clustering estático para todas las condiciones experimentales. La conclusión principal de estos resultados es que proporcionan evidencia para la aplicación del clustering jerárquico de una consulta específica a RI basada en una mayor eficacia.

Un trabajo de Tombros y Rijsbergen (2004) propuso una visión axiomática donde los documentos relevantes tienden a ser muy similares entre sí, por lo tanto, estos deberían aparecer en el mismo clúster. La propuesta presenta el uso de medidas de similitud sensibles a las consultas (QSSM: Query-Sensitive Similarity Measure) cuyo propósito es unir un par de documentos, que tienen atributos que se expresan en la consulta. Las medidas de similitud sensibles a las consultas pueden ser definidas a través de una función con dos componentes:

1) El primer componente corresponde a una similitud convencional entre dos documentos.
2) El segundo componente toma en consideración una similitud común para tres objetos: un par de documentos y la consulta.

Los autores probaron tres medidas de similitud sensibles a la consulta: M_1, M_2 y M_3. Dos medidas (M_1 y M_3) usaron una función diferente para combinar similitudes estáticas y variables (M_1 usó un producto de las dos fuentes mientras que M_3 utilizó una combinación lineal). La tercera medida sólo tuvo en cuenta términos comunes entre documentos y en términos de la consulta (medida M_2). Se han utilizado seis colecciones de documentos en los escenarios experimentales. Para determinar el grado de separación entre los documentos relevantes y los no relevantes, se ha aplicado la prueba del N-vecino más cercano (N-Nearest Neighbour test), en particular la prueba 5-NN. Los resultados finales indican que las medidas M_1 y M_3 son siempre significativamente más efectivas que la medida del coseno y no dependen fuertemente del largo de la consulta. De lo contrario, la medida M_2 es sensible a las variaciones del largo de la consulta, pero a pesar de ello, también trajo mejoras significativas sobre la medida del coseno en un gran número de condiciones experimentales. La conclusión principal de esta investigación es que el uso de medidas sensibles a la consulta para el cálculo de relaciones entre documentos es altamente efectivo.

Similarmente, Hasanzadeh y Keshavarzi (2009) señalaron que los documentos relevantes tienden a ser más similares entre sí que los documentos no relevantes, por lo que tienden a aparecer en los mismos clústeres. Por lo tanto, el clustering de una consulta específica se investigó mediante el uso de una QSSM en un entorno experimental. En términos generales, el sistema toma una consulta y una gran colección de documentos como entrada. En el primer paso, los documentos relevantes se encuentran y se muestran como una lista clasificada de acuerdo a su similitud con la consulta. Posteriormente, algunos de

los documentos que se encuentran en la parte superior de la lista junto con la consulta se agrupan utilizando la QSSM. Para llevar a cabo los experimentos, los autores emplearon la colección de documentos TIPSTER y el algoritmo K-means. La medida de similitud considera la medida del coseno entre dos documentos relevantes y el coseno entre los documentos y la consulta. A partir de las conclusiones, los autores afirmaron que el clustering de una consulta específica mejora la eficacia de las listas de clasificación.

Con un objetivo diferente, Baeza-Yates et al. (2004) desarrolló un método, que proporciona una lista de consultas relacionadas para una consulta enviada a un motor de búsqueda. Las consultas relacionadas corresponden a consultas previamente enviadas. El método se basa en un proceso de clustering de consultas, donde se identifican grupos de consultas semánticamente similares. Las preferencias históricas de los usuarios registrados en los Logs de consultas se utilizan para crear los clústeres. El proceso de clustering se basa en una representación del vector de peso de términos de consultas obtenidas considerando las URL de la consulta que han sido clickeadas. El método presenta dos ventajas: En primer lugar, determina las consultas relacionadas y en segundo lugar, clasifica las consultas según un criterio de relevancia. Las consultas se clasifican teniendo en cuenta dos criterios: (a) la similitud entre las consultas que pertenecen al clúster y la consulta de entrada, y (b) el soporte, que mide cuantas respuestas de la consulta han atraído la atención de los usuarios. Finalmente, a través de la combinación de las medidas (a) y (b), es factible definir el interés de una consulta recomendada. Para probar esta propuesta, los autores llevaron a cabo un conjunto de experimentos. Se utilizaron archivos Log de un motor de búsqueda. Los resultados empíricos mostraron mejoras en la precisión media.

Una interesante clasificación del clustering de consulta fue propuesta por Fu et al. (2004) que comparó diferentes medidas de similitud de consultas. En esta clasificación se han propuesto tres categorías:

a) Propuestas basadas en el contenido.
b) Propuestas basadas en la retroalimentación.
c) Propuestas basadas en los resultados.

Las propuestas basadas en el contenido comparan los vectores de términos de la consulta. En palabras simples, los términos comunes se pueden utilizar para caracterizar los clústeres de consultas. Para esta categoría se utilizaron varias funciones de similitud como la similitud del coseno y la similitud de Jaccard, por mencionar algunos. Los autores

sugirieron que este método no es conveniente en el contexto de los motores de búsqueda, debido a la longitud promedio de los términos de la consulta. En contraste, las propuestas basadas en la retroalimentación usan las selecciones de los usuarios en los resultados de búsqueda como medida de similitud. Estas propuestas dependen de los documentos clickeados por los usuarios usando los archivos Log. Por lo tanto, dos consultas son similares si promueven la selección de documentos similares. Se mencionan dos desventajas: la primera se da cuando el número de documentos relevantes es alto, y la segunda desventaja se produce cuando hay pocos documentos comunes. Las propuestas basadas en los resultados calculan la similitud entre las consultas calculando la superposición en los documentos devueltos para las consultas. Los documentos relevantes deben aparecer al principio de las listas resultantes. El principal inconveniente de este tipo de propuesta es que consume un alto tiempo de ejecución para los sistemas de búsquedas en línea. En los escenarios experimentales, se han considerado las URL para obtener los resultados empíricos. Los resultados finales mostraron buenos resultados cuando las tres propuestas se utilizan al mismo tiempo.

Más recientemente, Saravanakumar y Moturi (2011) propusieron un framework para identificar y resumir la semántica de la consulta del usuario. Al principio, los resultados se clasifican en grupos potenciales, posteriormente el usuario elige un grupo para volver a clasificar el resultado final. Por lo tanto, este framework ofrece al usuario un servicio de búsqueda personalizado. El framework propuesto se compone de un módulo de preprocesamiento y módulos clustering. En el módulo de preprocesamiento, la consulta es dada por el usuario donde se eliminan palabras no relevantes de la consulta y se proporciona la semántica teniendo en cuenta el contexto. Los módulos de clustering son responsables de dar la semántica de este modelo. Es importante enfatizar que el historial del usuario se utiliza para personalizar la información. Los intereses del usuario se almacenan en el perfil del usuario para mejorar la búsqueda.

Un nuevo problema interesante fue presentado por Niederberger et al. (2012). Los autores entregaron una propuesta de clustering y visualización, con el cual es posible apreciar al mismo tiempo que el dataset está aumentando de tamaño. Se ha utilizado un conjunto de documentos en alemán. Este conjunto comprende entre 300 y 1000 documentos con un largo promedio de 54 términos. Los resultados experimentales muestran que este método supera los algoritmos estándar de clustering con respecto a la confiabilidad de la clasificación en tiempo real. Los autores utilizaron cuatro conjuntos de datos para realizar

los experimentos, los resultados mostraron que la propuesta de los autores es capaz de reconstruir los cuatro conjuntos de datos, en cambio, el algoritmo K-means y el clustering jerárquico son capaces de reconstruir un solo conjunto de datos. Al mismo tiempo, esta propuesta proporciona una visualización innovadora para el problema de la reducción de la dimensionalidad.

Diferente de las métricas tradicionales utilizadas en el contexto de clustering de RI, la similitud sensible a la consulta como métrica ha alcanzado una posición relevante en el entorno de clustering en RI. Na (2013) sugirió un framework probabilístico que define la similitud sensible a la consulta arraigado en una co-relevancia probabilística, donde la similitud entre dos documentos es proporcional a la probabilidad de co-relevancia para una consulta específica. Se consideraron dos casos para determinar la co-relevancia. Primero, la relevancia de que un documento es independiente de la relevancia de otros documentos. Segundo, la relevancia de un documento depende de otros. Con el objetivo de demostrar esta propuesta, se han ejecutado varios escenarios experimentales utilizando las colecciones TREC (Text Retrieval Conferences) y la prueba del vecino más cercano. Los resultados finales de los autores mostraron que la propuesta de la medida de similitud sensible a la consulta tiene una mejor similitud basada en los términos.

A continuación se dará a conocer los algoritmos de clustering, que se van a implementar para realizar los experimentos, siendo estos el Single Link, Complete Link y el Average Link. La elección de estos algoritmos de clustering jerárquicos se debe a la popularidad y uso de éstos en la literatura de RI (Tombros, 2002).

3.3 Single Link

El algoritmo de clustering Single Link, se basa en agrupar los clústeres de forma ascendente, de tal manera que en un clúster dos objetos son similares entre sí, más que otros objetos que están en otros clúster.

En cada paso de este algoritmo, combina dos clústeres separados por la distancia más corta entre ellos, además Hartigan (1975) describe que los clústeres son encadenados en formas alargadas, donde los objetos muy separados están unidos entre sí por una cadena de objetos cercanos. Además, si los clusters son largos con altas densidades de objetos

dentro de cada grupo, entonces el algoritmo Single Link será mejor que el resto de los algoritmos jerárquicos.

3.4 Complete Link

El algoritmo Complete Link es una definición completamente opuesta al algoritmo Single Link, puesto que, combina dos clústeres separados por la distancia más larga entre ellos. En este algoritmo, los clústeres que se forman tienden a ser más pequeños y unidos estrechamente, ésto se debe a la forma en que son unidos, todo lo contrario al algoritmo Single Link. En el algoritmo Complete Link puede ocurrir que el vecino más cercano de un documento se encuentre en un grupo diferente, sin embargo, los vecinos más cercanos mutuos estarán siempre en el mismo grupo (Voorhees, 1985).

3.5 Average Link

En el algoritmo Average Link la similitud entre dos clústeres es el promedio de la similitud entre todos los pares de documentos, de esta forma, un documento del par de documentos pertenece a un clúster y el otro documento a otro clúster.

Los clústeres formados por el algoritmo Average Link no son tan alargados como los del algoritmo Single Link, ni tan unidos como en el algoritmo Complete Link. En el algoritmo Average Link la similitud se basa en un promedio, debido a esto no se puede deducir la similitud mínima o máxima entre los documentos en un clúster (Voorhees, 1985).

Sneath y Sokal (1973), afirmaron que el algoritmo Average Link es el más preferible de los algoritmos jerárquicos. Sin embargo, Williams junto con sus compañeros de trabajo (1971a) hicieron una crítica la cual indica que era más probable con este algoritmo formar grupos “no conformistas” (grupos cuyos miembros sólo comparten la propiedad de que son diferente a todo los demás, entre ellos).

3.6 Medida de similitud entre consultas y documentos (QDSM)

En esta sección se describe la medida QDSM (Query-Document Similarity Measure) propuesta por Gutiérrez-Soto (2016) que está inspirada en QSSM. Esta propuesta toma en consideración la distancia del coseno entre ambas consultas pasadas (consultas ya realizadas o consultas históricas) más la distancia proveída por el algoritmo LCS (Longest Common Subsequence) entre los documentos relevantes de ambas listas (para cada par de consultas). El LCS es un algoritmo que calcula la subsecuencia común más larga de dos arreglos (pueden ser cadenas), es decir, es la subsecuencia que se encuentra tanto en el primer arreglo, como en el segundo, y es de longitud máxima, además las subsecuencias no necesitan tener posiciones consecutivas en la secuencia original. Con la medida QDSM se busca proveer una mejor efectividad al momento de recuperar los documentos (donde al mismo tiempo sería posible mejorar la precisión).

3.6.1 Definición formal de QDSM

Sea:

- q : una consulta pasada.
- d : un documento.
- q' : una nueva consulta.
- D : un conjunto de documentos.
- Q : un conjunto finito de consultas pasadas.
- Q' : un conjunto finito de nuevas consultas.

Definición 1: Sea $V_N(q)$ un conjunto de N documentos recuperados dado una q, utilizando una medida como coseno entre una consulta q y documentos.

Definición 2: Sea $A(q)$ un conjunto finito de todos los documentos relevantes para la consulta q, tal que $A(q) \subset V_N(q)$.

Definición 3: Sea $A'(q)$ un conjunto finito de todos los documentos no relevantes para la consulta q, tal que $A'(q) \subset V_N(q)$.

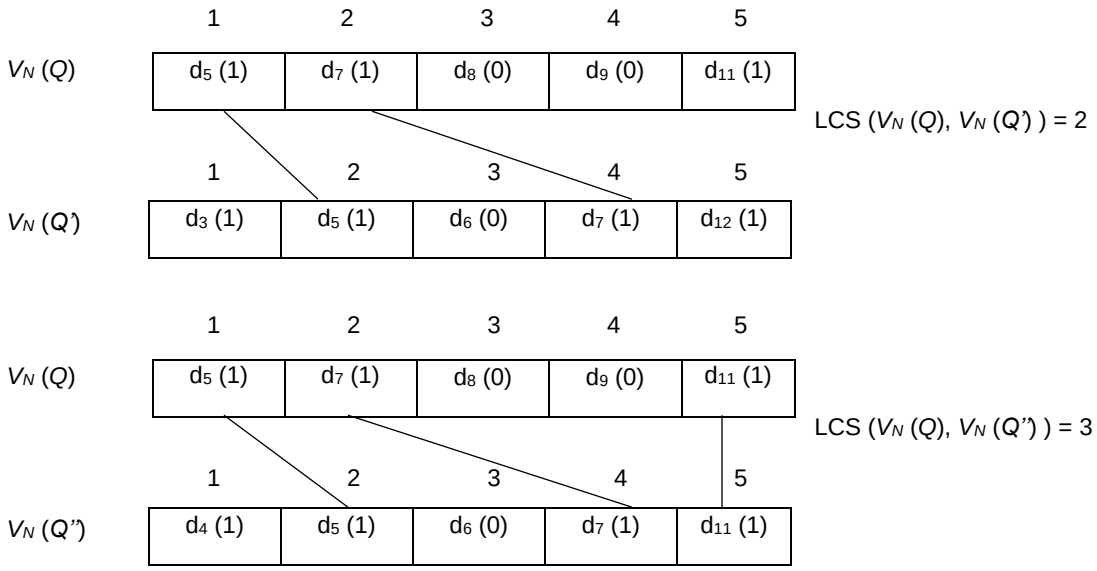
Dada una consulta *q*, tal que $q \in Q$, entonces es posible denotar c(*q*) = $q \cup V_N(q)$ como el conjunto de todos los documentos recuperados y la consulta *q*. Sea C = Uc(*q*) el conjunto de todos los documentos recuperados con sus respectivas consultas.

Definición 4: Sea LCS ($V_N(q)$, $V_N(q')$) $\rightarrow \mathbb{N}$, la función que es el resultado de aplicar el algoritmo de la subsecuencia común más larga entre dos listas de documentos para las consultas *q* y *q'*. Se aplica considerando sólo los documentos relevantes.

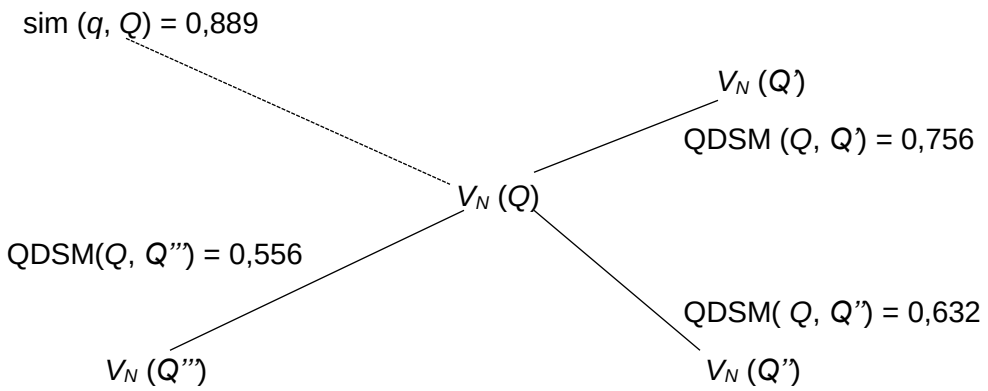
En la figura 3.1.a), los documentos relevantes contienen el valor 1, en el formato $d_{idDoc}(1)$, mientras que los documentos irrelevantes contienen el valor 0. La distancia proporcionada por el LCS ($V_N(q)$, $V_N(q')$) corresponde al match de los documentos relevantes.

Definición 5: QDSM (*q, q*') = $\frac{sim(q,q')+LCS(V_N(q),V_N(q'))}{1+Max(|A(q)|,|A(q')|)}$ corresponde a la medida de similitud entre consultas y documentos, que es usada para el clustering consulta-documento.

En la figura 3.1.b) $V_N(Q)$ corresponde a un centroide, así cada $V_N(q)$ corresponde a un objeto en un clúster. La QDSM (*q*, *q*') es la distancia usada para construir el clúster. La sim(*q*, *Q*) corresponde a la distancia entre una nueva consulta y el centroide. Básicamente, el clustering consulta-documento se apoya en dos pasos. Primero, la consulta ejecutada es almacenada con sus documentos (véase la Definición 1). Para construir los clústeres consulta-documento usando las consultas almacenadas, se usa la distancia QDSM (véase la Definición 5). En la figura 3.1.b), el centroide ($\overline{c_q}$), corresponde a V_N(Q), mientras que $V_N(Q')$, $V_N(Q'')$ y $V_N(Q''')$ son objetos del clúster. Es importante mencionar que cada vez que se añade un objeto a algún clúster, se actualiza el centroide. Así, desde el clúster con el centroide más similar (véase la Figura 3.1.b)) se utiliza para responder a una nueva consulta. Por lo tanto, para construir la lista de documentos para *q* usando el N-vecino más cercano, como N = 2 (véase la figura 3.1.b)), se utilizan los documentos relevantes de $V_N(Q')$ y $V_N(Q'')$ para responder a la nueva consulta.



a)



b)

Figura 3.1: La distancia usada para construir el clúster basado en centroides.

3.7 Entorno experimental

El entorno experimental en esta sección, está compuesto básicamente del dataset “AOL” (America Online). Este dataset contiene registros recolectados desde el 01/03/2006 hasta el 31/05/2006, que corresponde a un período de tres meses, además contiene 21.011.340 consultas diferentes en 36.389.567 transacciones con más de 650.000 usuarios. Cada registro contiene las siguientes piezas de información:

- AnonID: un número de identificación de usuario anónimo.
- Query: la consulta emitida por el usuario.
- QueryTime: la hora en que se envió la consulta para la búsqueda.
- ItemRank: Si el usuario hace click en un resultado de búsqueda, aparece el rank del elemento en el que hizo click, siendo este rank un número asignado por el motor de búsqueda, que consiste en la posición de la lista de los resultados en que el usuario hizo un click.
- ClickURL: si el usuario ha hecho click en un resultado de búsqueda, se muestra la parte del dominio de la URL en el resultado seleccionado, que representa un documento.

Si el usuario no hizo click en algún elemento luego de hacer la consulta, en el dataset no aparecerán las columnas de ItemRank y ClickURL y sólo se mostrarán las primeras tres columnas (AnonID, Query, QueryTime). En la Figura 3.2, se muestra un ejemplo de algunos datos que contiene el dataset “AOL” con su estructura ya mencionada.

AnonID	Query	QueryTime	ItemRank	ClickURL
142	rentdirect.com	2006-03-01 07:17:12		
142	westchester.gov	2006-03-20 03:55:57	1	http://www.westchestergov.com
217	lottery	2006-03-27 14:10:38	1	http://www.calottery.com
217	lottery	2006-03-27 16:34:59	1	http://www.calottery.com
1268	www.raindanceexpress.com	2006-03-18 20:13:01		
1326	holiday mansion houseboat	2006-03-29 17:14:01	5	http://www.everyboat.com
1326	back to the future	2006-04-01 17:59:28	1	http://www.imdb.com
1410	www.vanessascorner.com	2006-05-21 13:31:02		
2005	wnmu homepage	2006-03-01 00:46:55	2	http://www.wnmu.edu

Figura 3.2: Ejemplo de datos contenidos en el dataset “AOL”

En un primer paso de pre procesamiento, se filtraron las consultas que sólo contenían una URL, ya que esta URL representa un documento asociado a la consulta. En segundo lugar, se utiliza un documento que contiene las Stop-words, para poder eliminar términos que generalmente son artículos definidos e indefinidos (por ejemplo: el, la, los, las, un, unos, unas), símbolos, por mencionar algunos, que no son relevantes para la medida QDSM.

3.7.1 Documentos relevantes

Basado en el trabajo de Shen Jiang et al. (2008), se pueden clasificar los documentos asociados a las consultas de los usuarios si estos son “relevantes” o “no relevantes”. Primero, se deben organizar las consultas en sesiones, es decir, se establece que si una consulta se repite muchas veces en el mismo día realizada por el mismo usuario, se resumirá a una sola consulta que estará asociada con todos los documentos de las consultas repetidas.

Luego de organizar las consultas por sesiones, se consideran dos maneras de clasificar la relevancia de los documentos asociados a las consultas. La primera manera es considerar a todos los documentos asociados a la consulta como relevantes (*AllRel*) y la segunda manera consiste en sólo considerar el último documento asociado a la consulta en que el usuario hizo click como relevante (*LastRel*).

La Tabla 3.1 muestra la distribución de probabilidad como porcentaje de los documentos clasificados en *N* para determinar si un documento es relevante o no relevante. Esta *N* corresponde al ItemRank de cada URL asociada a la consulta.

N	*AllRel*	*LastRel*
≤ 20	43,65 %	57,02 %
≤ 120	51,06 %	62,78 %
≤ 300	53,95 %	64,93 %
> 300	47,05 %	35,07 %

Tabla 3.1 Probabilidad de que un documento sea relevante si su rank es *N* realizado por Jiang et al.

Para determinar la relevancia de un documento, se utiliza una probabilidad aleatoria, tomando en cuenta la Tabla 3.1, si el resultado de la probabilidad se encuentra dentro del porcentaje según el itemRank que le corresponda y si se toma en cuenta todos los documentos de la consulta o sólo al último, se determina si un documento es relevante. Con respecto a la distribución *AllRel,* se consideran dos variantes de éste. La primera variante denominada AllParcial considera que por cada documento asociado a la consulta junto con el resultado de la probabilidad aleatoria, se determina su relevancia considerando su propio itemRank de forma individual, es decir, todos los documentos de la consultas tienen una relevancia de forma individual utilizando la Tabla 3.1. En cambio, en la segunda variante denominada All se realiza un promedio de los itemRank de los documentos asociados de la consulta, a partir de este promedio y junto con el resultado de la probabilidad aleatoria, se puede determinar si todos los documentos son relevantes o no relevantes según la Tabla 3.1.

3.7.2 Aplicación de la medida de distancia del coseno

Luego de tener las consultas en sesiones y los documentos asociados con ellos, se crea una matriz que está compuesta por las consultas y todos los términos de todas las consultas, pero estos términos no pueden estar repetidos (ver Tabla 3.2). Si algún término no está contenido en una consulta, el valor con el que se presenta en la matriz es con un cero, en caso contrario, se representa con la cantidad en que el término aparece en la consulta, por ejemplo, en la Tabla 3.2, en la fila de la consulta Q_2 y en la columna del término “search”, esta casilla contiene un 2, debido a que ese término aparece dos veces en esa consulta. Con esta matriz las consultas pueden expresarse en función de unos vectores que recogen la frecuencia de aparición de los términos en las consultas. A continuación, se presentan las consultas Q_1, Q_2, Q_3 y Q_4 para ejemplificar en la Tabla 3.2, donde se muestra un ejemplo de cómo desarrollar la matriz de consultas y términos.

- Q_1: search spain football
- Q_2: search google search spain football cup
- Q_3: search spain
- Q_4: search spain google

	search	google	cup	football	spain
Q_1	1	0	0	1	1
Q_2	2	1	1	1	0
Q_3	1	0	0	0	1
Q_4	1	1	0	0	1

Tabla 3.2: Ejemplo de matriz entre términos y consultas.

De la misma forma se crea otra matriz que está compuesta por todos los documentos y los términos que son los mismos documentos pero sin que se repitan, para así determinar la frecuencia de aparición entre los documentos.

Después de tener la matriz de las consultas, se calcula el factor de peso para un término en una consulta que se define como combinación de la frecuencia del término, y la frecuencia inversa del documento. Este peso se calcula a través del logaritmo de la cantidad de filas de consultas que contenga la matriz tiendo como base a la cantidad de términos que aparecen en cada columna. Este valor del peso, reemplaza a los valores de la cantidad de los términos en la matriz, por ejemplo, el valor del peso del término “google” se reemplaza en todas las casillas de la columna que tengan un valor mayor a cero.

Siguiendo el ejemplo de la Tabla 3.2, a continuación en la Tabla 3.3, se calculan los pesos de la matriz y luego la modificación de ésta con los cálculos de los pesos en la matriz:

Log (search) = Log (4/4) = log (1) = 0

Log (google) = Log (4/2) = log (2) = 0,301

Log (cup) = Log (4/1) = log (4) = 0,602

Log (football) = Log (4/2) = log (2) = 0,301

Log (spain) = Log (4/3) = log (1,33) = 0,124

	search	google	cup	football	spain
Q_1	0	0	0	0,301	0,124
Q_2	0	0,301	0,602	0,301	0
Q_3	0	0	0	0	0,124
Q_4	0	0,301	0	0	0,124

Tabla 3.3: Ejemplo de matriz entre términos y consultas con los pesos calculados.

De la misma manera se calculan los pesos de los términos a la matriz de los documentos para luego tener los pesos en la matriz según corresponda.

Con la matriz de consultas y términos con sus pesos calculados, ahora corresponde sacar las similitud entre las consultas con la medida de distancia del coseno, para esto hay que multiplicar componente a componente de los vectores y sumar los resultados. El modo más sencillo de obtener la similitud es por medio del producto escalar de los vectores (es decir, multiplicando los componentes de cada vector y sumando los resultados). Siguiendo los ejemplos anteriores se calcula la similitud de la matriz de la Tabla 3.3:

Sim (Q_2, Q_1) = 0*0+0*0,301+0*0,602+0.301*0,301+0,124*0 = 0,090

Sim (Q_3, Q_1) = 0*0+0*0+0*0+0,301*0+0,124*0,124 = 0,015

Sim (Q_3, Q_2) = 0*0+0*0,301+0*0,602+0*0,301+0,124*0 = 0

Sim (Q_4, Q_1) = 0*0+0*0,301+0*0+0,301*0+0,124*0,124 = 0,015

Sim (Q_4, Q_2) = 0*0+0,301*0,301+0*0,602+0,301*0+0,124*0 = 0,090

Sim (Q_4, Q_3) = 0*0+0*0,301+0*0+0*0+0,124*0,124 = 0,015

En los cálculos de la similitud entre las consultas, se considera que el valor de la similitud para las mismas consultas es 1, por ejemplo entre Q_1 y Q_1, además si se calcula la similitud entre Q_2 y Q_1 será el mismo resultado que entre Q_1 y Q_2. También se calcula la similitud a

la matriz de los documentos, teniendo en cuenta las mismas consideraciones anteriormente señaladas. Con esta medida se puede observar las consultas que son más similares entre sí, ya que el valor fluctúa entre 0 y 1, el valor 0 significa que el par de consultas no tiene nada similar entre ellos siendo completamente distintos. En la Tabla 3.4 se muestra la matriz con la similitud entre las consultas ya calculadas:

	Q_1	Q_2	Q_3	Q_4
Q_1	1	0,090	0,015	0,015
Q_2	0,090	1	0	0,090
Q_3	0,015	0	1	0,015
Q_4	0,015	0,090	0,015	1

Tabla 3.4: Ejemplo de matriz con los cálculos de la similitud entre consultas.

3.7.3 Aplicación de la medida QDSM a la matriz de similitud entre consultas

Para la medida QDSM sólo se utiliza en la matriz de consultas, aplicando la Definición 5 de la medida QDSM, para cada par de consultas sin considerar las consultas repetidas como la similitud entre Q_1 y Q_1 y sólo la parte inferior de la diagonal de la matriz, ya que el orden de la similitud entre un par de consulta, como por ejemplo, Sim (Q_1, Q_2) es lo mismo que Sim (Q_2, Q_1). Con sólo considerar la parte inferior de la matriz se ahorran cálculos innecesarios de realizar si se considerara a toda la matriz. Esta medida se aplicará a las variantes All, AllParcial y Last, y así también habrá tres resultados distintos de QDSM, ya que según las variantes, toman en cuenta a los documentos de las consultas de maneras distintas.

Para la medida QDSM, además de requerir la similitud entre el par de consultas, se calcula subsecuencia común más larga (LCS) entre la relevancia de los documentos asociados en la consulta. Si un documento es relevante se le representa con la letra “R”, en cambio, si un documento no es relevante se representa con la letra “N”, pero en el cálculo de este algoritmo realmente sólo se toma en cuenta a los documentos relevantes, en este caso los

que están representados con la letra “R”. En el siguiente ejemplo se muestra un resultado del cálculo del algoritmo LCS:

- Q_1: tiene dos documentos, uno de ellos es relevante, su secuencia es {NR}.
- Q_2: tiene 6 documentos, tres de ellos son relevantes, su secuencia es {RNRRNN}.

LCS (Q_1, Q_2) = 1

Con el resultado anterior, se puede observar que sólo existe una subsecuencia común más larga entre los documentos relevantes, por esto el resultado es 1. Además de calcular el LCS, se necesita tener la cantidad máxima de documentos relevantes entre el par de consultas, siguiendo el ejemplo anterior, el máximo de documentos relevantes entre Q_1 y Q_2 es el valor 3.

A continuación en las Tablas 3.5, 3.6 y 3.7, se presentan algunos valores de cálculos del QDSM entre consultas, utilizando el dataset “AOL” y utilizando las variantes All, AllParcial y Last.

Pares	Coseno	QDSM All	LCS	Máximo relevantes
(Q_{22}, Q_{123})	0,109	0,369	1	2
(Q_{25}, Q_{26})	0,085	0,085	0	0
(Q_{41}, Q_{102})	0,0209	0,51	1	1

Tabla 3.5: Muestra de resultados con el dataset de la medida QDSM de la variante All.

Pares	Coseno	QDSM AllParcial	LCS	Máximo relevantes
(Q_{22}, Q_{123})	0,109	0,054	0	1
(Q_{25}, Q_{26})	0,085	0,542	1	1
(Q_{41}, Q_{102})	0,0209	0,0209	0	0

Tabla 3.6: Muestra de resultados con el dataset de la medida QDSM de la variante AllParcial.

Pares	Coseno	QDSM Last	LCS	Máximo relevantes
(Q_{22}, Q_{123})	0,109	0,554	1	1
(Q_{25}, Q_{26})	0,085	0,542	1	1
(Q_{41}, Q_{102})	0,0209	0,0209	0	0

Tabla 3.7: Muestra de resultados con el dataset de la medida QDSM de la variante Last.

3.7.4 Conclusiones de la medida QDSM

Respecto a las pequeñas muestras de los resultados mostrados anteriormente, la medida QDSM puede mejorar, igualar o incluso empeorar con respecto a la medida del coseno, ya que sólo dependerá de la relevancia de los documentos en el par de las consultas, precisamente se puede observar que la medida es mejor si el LCS es mayor a cero ($x > 0$). Si el número máximo de los documentos relevantes es distinto de cero ($y \neq 0$) y además el LCS es igual a cero ($x = 0$), el resultado de la medida QDSM será menor que la distancia del coseno. También los resultados variarán dependiendo de la variante que se utilize, puesto que cada una de ellas tiene una forma distinta de considerar si un documento es relevante. Por eso, en cada una de las Tablas (3.5 a 3.7) se puede observar que algunos resultados son iguales o también varían, debido a que el LCS y el máximo de documentos relevantes son distintos en cada tabla.

3.7.5 Implementación de los algoritmos de clustering

Con los resultados del QDSM en sus variantes y los resultados con la medida del coseno para los documentos, se implementan los algoritmos de clustering Single Link, Complete

Link y el Average Link, para formar clústeres con cada uno de los variantes de las consultas y formar clústeres con los documentos con la distancia de coseno.

3.8 Resultados experimentales

Se formaron dos clústeres para cada uno de los algoritmos de clustering, siendo estos los clústeres de los documentos considerando la medida del coseno y los clústeres de las consultas con sus tres variantes: All, AllParcial y Last, teniendo en cuenta la medida QDSM para formar estos clústeres de las consultas. En la Figura 3.3, se puede observar la idea principal para llevar a cabo los experimentos para comparar la medida QDSM con la medida del coseno, a través de la recuperación de documentos que se encuentran agrupados en los clústeres. Principalmente, es necesario aplicar una consulta ya conocida a los clústeres de las consultas, tal como se muestra en la Figura 3.3.a), con tal de que se encuentre la consulta pasada (consulta ya realizada) en los clústeres. Se contabiliza cuántos clústeres se tuvieron que recorrer hasta encontrar la última consulta del clúster que haga match con la consulta pasada. Luego de obtener las consultas recuperadas que estaban contenidas en los clústeres, se considera los documentos asociados de cada consulta recuperada, como se representa en la Figura 3.3.b), además se contabiliza si esos documentos asociados a cada consulta recuperada son relevantes. Posteriormente, con los documentos asociados de las consultas recuperadas, se procede a encontrar estos documentos en los clústeres de documentos, tal como se presenta en la Figura 3.3.c). Si el documento a buscar hizo match con un documento del clúster, se contabiliza sólo si el documento recuperado es relevante, también se contabiliza la cantidad de cuántos clústeres se recorrieron hasta encontrar el último documento contenido en un clúster que hizo match con el documento a encontrar. Es importante mencionar que estos experimentos se realizarán diez veces y se sacará un promedio de los resultados obtenidos, independientemente de las consultas pasadas que se quiera buscar en los clústeres de las consultas, puesto que la relevancia de los documentos se produce a través de una probabilidad aleatoria, para que así se produzcan resultados más claros del comportamiento de la medida del coseno y la medida QDSM.

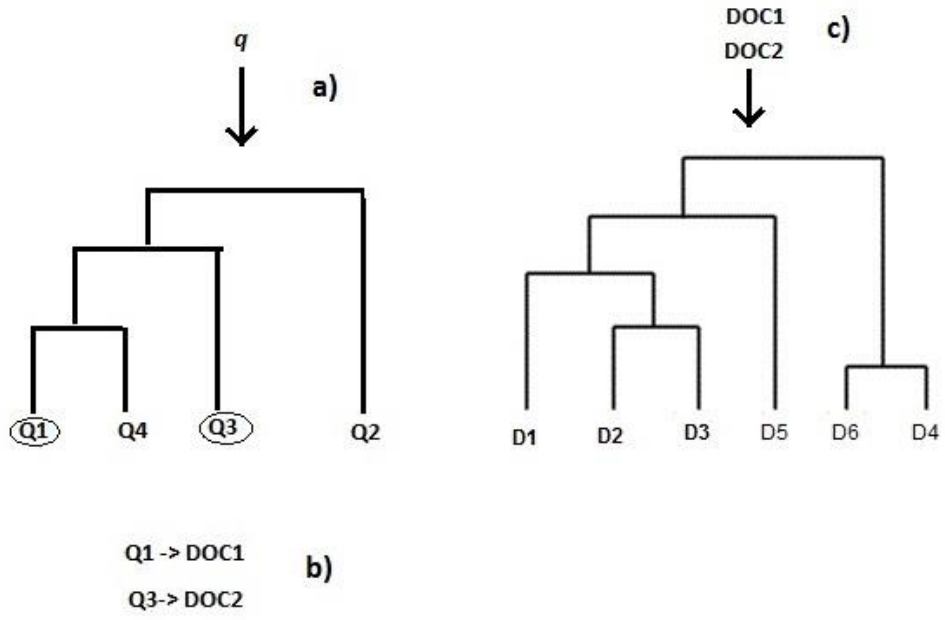


Figura 3.3: Diagrama que representa la forma de realizar los experimentos.

A continuación, los resultados que se obtuvieron en los experimentos están contenidos en las Tablas (A1 a A9) del Anexo 1 de forma detallada. La Tabla A1, está compuesta por cinco columnas. En la primera columna denominada “Total de consultas”, se puede observar la cantidad total de las consultas ya conocidas o consultas pasadas que se aplicaron a los clústeres de las consultas, de la misma forma que se presenta en la Figura 3.3.a). Estas cantidades de consultas se han realizado de forma ascendente. En la segunda columna de la Tabla denominada “N° relevantes All”, corresponde a la sumatoria total de todos los documentos relevantes que se recuperaron utilizando las consultas pasadas, como se prsenta en la Figura 3.3.b). En la tercera columna denominada “N° relevantes Coseno”, corresponde a la sumatoria total de todos los documentos relevantes que se encontraron en los clústeres de los documentos utilizando los documentos asociados de las consultas recuperadas, tal como se presenta en la Figura 3.3.c). En la cuarta columna denominada “Recorrido All”, corresponde a la sumatoria total de cuántos clústeres se recorrieron hasta que la consulta pasada haga match con el último clúster que contenga esa consulta. De la misma manera, en la quinta columna denominada “Recorrido Coseno”, corresponde a la sumatoria total de cuántos clústeres se recorrieron hasta que el documento asociado a la consulta recuperada haga match con el último clúster que contenga ese documento. Esta composición de la Tabla A1 del Anexo 1, se repite en las Tablas (A2 a la A9), con la

diferencia que la segunda y cuarta columna varía dependiendo de la variante que se esté utilizando.

En las Figuras (3.4 a la 3.12), se presenta de forma gráfica los datos contenidos de las Tablas del Anexo 1, para que así se pueda observar el comportamiento de las medidas QDSM y coseno, dependiendo de la variante que se utiliza y del algoritmo de clustering que se implementa. Estas figuras están compuestas de la siguiente forma: en el eje *x*, se encuentran las cantidades de las consultas pasadas que son utilizadas y en el eje *y* aparece las cantidades de los documentos recuperados. También, está representado de forma lineal la primera y segunda columna de las Tablas del Anexo 1, dependiendo de la variante y el algoritmo que se utilize. La comparación de la cantidad recorrida en los clústeres, no está representado de forma gráfica, pero los resultados en detalle se encuentran en la cuarta y quinta columna de las Tablas (A1 a la A9) del Anexo 1.

En la Figura 3.4, se puede observar que los documentos relevantes recuperados en el clústeres de las consultas son mayores que los documentos recuperados por los clústeres de los documentos. Además, mientras que vaya creciendo la cantidad de las consultas pasadas aplicadas, la serie de “N° relevantes Coseno” la cantidad de documentos relevantes se va acercando de forma gradual a la otra serie denominada “N° relevantes All”. En este caso, con la variante All y el algoritmo “Single Link”, la medida QDSM es mejor que la medida de coseno.

En la Tabla A1, comparando los recorridos de la columna “Recorrido All” y “Recorrido Coseno”, en ambos clústeres al recuperar los documentos, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos.

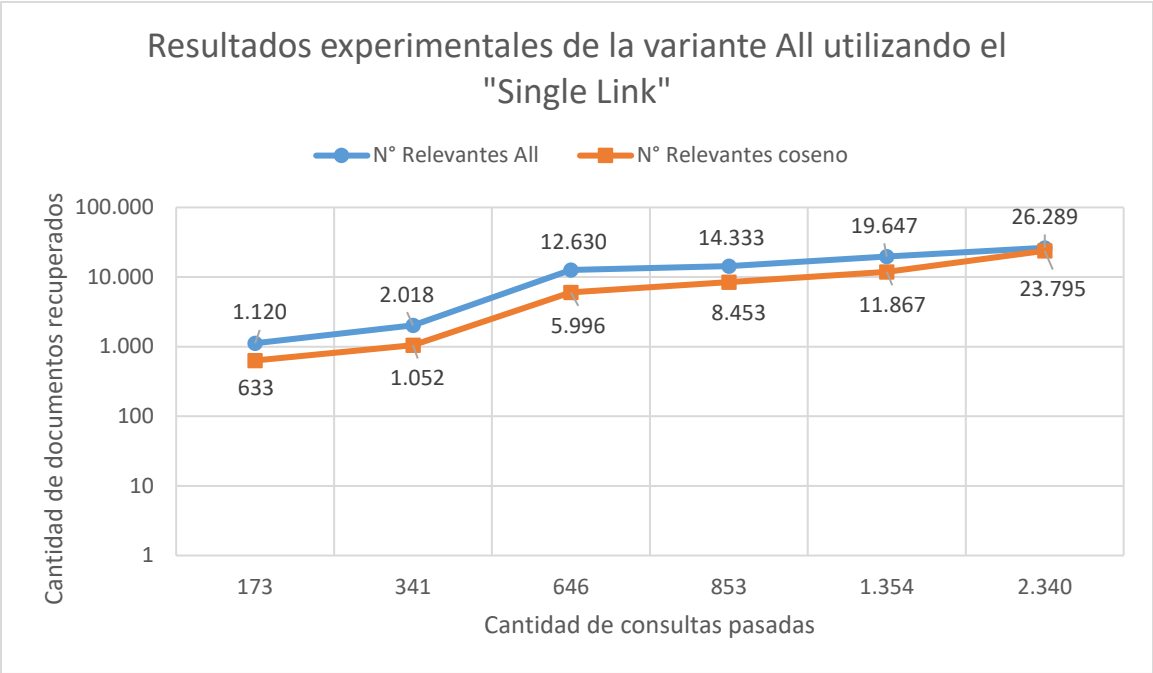


Figura 3.4: Gráfico de los resultados experimentales con la variante All utilizando el algoritmo "Single Link".

En la Figura 3.5, se puede observar que en este caso, con la variante AllParcial y el algoritmo "Single Link", la medida del coseno es mejor que la medida QDSM. Los documentos relevantes recuperados en el clústeres de los documentos son mayores que los documentos recuperados por los clústeres de las consultas. Además, se puede observar que en las cantidades mayores de consultas pasadas, se produce una gran distancia entre las series de la gráfica, por ejemplo, en las 2.340 consultas, se puede notar que la diferencia entre los documentos recuperados entre las dos series de la gráfica casi se sextuplica. En la Tabla A2, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, además se puede notar que la cantidad de clústeres recorridos con la variante All y AllParcial son similares.

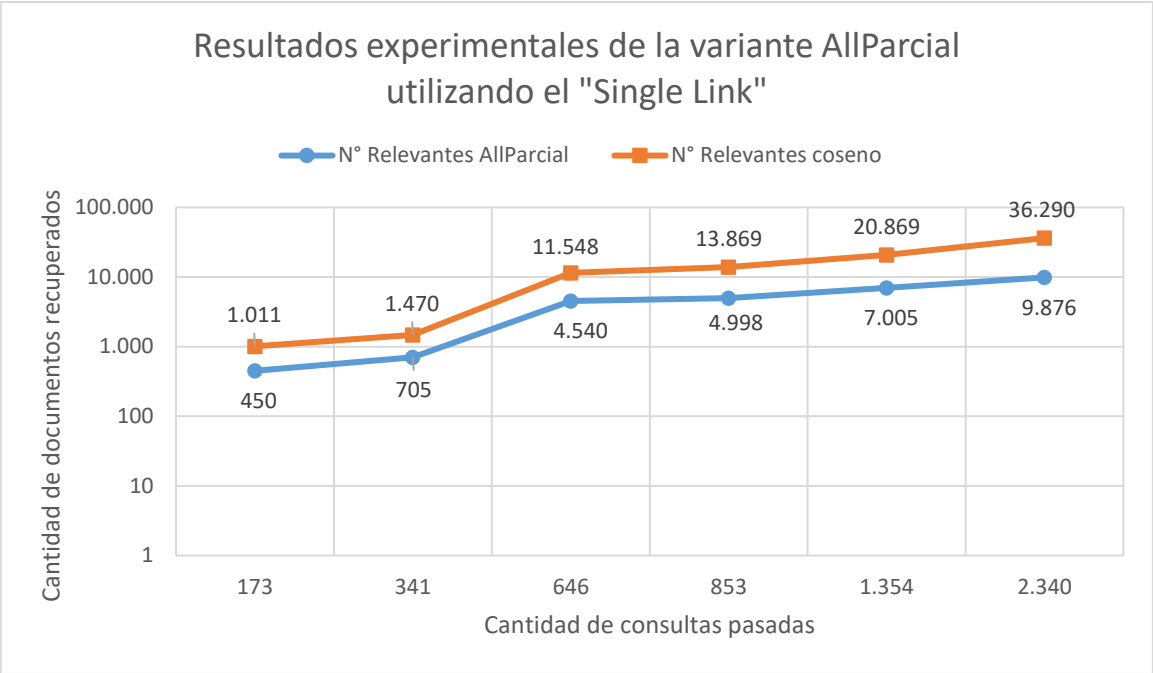


Figura 3.5: Gráfico de los resultados experimentales con la variante AllParcial utilizando el algoritmo "Single Link".

En la Figura 3.6, se observa que la medida del coseno es mejor que la medida QDSM. Los documentos relevantes recuperados en el clústeres de los documentos son superiores que los documentos recuperados por los clústeres de las consultas. Además, se puede notar que en las cantidades mayores de consultas pasadas, se produce una gran distancia entre las dos series de la gráfica, por ejemplo, en las 2.340 consultas, se puede notar que la diferencia entre los documentos recuperados entre las dos series de la gráfica se duplica, pero la diferencia es menor comparado con la variante AllParcial. En la Tabla A3, comparando los recorridos en ambos clústeres al recuperar los documentos, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos.

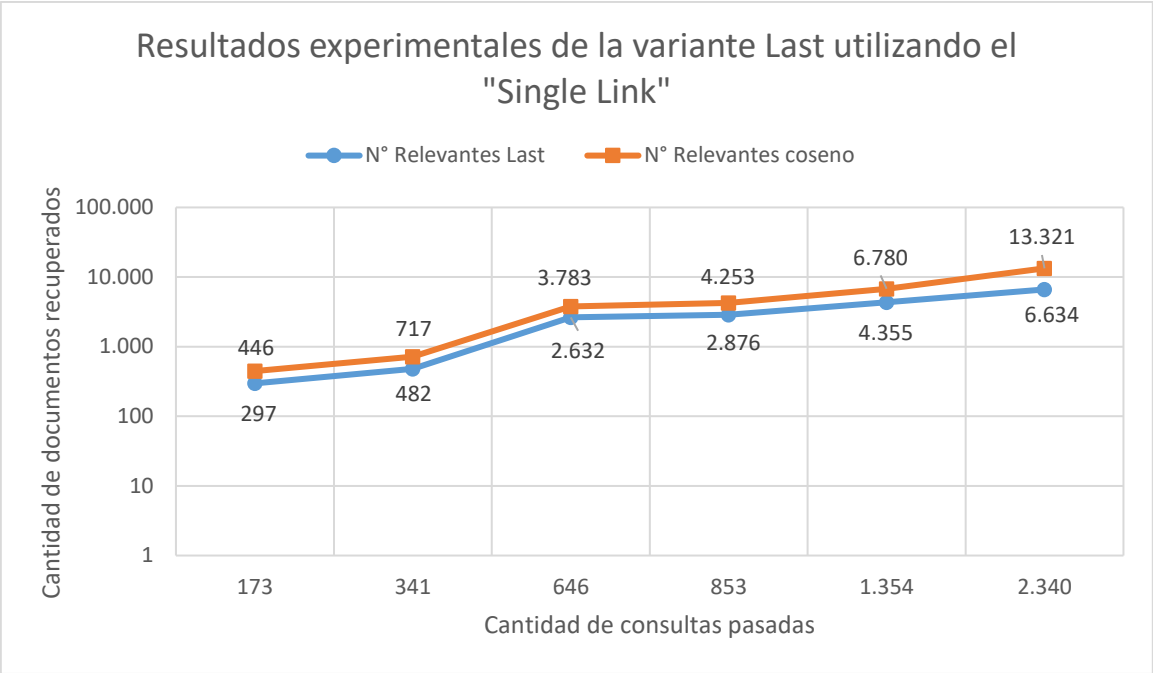


Figura 3.6: Gráfico de los resultados experimentales con la variante Last utilizando el algoritmo "Single Link".

En la Figura 3.7, se puede observar que en este caso, con la variante All y el algoritmo "Complete Link", la medida QDSM es mejor que la medida del coseno, ya que los documentos relevantes recuperados en el clústeres de las consultas son superiores que los documentos recuperados por los clústeres de los documentos. Además, se puede observar que mientras aumenta la cantidad de las consultas pasadas, las dos series de los gráficos se van acercando, pero la serie "N° relevantes All" siempre obtiene más cantidades de documentos recuperados que en la otra serie. En la Tabla A4, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, por ejemplo, con 2.340 consultas pasadas, el "Recorrido Coseno" triplica la cantidad de "Recorrido All".

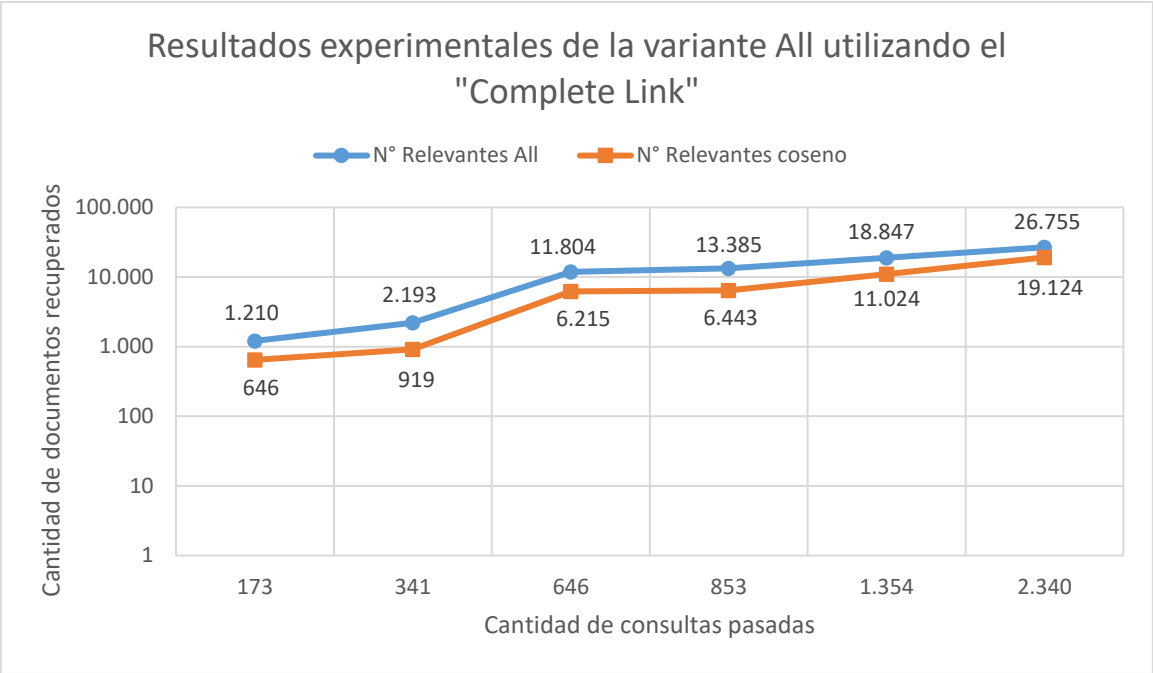


Figura 3.7: Gráfico de los resultados experimentales con la variante All utilizando el algoritmo "Complete Link".

En la Figura 3.8, se puede observar que en este caso, con la variante AllParcial y el algoritmo "Complete Link", la medida del coseno es mejor que la medida QDSM. Los documentos relevantes recuperados en el clústeres de los documentos son mayores que los documentos recuperados por los clústeres de las consultas. Además, se puede observar que en las cantidades mayores de consultas pasadas, se produce una gran distancia entre las series de la gráfica, por ejemplo, en las 2.340 consultas, se puede notar que la diferencia entre los documentos recuperados entre las dos series de la gráfica se triplica. En la Tabla A5, se puede notar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, por ejemplo, con 2.340 consultas pasadas, el "Recorrido Coseno" casi cuadriplica la cantidad de "Recorrido AllParcial".

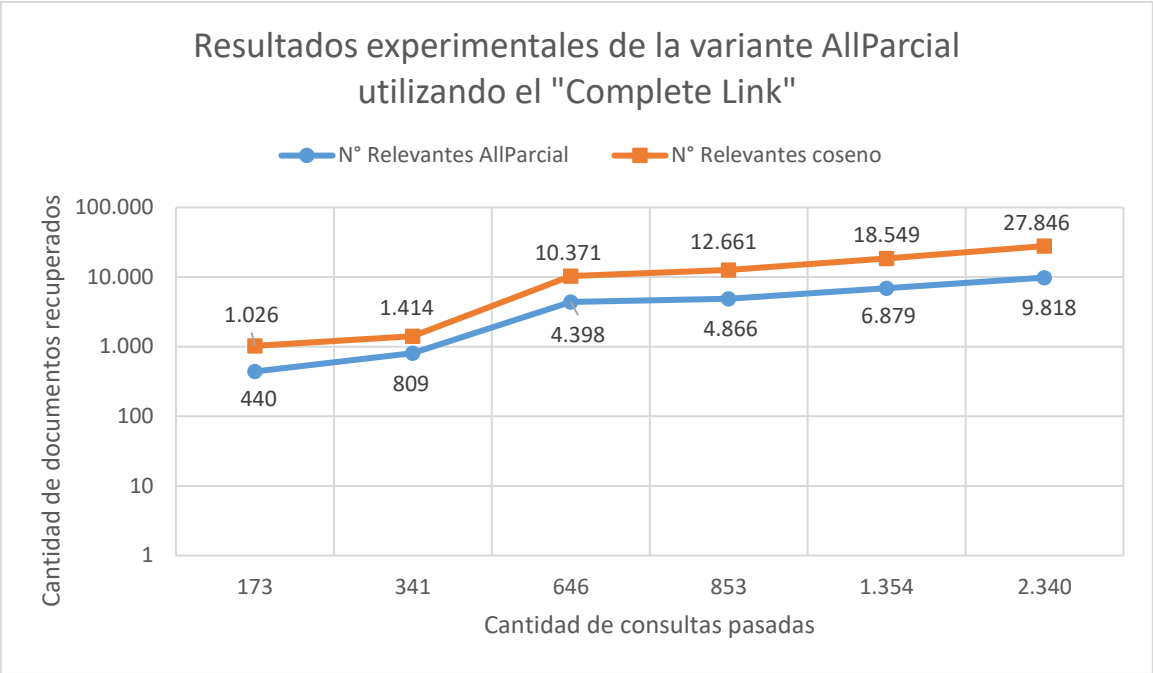


Figura 3.8: Gráfico de los resultados experimentales con la variante AllParcial utilizando el algoritmo "Complete Link".

En la Figura 3.9, se observa que la medida del coseno es mejor que la medida QDSM. Los documentos relevantes recuperados en el clústeres de los documentos son superiores que los documentos recuperados por los clústeres de las consultas. Además, se puede notar que en las cantidades mayores de consultas pasadas, se produce una mediana distancia entre las series de la gráfica. Por ejemplo, en las 2.340 consultas, se puede notar que la diferencia entre los documentos recuperados entre las dos series de la gráfica es por casi 4.000 documentos, pero la diferencia es menor comparado con la variante AllParcial. En la Tabla A6, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos.

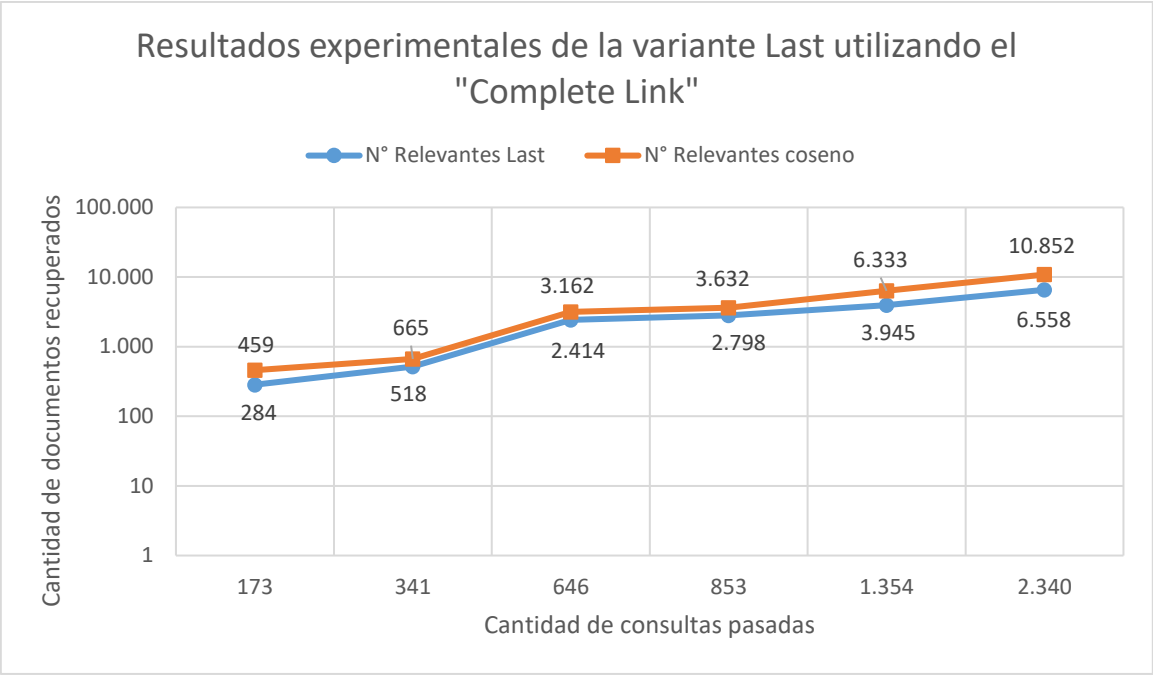


Figura 3.9: Gráfico de los resultados experimentales con la variante Last utilizando el algoritmo "Complete Link".

En la Figura 3.10, se puede observar que en este caso, con la variante All y el algoritmo "Average Link", la medida QDSM es mucho mejor que la medida del coseno, ya que los documentos relevantes recuperados en el clústeres de las consultas son muy superiores que los documentos recuperados por los clústeres de los documentos. También, se puede notar las diferencias entre las cantidades de documentos es muy grande, por ejemplo, en las 2.340 consultas, la diferencia entre la cantidad de los documentos recuperados es de veinte veces gracias a la implementación del algoritmo "Average Link". En la Tabla A7, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, por ejemplo, con 646 consultas pasadas, el "Recorrido Coseno" triplica la cantidad de "Recorrido All".

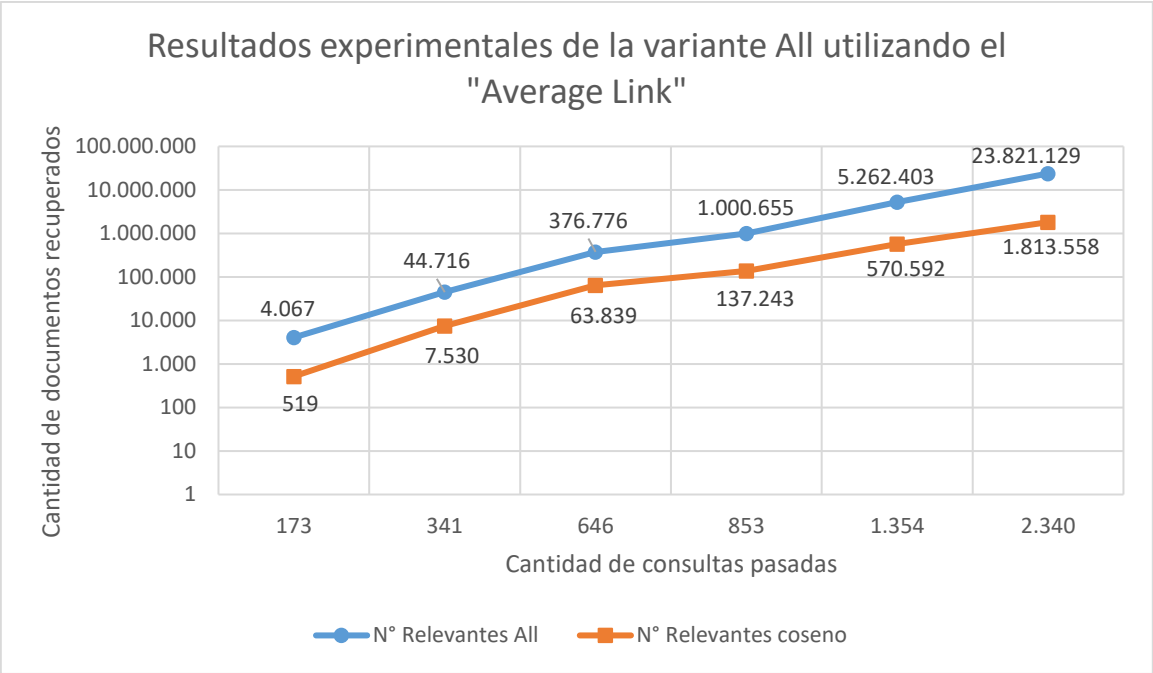


Figura 3.10: Gráfico de los resultados experimentales con la variante All utilizando el algoritmo "Average Link".

En la Figura 3.11, se puede observar que en este caso, con la variante AllParcial y el algoritmo "Average Link", la medida QDSM es mucho mejor que la medida del coseno, ya que los documentos relevantes recuperados en el clústeres de las consultas son muy superiores que los documentos recuperados por los clústeres de los documentos. También, se puede notar las diferencias entre las cantidades de documentos son superiores". Con el algortimo "Average Link", la serie "N° relevantes AllParcial" es mayor que la serie "N° relevantes Coseno", mientras que en los otros dos algoritmos de clustering ocurre de manera inversa. En la Tabla A8, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, por ejemplo, con 2.340 consultas pasadas, el "Recorrido Coseno" triplica la cantidad de "Recorrido AllParcial".

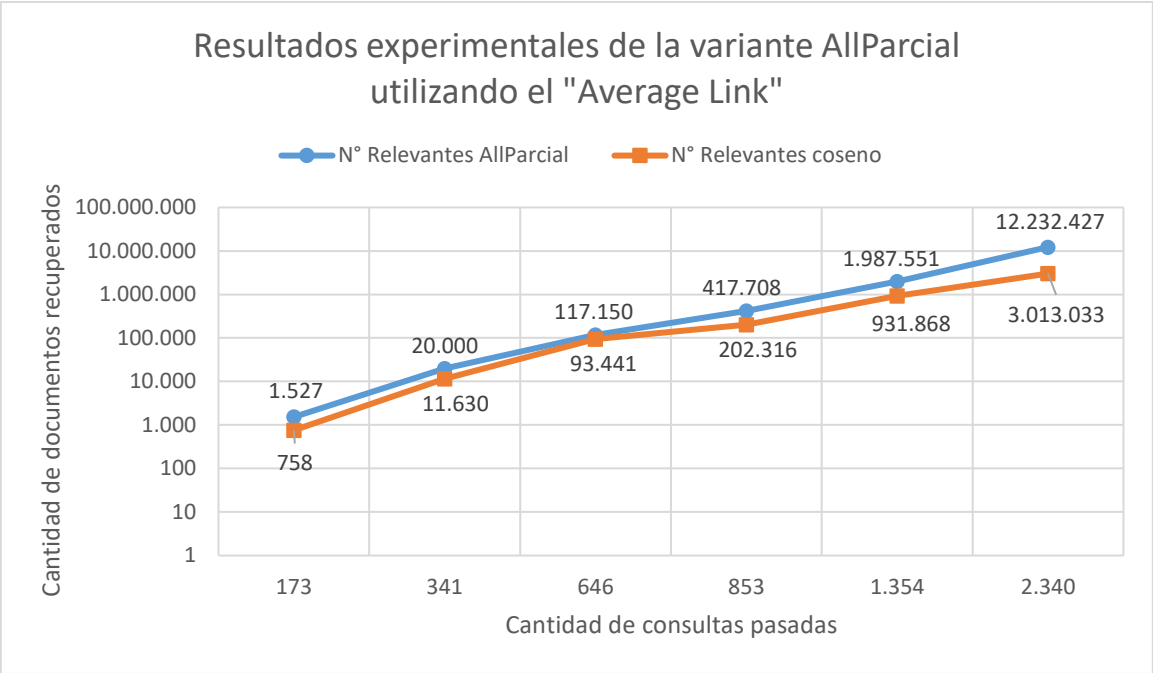


Figura 3.11: Gráfico de los resultados experimentales con la variante AllParcial utilizando el algoritmo “Average Link”.

En la Figura 3.12, se puede observar que en este caso, con la variante Last y el algoritmo “Average Link”, la medida QDSM es mucho mejor que la medida del coseno, ya que los documentos relevantes recuperados en el clústeres de las consultas son muy superiores que los documentos recuperados por los clústeres de los documentos. También, se puede notar las diferencias entre las cantidades de documentos son superiores”. Con el algortimo “Average Link”, la serie “N° relevantes Last” es mayor que la serie “N° relevantes Coseno”, mientras que en los otros dos algoritmos de clustering no ocurre de la misma manera, sino que ocurre de manera inversa. En la Tabla A9, se puede observar que en los clústeres de las consultas hubo una menor cantidad de clústeres recorridos que en los clústeres de los documentos, por ejemplo, con 2.340 consultas pasadas, el “Recorrido Coseno” triplica la cantidad de “Recorrido All”.

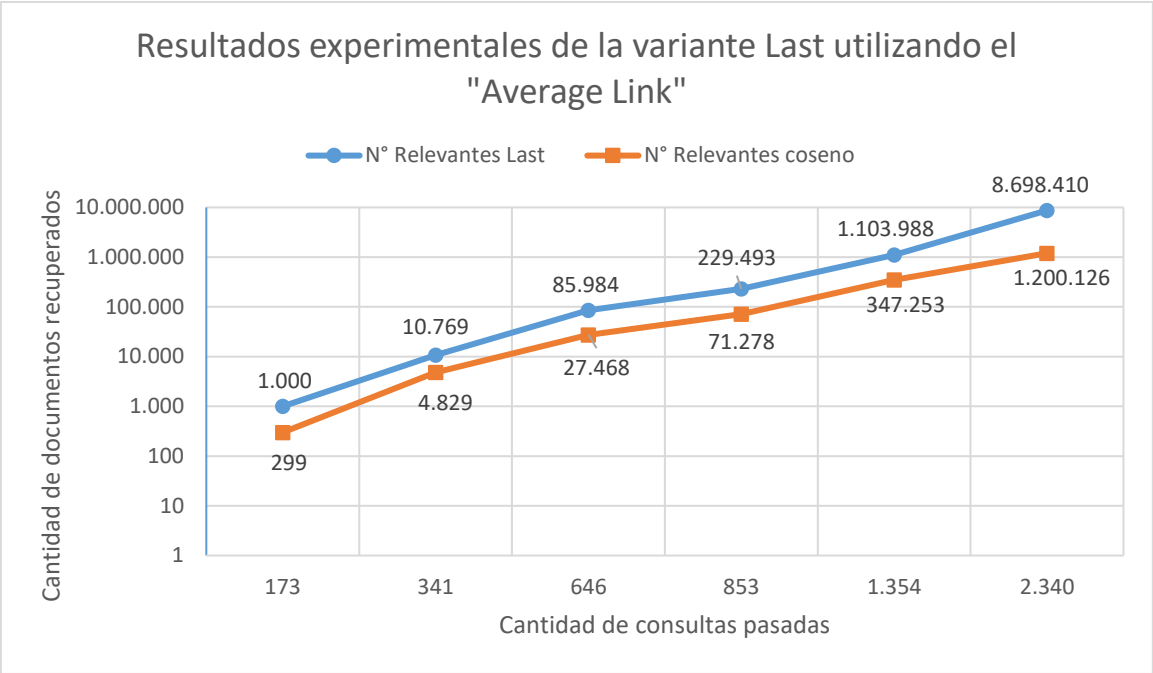


Figura 3.12: Gráfico de los resultados experimentales con la variante Last utilizando el algoritmo "Average Link".

3.9 Conclusiones de los experimentos

La medida QDSM toma en consideración a los documentos relevantes para cada consulta. Por lo tanto, dos consultas son similares teniendo en cuenta no sólo la similitud entre ellos, sino también cuántos documentos son similares en ambas listas. Las conclusiones por los experimentos finales que son representados en las Figuras (3.4 a 3.12), indican que la variante All es la mejor de todas frente a los demás, ya que considera que todos los documentos asociados a una consulta pueden ser relevantes o no, si estos son relevantes la medida QDSM mejorará de mejor forma. Con respecto a la variante AllParcial, se observa que se obtienen menos documentos relevantes frente a los documentos relevantes de los clústeres de los documentos, porque la relevancia de los documentos asociados a la consulta son considerados de forma individual. Por ejemplo, la consulta posee cinco documentos y sólo dos de estos son relevantes, ya que estos cinco documentos se tienen que buscar en los clústeres de los documentos, es más probable que los cinco documentos hagan match y sean relevantes a la vez, por lo que el AllParcial sale perjudicado. De la misma forma sucede en la variante del Last, que sólo toma en cuenta el último documento de las consultas. Por ejemplo, puede ocurrir que en los clústeres de las consultas se hayan

encontrado un documento y este es relevante, pero al buscarlo en los clústeres de los documentos, se puede observar que el documento puede hacer match en uno o más clústeres y la cantidad de los documentos relevantes puede ser la misma que la cantidad de los clústeres de las consultas o puede ser superior. Cabe destacar que en los tres algoritmos de clustering junto con las tres variantes, el recorrido para recuperar las consultas con sus documentos son menores que el recorrido para recuperar los documentos relevantes en los clústeres de los documentos. Con respecto al recorrido en los clústeres, se puede concluir que la medida QDSM se necesita menos tiempo en recuperar los documentos en comparación con la medida del coseno.

Con respecto a los algoritmos de clustering implementados, se puede observar que entre ellos, claramente el “Average Link” es el mejor, ya que supera con creces los documentos relevantes que se recuperaron en los clústeres de las consultas, esto se puede observar en las Figuras 3.10, 3.11 y 3.12. Además en este este algoritmo, la variante All es la que entrega mejores resultados. También, se puede observar que el “Average Link” incluso mejora las variantes AllParcial y Last, puesto que se recuperan más documentos relevantes de los clústeres de las consultas en comparación con los documentos recuperados de los clústeres de los documentos, pero la ejecución de este algoritmo es más lenta con respecto a los otros dos algoritmos de clusering, ya que se requiere más trabajo en realizar los clústeres.

Respecto al algoritmo “Single Link” y el “Complete Link”, el algoritmo “Single Link” es levemente superior, esto se puede notar en las Figuras 3.4, 3.7 y 3.10, indicando que los documentos recuperados son mayores con respecto de los documentos recuperados utilizando el algoritmo “Complete Link”. Entre el algoritmo “Single Link” y el “Complete Link”, existe muy poca diferencia de documentos recuperados entre ellos, pero el algoritmo “Single Link” es levemente superior que el algoritmo “Complete Link”, esto se puede observar en las Figuras 3.4 y 3.7.

Finalmente, se puede concluir que la medida QDSM es mejor que la medida de distancia del coseno en ciertos casos. Aplicando los algoritmos de clustering del “Single Link” y el “Complete Link”, la medida QDSM es mejor sólo si se considera la variante All, que determina la totalidad de los documentos de una consulta si son relevantes o no relevantes. En cambio, utilizando el algoritmo de clustering “Average Link”, la medida QDSM es mejor que la medida de distancia del coseno, incluso en las tres variantes, dado que además este algoritmo es el mejor comparado a los otros dos.

Capítulo 4: Conclusiones

En la literatura de RI se pueden encontrar diversas contribuciones tales como propuestas de indexación, funciones de comparación y modelos formales. Sin embargo, pocas propuestas toman ventaja de las búsquedas realizadas en el pasado. Además, la mayoría de estas propuestas se encuentran en el dominio de los motores de búsqueda web, que se basan en consultas históricas almacenadas en archivos de registro. Sin embargo, la mayoría de estas propuestas se refieren a consultas repetitivas.

Actualmente, pocos trabajos se ocupan del uso de consultas pasadas similares. Por lo tanto, esta investigación consiste en comparar a través de resultados empíricos, la medida de distancia QDSM con la distancia del coseno sobre algoritmos de clustering, con el propósito de mejorar la efectividad en la recuperación de los documentos. Para lograr esto, se ven implicados estos tres objetivos:

- Se realiza un estudio del estado del arte y de los algoritmos más utilizados en el contexto de la RI.
- Encontrar un dataset adecuado para poder implementar un algoritmo de clustering.
- Obtener los resultados empíricos que contrastan la medida QDSM con la medida del coseno.

La contribución general de este proyecto de título fué obtener resultados empíricos que comparen la medida QDSM con la medida del coseno, implementando los algoritmos de clustering. A continuación, en los siguientes párrafos se van a enumerar las contribuciones de esta tesis con respecto a los objetivos de ésta.

4.1 Contribuciones de la investigación

4.1.1 Estudio del estado del arte y de los algoritmos más utilizados en el contexto de RI

En el Capítulo 2, se da a conocer sobre la recuperación de la información, como la definición, los modelos que han sido desarrollados, siendo el más utilizado el modelo de espacio vectorial, entre otras cosas. En la Sección 2.3 da a conocer la medida de distancia del coseno, que indica si una consulta posee similitud con un término.

En el Capítulo 3, se explica en que consiste el clustering, además se presentan diferentes propuestas de la literatura que trata del clustering de consultas, que toman en cuenta el contexto en el que se utiliza la similitud. También se da a conocer sobre el algoritmo de clustering llamado Single Link, siendo éste uno de los más utilizados para formar clústeres.

4.1.2 Encontrar un dataset adecuado para poder implementar un algoritmo de clustering

En la Sección 3.7, se da a conocer el dataset apropiado para poder implementar el algoritmo de clustering. Este dataset posee miles de datos para poder trabajar en él, además este dataset está bien estructurado, junto con la explicación de todas las piezas de información que posee. Cabe mencionar que finalmente se implementaron tres algoritmos de clustering siendo estos el Single Link, Complete Link y el Average Link, para poder contrastar la medida de distancia del coseno con la medida QDSM y así poder ampliar más los experimentos y tener resultados más variados.

4.1.3 Obtener los resultados empíricos que contrastan la medida QDSM con la medida del coseno

La principal contribución de esta sección es que se demuestra que con la medida QDSM se obtienen mejores resultados en recuperar los documentos relevantes en comparación con la medida del coseno en ciertos casos. En la Sección 3.8 de los resultados experimentales, se observa que los resultados mejoran solo si existen documentos relevantes suficientes para mejorar la medida QDSM frente a la distancia del coseno, generalmente ocurre cuando se utiliza la variante All, en el que considera que todos los documentos asociados a una consulta pueden ser relevantes o no relevantes. Además, con los tres algoritmos de clustering que se implementaron se observa que la medida QDSM es mejor si se consideran a todos los documentos de una consulta relevantes o no relevantes.

4.2 Contribuciones futuras

Este proyecto de título buscó comparar a través de resultados empíricos, la medida de distancia QDSM con la distancia del coseno. Con los experimentos realizados se pudo demostrar que la medida QDSM es mejor en ciertos casos, sin embargo, a continuación se presentarán algunas sugerencias que pueden ampliar la comparación de estas dos medidas.

4.2.1 Realizar nuevos experimentos con un nuevo dataset

En la Sección 3.7, el entorno experimental presenta un dataset que fue realizado en el año 2006, por lo que poder ser interesante buscar un dataset más actual y que pudiera tener si los documentos que se encuentran son o no relevantes para los usuarios, para que no se tenga que realizar una probabilidad si un documento es relevantes y que venga ya asignado en el dataset.

4.2.2 Implementar más algoritmos de clustering

En el Capítulo 3, se presentaron tres algoritmos de clustering para poder realizar los experimentos, pero se podrían aplicar más algoritmos como por ejemplo el algoritmo Ward's, Bisecting K-Means, entre otros, para así observar el comportamiento de los resultados si se aplicarían estos algoritmos de clustering, observar si la medida QDSM sigue siendo superior que la medida de distancia del coseno, y así poderlo comparar con los algoritmos que ya se implementaron.

4.2.3 Experimentar con una mayor cantidad de consultas para comparar las medidas

En la Sección 3.8, de los resultados experimentales se entregaron resultados con una cierta cantidad de consultas, pero estas consultas no son una cantidad muy grande. Se podría aumentar la cantidad de estas consultas para poder ver el comportamiento de la medida QDSM frente a la medida de distancia del coseno, y así poder tener un gran volumen de datos que puedan sugerir que la medida QDSM es mejor.

Bibliografía

Anderberg, M. R. (1973). Cluster Analysis for Applications. Academic Press.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In Proceedings of the 2004 International Conference on Current Trends in Database Technology, EDBT'04, páginas 588– 596, Berlin, Heidelberg. Springer-Verlag.

Barry, C. L. (1994). User-defined relevance criteria : an exploratory study. Journal of the American Society for Information Science, 45 :149–159.

Bharat, K. and Henzinger, M. R. (1998). Improved algorithms for topic distillation in a hyperlinked environment. In In SIGIR Conference on Research and Development in Information Retrieval.

Borlund, P. y Ingwersen, P. (1997). The development of a method for the evaluation of interactive information retrieval systems. Journal of Documentation, 53 :225–250.

Broder, A. (2002). A taxonomy of web search. SIGIR Forum, 36(2) :3–10.

Cetintas, S., Si, L., y Yuan, H. (2011). Using past queries for resource selection in distributed information retrieval. Technical Report 1743, Department of Computer Science, Purdue University.

Chowdhury, G. (2010). Introduction to Modern Information Retrieval, Third Edition. Facet Publishing, 3rd edition.

Fitzpatrick, L. y Dent, M. (1997). Automatic feedback using past queries : Social searching ? In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '97, páginas 306–313, New York, NY, USA. ACM.

Fu, L., Dion, D. H., y Schubert, S. S. (2004). The effect of similarity measures on the quality of query clusters. Journal of Information Science, 30(5) :396– 407.

Fuhr, N. y Lalmas, M. (2007). Advances in xml retrieval: The inex initiative. In Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries, IWRIDL '06, páginas 16 :1–16 :6, New York, NY, USA. ACM.

Good, I.J. (1958). Speculations concerning information retrieval. Research report PC-78, IBM Research Centre, Yorktown Heights, New York.

Gordon, A. D. (1987). A Review of Hierarchical Classification. Journal of the Royal Statistical Society. Series A (General), 150(2) :119–137.

Gutiérrez-Soto, C. (2016). Exploring the Reuse of Past Search Results in Information Retrieval.

Harman, D. (1993). Overview of the first trec conference. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93, páginas 36–47, New York, NY, USA. ACM.

Hartigan, J. A. (1975). Clustering Algorithms. John Wiley & Sons, Inc., New York, NY, USA, 99th edition.

Hasanzadeh, S. y Keshavarzi, A. (2009). Application of query sensitive similarity measure in ir systems. In Proceedings of the 2009 Third Asia International Conference on Modelling and Simulation, AMS '09, pages 73–78, Washington, DC, USA. IEEE Computer Society.

Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '96, New York, NY, USA, ACM (1996) 76–84

Hust, A. (2004). Introducing Query Expansion Methods for Collaborative Information Retrieval. In Dengel, A., Junker, M., and Weisbecker, A., editors, Reading and Learning - Adaptive Content Recognition, volume 2956 of Lecture Notes in Computer Science, pages 252–280, Berlin, Heidelberg, New York. Springer-Verlag.

Jiang, S., Zilles, S., Holte, R. (2008). Empirical analysis of the rank distribution of relevant documents in web search.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 :11–21.

Keen, E. M. (1992). Presenting results of experimental retrieval comparisons. Inf. Process. Manage., 28(4) :491–502.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. J. ACM, 46(5):604–632.

Lewis, D. D. y Jones, K. S. (1996). Natural language processing for information retrieval. Commun. ACM, 39(1) :92–101.

Liu, X. y Croft, W. B. (2002). Passage retrieval based on language models. In Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM '02, páginas 375–382, New York, NY, USA. ACM.

Luhn, H.P. (1958). The automatic creation of literature abstracts. IBM Journal of Research and Development.

Na, S.-H. (2013). Probabilistic co-relevance for query-sensitive similarity measurement in information retrieval. Inf. Process. Manage., 49(2) :558–575.

Niederberger, T., Stoop, N., Christen, M., y Ott, T. (2012). Hebbian Principal Component Clustering for Information Retrieval on a Crowdsourcing Platform. Nonlinear Dynamics of Electronic Systems, Proceedings of NDES 2012, páginas 1–4.

Ouksel, A. (2002). Mining the world wide web : An information search approach by george chang, marcus j. healey (editor), james a. m. mchugh, jason t. l. wang. SIGMOD Rec., 31(2) :69–70.

Preece, S.E. (1973). Clustering as an output option. Proceedings of the American Society for Information Science, 10:189-190.

Reid, J. (2000). A task-oriented non-interactive evaluation methodology for. Information Retrieval Systems, Information Retrieval, 2 :115–129.

Rijsbergen, C. J. V. (1979). Information Retrieval. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Robertson, S. E., van Rijsbergen, C. J., y Porter, M. F. (1981). Probabilistic models of indexing and searching. In Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval, SIGIR '80, páginas 35–56, Kent, UK, UK. Butterworth & Co.

Ruthven, I., y White, R. W., editors, ECIR, volume 4956 of Lecture Notes in Computer Science, páginas 446–453. Springer.

Salton, G. y McGill, M. J. (1986). Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA.

Salton, G. (1971). The SMART Retrieval System and ;Experiments in Automatic Document Processing. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Salton, G. y Buckley, C. (1987). Term weighting approaches in automatic text retrieval. Technical report, Cornell University, Ithaca, NY, USA.

Salton, G., Allan, J., y Buckley, C. (1993). Approaches to passage retrieval in full text information systems. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93, páginas 49–58, New York, NY, USA. ACM.

Salton, G. y McGill, M. (1983). Introduction to Modern Information Retrieval. McGraw-Hill computer science series. McGraw-Hill International.

Salton, G. (1971). The SMART Retrieval System and ;Experiments in Automatic Document Processing. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Salton, G., Wong, A., y Yang, C. S. (1975). A vector space model for automatic indexing.Commun. ACM, 18(11) : 613–620.

Saravanakumar, K. y Moturi, M. (2011). Article : Semantic based personalized framework for information retrieval. International Journal of Computer Applications, 20(4) :14–17.

Schamber, L., Eisenberg, M., y Nilan, M. S. (1990). A re-examination of relevance : Toward a dynamic, situational definition. Inf. Process. Manage., 26(6) :755–776.

Sneath, P. y Sokal, R. (1973). Numerical Taxonomy : The Principles and Practice of Numerical Classification. A Series of books in biology. W. H. Freeman.

Sparck Jones, K. y Willett, P., editors (1997a). Readings in Information Retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Sparck Jones, K. y van Rijsbergen, C.J. (1976). Information retrieval test collections. Journal of Documentation.

Tombros, A (2002).The effectiveness of query-based hierarchic clustering of documents for information retrieval.

Tombros, A., Villa, R., Rijsbergen, C. J. V. (2002). The effectiveness of query-specific hierarchic clustering. In in information retrieval. Information Processing and Management, páginas 559–582.

Tombros, A., van Rijsbergen, C. J. (2004). Query-sensitive similarity measures for information retrieval. Knowl. Inf. Syst., 6(5) :617–642.

Van Rijsbergen, C. J. (1986). A new theoretical framework for information retrieval. SIGIR Forum, 21(1-2) :23–29.

Voorhees, E. M. (1985). The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval. PhD thesis, Cornell University.

Willett, P. (1988). Recent trends in hierarchic document clustering : A critical review. Inf. Process. Manage., 24(5) :577–597.

Anexo 1: Tablas de los resultados experimentales

Total de consultas	N° Relevantes All	N° Relevantes Coseno	Recorrido All	Recorrido Coseno
173	1.120	633	2.067	3.257
341	2.018	1.052	11.991	15.746
646	12.630	5.996	61.782	83.993
853	14.333	8.453	118.018	161.782
1.354	19.647	11.867	343.998	425.890
2.340	28.289	23.795	1.093.866	4.148.057

Tabla A1: Resultados experimentales con la variante All utilizando el algoritmo "Single Link".

Total de consultas	N° Relevantes AllParcial	N° Relevantes Coseno	Recorrido AllParcial	Recorrido Coseno
173	450	1.011	2.263	3.250
341	705	1.470	11.972	15.727
646	4.540	11.548	62.465	83.956
853	4.998	13.869	120.241	160.824
1.354	7.005	20.869	346.251	426.991
2.340	9.876	36.290	1.107.254	4.142.041

Tabla A2: Resultados experimentales con la variante AllParcial utilizando el algoritmo "Single Link".

Total de consultas	N° Relevantes Last	N° Relevantes Coseno	Recorrido Last	Recorrido Coseno
173	297	446	2.186	3.218
341	482	717	11.866	14.993
646	2.632	3.783	59.556	77.544
853	2.876	4.253	111.816	145.503
1.354	4.355	6.780	334.048	385.044
2.340	6.634	13.321	1.125.440	4.001.192

Tabla A3: Resultados experimentales con la variante Last utilizando el algoritmo “Single Link”.

Total de consultas	N° Relevantes All	N° Relevantes Coseno	Recorrido All	Recorrido Coseno
173	1.210	646	2.316	6.917
341	2.193	919	13.608	48.826
646	11.804	6.215	65.817	280.471
853	13.385	6.443	124.971	542.002
1.354	18.847	11.024	356.919	1.472.322
2.340	26.755	19.124	1.011.942	3.845.730

Tabla A4: Resultados experimentales con la variante All utilizando el algoritmo “Complete Link”.

Total de consultas	N° Relevantes AllParcial	N° Relevantes Coseno	Recorrido AllParcial	Recorrido Coseno
173	440	1.026	2.379	6.952
341	809	1.414	13.900	49.149
646	4.398	10.371	67.567	279.713
853	4.866	12.661	127.096	539.224
1.354	6.879	18.549	364.666	1.468.346
2.340	9.818	27.846	981.066	3.818.593

Tabla A5: Resultados experimentales con la variante AllParcial utilizando el algoritmo “Complete Link”.

Total de consultas	N° Relevantes Last	N° Relevantes Coseno	Recorrido Last	Recorrido Coseno
173	284	459	2.063	6.692
341	518	665	13.241	46.947
646	2.414	3.162	68.505	265.142
853	2.798	3.632	128.623	502.450
1354	3.945	6.333	355.374	1.374.507
2340	6.558	10.852	999.627	3.634.567

Tabla A6: Resultados experimentales con la variante Last utilizando el algoritmo “Complete Link”.

Total de consultas	N° Relevantes All	N° Relevantes Coseno	Recorrido All	Recorrido Coseno
173	4.067	519	4.312	8.236
341	44.716	7.530	27.822	77.669
646	376.776	63.839	132.664	432.753
853	1.000.655	137.243	262.766	822.853
1.354	5.262.403	570.592	807.470	2.298.742
2.340	23.821.129	1.813.558	2.373.187	6.357.879

Tabla A7: Resultados experimentales con la variante All utilizando el algoritmo “Average Link”.

Total de consultas	N° Relevantes AllParcial	N° Relevantes Coseno	Recorrido AllParcial	Recorrido Coseno
173	1.527	758	4.187	9.810
341	20.000	11.630	27.889	78.165
646	117.150	93.441	133.273	437.487
853	417.708	202.316	266.824	826.860
1.354	1.987.551	931.868	806.392	2.302.584
2.340	12.232.427	3.013.033	2.375.520	6.495.996

Tabla A8: Resultados experimentales con la variante AllParcial utilizando el algoritmo “Average Link”.

Total de consultas	N° Relevantes Last	N° Relevantes Coseno	Recorrido Last	Recorrido Coseno
173	1.000	299	3.529	7.231
341	10.769	4.829	23.932	64.732
646	85.984	27.468	123.195	391.467
853	229.493	71.278	243.609	736.114
1.354	1.103.988	347.253	763.772	2.116.396
2.340	8.698.410	1.200.126	2.291.057	6.090.714

Tabla A9: Resultados experimentales con la variante Last utilizando el algoritmo “Average Link”.

Anexo 2: Código fuente del programa

```
/***Autor: Daniel Ibacache Soto
Este código corresponde al programa creado para realizar los experimentos
Compilación: La forma de compilar este programa en Linux es gcc codigo.c -lm -o programa
Se ejecuta  de la siguiente forma:  ./programa ***/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>
struct Queries{

  char id[2000];
  char query[2000];
  char time[2000];
  char itemRank[2000];
  char url[2000];
  char disponible;
  struct Queries *sgte;

};
typedef struct Queries QRY;
typedef QRY *QUERY;

struct Matriz_Docu{
  char *Doc;
  char All;
  char Promedio;
```

```
  char Last;
  int cantidad_all_promedio;
  int relevancia_All;
  int relevancia_Promedio;
  int relevancia_Last;
};


struct Matriz_relevancia_Docu{
  char *Doc;
  char All;
  char Promedio;
  char Last;
  int cantidad_all_promedio;
};


struct Matriz_similaridad{
  float valor;
};

struct Queries_con_URLS{

  char *queries[200];
  char *terminos[200];
  char *terminos2[200];
  char url[200];
  char *itemRank[200];
  int cantidad;
  int cantidad_terminos;
  int cantidad_terminos_no_repetidos;
```

```
  struct Queries_con_URLS *sgte;
};
typedef struct Queries_con_URLS QRY_URLS;
typedef QRY_URLS *QUERY_URLS;


struct Queries_con_ID{

  char id[200];
  char time[200];
  char query[2000];
  char *terminos[2000];
  int cant_All;
  int cant_Promedio;
  char LastRel;
  char AllRel[200];
  char promedioAll;
  char *url[2000];
  char *itemRank[2000];
  int cantidad_url;
  int cantidad_terminos;
  int cantidad_terminos_no_repetidos;
  struct Queries_con_ID *sgte;
};
typedef struct Queries_con_ID QRY_ID;
typedef QRY_ID *QUERY_ID;

struct Cluster_Coseno{

  int doc1;
  int doc2;
```

```
  float valor;
  struct Cluster_Coseno *sgte;
};
typedef struct Cluster_Coseno CLUSTER_COSENO;
typedef CLUSTER_COSENO *Cluster_coseno;

struct Cluster_qdsm_promedio{

  int query1;
  int query2;
  char *query1_string;
  char *query2_string;
  float valor;
  struct Cluster_qdsm_promedio *sgte;
};
typedef struct Cluster_qdsm_promedio CLUSTER_QDSM_PROMEDIO;
typedef CLUSTER_QDSM_PROMEDIO *Cluster_Qdsm_Promedio;

struct Cluster_qdsm_last{

  int query1;
  int query2;
  char *query1_string;
  char *query2_string;
  float valor;
  struct Cluster_qdsm_last *sgte;
};
typedef struct Cluster_qdsm_last CLUSTER_QDSM_LAST;
typedef CLUSTER_QDSM_LAST *Cluster_Qdsm_Last;

struct Cluster_qdsm_all{
```

```
  int query1;
  int query2;
  float valor;
  char *query1_string;
  char *query2_string;
  struct Cluster_qdsm_all *sgte;
};
typedef struct Cluster_qdsm_all CLUSTER_QDSM_ALL;
typedef CLUSTER_QDSM_ALL *Cluster_Qdsm_All;

struct Cluster_qdsm{

  int query1;
  int query2;
  float valor;
  char *query1_string;
  char *query2_string;
  struct Cluster_qdsm *sgte;
};
typedef struct Cluster_qdsm CLUSTER_QDSM;
typedef CLUSTER_QDSM *Cluster_Qdsm;


struct iInicial{
  int i;
  struct iInicial *sgte;
};

struct jInicial{
  int j;
```

```
  struct jInicial *sgte;
};

struct Elem{

  int indice;
  int idDoc;
  int Topic;
  float TermDoc[2];
  int MarcadoFil;
  int MarcadoCol;
  float valorMatriz;
  int Marcado;
  int t;
  struct iInicial *IINICIAL;
  struct jInicial *JINICIAL;
};

struct Elem *Arr;
struct Elem **Matriz;
struct Elem **Matriz2;

struct Lista{

  int idDoc[20];
  int t;
  float min;
  int n;
  struct iInicial *IINICIAL;
  struct jInicial *JINICIAL;
  struct Lista *sgte;
```

```
};

struct Cluster{
  struct Lista *L; //esta es la lista de los cluster
  struct Cluster *sgte;
  int n;
};



/* Función que acorta una frase desde una posicion inicial hata una posicion final*/
void substring(char string[], char substring[], int posicion, int largo) {
  int contador = 0;
  while (contador < largo) {
    substring[contador] = string[posicion+contador];
    contador++;
  }
  substring[contador] = '\0';
}

/* Función que inserta la consulta, id, tiempo, url y rank a la estructura*/
QUERY insertar(char aux[],QUERY query, char aux2[],char aux3[],char aux4[],char
aux5[],char aux6[],char aux7[],char aux8[],char aux9[],char aux10[]){


  int i,largo,largo2,largo3,largo4,largo5;
  largo=strlen(aux);
  if(query==NULL){
    query=(QUERY)malloc(sizeof(QRY));
    for(i=0;i<largo;i++){
      if(aux[i]=='\t'){
        substring(aux,aux2,0,i);
        substring(aux,aux3,i+1,largo-i-1);
        break;
```

```
  }
}
largo2=strlen(aux3);
for(i=0;i<largo2;i++){
  if(aux3[i]=='\t'){
    substring(aux3,aux4,0,i);
    substring(aux3,aux5,i+1,largo2-i-1);
    break;
  }
}
largo3=strlen(aux5);
for(i=0;i<largo3;i++){
  if(aux5[i]==':'){
    substring(aux5,aux6,0,i+6);
    substring(aux5,aux7,i+7,largo3-i-7);
    break;
  }
}
largo4=strlen(aux7);
if(largo4==2){
  query->itemRank[0]=' ';
  query->url[0]=' ';
  query->disponible='n';
}else{
  for(i=0;i<largo4;i++){
    if(aux7[i]=='\t'){
      substring(aux7,aux8,0,i);
      substring(aux7,aux9,i+1,largo4-i-1);
      break;
    }
  }
```

```
      strcpy(query->itemRank,aux8);
      strcpy(query->url,aux9);
      query->disponible='s';
    }
    query->sgte=NULL;
    strcpy(query->id,aux2);
    strcpy(query->query,aux4);
    largo5=strlen(aux6);
    for(i=0;i<largo5;i++){
      if(aux6[i]=='\t' || aux6[i]==' ' ){
        substring(aux6,aux10,0,i);
        break;
      }

    }
    strcpy(query->time,aux10);
    return(query);
  }
  else{
    query->sgte=insertar(aux,query->sgte,aux2,aux3,aux4,aux5, aux6,
aux7,aux8,aux9,aux10);
  }
}

/* Función que filtra las consulta que tengan una url */
void solo_validos(QUERY query,char *queries[],char *urls[],char *itemRank[],char
*id[],char *time[],int i,char *all_Documentos[]){

  char queries_aux[210];
  int largo;
  if(query!=NULL){
    if(query->disponible=='s'){
```

```
      strcpy(queries_aux,query->query);
      largo=strlen(queries_aux);
      if(largo==1 && (queries_aux[0]=='-' || queries_aux[0]==')' || queries_aux[0]=='?' ||
queries_aux[0]== '-' || queries_aux[0]==';' || queries_aux[0]==',' || queries_aux[0]=='"' ||
queries_aux[0]==':' || queries_aux[0]=='*' || queries_aux[0]=='+' || queries_aux[0]=='/') ){
        solo_validos(query->sgte,queries,urls,itemRank,id,time,i,all_Documentos);
      }else{
        queries[i]=query->query;
        urls[i]=query->url;
        itemRank[i]=query->itemRank;
        id[i]=query->id;
        time[i]=query->time;
        solo_validos(query->sgte,queries,urls,itemRank,id,time,++i,all_Documentos);}


    }
    else{
      solo_validos(query->sgte,queries,urls,itemRank,id,time,i,all_Documentos);
    }
  }
}



QUERY_ID  urls_con_id(QUERY_ID query_id,char *queries[],char *urls[],char
*itemRank[],char *id[],char *time[],int i,int contador5,char query_aux[],char time_aux[],int
cantidad_url,int j[],struct Matriz_Docu **Documentos,int contador_documentos[],int
posicion[],char auxiliar[],char auxiliar2[],char *auxiliar3[]){


  int contador2=0,contador=0,contador3=0;
  if (contador<=i)
  {
    if (j[0]==0)
    {
      contador=j[0];
```

```
query_id=(QUERY_ID)malloc(sizeof(QRY_ID));
query_id->sgte=NULL;
strcpy(query_id->id,id[contador]);
strcpy(query_id->query,queries[contador]);
strcpy(query_id->time,time[contador]);
strcpy(query_aux,queries[contador]);
strcpy(time_aux,time[contador]);
query_id->cantidad_url=cantidad_url;
query_id->url[contador3]=urls[contador];
query_id->itemRank[contador3]=itemRank[contador];
Documentos[posicion[0]][contador3].Doc=query_id->url[contador3];
contador_documentos[posicion[0]]=contador3;
++posicion[0];
}else{
  if (query_id==NULL)
  {
    contador=j[0];
    contador2=contador;
    query_id=(QUERY_ID)malloc(sizeof(QRY_ID));
    query_id->sgte=NULL;
    cantidad_url=1;
    strcpy(query_id->id,id[contador]);
    strcpy(query_id->query,queries[contador]);
    strcpy(query_id->time,time[contador]);
    strcpy(query_aux,queries[contador]);
    strcpy(time_aux,time[contador]);
    query_id->url[contador3]=urls[contador];
    query_id->itemRank[contador3]=itemRank[contador];
    Documentos[posicion[0]][contador3].Doc=query_id->url[contador3];
    ++contador2;
    ++contador3;
```

```
        if(contador2<i){
          if (contador2==i-1)
          {
            if (strcmp(query_aux,queries[contador2])==0 &&
strcmp(time_aux,time[contador2])==0)
            {
              ++cantidad_url;
              query_id->url[contador3]=urls[contador2];
              query_id->itemRank[contador3]=itemRank[contador2];
              Documentos[posicion[0]][contador3].Doc=query_id->url[contador3];
              ++contador2;
              ++contador3;
               contador_documentos[posicion[0]]=contador3;
            }
          }else{
            while(strcmp(query_aux,queries[contador2])==0 &&
strcmp(time_aux,time[contador2])==0){
              ++cantidad_url;
              query_id->url[contador3]=urls[contador2];
              query_id->itemRank[contador3]=itemRank[contador2];
              Documentos[posicion[0]][contador3].Doc=query_id->url[contador3];
              ++contador2;
              ++contador3;
            }
            contador_documentos[posicion[0]]=contador3;
            ++posicion[0];
          }
        }
        --contador2;
        query_id->cantidad_url=cantidad_url;
        contador=contador2;
        j[0]=contador2;
```

```
       }else{
         query_id->sgte=urls_con_id(query_id-
>sgte,queries,urls,itemRank,id,time,i,contador3,query_aux,time_aux,cantidad_url,j,Docum
entos,contador_documentos,posicion,auxiliar,auxiliar2,auxiliar3);
       }
     }
     return(query_id);
  }
}

void Sacar_terminos_repetidos(QUERY_URLS query){

  int contador_auxiliar=0,contador_aux=0,cont;
  char *auxiliar4[2000],*terminos_aux=NULL;
  if(query!=NULL){
    contador_auxiliar=query->cantidad_terminos;
    for (int i=0;i<contador_auxiliar;i++){
      cont=0;
      terminos_aux=query->terminos[i];
      for (int k=0;k<contador_auxiliar;k++){
        if(strcmp(query->terminos[k],terminos_aux)==0){
          for(int l=0;l<contador_aux;l++){
            if(strcmp(auxiliar4[l],terminos_aux)==0){
              cont++;
              break;
            }
          }
          if(cont==0){
            auxiliar4[contador_aux]=(char *)malloc(sizeof(strlen(terminos_aux)));
            auxiliar4[contador_aux]=terminos_aux;
            query->terminos2[contador_aux]=terminos_aux;
            contador_aux++;
```

```
            cont=0;
          }
        }
      }
      query->cantidad_terminos_no_repetidos=contador_aux;
    }
    Sacar_terminos_repetidos(query->sgte);
  }
}

void sacar_Documentos_repetidos(char *all_Documentos[],char
*Documentos_no_repetidos[],int contador,int conta_Doc[]){

  int contador_aux=0,cont;
  char *auxiliar4[5000],*documentos_aux=NULL;
  for (int i=0;i<contador;i++) {
    cont=0;
    documentos_aux=all_Documentos[i];
    for (int k=0;k<contador;k++){
      if(strcmp(all_Documentos[k],documentos_aux)==0){
        for(int l=0;l<contador_aux;l++){
          if(strcmp(auxiliar4[l],documentos_aux)==0){
            cont++;
            break;
          }
        }
        if(cont==0){
          auxiliar4[contador_aux]=(char *)malloc(sizeof(strlen(documentos_aux)));
          auxiliar4[contador_aux]=documentos_aux;
          Documentos_no_repetidos[contador_aux]=documentos_aux;
          contador_aux++;
          ++conta_Doc[0];
```

```
          cont=0;
        }
      }
    }
  }
}


void poner_todos_documentos(struct Matriz_Docu **Documentos,struct
Matriz_relevancia_Docu **relevancia_Doc,int contador_documentos[],int contador){


  int pos=0;
  for (int i = 0; i < contador; ++i)
  {
    if(contador_documentos[i]==0){
      relevancia_Doc[0][pos].Doc=Documentos[i][0].Doc;
      relevancia_Doc[0][pos].All=Documentos[i][0].All;
      relevancia_Doc[0][pos].Promedio=Documentos[i][0].Promedio;
      relevancia_Doc[0][pos].Last=Documentos[i][0].Last;
      ++pos;
    }else{
      for (int j = 0; j < contador_documentos[i]; ++j)
      {
        relevancia_Doc[0][pos].Doc=Documentos[i][j].Doc;
        relevancia_Doc[0][pos].All=Documentos[i][j].All;
        relevancia_Doc[0][pos].Promedio=Documentos[i][j].Promedio;
        relevancia_Doc[0][pos].Last=Documentos[i][j].Last;
        ++pos;
      }
    }
  }
}
```

```
void sacar_stop_words(QUERY_URLS query,char aux[]){

  char aux2[2000];
  strcpy(aux2,aux);
  if(query!=NULL){
    for (int i = 0; i <query->cantidad_terminos; ++i)
    {
      if(strcmp(aux2,query->terminos[i])==0){
        for (int j= i; j<query->cantidad_terminos; ++j)
        {
          query->terminos[j]=query->terminos[j+1];
        }
        query->terminos[query->cantidad_terminos]=NULL;
        query->cantidad_terminos=query->cantidad_terminos-1;
      }else{
        continue;
      }
    }
    sacar_stop_words(query->sgte,aux);
  }
}


void sacar_stop_words2(QUERY_URLS query,char *aux){

  int largo_auxiliar,largo3;
  char auxiliar[2000],aux3[2000];
  char *unir=NULL,*espacio=" ";
  char *temp[200];
```

```
int x=0,j=0,t=0;
if(query!=NULL){
  while(query->queries[j]!=NULL)
  {
    unir=NULL;
    x=0;
    strcpy(auxiliar,query->queries[j]);
    largo_auxiliar=strlen(auxiliar);
    for (int l = 0; l < largo_auxiliar; ++l)
    {
      if((auxiliar[l]==' ' || auxiliar[l]=='\t' )&& auxiliar[l]!='\0'){
        substring(auxiliar,aux3,0,l);
        temp[x]=(char *)malloc(sizeof(strlen(aux3)));
        strcpy(temp[x],aux3);
        largo_auxiliar=largo_auxiliar-l-1;
        substring(auxiliar,auxiliar,l+1,largo_auxiliar);
        l=0;
        ++x;
      }else{
        if(l+1==largo_auxiliar){
          temp[x]=(char *)malloc(sizeof(strlen(auxiliar)));
          strcpy(temp[x],auxiliar);
          ++x;
        }
      }
    }
    while(t<x){
      ++t;
    }
    for (int m = 0; m < x; ++m)
    {
```

```
  if(strcmp(aux,temp[m])==0){
    for (int k= m; k<x; ++k)
    {
      temp[k]=temp[k+1];
    }
    temp[x]=NULL;
    --x;
  }
}
t=0;
while(t<x){
  ++t;
}
for (int p = 0; p < x; ++p)
{
  if(p==0){
    unir=temp[p];
    strncat(unir,espacio,1);
  }else{
    if(p+1==x){
      largo3=strlen(temp[p]);
      strncat(unir,temp[p],largo3);
    }else{
      largo3=strlen(temp[p]);
      strncat(unir,temp[p],largo3);
      strncat(unir,espacio,1);
    }
  }
}
strcpy(query->queries[j],unir);
++j;
```

```
    }
    sacar_stop_words2(query->sgte,aux);
  }
}


void sacar_stop_words21(QUERY_ID query,char *aux, int total[],char auxiliar[],char
auxiliar2[],char *auxiliar3[],char aux3[],char *temp[],char *unir){


  int largo_auxiliar,largo3;
  int x=0,j=0,t=0;
  unir=NULL;
  char *espacio=" ";
  if(query!=NULL){
    unir=NULL;
    x=0;
    strcpy(auxiliar,query->query);
    largo_auxiliar=strlen(auxiliar);
    for (int l = 0; l < largo_auxiliar; ++l)
    {
      if((auxiliar[l]==' ' || auxiliar[l]=='\t' )&& auxiliar[l]!='\0'){
        substring(auxiliar,aux3,0,l);
        temp[x]=NULL;
        temp[x]=(char *)malloc(largo_auxiliar*sizeof(strlen(aux3)));
        strcpy(temp[x],aux3);
        largo_auxiliar=largo_auxiliar-l-1;
        substring(auxiliar,auxiliar,l+1,largo_auxiliar);
        l=0;
        ++x;
      }else{
        if(l+1==largo_auxiliar){
          temp[x]=NULL;
          temp[x]=(char *)malloc(largo_auxiliar*sizeof(strlen(auxiliar)));
```

```
      strcpy(temp[x],auxiliar);
      ++x;
    }
  }
}
if(x==1){
  strcpy(query->query,temp[0]);
}else{
  while(t<x){
    ++t;
  }
  for (int m = 0; m < x; ++m)
  {
    if(strcmp(aux,temp[m])==0){
      for (int k= m; k<x; ++k)
      {
        temp[k]=temp[k+1];
      }
      temp[x]=NULL;
      --x;
    }
  }
  t=0;
  while(t<x){
    ++t;
  }
  for (int p = 0; p < x; ++p)
  {
    if(p==0){
      unir=temp[p];
      strncat(unir,espacio,1);
```

```
    }else{
      if(p+1==x){
        largo3=strlen(temp[p]);
        strncat(unir,temp[p],largo3);
        }else{
          largo3=strlen(temp[p]);
          strncat(unir,temp[p],largo3);
          strncat(unir,espacio,1);
        }
    }
  }
  strcpy(query->query,unir);
}
int contador_auxiliar=0;
strcpy(auxiliar,query->query);
largo_auxiliar=strlen(auxiliar);
for ( int k = 0; k < largo_auxiliar;k++)
{
  if((auxiliar[k]=='\t' || auxiliar[k]==' ')&& auxiliar[k]!='\0'){
    substring(auxiliar,auxiliar2,0,k);
    auxiliar3[contador_auxiliar]=NULL;
    auxiliar3[contador_auxiliar]=(char *)malloc(largo_auxiliar*sizeof(strlen(auxiliar2)));
    strcpy((char *)auxiliar3[contador_auxiliar],(char *)auxiliar2);
    query->terminos[contador_auxiliar]=auxiliar3[contador_auxiliar];
    contador_auxiliar=contador_auxiliar+1;
    largo_auxiliar=largo_auxiliar-k-1;
    substring(auxiliar,auxiliar,k+1,largo_auxiliar);
    k=0;
  }else{
    if(k+1==largo_auxiliar){
      auxiliar3[contador_auxiliar]=NULL;
```

```
        auxiliar3[contador_auxiliar]=(char *)malloc(largo_auxiliar*sizeof(strlen(auxiliar)));;
        strcpy((char *)auxiliar3[contador_auxiliar],(char *)auxiliar);
        query->terminos[contador_auxiliar]=auxiliar3[contador_auxiliar];
        contador_auxiliar=contador_auxiliar+1;
       }
       else continue;
     }
   }
   query->cantidad_terminos=contador_auxiliar;
   total[0]=total[0]+contador_auxiliar;
   ++j;
   sacar_stop_words21(query->sgte,aux,total,auxiliar,auxiliar2,auxiliar3,aux3,temp,unir);
  }
}


void sacar_terminos(QUERY_ID query,char *terminos[],int cantidad[]){

  if(query!=NULL){
    for (int i = 0; i <query->cantidad_terminos; ++i)
    {
      terminos[cantidad[0]]=query->terminos[i];
      ++cantidad[0];
    }
    sacar_terminos(query->sgte,terminos,cantidad);
  }
}

void sacar_terminos_repetidos(char *terminos[],char *terminos_final[],int cantidad,int
cantidad_final[]){

  int contador_auxiliar=0,cont=0,contador_aux=0;
```

```
  char *terminos_aux=NULL;
  char *auxiliar4[3000];
  contador_auxiliar=cantidad;
  for (int i=0;i<contador_auxiliar;i++) {
    cont=0;
    terminos_aux=terminos[i];
    for (int k=0;k<contador_auxiliar;k++){
      if ( strcmp(terminos[k],terminos_aux)==0){
        for(int l=0;l<contador_aux;l++){
          if (strcmp(auxiliar4[l],terminos_aux)==0){
            cont++;
            break;
          }
        }
        if(cont==0){
          auxiliar4[contador_aux]=(char *)malloc(sizeof(strlen(terminos_aux)));
          auxiliar4[contador_aux]=terminos_aux;
          terminos_final[contador_aux]=terminos_aux;
          contador_aux++;
          cont=0;
        }
      }
    }
  }
  cantidad_final[0]=contador_aux;
}

void sacar_query(QUERY_ID query,char *query_final[],int total_query[]){

  if(query!=NULL){
    query_final[total_query[0]]=(char *)query->query;
```

```
    ++total_query[0];
    sacar_query(query->sgte,query_final,total_query);
  }
}



void probabilidad_relevancia(QUERY_ID query,struct Matriz_Docu **relevancia_Doc,int
paso[]){


  if(query!=NULL){
    int cantidad=0,rank=0;
    int cant_All,cant_Promedio;
    cant_All=0;cant_Promedio=0;
    int valor=0;
    cantidad=query->cantidad_url;
    relevancia_Doc[paso[0]][0].cantidad_all_promedio=cantidad;
    for (int i = 0; i < cantidad; ++i)
    {
      valor=0;
      /* Valor aleatorio para saber si es relevante según la tabla de probabilidad*/
      valor=1+(rand()%100);
      if(cantidad==1){
        rank=atoi(query->itemRank[i]);
        if(rank<=20){
          if(valor<=44){
            query->AllRel[i]='R';
            relevancia_Doc[paso[0]][0].All='R';
            ++cant_All;
          }else{
            query->AllRel[i]='N';
            relevancia_Doc[paso[0]][0].All='N';
          }
```

```
  if(valor<=57){
    query->LastRel='R';
    relevancia_Doc[paso[0]][0].Last='R';
  }else{
    query->LastRel='N';
    relevancia_Doc[paso[0]][0].Last='N';
  }
 }else{
  if(rank<=120){
    if(valor<=51){
      query->AllRel[i]='R';
      relevancia_Doc[paso[0]][0].All='R';
      ++cant_All;
    }else{
      query->AllRel[i]='N';
      relevancia_Doc[paso[0]][0].All='N';
    }
    if(valor<=63){
      query->LastRel='R';
      relevancia_Doc[paso[0]][0].Last='R';
    }else{
      query->LastRel='N';
      relevancia_Doc[paso[0]][0].Last='N';
    }
  }else{
    if(rank<=300){
      if(valor<=54){
        query->AllRel[i]='R';
        relevancia_Doc[paso[0]][0].All='R';
        ++cant_All;
      }else{
```

```
          query->AllRel[i]='N';
          relevancia_Doc[paso[0]][0].All='N';
        }
        if(valor<=65){
          query->LastRel='R';
          relevancia_Doc[paso[0]][0].Last='R';
        }else{
          query->LastRel='N';
          relevancia_Doc[paso[0]][0].Last='N';
        }
      }else{
        if(valor<=47){
          query->AllRel[i]='R';
          relevancia_Doc[paso[0]][0].All='R';
          ++cant_All;
        }else{
          query->AllRel[i]='N';
          relevancia_Doc[paso[0]][0].All='N';
        }
        if(valor<=35){
          query->LastRel='R';
          relevancia_Doc[paso[0]][0].Last='R';
        }else{
          query->LastRel='N';
          relevancia_Doc[paso[0]][0].Last='N';
        }
      }
    }
  }
}else{
  if(i+1==cantidad){
```

```
rank=atoi(query->itemRank[i]);
if(rank<=20){
  if(valor<=44){
    query->AllRel[i]='R';
    relevancia_Doc[paso[0]][i].All='R';
    ++cant_All;
  }else{
    query->AllRel[i]='N';
    relevancia_Doc[paso[0]][i].All='N';
  }
  if(valor<=57){
    query->LastRel='R';
    relevancia_Doc[paso[0]][i].Last='R';
  }else{
    query->LastRel='N';
    relevancia_Doc[paso[0]][i].Last='N';
  }
}else{
  if(rank<=120){
    if(valor<=51){
      query->AllRel[i]='R';
      relevancia_Doc[paso[0]][i].All='R';
      ++cant_All;
    }else{
    query->AllRel[i]='N';
    relevancia_Doc[paso[0]][i].All='N';
    }
    if(valor<=63){
      query->LastRel='R';
      relevancia_Doc[paso[0]][i].Last='R';
    }else{
```

```
    query->LastRel='N';
    relevancia_Doc[paso[0]][i].Last='N';
  }
}else{
  if(rank<=300){
    if(valor<=54){
      query->AllRel[i]='R';
      relevancia_Doc[paso[0]][i].All='R';
      ++cant_All;
    }else{
      query->AllRel[i]='N';
      relevancia_Doc[paso[0]][i].All='N';
    }
    if(valor<=65){
      query->LastRel='R';
      relevancia_Doc[paso[0]][i].Last='R';
    }else{
      query->LastRel='N';
      relevancia_Doc[paso[0]][i].Last='N';
    }
  }else{
    if(valor<=47){
      query->AllRel[i]='R';
      relevancia_Doc[paso[0]][i].All='R';
      ++cant_All;
    }else{
      query->AllRel[i]='N';
      relevancia_Doc[paso[0]][i].All='N';
    }
    if(valor<=35){
      query->LastRel='R';
```

```
            relevancia_Doc[paso[0]][i].Last='R';
            }else{
              query->LastRel='N';
              relevancia_Doc[paso[0]][i].Last='N';
            }
          }
        }
      }
    }else{
      rank=atoi(query->itemRank[i]);
      valor=1+(rand()%100);
      if(rank<=20){
        if(valor<=44){
          query->AllRel[i]='R';
          relevancia_Doc[paso[0]][i].All='R';
          ++cant_All;
        }else{
          query->AllRel[i]='N';
          relevancia_Doc[paso[0]][i].All='R';
        }
      }else{
        if(rank<=120){
          if(valor<=51){
            query->AllRel[i]='R';
            relevancia_Doc[paso[0]][i].All='R';
            ++cant_All;
          }else{
            query->AllRel[i]='N';
            relevancia_Doc[paso[0]][i].All='N';
          }
        }else{
```

```
          if(rank<=300){
            if(valor<=54){
              query->AllRel[i]='R';
              relevancia_Doc[paso[0]][i].All='R';
              ++cant_All;
            }else{
              query->AllRel[i]='N';
              relevancia_Doc[paso[0]][i].All='N';
            }
          }else{
            if(valor<=47){
              query->AllRel[i]='R';
              relevancia_Doc[paso[0]][i].All='R';
              ++cant_All;
            }else{
              query->AllRel[i]='N';
              relevancia_Doc[paso[0]][i].All='N';
            }
          }
        }
      }
    }
  }
}
query->cant_All=cant_All;
relevancia_Doc[paso[0]][0].relevancia_All=cant_All;
int puede=0;
float suma_rel=0;
for (int i = 0; i < cantidad; ++i)
{
  valor=0;
```

```
if(cantidad==1){
  puede=1;
  rank=atoi(query->itemRank[i]);
  if(rank<=20){
    valor=1+(rand()%100);
    if(valor<=44){
      query->promedioAll='R';
      relevancia_Doc[paso[0]][0].Promedio='R';
      ++cant_Promedio;
    }else{
      query->promedioAll='N';
      relevancia_Doc[paso[0]][0].Promedio='N';
    }
  }else{
    if(rank<=120){
      valor=1+rand()%100;
      if(valor<=51){
        query->promedioAll='R';
        relevancia_Doc[paso[0]][0].Promedio='R';
        ++cant_Promedio;
      }else{
        query->promedioAll='N';
        relevancia_Doc[paso[0]][0].Promedio='N';
      }
    }else{
      if(rank<=300){
        valor=1+rand()%100;
        if(valor<=54){
          query->promedioAll='R';
          relevancia_Doc[paso[0]][0].Promedio='R';
          ++cant_Promedio;
```

```
          }else{
            query->promedioAll='N';
            relevancia_Doc[paso[0]][0].Promedio='N';
          }
        }else{
          valor=1+rand()%100;
          if(valor<=47){
            query->promedioAll='R';
            relevancia_Doc[paso[0]][0].Promedio='R';
            ++cant_Promedio;
          }else{
            query->promedioAll='N';
            relevancia_Doc[paso[0]][0].Promedio='N';
          }
        }
      }
    }else{
      rank=atoi(query->itemRank[i]);
      suma_rel=suma_rel+rank;
    }
  }
  if(puede!=1){
    suma_rel=suma_rel/cantidad;
    valor=0;
    char rele;
    if(suma_rel<=20){
      valor=1+(rand()%100);
      if(valor<=44){
        query->promedioAll='R';
        rele='R';
```

```
    ++cant_Promedio;
  }else{
    query->promedioAll='N';
    rele='N';
  }
}else{
  if(suma_rel<=120){
    valor=1+rand()%100;
    if(valor<=51){
      query->promedioAll='R';
      rele='R';
      ++cant_Promedio;
    }else{
      query->promedioAll='N';
     rele='N';
    }
  }else{
    if(suma_rel<=300){
      valor=1+rand()%100;
      if(valor<=54){
        query->promedioAll='R';
        rele='R';
        ++cant_Promedio;
      }else{
        query->promedioAll='N';
        rele='N';
      }
    }else{
      valor=1+rand()%100;
      if(valor<=47){
        query->promedioAll='R';
```

```
            rele='R';
            ++cant_Promedio;
          }else{
            query->promedioAll='N';
            rele='N';
          }
        }
      }
     }
     for (int i = 0; i < cantidad; ++i)
     {
       relevancia_Doc[paso[0]][i].Promedio=rele;
     }
   }
   query->cant_Promedio=cant_Promedio;
   relevancia_Doc[paso[0]][0].relevancia_Promedio=cant_Promedio;
   if(query->LastRel=='R'){
     relevancia_Doc[paso[0]][0].relevancia_Last=1;
   }else{
     relevancia_Doc[paso[0]][0].relevancia_Last=0;
   }
   ++paso[0];
   probabilidad_relevancia(query->sgte,relevancia_Doc,paso);
  }
}


void guardar_relevancia(QUERY_ID query,char matriz[][300],char vector[],char
vector2[],int cant_All[],int cant_Promedio[],int cantidad,int cont,int cantidad_url[]){


  if(query!=NULL){
    for (int i = 0; i < query->cantidad_url; ++i)
    {
```

```
      matriz[cont][i]=query->AllRel[i];
    }
    vector[cont]=query->LastRel;
    vector2[cont]=query->promedioAll;
    cant_All[cont]=query->cant_All;
    cant_Promedio[cont]=query->cant_Promedio;
    cantidad_url[cont]=query->cantidad_url;
    guardar_relevancia(query-
>sgte,matriz,vector,vector2,cant_All,cant_Promedio,cantidad,++cont,cantidad_url);
  }
}

int lcs( char *X, char *Y, int m, int n );
int max(int a, int b);
float max2(float a, float b);

Cluster_coseno insertar_cluster_coseno_Single(Cluster_coseno cluster, int pos_x,int
pos_y,float valor){

  if(cluster==NULL){
    cluster=(Cluster_coseno)malloc(sizeof(CLUSTER_COSENO));
    cluster->sgte=NULL;
    cluster->doc1=pos_x;
    cluster->doc2=pos_y;
    cluster->valor=valor;
  }else{
    cluster->sgte=insertar_cluster_coseno_Single(cluster->sgte,pos_x,pos_y,valor);
  }
  return(cluster);
}

Cluster_Qdsm insertar_cluster_QDSM(Cluster_Qdsm cluster, int pos_x,int pos_y,float
valor,char *queries[]){
```

```
  if(cluster==NULL){
    cluster=(Cluster_Qdsm)malloc(sizeof(CLUSTER_QDSM));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
  }else{
    cluster->sgte=insertar_cluster_QDSM(cluster->sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}

Cluster_Qdsm_Promedio insertar_cluster_promedio_Single(Cluster_Qdsm_Promedio
cluster, int pos_x,int pos_y,float valor,char *queries[]){

  if(cluster==NULL){
    cluster=(Cluster_Qdsm_Promedio)malloc(sizeof(CLUSTER_QDSM_PROMEDIO));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
  }else{
    cluster->sgte=insertar_cluster_promedio_Single(cluster-
>sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}
```

```
Cluster_Qdsm_All insertar_cluster_All_Single(Cluster_Qdsm_All cluster, int pos_x,int
pos_y,float valor,char *queries[]){


  if(cluster==NULL){
    cluster=(Cluster_Qdsm_All)malloc(sizeof(CLUSTER_QDSM_ALL));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
  }else{
    cluster->sgte=insertar_cluster_All_Single(cluster->sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}


Cluster_Qdsm_Last insertar_cluster_Last_Single(Cluster_Qdsm_Last cluster, int pos_x,int
pos_y,float valor,char *queries[]){


  if(cluster==NULL){
    cluster=(Cluster_Qdsm_Last)malloc(sizeof(CLUSTER_QDSM_LAST));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
  }else{
    cluster->sgte=insertar_cluster_Last_Single(cluster->sgte,pos_x,pos_y,valor,queries);
  }
```

```
  return(cluster);
}

Cluster_coseno insertar_cluster_coseno_Complete(Cluster_coseno cluster, int pos_x,int
pos_y,float valor){

  if(cluster==NULL){
    cluster=(Cluster_coseno)malloc(sizeof(CLUSTER_COSENO));
    cluster->sgte=NULL;
    cluster->doc1=pos_x;
    cluster->doc2=pos_y;
    cluster->valor=valor;
  }else{
    cluster->sgte=insertar_cluster_coseno_Complete(cluster->sgte,pos_x,pos_y,valor);
  }
  return(cluster);
}

Cluster_Qdsm insertar_cluster_QDSM_Complete(Cluster_Qdsm cluster, int pos_x,int
pos_y,float valor,char *queries[]){

  if(cluster==NULL){
    cluster=(Cluster_Qdsm)malloc(sizeof(CLUSTER_QDSM));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
  }else{
    cluster->sgte=insertar_cluster_QDSM_Complete(cluster-
>sgte,pos_x,pos_y,valor,queries);
  }
```

```
  return(cluster);
}

Cluster_Qdsm_Promedio insertar_cluster_promedio_Complete(Cluster_Qdsm_Promedio
cluster, int pos_x,int pos_y,float valor,char *queries[]){

  if(cluster==NULL){
    cluster=(Cluster_Qdsm_Promedio)malloc(sizeof(CLUSTER_QDSM_PROMEDIO));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];

  }else{
    cluster->sgte=insertar_cluster_promedio_Complete(cluster-
>sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}

Cluster_Qdsm_All insertar_cluster_All_Complete(Cluster_Qdsm_All cluster, int pos_x,int
pos_y,float valor,char *queries[]){

  if(cluster==NULL){
    cluster=(Cluster_Qdsm_All)malloc(sizeof(CLUSTER_QDSM_ALL));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];
```

```
  }else{
    cluster->sgte=insertar_cluster_All_Complete(cluster->sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}

Cluster_Qdsm_Last insertar_cluster_Last_Complete(Cluster_Qdsm_Last cluster, int
pos_x,int pos_y,float valor,char *queries[]){

  if(cluster==NULL){
    cluster=(Cluster_Qdsm_Last)malloc(sizeof(CLUSTER_QDSM_LAST));
    cluster->sgte=NULL;
    cluster->query1=pos_x;
    cluster->query2=pos_y;
    cluster->valor=valor;
    cluster->query1_string=queries[pos_x];
    cluster->query2_string=queries[pos_y];

  }else{
    cluster->sgte=insertar_cluster_Last_Complete(cluster->sgte,pos_x,pos_y,valor,queries);
  }
  return(cluster);
}



void buscar_QDSM_AllParcial(Cluster_Qdsm_All cluster,int cuenta_All[],int sumador[],int
conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){

  if(cluster!=NULL){
    ++sumador[0];
    if((strcmp(cluster->query1_string,queries[query])==0)|| (strcmp(cluster-
>query2_string,queries[query])==0)){
      recorrido[0]=sumador[0];
```

```
      almacenado[conta_query[0]]=cluster->query1;
      ++conta_query[0];
      almacenado[conta_query[0]]=cluster->query2;
      ++conta_query[0];
      ++cuenta_All[0];
    }
    buscar_QDSM_AllParcial(cluster->sgte,cuenta_All,sumador,conta_query,query,queries,almacenado,recorrido);
  }
}

void buscar_QDSM_All(Cluster_Qdsm_Promedio cluster,int cuenta_Promedio[],int sumador[],int conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){

  if(cluster!=NULL){
    ++sumador[0];
    if((strcmp(cluster->query1_string,queries[query])==0)|| (strcmp(cluster->query2_string,queries[query])==0)){
      recorrido[0]=sumador[0];
      almacenado[conta_query[0]]=cluster->query1;
      ++conta_query[0];
      almacenado[conta_query[0]]=cluster->query2;
      ++conta_query[0];
      ++cuenta_Promedio[0];
    }
    buscar_QDSM_All(cluster->sgte,cuenta_Promedio,sumador,conta_query,query,queries,almacenado,recorrido);
  }
}

void buscar_QDSM_Last(Cluster_Qdsm_Last cluster,int cuenta_Promedio[],int sumador[],int conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){
```

```
  if(cluster!=NULL){
    ++sumador[0];
    if((strcmp(cluster->query1_string,queries[query])==0)|| (strcmp(cluster-
>query2_string,queries[query])==0)){
      recorrido[0]=sumador[0];
      almacenado[conta_query[0]]=cluster->query1;
      ++conta_query[0];
      almacenado[conta_query[0]]=cluster->query2;
      ++conta_query[0];
      ++cuenta_Promedio[0];
    }
    buscar_QDSM_Last(cluster-
>sgte,cuenta_Promedio,sumador,conta_query,query,queries,almacenado,recorrido);
  }
}

void buscar_QDSM_AllParcial_Average(struct Cluster *cluster,int cuenta_All[],int
sumador[],int conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){

  if(cluster!=NULL){
    if(cluster->L!=NULL){
      int cont1,cont2;
      cont1=0;cont2=0;
      ++sumador[0];
      struct iInicial *auxI=cluster->L->IINICIAL;
      struct jInicial *auxJ=cluster->L->JINICIAL;
      while(auxI != NULL){
        ++cont1;
        auxI=auxI->sgte;
      }
      while(auxJ != NULL){
      ++cont2;
      auxJ=auxJ->sgte;
```

```
}
int conta;
conta=cont1+cont2;
char *vector[conta+1];
int vector2[conta+1];
int contador=0;
struct iInicial *auxI2=cluster->L->IINICIAL;
struct jInicial *auxJ2=cluster->L->JINICIAL;
while(auxI2 != NULL){
  vector[contador]=queries[auxI2->i];
  vector2[contador]=auxI2->i;
  ++contador;
  auxI2=auxI2->sgte;
}
while(auxJ2 != NULL){
  vector[contador]=queries[auxJ2->j];
  vector2[contador]=auxJ2->j;
  ++contador;
  auxJ2=auxJ2->sgte;
}
for (int i = 0; i < conta; ++i)
{
  if((strcmp(vector[i],queries[query])==0)){
    recorrido[0]=sumador[0];
    for (int j = 0; j < conta; ++j)
    {
      almacenado[conta_query[0]]=vector2[j];
      ++conta_query[0];
      ++cuenta_All[0];
    }
    break;
```

```
      }
     }
     buscar_QDSM_AllParcial_Average(cluster-
>sgte,cuenta_All,sumador,conta_query,query,queries,almacenado,recorrido);
   }
  }
}


void buscar_QDSM_All_Average(struct Cluster *cluster,int cuenta_Promedio[],int
sumador[],int conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){


 if(cluster!=NULL){
   if(cluster->L!=NULL){
     int cont1,cont2;
     cont1=0;cont2=0;
     ++sumador[0];
     struct iInicial *auxI=cluster->L->IINICIAL;
     struct jInicial *auxJ=cluster->L->JINICIAL;
     while(auxI != NULL){
       ++cont1;
       auxI=auxI->sgte;
     }
     while(auxJ != NULL){
       ++cont2;
       auxJ=auxJ->sgte;
     }
     int conta;
     conta=cont1+cont2;
     char *vector[conta+1];
     int vector2[conta+1];
     int contador=0;
     struct iInicial *auxI2=cluster->L->IINICIAL;
```

```
      struct jInicial *auxJ2=cluster->L->JINICIAL;
      while(auxI2 != NULL){
        vector[contador]=queries[auxI2->i];
        vector2[contador]=auxI2->i;
        ++contador;
        auxI2=auxI2->sgte;
      }
      while(auxJ2 != NULL){
        vector[contador]=queries[auxJ2->j];
        vector2[contador]=auxJ2->j;
        ++contador;
        auxJ2=auxJ2->sgte;
      }
      for (int i = 0; i < conta; ++i)
      {
        if((strcmp(vector[i],queries[query])==0)){
          recorrido[0]=sumador[0];
          for (int j = 0; j < conta; ++j)
          {
            almacenado[conta_query[0]]=vector2[j];
            ++conta_query[0];
            ++cuenta_Promedio[0];
          }
          break;
        }
      }
      buscar_QDSM_All_Average(cluster-
>sgte,cuenta_Promedio,sumador,conta_query,query,queries,almacenado,recorrido);
    }
  }
}
```

```
void buscar_QDSM_Last_Average(struct Cluster *cluster,int cuenta_Promedio[],int
sumador[],int conta_query[],int query,char *queries[],int almacenado[],int recorrido[]){


  if(cluster!=NULL){
    if(cluster->L!=NULL){
      int cont1,cont2;
      cont1=0;cont2=0;
      ++sumador[0];
      struct iInicial *auxI=cluster->L->IINICIAL;
      struct jInicial *auxJ=cluster->L->JINICIAL;
      while(auxI != NULL){
        ++cont1;
        auxI=auxI->sgte;
      }
      while(auxJ != NULL){
        ++cont2;
        auxJ=auxJ->sgte;
      }
      int conta;
      conta=cont1+cont2;
      char *vector[conta+1];
      int vector2[conta+1];
      int contador=0;
      struct iInicial *auxI2=cluster->L->IINICIAL;
      struct jInicial *auxJ2=cluster->L->JINICIAL;
      while(auxI2 != NULL){
        vector[contador]=queries[auxI2->i];
        vector2[contador]=auxI2->i;
        ++contador;
        auxI2=auxI2->sgte;
      }
      while(auxJ2 != NULL){
```

```
      vector[contador]=queries[auxJ2->j];
      vector2[contador]=auxJ2->j;
      ++contador;
      auxJ2=auxJ2->sgte;
    }
    for (int i = 0; i < conta; ++i)
    {
      if((strcmp(vector[i],queries[query])==0)){
        recorrido[0]=sumador[0];
        for (int j = 0; j < conta; ++j)
        {
          almacenado[conta_query[0]]=vector2[j];
          ++conta_query[0];
          ++cuenta_Promedio[0];
        }
        break;
      }
    }
    buscar_QDSM_Last_Average(cluster-
>sgte,cuenta_Promedio,sumador,conta_query,query,queries,almacenado,recorrido);
   }
  }
}


void buscar_Coseno_AllParcial(Cluster_coseno cluster,struct Matriz_relevancia_Docu
**todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int
cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int
contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    int esta=0,esta2=0,cas=0,sale=0,sale2=0;
    ++sumador[0];
    for (int i = 0; i < contador_query; ++i)
```

```
  {
    if (sale==1)
    {
      break;
    }else{
      cas=contador_documentos[queries_almacenadas[i]];
      for (int j = 0; j < cas; ++j)
      {
        if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][cluster-
>doc1].Doc)==0){
          esta=1;
          recorrido[0]=sumador[0];
          if(todo_Doc[0][cluster->doc1].All=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
            ++no_relevante[0];
          }
          sale=1;
          break;
        }
      }
    }

  }
  if(esta==0){
    if(todo_Doc[0][cluster->doc1].All=='R'){
      ++relevantes_no_query[0];
    }
  }
  for (int i = 0; i < contador_query; ++i)
  {
    if (sale2==1)
```

```
    {
      break;
    }else{
      cas=contador_documentos[queries_almacenadas[i]];
      for (int j = 0; j < cas; ++j)
      {
        if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][cluster-
>doc2].Doc)==0){
          esta2=1;
          recorrido[0]=sumador[0];
          if(todo_Doc[0][cluster->doc2].All=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
            ++no_relevante[0];
          }
          sale2=1;
          break;
        }
      }
    }
  }
  if(esta2==0){
    if(todo_Doc[0][cluster->doc2].All=='R'){
      ++relevantes_no_query[0];
    }
  }
  if(esta==1 && esta2==0){
    ++cuenta_coseno[0];
  }else{
    if(esta==1 && esta2==0){
      ++cuenta_coseno[0];
    }else{
```

```
      if(esta==1 && esta2==1){
        ++cuenta_coseno[0];
      }
     }
   }
   buscar_Coseno_AllParcial(cluster->sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no_relevante);
  }
}



void buscar_Coseno_All(Cluster_coseno cluster,struct Matriz_relevancia_Docu **todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    int esta=0,esta2=0,cas=0,sale=0,sale2=0;
    ++sumador[0];
    for (int i = 0; i < contador_query; ++i)
    {
      if (sale==1)
      {
        break;
      }else{
        cas=contador_documentos[queries_almacenadas[i]];
        for (int j = 0; j < cas; ++j)
        {
          if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][cluster->doc1].Doc)==0){
            esta=1;
            recorrido[0]=sumador[0];
```

```
          if(todo_Doc[0][cluster->doc1].Promedio=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
            ++no_relevante[0];
          }
          sale=1;
          break;
        }
      }
    }
  }
  if(esta==0){
    if(todo_Doc[0][cluster->doc1].Promedio=='R'){
      ++relevantes_no_query[0];
    }
  }
  for (int i = 0; i < contador_query; ++i)
  {
    if (sale2==1)
    {
      break;
    }else{
      cas=contador_documentos[queries_almacenadas[i]];
      for (int j = 0; j < cas; ++j)
      {
        if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][cluster-
>doc2].Doc)==0){
          esta2=1;
          recorrido[0]=sumador[0];
          if(todo_Doc[0][cluster->doc2].Promedio=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
```

```
            ++no_relevante[0];
          }
          sale2=1;
          break;
        }
      }
    }
  }
  if(esta2==0){
    if(todo_Doc[0][cluster->doc2].Promedio=='R'){
      ++relevantes_no_query[0];
    }
  }
  if(esta==1 && esta2==0){
    ++cuenta_coseno[0];
  }else{
    if(esta==1 && esta2==0){
      ++cuenta_coseno[0];
    }else{
      if(esta==1 && esta2==1){
        ++cuenta_coseno[0];
      }
    }
  }
  buscar_Coseno_All(cluster->sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no_relevante);
 }
}
```

```
void buscar_Coseno_Last(Cluster_coseno cluster,struct Matriz_relevancia_Docu
**todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int
cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int
contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    int esta=0,esta2=0,cas=0,sale=0,sale2=0;
    ++sumador[0];
    for (int i = 0; i < contador_query; ++i)
    {
      if (sale==1)
      {
        break;
      }else{
        cas=contador_documentos[queries_almacenadas[i]]-1;
        if(strcmp(Documentos[queries_almacenadas[i]][cas].Doc,todo_Doc[0][cluster-
>doc1].Doc)==0){
          esta=1;
          recorrido[0]=sumador[0];
          if(todo_Doc[0][cluster->doc1].Last=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
            ++no_relevante[0];
          }
          sale=1;
          break;
        }
      }
    }
    if(esta==0){
      if(todo_Doc[0][cluster->doc1].All=='R'){
        ++relevantes_no_query[0];
      }
```

```
    }
    for (int i = 0; i < contador_query; ++i)
    {
      if (sale2==1)
      {
        break;
      }else{
        cas=contador_documentos[queries_almacenadas[i]]-1;
        if(strcmp(Documentos[queries_almacenadas[i]][cas].Doc,todo_Doc[0][cluster-
>doc2].Doc)==0){
          esta2=1;
          recorrido[0]=sumador[0];
          if(todo_Doc[0][cluster->doc2].Last=='R'){
            ++cantidad_relevancia_coseno[0];
          }else{
            ++no_relevante[0];
          }
          sale2=1;
          break;
        }
      }
    }
    if(esta2==0){
      if(todo_Doc[0][cluster->doc2].Last=='R'){
        ++relevantes_no_query[0];
      }
    }
    if(esta==1 && esta2==0){
      ++cuenta_coseno[0];
    }else{
      if(esta==1 && esta2==0){
        ++cuenta_coseno[0];
```

```
      }else{
        if(esta==1 && esta2==1){
          ++cuenta_coseno[0];
        }
      }
    }
    buscar_Coseno_Last(cluster->sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no_relevante);
  }
}



void buscar_Coseno_AllParcial_Average(struct Cluster *cluster,struct Matriz_relevancia_Docu **todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    if(cluster->L!=NULL){
      int cont1,cont2;
      cont1=0;cont2=0;
      struct iInicial *auxI=cluster->L->IINICIAL;
      struct jInicial *auxJ=cluster->L->JINICIAL;
      while(auxI != NULL){
        ++cont1;
        auxI=auxI->sgte;
      }
      while(auxJ != NULL){
        ++cont2;
        auxJ=auxJ->sgte;
      }
```

```
int conta;
conta=cont1+cont2;
int vector2[conta+1];
int contador=0;
struct iInicial *auxI2=cluster->L->IINICIAL;
struct jInicial *auxJ2=cluster->L->JINICIAL;
while(auxI2 != NULL){
  vector2[contador]=auxI2->i;
  ++contador;
  auxI2=auxI2->sgte;
}
while(auxJ2 != NULL){
  vector2[contador]=auxJ2->j;
  ++contador;
  auxJ2=auxJ2->sgte;
}
int esta=0,cas=0,sale=0,sale2=0;
++sumador[0];
for (int i = 0; i < contador_query; ++i)
{
  if (sale==1)
  {
    break;
  }else{
    cas=contador_documentos[queries_almacenadas[i]];
    for (int j = 0; j < cas; ++j)
    {
      if(sale2==1){
        break;
      }else{
        for (int k = 0; k < conta; ++k)
```

```
            {

if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][vector2[k]].Doc)==0){
                esta=1;
                ++cuenta_coseno[0];
                recorrido[0]=sumador[0];
                if(todo_Doc[0][vector2[k]].All=='R'){
                  ++cantidad_relevancia_coseno[0];
                }else{
                  ++no_relevante[0];
                }
                sale=1;
                sale2=1;
                break;
              }
            }
          }
        }
      }
      if(esta==0){
        for (int k = 0; k < conta; ++k)
        {
          if(todo_Doc[0][vector2[k]].All=='R'){
            ++relevantes_no_query[0];
          }
        }
      }
      buscar_Coseno_AllParcial_Average(cluster-
>sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_rel
evancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no
_relevante);
    }
```

```
  }
}



void buscar_Coseno_All_Average(struct Cluster *cluster,struct Matriz_relevancia_Docu
**todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int
cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int
contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    if(cluster->L!=NULL){
      int cont1,cont2;
      cont1=0;cont2=0;
      struct iInicial *auxI=cluster->L->IINICIAL;
      struct jInicial *auxJ=cluster->L->JINICIAL;
      while(auxI != NULL){
        ++cont1;
        auxI=auxI->sgte;
      }
      while(auxJ != NULL){
        ++cont2;
        auxJ=auxJ->sgte;
      }
      int conta;
      conta=cont1+cont2;
      int vector2[conta+1];
      int contador=0;
      struct iInicial *auxI2=cluster->L->IINICIAL;
      struct jInicial *auxJ2=cluster->L->JINICIAL;
      while(auxI2 != NULL){
        vector2[contador]=auxI2->i;
        ++contador;
        auxI2=auxI2->sgte;
```

```
    }
    while(auxJ2 != NULL){
      vector2[contador]=auxJ2->j;
      ++contador;
      auxJ2=auxJ2->sgte;
    }
    int esta=0,cas=0,sale=0,sale2=0;
    ++sumador[0];
    for (int i = 0; i < contador_query; ++i)
    {
      if (sale==1)
      {
        break;
      }else{
        cas=contador_documentos[queries_almacenadas[i]];
        for (int j = 0; j < cas; ++j)
        {
          if(sale2==1){
            break;
          }else{
            for (int k = 0; k < conta; ++k)
            {

if(strcmp(Documentos[queries_almacenadas[i]][j].Doc,todo_Doc[0][vector2[k]].Doc)==0){
                esta=1;
                ++cuenta_coseno[0];
                recorrido[0]=sumador[0];
                if(todo_Doc[0][vector2[k]].Promedio=='R'){
                  ++cantidad_relevancia_coseno[0];
                }else{
                   ++no_relevante[0];
                }
```

```
                sale=1;
                sale2=1;
                break;
              }
            }
          }
        }
      }
      if(esta==0){
        for (int k = 0; k < conta; ++k)
        {
          if(todo_Doc[0][vector2[k]].Promedio=='R'){
          ++relevantes_no_query[0];
          }
        }

      }
      buscar_Coseno_All_Average(cluster->sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no_relevante);
    }
  }
}



void buscar_Coseno_Last_Average(struct Cluster *cluster,struct Matriz_relevancia_Docu **todo_Doc,struct Matriz_Docu **Documentos,int contador_documentos[],int sumador[],int cuenta_coseno[],int cantidad_relevancia_coseno[],int relevantes_no_query[],int contador_query,int queries_almacenadas[],int recorrido[],int no_relevante[]){


  if(cluster!=NULL){
    if(cluster->L!=NULL){
```

```
int cont1,cont2;
cont1=0;cont2=0;
struct iInicial *auxI=cluster->L->IINICIAL;
struct jInicial *auxJ=cluster->L->JINICIAL;
while(auxI != NULL){
  ++cont1;
  auxI=auxI->sgte;
}
while(auxJ != NULL){
  ++cont2;
  auxJ=auxJ->sgte;
}
int conta;
conta=cont1+cont2;
int vector2[conta+1];
int contador=0;
struct iInicial *auxI2=cluster->L->IINICIAL;
struct jInicial *auxJ2=cluster->L->JINICIAL;
while(auxI2 != NULL){
  vector2[contador]=auxI2->i;
  ++contador;
  auxI2=auxI2->sgte;
}
while(auxJ2 != NULL){
  vector2[contador]=auxJ2->j;
  ++contador;
  auxJ2=auxJ2->sgte;
}
int esta=0,cas=0,sale=0,sale2=0;
++sumador[0];
for (int i = 0; i < contador_query; ++i)
```

```
    {
      if (sale==1)
      {
        break;
      }else{
        cas=contador_documentos[queries_almacenadas[i]]-1;
        for (int j = 0; j < cas; ++j)
        {
          if(sale2==1){
            break;
          }else{
            for (int k = 0; k < conta; ++k)
            {

if(strcmp(Documentos[queries_almacenadas[i]][cas].Doc,todo_Doc[0][vector2[k]].Doc)==0)
{
                esta=1;
                ++cuenta_coseno[0];
                recorrido[0]=sumador[0];
                if(todo_Doc[0][vector2[k]].Last=='R'){
                  ++cantidad_relevancia_coseno[0];
                }else{
                  ++no_relevante[0];
                }
                sale=1;
                sale2=1;
                break;
              }
            }
          }
        }
      }
```

```
      }
      if(esta==0){
        for (int k = 0; k < conta; ++k)
        {
          if(todo_Doc[0][vector2[k]].Last=='R'){
            ++relevantes_no_query[0];
          }
        }
      }
      buscar_Coseno_Last_Average(cluster-
>sgte,todo_Doc,Documentos,contador_documentos,sumador,cuenta_coseno,cantidad_rel
evancia_coseno,relevantes_no_query,contador_query,queries_almacenadas,recorrido,no
_relevante);
    }
  }
}




struct Cluster *InsertarC(struct Cluster *C, struct Lista *H, int n){


  if(C==NULL)
  {
    C=(struct Cluster *)malloc(sizeof(struct Cluster));
    C->L=H;
    C->n=n;
    C->sgte=NULL;
  }
  else C->sgte=InsertarC(C->sgte,H,n);
  return(C);
}
```

```
void juntarMatriz(struct Elem **M, struct Elem **M2,int iE,int jE,int n){


 int K;
 int L;
 for (int i = 0; i < n; ++i)
 {
  for (int j = 0; j < n && i>j; ++j)
  {
    K=i;
    L=j;
    if (i == jE+1)
    {
      K=iE;
    }
    if (i > jE+1 && i <= iE)
    {
      K=i-1;
    }
    if (j == jE+1)
    {
      L=iE;
    }
    if (j > jE+1 && j <= iE)
    {
      L=j-1;
    }
    if (K>L)
    {
      M2[i][j].valorMatriz=M[K][L].valorMatriz;
      M2[i][j].IINICIAL=M[K][L].IINICIAL;//iInicial y jInicial son para no perder la referencia
de los documentos
      M2[i][j].JINICIAL=M[K][L].JINICIAL;
```

```
      }
      else{

        M2[i][j].valorMatriz=M[L][K].valorMatriz;
        M2[i][j].IINICIAL=M[L][K].JINICIAL;//iInicial y jInicial son para no perder la referencia de los documentos
        M2[i][j].JINICIAL=M[L][K].IINICIAL;//el i y el j estan cambiados para arreglar el problema que nos surgía al final
      }
    }
  }
}

void colocarNoMarcadosYMarcados(struct Elem **M, struct Elem **M2,int iE,int jE,int n){

  for (int i = 0; i < n; ++i)
  {
    for (int j = 0; j < n && i>j; ++j)
    {
      if (i != iE && i != jE && j != jE && j != iE)
      {
        if (i > iE && j < jE)
        {
          M2[i-1][j].valorMatriz=M[i][j].valorMatriz;
          M2[i-1][j].IINICIAL=M[i][j].IINICIAL;
          M2[i-1][j].JINICIAL=M[i][j].JINICIAL;
        }
        if (i > iE && j > jE)
        {
          M2[i-1][j-1].valorMatriz=M[i][j].valorMatriz;
          M2[i-1][j-1].IINICIAL=M[i][j].IINICIAL;
          M2[i-1][j-1].JINICIAL=M[i][j].JINICIAL;
```

```
  }
  if (i < iE && j > jE)
  {
    M2[i][j-1].valorMatriz=M[i][j].valorMatriz;
    M2[i][j-1].IINICIAL=M[i][j].IINICIAL;
    M2[i][j-1].JINICIAL=M[i][j].JINICIAL;
  }
  if (i < iE && j < jE)
  {
    M2[i][j].valorMatriz=M[i][j].valorMatriz;
    M2[i][j].IINICIAL=M[i][j].IINICIAL;
    M2[i][j].JINICIAL=M[i][j].JINICIAL;
  }
}
else//Marcados
{
  if (!((i == jE || i == iE ) && (j == jE || j == iE)))
  {
    if (i == jE)
    {
      struct iInicial *iAux=M[i][j].IINICIAL;
      while(iAux->sgte != NULL)
      {
        iAux=iAux->sgte;
      }
      iAux->sgte=M[i+1][j].IINICIAL;
      M2[i][j].valorMatriz=(M[i][j].valorMatriz + M[i+1][j].valorMatriz) / 2;
      M2[i][j].IINICIAL=M[i][j].IINICIAL;
      M2[i][j].JINICIAL=M[i][j].JINICIAL;
    }
    if (j == jE)
```

```
        {
          struct jInicial *jAux=M[i][j].JINICIAL;
          while(jAux->sgte != NULL)
          {
            jAux=jAux->sgte;
          }
          jAux->sgte=M[i][j+1].JINICIAL;
          M2[i-1][j].valorMatriz=(M[i][j].valorMatriz + M[i][j+1].valorMatriz) / 2;
          M2[i-1][j].IINICIAL=M[i][j].IINICIAL;
          M2[i-1][j].JINICIAL=M[i][j].JINICIAL;
        }


      }
     }
    }
  }
}

struct Lista *CrearLista(struct Elem **M,struct Cluster *C,struct Lista *H,struct Elem *A,int
N,int t, float min){


  for (int i = 0; i < N; ++i)
  {
    for (int j = 0; j < N && i>j ; ++j)
    {
      if (M[i][j].Marcado == 1 && H == NULL)
      {
        H=(struct Lista *)malloc(sizeof(struct Lista));
        H->t=t;
        H->min=M[i][j].valorMatriz;
        H->sgte=NULL;
        H->IINICIAL=M[i][j].IINICIAL;
```

```
      H->JINICIAL=M[i][j].JINICIAL;
      M[i][j].Marcado = 2;
     }
     else{
      if (M[i][j].Marcado == 1 && H != NULL)
      {
       C=InsertarC(C,H,1);
       H=NULL;
       H=(struct Lista *)malloc(sizeof(struct Lista));
       H->t=t;
       H->min=M[i][j].valorMatriz;
       H->IINICIAL=M[i][j].IINICIAL;
       H->JINICIAL=M[i][j].JINICIAL;
       H->sgte=NULL;
       M[i][j].Marcado = 2;
      }
     }
    }
  }
  return(H);
}



void experimentos(char *query_final[],char *terminos_final[],int total_query,int
total_termino,char matriz_All[][300],char vector_Last[],char vector_promedioAll[],int
cant_url[],int cant_All[],int cant_Promedio[],int total_Doc,int no_repetido_Doc,char
*all_Documentos[],char *Documentos_no_repetidos[],struct Matriz_Docu
**Documentos,struct Matriz_relevancia_Docu **todo_Doc,int contador_documentos[]){


  int largo_auxiliar,contador_auxiliar=0;
  char auxiliar[5000],auxiliar2[5000],*auxiliar3[5000];
  int l=0;
  char *queries[2000];
```

```
  struct Matriz_similaridad **matriz;
  struct Matriz_similaridad **matriz_Doc1;
  struct Matriz_similaridad **matriz_similaridad_Doc;
  struct Matriz_similaridad **matriz_similaridad;
  /* Memoria para las matrices*/
  if(total_query>=total_termino){
    matriz=(struct Matriz_similaridad **)malloc((total_query+30)*sizeof(struct
Matriz_similaridad *));
    matriz_Doc1=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
    matriz_similaridad_Doc=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
    matriz_similaridad=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
    for(int i=0;i<total_query+1;i++){
      matriz_similaridad[i]=(struct Matriz_similaridad *)malloc((total_query+1)*sizeof(struct
Matriz_similaridad));
    }
    for(int i=0;i<total_query+1;i++){
      matriz[i]=(struct Matriz_similaridad *)malloc((total_query+20)*sizeof(struct
Matriz_similaridad));
    }
    for(int i=0;i<total_Doc+1;i++){
      matriz_Doc1[i]=(struct Matriz_similaridad *)malloc((total_Doc+1)*sizeof(struct
Matriz_similaridad));
      matriz_similaridad_Doc[i]=(struct Matriz_similaridad
*)malloc((total_Doc+1)*sizeof(struct Matriz_similaridad));
    }

  }else{
    matriz=(struct Matriz_similaridad **)malloc((total_termino+1)*sizeof(struct
Matriz_similaridad *));
    matriz_Doc1=(struct Matriz_similaridad **)malloc(total_termino*sizeof(struct
Matriz_similaridad *));
    matriz_similaridad_Doc=(struct Matriz_similaridad **)malloc(total_termino*sizeof(struct
Matriz_similaridad *));
```

```
   matriz_similaridad=(struct Matriz_similaridad **)malloc(total_termino*sizeof(struct
Matriz_similaridad *));

   for(int i=0;i<total_query+1;i++){

   matriz_similaridad[i]=(struct Matriz_similaridad *)malloc((total_termino+1)*sizeof(struct
Matriz_similaridad));

   }

   for(int i=0;i<total_termino+1;i++){

   matriz[i]=(struct Matriz_similaridad *)malloc((total_termino+1)*sizeof(struct
Matriz_similaridad));

   }

   for(int i=0;i<total_Doc+1;i++){

     matriz_Doc1[i]=(struct Matriz_similaridad *)malloc((total_Doc+1)*sizeof(struct
Matriz_similaridad));

     matriz_similaridad_Doc[i]=(struct Matriz_similaridad
*)malloc((total_Doc+1)*sizeof(struct Matriz_similaridad));

   }


 }

 for (int i = 0; i < total_query; ++i)

 {

   for (int j = 0; j < total_termino; ++j)

   {

     matriz[i][j].valor=0;

   }

 }

/**MATRIZ de los documenots se llena con ceros **/

 for (int i = 0; i < total_Doc; ++i)

 {

   for (int j = 0; j < no_repetido_Doc; ++j)

   {

      matriz_Doc1[i][j].valor=0;


   }
```

```
  }
/* Diagonal de la matriz se llenan con 1 */
  for (int i = 0; i < total_Doc; ++i)
  {
    for (int j = 0; j < no_repetido_Doc; ++j)
    {
      if (strcmp(all_Documentos[i],Documentos_no_repetidos[j])==0)
      {
        matriz_Doc1[i][j].valor=1;
      }
    }
  }
  float logaritmo_Doc[no_repetido_Doc+1];
  for (int i = 0; i < no_repetido_Doc; ++i)
  {
    int cont2=0;
    float division2=0;
    for (int j = 0; j < total_Doc; ++j)
    {
      if(matriz_Doc1[j][i].valor>0){
        ++cont2;
      }
    }

    division2=total_Doc/cont2;
    logaritmo_Doc[i]=log10(division2);
  }
  for (int i = 0; i < total_Doc; ++i)
  {
    for (int j = 0; j < no_repetido_Doc; ++j)
    {
```

```
    if (matriz_Doc1[i][j].valor>0)
    {
      matriz_Doc1[i][j].valor=logaritmo_Doc[j];
    }
  }
}
for (int i = 0; i < total_Doc; ++i)
{
  for (int j = 0; j < total_Doc; ++j)
  {
    if (i==j)
    {
      matriz_similaridad_Doc[i][j].valor=1;
    }else{
      matriz_similaridad_Doc[i][j].valor=0;
    }
  }
}
float suma2;
for (int i = 0; i < total_Doc; ++i)
{
  for (int j = 0; j < total_Doc; ++j)
  {
    if (i!=j)
    {
      suma2=0;
      for (int k = 0; k < no_repetido_Doc; ++k)
      {
        suma2=matriz_Doc1[i][k].valor*matriz_Doc1[j][k].valor+suma2;
      }
      matriz_similaridad_Doc[i][j].valor=suma2/100;
```

```
    }
   }
  }
/**-------------------------------------------------------**/
  for (int k = 0; k < total_query; ++k)
  {
   contador_auxiliar=0;
   strcpy(auxiliar,query_final[k]);
   largo_auxiliar=strlen(auxiliar);
   for ( int t = 0; t < largo_auxiliar;t++)
   {
    if((auxiliar[t]=='\t' || auxiliar[t]==' ')&& auxiliar[t]!='\0'){
     substring(auxiliar,auxiliar2,0,t);
     auxiliar3[contador_auxiliar]=(char *)malloc(largo_auxiliar*sizeof(strlen(auxiliar2)));
     strcpy((char *)auxiliar3[contador_auxiliar],(char *)auxiliar2);
     queries[contador_auxiliar]=auxiliar3[contador_auxiliar];
     contador_auxiliar=contador_auxiliar+1;
     largo_auxiliar=largo_auxiliar-t-1;
     substring(auxiliar,auxiliar,t+1,largo_auxiliar);
     t=0;
    }else{
     if(t+1==largo_auxiliar){
      auxiliar3[contador_auxiliar]=(char *)malloc(largo_auxiliar*sizeof(strlen(auxiliar)));;
      strcpy((char *)auxiliar3[contador_auxiliar],(char *)auxiliar);
      queries[contador_auxiliar]=auxiliar3[contador_auxiliar];
      contador_auxiliar=contador_auxiliar+1;
     }
     else continue;
    }
   }
   for (int l = 0; l < total_termino; ++l)
```

```
  {
    int valor=0;
    for (int h = 0; h < contador_auxiliar; ++h)
    {
      if (strcmp(queries[h],terminos_final[l])==0)
      {
        ++valor;
        matriz[k][l].valor=valor;
      }
    }
  }
}
float logaritmo[total_termino+1];
for (int i = 0; i < total_termino; ++i)
{
  int cont=0;
  float division=0;
  for (int j = 0; j < total_query; ++j)
  {
    if(matriz[j][i].valor>0){
      ++cont;
    }
  }
  division=total_query/cont;
  logaritmo[i]=log10(division);
}
for (int i = 0; i < total_query; ++i)
{
  for (int j = 0; j < total_termino; ++j)
  {
    if (matriz[i][j].valor>0)
```

```
    {
      matriz[i][j].valor=logaritmo[j];
    }
  }
}
for (int i = 0; i < total_query; ++i)
{
  for (int j = 0; j < total_query; ++j)
  {
    if (i==j)
    {
      matriz_similaridad[i][j].valor=1;
    }else{
      matriz_similaridad[i][j].valor=0;
    }
  }
}
float suma;
for (int i = 0; i < total_query; ++i)
{
  for (int j = 0; j < total_query; ++j)
  {
    if (i!=j)
    {
      suma=0;
      for (int k = 0; k < total_termino; ++k)
      {
        suma=matriz[i][k].valor*matriz[j][k].valor+suma;
      }
      matriz_similaridad[i][j].valor=suma/100;
    }
```

```
  }
 }
 char
*unir_All[total_query+1],*unir_All2[total_query+1],*unir_All3[total_query+1],*unir_Last[total
_query+1],*unir_promedioAll[total_query+1],*unir=NULL,auxin[0],auxi2[0],auxi3[0];;
 int largo_last=0;
 for (int i = 0; i < total_query; ++i)
 {
  auxin[0]=vector_Last[i];
  largo_last=strlen(auxin);
  unir_Last[i]=(char *)malloc(sizeof(strlen(auxin)));
  strcpy(unir_Last[i],auxin);
 }
 int cantidad_Promedio=0;
 for (int i = 0; i < total_query; ++i)
 {
  cantidad_Promedio=0;
  if(cant_url[i]==1){
   auxi2[0]=vector_promedioAll[i];
   largo_last=strlen(auxi2);
   unir_promedioAll[i]=(char *)malloc(sizeof(strlen(auxi2)));
   strcpy(unir_promedioAll[i],auxi2);
  }else{
   for (int j = 0; j < cant_url[i]; ++j)
   {
    if(j==0){
     if(vector_promedioAll[i]=='R'){
      auxi2[0]='R';
      ++cantidad_Promedio;
     }else{
      auxi2[0]='N';
     }
```

```
        largo_last=strlen(auxi2);
        unir_All2[i]=(char *)malloc(sizeof(strlen(auxi2)));
        strcpy(unir_All2[i],auxi2);
        unir=unir_All2[i];
      }else{
        auxi2[0]=vector_promedioAll[i];
        if(auxi2[0]=='R'){
          ++cantidad_Promedio;
        }
        largo_last=strlen(auxi2);
        unir_All2[i]=(char *)malloc(sizeof(strlen(auxi2)));
        strcpy(unir_All2[i],auxi2);
        strncat(unir,unir_All2[i],largo_last);
      }
      unir_promedioAll[i]=(char *)malloc(sizeof(strlen(unir)));
      unir_promedioAll[i]=NULL;
      unir_promedioAll[i]=unir;
    }
  }
}
int largo_last2=0;
char auxi4[0],auxi5[0],*unir3=NULL;
unir=NULL;
for (int i = 0; i < total_query; ++i)
{
  if(cant_url[i]==-11){
    printf("M:%c\n",matriz_All[i][0]);
    auxi5[0]=matriz_All[i][0];
    unir3=auxi5;
    largo_last=strlen(auxi5);
    substring(auxi4,auxi5,0,3);
```

```
    largo_last2=strlen(unir3);
    unir_All[i]=(char *)malloc(sizeof(auxi5));
    strcpy(unir_All[i],auxi5);
  }else{
    for (int j = 0; j < cant_url[i]; ++j)
    {
      if(j==0){
        auxi3[0]=matriz_All[i][j];
        largo_last2=strlen(auxi3);
        unir_All3[i]=(char *)malloc(sizeof(strlen(auxi3)));
        strcpy(unir_All3[i],auxi3);
        unir=unir_All3[i];
      }else{
        auxi3[0]=matriz_All[i][j];
        largo_last2=strlen(auxi3);
        strncat(unir,auxi3,largo_last2);
      }
      unir_All[i]=(char *)malloc(sizeof(strlen(unir)));
      unir_All[i]=NULL;
      unir_All[i]=unir;
    }
  }
}
int relevancia_All[total_query+1],relevancia_Last[total_query+1],suma_rel=0;
for (int i = 0; i < total_query; ++i)
{
  if(vector_Last[i]=='R'){
    relevancia_Last[i]=1;
  }else{
    relevancia_Last[i]=0;
  }
```

```
  }
  for (int i = 0; i < total_query; ++i)
  {
    suma_rel=0;
    for (int j = 0; j < total_query; ++j)
    {
      if(cant_url[i]==1){
        if(matriz_All[i][0]=='R'){
          suma_rel=1;
          }
      }else{
        if(matriz_All[i][j]=='R'){
          suma_rel=suma_rel+1;
        }
      }
      relevancia_All[i]=suma_rel;
    }
  }
  /**QDSM ALLParcial **/
  float valor_lcs=0,maximo=0;
  char relevancia_A[200],relevancia_B[200],*A=NULL,*B=NULL;
  int largo_A,largo_B,contador_inicial=33,conta2=0;
  struct Matriz_similaridad **QDSM_All;
  struct Matriz_similaridad **QDSM_promedioAll;
  struct Matriz_similaridad **QDSM_Last;
  QDSM_All=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
  QDSM_promedioAll=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
  QDSM_Last=(struct Matriz_similaridad **)malloc(total_query*sizeof(struct
Matriz_similaridad *));
  for(int i=0;i<total_query+1;i++){
```

```
   QDSM_All[i]=(struct Matriz_similaridad *)malloc((total_query+1)*sizeof(struct
Matriz_similaridad));
   QDSM_promedioAll[i]=(struct Matriz_similaridad *)malloc((total_query+1)*sizeof(struct
Matriz_similaridad));
   QDSM_Last[i]=(struct Matriz_similaridad *)malloc((total_query+1)*sizeof(struct
Matriz_similaridad));
 }
 for (int i = 0; i < total_query; ++i)
 {
  for (int j = 0; j < total_query; ++j)
  {
    contador_inicial=33;
    if(i<j){
      if(matriz_similaridad[i][j].valor!=0){
        largo_A=strlen(unir_All[i]);
        largo_B=strlen(unir_All[j]);
        strcpy(relevancia_A,unir_All[i]);
        strcpy(relevancia_B,unir_All[j]);
        for (int k = 0; k < largo_A; ++k)
        {
          if(relevancia_A[k]=='N'){
            if(contador_inicial==82 || contador_inicial==78 ){
              ++contador_inicial;
            }
            relevancia_A[k]=contador_inicial;
            ++contador_inicial;
            conta2=contador_inicial;
          }
        }
        for (int l = 0; l < largo_B; ++l)
        {
          if(relevancia_B[l]=='N'){
            if(conta2==82 || conta2==78 ){
```

```
            ++conta2;
          }
          relevancia_B[l]=conta2;
          ++conta2;
        }
      }
      A=relevancia_A;
      B=relevancia_B;
      valor_lcs=lcs(A,B,strlen(A),strlen(B));
      maximo=max(relevancia_All[i],relevancia_All[j]);
      QDSM_All[i][j].valor=(matriz_similaridad[i][j].valor+valor_lcs)/(1+maximo);
      QDSM_All[j][i].valor=QDSM_All[i][j].valor;
    }else{
      QDSM_All[i][j].valor=0;
    }
   }else{
    if(i==j){
      QDSM_All[i][j].valor=0;
    }
   }
  }
 }
/** QDSM LAST**/
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j){
        if(matriz_similaridad[i][j].valor!=0){
          if(vector_Last[i]=='R' &&  vector_Last[j]=='R'){
            valor_lcs=1;
```

```
        }else{
          if(vector_Last[i]=='R' &&  vector_Last[j]!='R'){
            valor_lcs=0;
          }else{
            if(vector_Last[i]!='R' &&  vector_Last[j]=='R'){
              valor_lcs=0;
            }else{
              valor_lcs=0;
            }
          }
        }
        maximo=max(relevancia_Last[i],relevancia_Last[j]);
        QDSM_Last[i][j].valor=(matriz_similaridad[i][j].valor+valor_lcs)/(1+maximo);
        QDSM_Last[j][i].valor=QDSM_Last[i][j].valor;
      }else{
        QDSM_Last[i][j].valor=0;
      }
    }else{
      if(i==j){
        QDSM_Last[i][j].valor=0;
      }
    }
  }
 }
/** QDSM ALL**/
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i!=j){
        contador_inicial=33;
```

```
if(matriz_similaridad[i][j].valor!=0){
  largo_A=strlen(unir_promedioAll[i]);
  largo_B=strlen(unir_promedioAll[j]);
  strcpy(relevancia_A,unir_promedioAll[i]);
  strcpy(relevancia_B,unir_promedioAll[j]);
  for (int k = 0; k < largo_A; ++k)
  {
    if(relevancia_A[k]=='N'){
      if(contador_inicial==82 || contador_inicial==78 ){
        ++contador_inicial;
      }
      relevancia_A[k]=contador_inicial;
      ++contador_inicial;
      conta2=contador_inicial;
    }
  }
  for (int l = 0; l < largo_B; ++l)
  {
    if(relevancia_B[l]=='N'){
      if(conta2==82 || conta2==78 ){
        ++conta2;
      }
      relevancia_B[l]=conta2;
      ++conta2;
    }
  }
  A=relevancia_A;
  B=relevancia_B;
  valor_lcs=lcs(A,B,strlen(A),strlen(B));
  if(vector_promedioAll[i]=='R' &&  vector_promedioAll[j]=='R'){
    maximo=max(cant_url[i],cant_url[j]);
```

```
        }else{
          if(vector_promedioAll[i]=='R' &&  vector_promedioAll[j]!='R'){
            maximo=cant_url[i];
          }else{
            if(vector_promedioAll[i]!='R' &&  vector_promedioAll[j]=='R'){
              maximo=cant_url[j];
            }else{
              maximo=0;
            }
          }
        }
        QDSM_promedioAll[i][j].valor=(matriz_similaridad[i][j].valor+valor_lcs)/(1+maximo);
        QDSM_promedioAll[j][i].valor=QDSM_promedioAll[i][j].valor;
      }else{
        QDSM_promedioAll[i][j].valor=0;
      }
    }else{
      if(i==j){
        QDSM_promedioAll[i][j].valor=0;
      }
    }
  }
}
/*CLUSTERS SINGLE LINK*/
  int marcado_x[total_Doc+2],marcado_y[total_Doc+2];
  for (int i = 0; i < total_Doc; ++i)
  {
    marcado_x[i]=0;
    marcado_y[i]=0;
  }
/*CLUSTER COSENO*/
```

```
 float min=-1;
 int pos_x=0,pos_y=0;
 Cluster_coseno cluster_coseno_Single=NULL;
 while(1){
  min=-1;
  for (int i = 0; i < total_Doc; ++i)
  {
   for (int j = 0; j < total_Doc; ++j)
   {
    if(i<j && i!=j){
     if(marcado_x[i]!=1 && marcado_x[j]!=1){
      if(min>matriz_similaridad_Doc[i][j].valor){
      }else{
       if(min<matriz_similaridad_Doc[i][j].valor){
        min=matriz_similaridad_Doc[i][j].valor;
        pos_x=i;
        pos_y=j;
       }
      }
     }
    }
   }
  }
  if(min==0){
   break;
  }else{
   marcado_x[pos_x]=1;
   marcado_y[pos_y]=1;

cluster_coseno_Single=insertar_cluster_coseno_Single(cluster_coseno_Single,pos_x,pos
_y,min);
  }
```

```
}
for (int i = 0; i < total_query; ++i)
{
  marcado_x[i]=0;
  marcado_y[i]=0;
}
Cluster_Qdsm cluster_QDSM=NULL;
min=-1;
pos_x=0,pos_y=0;
while(1){
  min=-1;
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j && i!=j){
        if(marcado_x[i]!=1 && marcado_y[j]!=1){
          if(min>matriz_similaridad[i][j].valor){
          }else{
            if(min<matriz_similaridad[i][j].valor){
              min=matriz_similaridad[i][j].valor;
              pos_x=i;
              pos_y=j;
            }
          }
        }
      }
    }
  }
  if(min==0){
    break;
```

```
  }else{
    marcado_x[pos_x]=1;
    marcado_y[pos_y]=1;
    cluster_QDSM=insertar_cluster_QDSM(cluster_QDSM,pos_x,pos_y,min,query_final);
  }
}
for (int i = 0; i < total_query; ++i)
{
  marcado_x[i]=0;
  marcado_y[i]=0;
}
/*Cluster ALL */
Cluster_Qdsm_Promedio cluster_promedio_Single=NULL;
while(1){
  min=-1;
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j && i!=j){
        if(marcado_x[i]!=1 && marcado_y[j]!=1){
          if(min>QDSM_promedioAll[i][j].valor){
          }else{
            if(min<QDSM_promedioAll[i][j].valor){
              min=QDSM_promedioAll[i][j].valor;
              pos_x=i;
              pos_y=j;
            }
          }
        }
      }
```

```
      }
    }
    if(min==0){
      break;
    }else{
      marcado_x[pos_x]=1;
      marcado_y[pos_y]=1;

cluster_promedio_Single=insertar_cluster_promedio_Single(cluster_promedio_Single,pos
_x,pos_y,min,query_final);
    }
  }
  for (int i = 0; i < total_query; ++i)
  {
    marcado_x[i]=0;
    marcado_y[i]=0;
  }
  /*Cluster ALLParcial */
  Cluster_Qdsm_All cluster_All_Single=NULL;
  while(1){
    min=-1;
    for (int i = 0; i < total_query; ++i)
    {
      for (int j = 0; j < total_query; ++j)
      {
        if(i<j && i!=j){
          if(marcado_x[i]!=1 && marcado_y[j]!=1){
            if(min>QDSM_All[i][j].valor){
            }else{
              if(min<=QDSM_All[i][j].valor){
                min=QDSM_All[i][j].valor;
                pos_x=i;
```

```
            pos_y=j;
          }
        }
      }
    }
  }
  if(min==0){
    break;
  }else{
    marcado_x[pos_x]=1;
    marcado_y[pos_y]=1;

cluster_All_Single=insertar_cluster_All_Single(cluster_All_Single,pos_x,pos_y,min,query_f
inal);
  }
 }
 for (int i = 0; i < total_query; ++i)
 {
   marcado_x[i]=0;
  marcado_y[i]=0;
 }
 /*Cluster Last */
 Cluster_Qdsm_Last cluster_Last_Single=NULL;
 while(1){
  min=-1;
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j && i!=j){
        if(marcado_x[i]!=1 && marcado_y[j]!=1){
```

```
          if(min>QDSM_Last[i][j].valor){
          }else{
            if(min<=QDSM_Last[i][j].valor){
              min=QDSM_Last[i][j].valor;
              pos_x=i;
              pos_y=j;
            }
          }
        }
      }
    }
   if(min<0.0001){
     break;
   }else{
     marcado_x[pos_x]=1;
     marcado_y[pos_y]=1;
     if(pos_x!=0 && pos_y!=132){

cluster_Last_Single=insertar_cluster_Last_Single(cluster_Last_Single,pos_x,pos_y,min,qu
ery_final);
     }
   }
  }
/*** FIN CLUSTERS SINGLE LINK**/


/**  CLUSTERS COMPLETE LINK **/
  for (int i = 0; i < total_Doc; ++i)
  {
    marcado_x[i]=0;
    marcado_y[i]=0;
  }
```

```
/*CLUSTER COSENO*/
  pos_x=0,pos_y=0;
  Cluster_coseno cluster_coseno_Complete=NULL;
  while(1){
    min=10000;
    pos_x=-1;
    for (int i = 0; i < total_Doc; ++i)
    {
      for (int j = 0; j < total_Doc; ++j)
      {
        if(i<j && i!=j){
          if(marcado_x[i]!=1 && marcado_x[j]!=1){
            if(min>matriz_similaridad_Doc[i][j].valor && matriz_similaridad_Doc[i][j].valor!=0){
              min=matriz_similaridad_Doc[i][j].valor;
              pos_x=i;
              pos_y=j;
            }
          }
        }
      }
    }
    if(pos_x==-1){
      break;
    }else{
      marcado_x[pos_x]=1;
      marcado_y[pos_y]=1;

cluster_coseno_Complete=insertar_cluster_coseno_Complete(cluster_coseno_Complete,
pos_x,pos_y,min);
    }
  }
  for (int i = 0; i < total_query; ++i)
```

```
{
  marcado_x[i]=0;
  marcado_y[i]=0;
}
/*Cluster All*/
Cluster_Qdsm_Promedio cluster_promedio_Complete=NULL;
while(1){
  min=10000;
  pos_x=-1;
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j && i!=j){
        if(marcado_x[i]!=1 && marcado_y[j]!=1){
          if(min>QDSM_promedioAll[i][j].valor && QDSM_promedioAll[i][j].valor!=0){
            min=QDSM_promedioAll[i][j].valor;
            pos_x=i;
            pos_y=j;
          }
        }
      }
    }
  }
  if(pos_x==-1){
    break;
  }else{
    marcado_x[pos_x]=1;
    marcado_y[pos_y]=1;

cluster_promedio_Complete=insertar_cluster_promedio_Complete(cluster_promedio_Com
plete,pos_x,pos_y,min,query_final);
```

```
  }
}
for (int i = 0; i < total_query; ++i)
{
  marcado_x[i]=0;
  marcado_y[i]=0;
}
/*Cluster AllParcial*/
Cluster_Qdsm_All cluster_All_Complete=NULL;
while(1){
  min=10000;
  pos_x=-1;
  for (int i = 0; i < total_query; ++i)
  {
    for (int j = 0; j < total_query; ++j)
    {
      if(i<j && i!=j){
        if(marcado_x[i]!=1 && marcado_y[j]!=1){
          if(min>QDSM_All[i][j].valor && QDSM_All[i][j].valor!=0){
            min=QDSM_All[i][j].valor;
            pos_x=i;
            pos_y=j;
          }
        }
      }
    }
  }
  if(pos_x==-1){
    break;
  }else{
    marcado_x[pos_x]=1;
```

```
      marcado_y[pos_y]=1;

cluster_All_Complete=insertar_cluster_All_Complete(cluster_All_Complete,pos_x,pos_y,m
in,query_final);
    }
  }
  for (int i = 0; i < total_query; ++i)
  {
    marcado_x[i]=0;
    marcado_y[i]=0;
  }
  /*CLuster Last*/
  Cluster_Qdsm_Last cluster_Last_Complete=NULL;
  while(1){
    min=10000;
    pos_x=-1;
    for (int i = 0; i < total_query; ++i)
    {
      for (int j = 0; j < total_query; ++j)
      {
        if(i<j && i!=j){
          if(marcado_x[i]!=1 && marcado_y[j]!=1){
            if(min>QDSM_Last[i][j].valor && QDSM_Last[i][j].valor!=0){
              min=QDSM_Last[i][j].valor;
              pos_x=i;
              pos_y=j;
            }
          }
        }
      }
    }
    if(pos_x==-1){
```

```
      break;
    }else{
      marcado_x[pos_x]=1;
      marcado_y[pos_y]=1;
      if(pos_x!=0 && pos_y!=132){

cluster_Last_Complete=insertar_cluster_Last_Complete(cluster_Last_Complete,pos_x,pos_y,min,query_final);
      }
    }
  }
/**  FIN CLUSTERS COMPLETE LINK **/


/* CLUSTERS AVERAGE  */
/* CLUSTER COSENO AVERAGE */
  int N,i,n,t,j,k;
  if(total_query>=total_Doc){
    N=total_query;
  }else{
    N=total_Doc;
  }
  struct Lista *Lista_Coseno_Average=NULL;
  struct Cluster *Cluster_Coseno_Average=NULL;
  Arr=(struct Elem*)malloc(N*sizeof(struct Elem));
  for(int i=0;i<N;i++)
  {
    Arr[i].MarcadoFil=0;
    Arr[i].MarcadoCol=0;
    Arr[i].t=0;
  }
  Matriz=(struct Elem **)malloc(N*sizeof(struct Elem));
  for(i=0;i<N ;i++)
```

```
  Matriz[i]=(struct Elem*)malloc(N*sizeof(struct Elem));
Matriz2=(struct Elem **)malloc(N*sizeof(struct Elem));
for(i=0;i<N ;i++)
  Matriz2[i]=(struct Elem*)malloc(N*sizeof(struct Elem));
N=total_Doc;
for(int i=0;i<N;i++)
  for(int j=0;j<N && i>j; j++)
  {
    struct iInicial *iAux=NULL;
    struct jInicial *jAux=NULL;
    iAux=(struct iInicial *)malloc(sizeof(struct iInicial));
    jAux=(struct jInicial *)malloc(sizeof(struct jInicial));
    iAux->i = i;
    iAux->sgte = NULL;
    jAux->j = j;
    jAux->sgte = NULL;
    Matriz[i][j].valorMatriz =matriz_similaridad_Doc[i][j].valor;
    Matriz[i][j].Marcado=0;
    Matriz[i][j].IINICIAL=iAux;
    Matriz[i][j].JINICIAL=jAux;
  }
for(int i=0;i<N;i++)//inicializando matriz2
  for(int j=0;j<N && i>j; j++)
  {
    Matriz2[i][j].valorMatriz=0;
    Matriz2[i][j].Marcado=0;
  }
n=total_Doc;
l=-1;
k=-1;
t=1;
```

```
min=-1;
while(1)
{
 l=-1;
 min=-1;
 for(i=0;i<n;i++)
  for(j=0;j<n && i>j ;j++)
   {
    if(Matriz[i][j].valorMatriz > min && Matriz[i][j].valorMatriz > 0)
    {
     l=i;
     k=j;
     min=Matriz[i][j].valorMatriz;
    }
    else continue;
   }
 if(l==-1){
  break;
 }
 else if(l!=-1)
 {
  if(k-l != 1 && k-l != -1){
   juntarMatriz(Matriz,Matriz2,l,k,n);
   for (i = 0; i < n; ++i)
   {
    for (j = 0; j < n && i>j; ++j)
    {
     Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
     Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
     Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
    }
```

```
      }
      continue;
    }
    if(Arr[l].MarcadoFil==0)
    {
      Arr[l].MarcadoFil=1;
      Arr[l].t=t;
    }
    if(Arr[k].MarcadoCol==0)
    {
      Arr[k].MarcadoCol=1;
      Arr[k].t=t;
    }
    Matriz[l][k].Marcado=1;
    Matriz[l][k].t=t;
    Lista_Coseno_Average=NULL;

Lista_Coseno_Average=CrearLista(Matriz,Cluster_Coseno_Average,Lista_Coseno_Avera
ge, Arr, n, t, min);
    if (Lista_Coseno_Average != NULL)
    {
      Cluster_Coseno_Average=InsertarC(Cluster_Coseno_Average,
Lista_Coseno_Average,1);
    }
    colocarNoMarcadosYMarcados(Matriz,Matriz2,l,k,n);
    for (i = 0; i < n-1; ++i)
    {
      for (j = 0; j < n-1 && i>j; ++j)
      {
        Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
        Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
        Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
      }
```

```
      }
      n--;
    }
  }
/* Cluster AllParcial */
  N=total_query;
  for(int i=0;i<N;i++)
  {
  Arr[i].MarcadoFil=0;
  Arr[i].MarcadoCol=0;
  Arr[i].t=0;
  }
  struct Lista *Lista_All_Average=NULL;
  struct Cluster *Cluster_All_Average=NULL;
  for(int i=0;i<N;i++)
    for(int j=0;j<N && i>j; j++)
    {
      Matriz[i][j].valorMatriz =0;
      Matriz[i][j].Marcado=0;
      Matriz[i][j].IINICIAL=NULL;
      Matriz[i][j].JINICIAL=NULL;
    }
  for(int i=0;i<N;i++)
    for(int j=0;j<N && i>j; j++)
    {
      struct iInicial *iAux=NULL;
      struct jInicial *jAux=NULL;
      iAux=(struct iInicial *)malloc(sizeof(struct iInicial));
      jAux=(struct jInicial *)malloc(sizeof(struct jInicial));
      iAux->i = i;
      iAux->sgte = NULL;
```

```
    jAux->j = j;
    jAux->sgte = NULL;
    Matriz[i][j].valorMatriz =QDSM_All[i][j].valor;
    Matriz[i][j].Marcado=0;
    Matriz[i][j].IINICIAL=iAux;
    Matriz[i][j].JINICIAL=jAux;
  }
for(int i=0;i<N;i++)
  for(int j=0;j<N && i>j; j++)
  {
    Matriz2[i][j].valorMatriz=0;
    Matriz2[i][j].Marcado=0;
  }
n=total_query;
l=-1;
k=-1;
t=1;
min=-1;
while(1)
{
  l=-1;
  min=-1;
  for(i=0;i<n;i++)
    for(j=0;j<n && i>j ;j++)
    {
      if(Matriz[i][j].valorMatriz > min && Matriz[i][j].valorMatriz > 0)
      {
        l=i;
        k=j;
        min=Matriz[i][j].valorMatriz;
      }
```

```
    else continue;
  }
if(l==-1){
  break;
}
else if(l!=-1)
{
  if(k-l != 1 && k-l != -1){
    juntarMatriz(Matriz,Matriz2,l,k,n);
    for (i = 0; i < n; ++i)
    {
      for (j = 0; j < n && i>j; ++j)
      {
        Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
        Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
        Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
      }
    }
    continue;
  }
  if(Arr[l].MarcadoFil==0)
  {
    Arr[l].MarcadoFil=1;
    Arr[l].t=t;
  }
  if(Arr[k].MarcadoCol==0)
  {
    Arr[k].MarcadoCol=1;
    Arr[k].t=t;
  }
  Matriz[l][k].Marcado=1;
```

```
      Matriz[l][k].t=t;
      Lista_All_Average=NULL;
      Lista_All_Average=CrearLista(Matriz,Cluster_All_Average,Lista_All_Average, Arr, n, t,
min);
      if (Lista_All_Average != NULL)
      {
        Cluster_All_Average=InsertarC(Cluster_All_Average, Lista_All_Average,1);
      }
      colocarNoMarcadosYMarcados(Matriz,Matriz2,l,k,n);
      for (i = 0; i < n-1; ++i)
      {
        for (j = 0; j < n-1 && i>j; ++j)
        {
          Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
          Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
          Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
        }
      }
      n--;
    }
  }
/* Cluster All */
  for(int i=0;i<N;i++)
  {
    Arr[i].MarcadoFil=0;
    Arr[i].MarcadoCol=0;
    Arr[i].t=0;
  }
  struct Lista *Lista_AllPromedio_Average=NULL;
  struct Cluster *Cluster_AllPromedio_Average=NULL;
  for(int i=0;i<N;i++)
    for(int j=0;j<N && i>j; j++)
```

```
  {
    Matriz[i][j].valorMatriz =0;
    Matriz[i][j].Marcado=0;
    Matriz[i][j].IINICIAL=NULL;
    Matriz[i][j].JINICIAL=NULL;
  }
for(int i=0;i<N;i++)
  for(int j=0;j<N && i>j; j++)
  {
    struct iInicial *iAux=NULL;
    struct jInicial *jAux=NULL;
    iAux=(struct iInicial *)malloc(sizeof(struct iInicial));
    jAux=(struct jInicial *)malloc(sizeof(struct jInicial));
    iAux->i = i;
    iAux->sgte = NULL;
    jAux->j = j;
    jAux->sgte = NULL;
    Matriz[i][j].valorMatriz =QDSM_promedioAll[i][j].valor;
    Matriz[i][j].Marcado=0;
    Matriz[i][j].IINICIAL=iAux;
    Matriz[i][j].JINICIAL=jAux;
  }
for(int i=0;i<N;i++)
  for(int j=0;j<N && i>j; j++)
  {
    Matriz2[i][j].valorMatriz=0;
    Matriz2[i][j].Marcado=0;
  }
n=total_query;
l=-1;
k=-1;
```

```
t=1;
min=-1;
while(1)
{
  l=-1;
  min=-1;
  for(i=0;i<n;i++)
    for(j=0;j<n && i>j ;j++)
    {
      if(Matriz[i][j].valorMatriz > min && Matriz[i][j].valorMatriz > 0)
      {
        l=i;
        k=j;
        min=Matriz[i][j].valorMatriz;
      }
      else continue;
    }
  if(l==-1){
    break;
  }
  else if(l!=-1)
  {
    if(k-l != 1 && k-l != -1){
      juntarMatriz(Matriz,Matriz2,l,k,n);
      for (i = 0; i < n; ++i)
      {
        for (j = 0; j < n && i>j; ++j)
        {
          Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
          Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
          Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
```

```
      }
     }
     continue;
    }
    if(Arr[l].MarcadoFil==0)
    {
     Arr[l].MarcadoFil=1;
     Arr[l].t=t;
    }
    if(Arr[k].MarcadoCol==0)
    {
     Arr[k].MarcadoCol=1;
     Arr[k].t=t;
    }
    Matriz[l][k].Marcado=1;
    Matriz[l][k].t=t;
    Lista_AllPromedio_Average=NULL;

Lista_AllPromedio_Average=CrearLista(Matriz,Cluster_AllPromedio_Average,Lista_AllPro
medio_Average,Arr,n,t,min);
    if (Lista_AllPromedio_Average != NULL)
    {
      Cluster_AllPromedio_Average=InsertarC(Cluster_AllPromedio_Average,
Lista_AllPromedio_Average,1);
    }
    colocarNoMarcadosYMarcados(Matriz,Matriz2,l,k,n);
    for (i = 0; i < n-1; ++i)
    {
     for (j = 0; j < n-1 && i>j; ++j)
     {
       Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
       Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
       Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
```

```
        }
       }
       n--;
     }
   }
 /* CLUSTER LAST */
   for(int i=0;i<N;i++)
   {
     Arr[i].MarcadoFil=0;
     Arr[i].MarcadoCol=0;
     Arr[i].t=0;
   }
   struct Lista *Lista_Last_Average=NULL;
   struct Cluster *Cluster_Last_Average=NULL;
   for(int i=0;i<N;i++)
     for(int j=0;j<N && i>j; j++)
     {
       Matriz[i][j].valorMatriz =0;
       Matriz[i][j].Marcado=0;
       Matriz[i][j].IINICIAL=NULL;
       Matriz[i][j].JINICIAL=NULL;
     }
   for(int i=0;i<N;i++)
     for(int j=0;j<N && i>j; j++)
     {
       struct iInicial *iAux=NULL;
       struct jInicial *jAux=NULL;
       iAux=(struct iInicial *)malloc(sizeof(struct iInicial));
       jAux=(struct jInicial *)malloc(sizeof(struct jInicial));
       iAux->i = i;
       iAux->sgte = NULL;
```

```
    jAux->j = j;
    jAux->sgte = NULL;
    Matriz[i][j].valorMatriz =QDSM_Last[i][j].valor;
    Matriz[i][j].Marcado=0;
    Matriz[i][j].IINICIAL=iAux;
    Matriz[i][j].JINICIAL=jAux;
  }
for(int i=0;i<N;i++)
  for(int j=0;j<N && i>j; j++)
  {
    Matriz2[i][j].valorMatriz=0;
    Matriz2[i][j].Marcado=0;
  }
n=total_query;
l=-1;
k=-1;
t=1;
min=-1;
while(1)
{
  l=-1;
  min=-1;
  for(i=0;i<n;i++)
    for(j=0;j<n && i>j ;j++)
    {
      if(Matriz[i][j].valorMatriz > min && Matriz[i][j].valorMatriz > 0)
      {
        l=i;
        k=j;
        min=Matriz[i][j].valorMatriz;
      }
```

```
    else continue;
  }
if(l==-1){
  break;
}
else if(l!=-1)
{
  if(k-l != 1 && k-l != -1){
    juntarMatriz(Matriz,Matriz2,l,k,n);
    for (i = 0; i < n; ++i)
    {
      for (j = 0; j < n && i>j; ++j)
      {
        Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
        Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
        Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
      }
    }
    continue;
  }
  if(Arr[l].MarcadoFil==0)
  {
    Arr[l].MarcadoFil=1;
    Arr[l].t=t;
  }
  if(Arr[k].MarcadoCol==0)
  {
    Arr[k].MarcadoCol=1;
    Arr[k].t=t;
  }
  Matriz[l][k].Marcado=1;
```

```
      Matriz[l][k].t=t;
      Lista_Last_Average=NULL;
      Lista_Last_Average=CrearLista(Matriz,Cluster_Last_Average,Lista_Last_Average,
Arr, n, t, min);
      if (Lista_Last_Average != NULL)
      {
        Cluster_Last_Average=InsertarC(Cluster_Last_Average, Lista_Last_Average,1);
      }
      colocarNoMarcadosYMarcados(Matriz,Matriz2,l,k,n);
      for (i = 0; i < n-1; ++i)
      {
        for (j = 0; j < n-1 && i>j; ++j)
        {
          Matriz[i][j].valorMatriz=Matriz2[i][j].valorMatriz;
          Matriz[i][j].IINICIAL=Matriz2[i][j].IINICIAL;
          Matriz[i][j].JINICIAL=Matriz2[i][j].JINICIAL;
        }
      }
      n--;
    }
  }
/* FIN CLUSTERS AVERAGE */


/* Comprobacion clustering coseno-AllParcial  */
  int
sumador[0],cuenta_All[0],queries_almacenadas[500],conta_query[0],recorrido[0],cuenta_P
romedio[0],cuenta_Last[0],no_relevante[0];
  int rele_total_All=0,rele_total_Promedio=0;
  k=0;
  int cuenta_coseno[0],cantidad_relevancia_coseno[0],relevantes_no_query[0];
  int
resumen_relevantes_All,resumen_relevantes_coseno_All,recorrido_All,recorrido_coseno_
All,aux_recorrido,aux_relevantes,total_All_queries;
```

```
 resumen_relevantes_All=0;resumen_relevantes_coseno_All=0;total_All_queries=0;
 recorrido_All=0;recorrido_coseno_All=0;aux_recorrido=0;aux_relevantes=0;
 for (k = 0; k < total_query;++k)
 {
  cuenta_All[0]=0;
  sumador[0]=0;
  conta_query[0]=0;
  recorrido[0]=0;

buscar_QDSM_AllParcial(cluster_All_Single,cuenta_All,sumador,conta_query,k,query_fin
al,queries_almacenadas,recorrido);
  if(recorrido[0]!=0){
    rele_total_All=0;
    for (int i = 0; i < conta_query[0]; ++i)
    {
      rele_total_All=rele_total_All+Documentos[k][0].relevancia_All;
    }
    aux_recorrido=recorrido[0];
    aux_relevantes=rele_total_All;
    no_relevante[0]=0;
    cuenta_coseno[0]=0;
    sumador[0]=0;
    cantidad_relevancia_coseno[0]=0;
    relevantes_no_query[0]=0;
    recorrido[0]=0;

buscar_Coseno_AllParcial(cluster_coseno_Single,todo_Doc,Documentos,contador_docu
mentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,conta
_query[0],queries_almacenadas,recorrido,no_relevante);
    if(recorrido[0]!=0){
      recorrido_coseno_All=recorrido_coseno_All+recorrido[0];

resumen_relevantes_coseno_All=resumen_relevantes_coseno_All+cantidad_relevancia_c
oseno[0];
```

```
        resumen_relevantes_All=resumen_relevantes_All+aux_relevantes;
        recorrido_All=recorrido_All+aux_recorrido;
        ++total_All_queries;
      }
    }
  }
/*  Comprobacion clustering coseno-All*/
  int
resumen_relevantes_Promedio,resumen_relevantes_coseno_Promedio,recorrido_Promed
io,recorrido_coseno_Promedio,total_Promedio_queries;

resumen_relevantes_Promedio=0;resumen_relevantes_coseno_Promedio=0;total_Prome
dio_queries=0;

recorrido_Promedio=0;recorrido_coseno_Promedio=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for ( k= 0; k < total_query; ++k)
  {
    for (int i = 0; i < conta_query[0]; ++i)
    {
      queries_almacenadas[i]=0;
    }
    conta_query[0]=0;
    sumador[0]=0;
    recorrido[0]=0;
    cuenta_Promedio[0]=0;

buscar_QDSM_All(cluster_promedio_Single,cuenta_Promedio,sumador,conta_query,k,qu
ery_final,queries_almacenadas,recorrido);
    if(recorrido[0]!=0){
      rele_total_Promedio=0;
      for (int i = 0; i < conta_query[0]; ++i)
      {
```

```
rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Promedio+contad
or_documentos[k];
      }
      aux_recorrido=recorrido[0];
      aux_relevantes=rele_total_Promedio;
      cuenta_coseno[0]=0;
      sumador[0]=0;
      cantidad_relevancia_coseno[0]=0;
      relevantes_no_query[0]=0;
      recorrido[0]=0;
      no_relevante[0]=0;

buscar_Coseno_All(cluster_coseno_Single,todo_Doc,Documentos,contador_documentos,
sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,conta_query[0
],queries_almacenadas,recorrido,no_relevante);
      if(recorrido[0]!=0){
        recorrido_coseno_Promedio=recorrido_coseno_Promedio+recorrido[0];

resumen_relevantes_coseno_Promedio=resumen_relevantes_coseno_Promedio+cantida
d_relevancia_coseno[0];
        resumen_relevantes_Promedio=resumen_relevantes_Promedio+aux_relevantes;
        recorrido_Promedio=recorrido_Promedio+aux_recorrido;
        ++total_Promedio_queries;
      }
    }
  }
/*  Comprobacion clustering coseno-Last  */
  int
resumen_relevantes_Last,resumen_relevantes_coseno_Last,recorrido_Last,recorrido_cos
eno_Last,total_Last_queries;
  resumen_relevantes_Last=0;resumen_relevantes_coseno_Last=0;total_Last_queries=0;
  recorrido_Last=0;recorrido_coseno_Last=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for (k= 0; k < total_query; ++k)
```

```
 {
   for (int i = 0; i < conta_query[0]; ++i)
   {
     queries_almacenadas[i]=0;
   }
   queries_almacenadas[0]=0;
   conta_query[0]=0;
   sumador[0]=0;
   recorrido[0]=0;
   no_relevante[0]=0;
   cuenta_Last[0]=0;

buscar_QDSM_Last(cluster_Last_Single,cuenta_Last,sumador,conta_query,k,query_final,
queries_almacenadas,recorrido);
   if(recorrido[0]!=0){
     rele_total_Promedio=0;
     for (int i = 0; i < conta_query[0]; ++i)
     {
       rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Last;
     }
     aux_recorrido=recorrido[0];
     aux_relevantes=rele_total_Promedio;
     cuenta_coseno[0]=0;
     sumador[0]=0;
     cantidad_relevancia_coseno[0]=0;
     relevantes_no_query[0]=0;
     recorrido[0]=0;
     no_relevante[0]=0;

buscar_Coseno_Last(cluster_coseno_Single,todo_Doc,Documentos,contador_documento
s,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,conta_query
[0],queries_almacenadas,recorrido,no_relevante);
     if(recorrido[0]!=0){
```

```
        recorrido_coseno_Last=recorrido_coseno_Last+recorrido[0];

resumen_relevantes_coseno_Last=resumen_relevantes_coseno_Last+cantidad_relevanci
a_coseno[0];
        resumen_relevantes_Last=resumen_relevantes_Last+aux_relevantes;
        recorrido_Last=recorrido_Last+aux_recorrido;
        ++total_Last_queries;
      }
    }
  }
/*RESUMENES*/
  printf("\n");
  printf("Metodo Single Link\n");
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes All:%d\n",resumen_relevantes_Promedio);
  printf("Total recorrido All:%d\n",recorrido_Promedio );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_Promedio);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Promedio );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes AllParcial:%d\n",resumen_relevantes_All);
  printf("Total recorrido AllParcial:%d\n",recorrido_All );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_All);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_All );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes Last:%d\n",resumen_relevantes_Last);
  printf("Total recorrido Last:%d\n",recorrido_Last );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_Last);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Last );
/*** COMPLETE LINK**/
```

```
 /*  Comprobacion clustering coseno-AllParcial*/
 resumen_relevantes_All=0;resumen_relevantes_coseno_All=0;total_All_queries=0;
 recorrido_All=0;recorrido_coseno_All=0;aux_recorrido=0;aux_relevantes=0;
 for (k = 0; k < total_query;++k)
 {
   cuenta_All[0]=0;
   sumador[0]=0;
   conta_query[0]=0;
   recorrido[0]=0;

buscar_QDSM_AllParcial(cluster_All_Complete,cuenta_All,sumador,conta_query,k,query_
final,queries_almacenadas,recorrido);
   if(recorrido[0]!=0){
     rele_total_All=0;
     for (int i = 0; i < conta_query[0]; ++i)
     {
       rele_total_All=rele_total_All+Documentos[k][0].relevancia_All;
     }
     aux_recorrido=recorrido[0];
     aux_relevantes=rele_total_All;
     no_relevante[0]=0;
     cuenta_coseno[0]=0;
     sumador[0]=0;
     cantidad_relevancia_coseno[0]=0;
     relevantes_no_query[0]=0;
     recorrido[0]=0;

buscar_Coseno_AllParcial(cluster_coseno_Complete,todo_Doc,Documentos,contador_do
cumentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,con
ta_query[0],queries_almacenadas,recorrido,no_relevante);
     if(recorrido[0]!=0){
       recorrido_coseno_All=recorrido_coseno_All+recorrido[0];
```

```
resumen_relevantes_coseno_All=resumen_relevantes_coseno_All+cantidad_relevancia_c
oseno[0];
        resumen_relevantes_All=resumen_relevantes_All+aux_relevantes;
        recorrido_All=recorrido_All+aux_recorrido;
        ++total_All_queries;
      }
    }
  }
/*Comprobacion clustering coseno-All*/

resumen_relevantes_Promedio=0;resumen_relevantes_coseno_Promedio=0;total_Prome
dio_queries=0;

recorrido_Promedio=0;recorrido_coseno_Promedio=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for ( k= 0; k < total_query; ++k)
  {
    for (int i = 0; i < conta_query[0]; ++i)
    {
      queries_almacenadas[i]=0;
    }
    conta_query[0]=0;
    sumador[0]=0;
    recorrido[0]=0;
    cuenta_Promedio[0]=0;

buscar_QDSM_All(cluster_promedio_Complete,cuenta_Promedio,sumador,conta_query,k
,query_final,queries_almacenadas,recorrido);
    if(recorrido[0]!=0){
      rele_total_Promedio=0;
      for (int i = 0; i < conta_query[0]; ++i)
      {
```

```
rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Promedio+contad
or_documentos[k];
      }
      aux_recorrido=recorrido[0];
      aux_relevantes=rele_total_Promedio;
      cuenta_coseno[0]=0;
      sumador[0]=0;
      cantidad_relevancia_coseno[0]=0;
      relevantes_no_query[0]=0;
      recorrido[0]=0;
      no_relevante[0]=0;

buscar_Coseno_All(cluster_coseno_Complete,todo_Doc,Documentos,contador_document
os,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,conta_quer
y[0],queries_almacenadas,recorrido,no_relevante);
      if(recorrido[0]!=0){
        recorrido_coseno_Promedio=recorrido_coseno_Promedio+recorrido[0];

resumen_relevantes_coseno_Promedio=resumen_relevantes_coseno_Promedio+cantida
d_relevancia_coseno[0];
        resumen_relevantes_Promedio=resumen_relevantes_Promedio+aux_relevantes;
        recorrido_Promedio=recorrido_Promedio+aux_recorrido;
        ++total_Promedio_queries;
      }
    }
  }
/*  Comprobacion clustering coseno-Last  */
  resumen_relevantes_Last=0;resumen_relevantes_coseno_Last=0;total_Last_queries=0;
  recorrido_Last=0;recorrido_coseno_Last=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for (k= 0; k < total_query; ++k)
  {
    for (int i = 0; i < conta_query[0]; ++i)
```

```
    {
      queries_almacenadas[i]=0;
    }
    queries_almacenadas[0]=0;
    conta_query[0]=0;
    sumador[0]=0;
    recorrido[0]=0;
    no_relevante[0]=0;
    cuenta_Last[0]=0;

buscar_QDSM_Last(cluster_Last_Complete,cuenta_Last,sumador,conta_query,k,query_fi
nal,queries_almacenadas,recorrido);
    if(recorrido[0]!=0){
      rele_total_Promedio=0;
      for (int i = 0; i < conta_query[0]; ++i)
      {
        rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Last;
      }
      aux_recorrido=recorrido[0];
      aux_relevantes=rele_total_Promedio;
      cuenta_coseno[0]=0;
      sumador[0]=0;
      cantidad_relevancia_coseno[0]=0;
      relevantes_no_query[0]=0;
      recorrido[0]=0;
      no_relevante[0]=0;

buscar_Coseno_Last(cluster_coseno_Complete,todo_Doc,Documentos,contador_docume
ntos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,conta_qu
ery[0],queries_almacenadas,recorrido,no_relevante);
      if(recorrido[0]!=0){
        recorrido_coseno_Last=recorrido_coseno_Last+recorrido[0];
```

```
resumen_relevantes_coseno_Last=resumen_relevantes_coseno_Last+cantidad_relevanci
a_coseno[0];
        resumen_relevantes_Last=resumen_relevantes_Last+aux_relevantes;
        recorrido_Last=recorrido_Last+aux_recorrido;
        ++total_Last_queries;
      }
    }
  }
/*RESUMENES*/
  printf("\n");
  printf("Metodo Complete Link\n");
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes All Complete:%d\n",resumen_relevantes_Promedio);
  printf("Total recorrido All:%d\n",recorrido_Promedio );
  printf("Total relevantes Coseno Complete:%d\n",
resumen_relevantes_coseno_Promedio);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Promedio );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes AllParcial Complete:%d\n",resumen_relevantes_All);
  printf("Total recorrido AllParcial:%d\n",recorrido_All );
  printf("Total relevantes Coseno Complete:%d\n", resumen_relevantes_coseno_All);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_All );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes Last Complete:%d\n",resumen_relevantes_Last);
  printf("Total recorrido Last:%d\n",recorrido_Last );
  printf("Total relevantes Coseno Complete:%d\n", resumen_relevantes_coseno_Last);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Last );
/*** AVERAGE LINK**/
  /*  Comprobacion clustering coseno-AllParcial*/
```

```
 resumen_relevantes_All=0;resumen_relevantes_coseno_All=0;total_All_queries=0;
 recorrido_All=0;recorrido_coseno_All=0;aux_recorrido=0;aux_relevantes=0;
 for (k = 0; k < total_query;++k)
 {
  cuenta_All[0]=0;
  sumador[0]=0;
  conta_query[0]=0;
  recorrido[0]=0;

buscar_QDSM_AllParcial_Average(Cluster_All_Average,cuenta_All,sumador,conta_query
,k,query_final,queries_almacenadas,recorrido);
  if(recorrido[0]!=0){
    rele_total_All=0;
    for (int i = 0; i < conta_query[0]; ++i)
    {
      rele_total_All=rele_total_All+Documentos[k][0].relevancia_All;
    }
    aux_recorrido=recorrido[0];
    aux_relevantes=rele_total_All;
    no_relevante[0]=0;
    cuenta_coseno[0]=0;
    sumador[0]=0;
    cantidad_relevancia_coseno[0]=0;
    relevantes_no_query[0]=0;
    recorrido[0]=0;

buscar_Coseno_AllParcial_Average(Cluster_Coseno_Average,todo_Doc,Documentos,con
tador_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_q
uery,conta_query[0],queries_almacenadas,recorrido,no_relevante);
    if(recorrido[0]!=0){
      recorrido_coseno_All=recorrido_coseno_All+recorrido[0];

resumen_relevantes_coseno_All=resumen_relevantes_coseno_All+cantidad_relevancia_c
oseno[0];
```

```
        resumen_relevantes_All=resumen_relevantes_All+aux_relevantes;
        recorrido_All=recorrido_All+aux_recorrido;
        ++total_All_queries;
      }
    }
  }
/*  Comprobacion clustering coseno-All*/

resumen_relevantes_Promedio=0;resumen_relevantes_coseno_Promedio=0;total_Prome
dio_queries=0;

recorrido_Promedio=0;recorrido_coseno_Promedio=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for ( k= 0; k < total_query; ++k)
  {
    for (int i = 0; i < conta_query[0]; ++i)
    {
      queries_almacenadas[i]=0;
    }
    conta_query[0]=0;
    sumador[0]=0;
    recorrido[0]=0;
    cuenta_Promedio[0]=0;

buscar_QDSM_All_Average(Cluster_AllPromedio_Average,cuenta_Promedio,sumador,co
nta_query,k,query_final,queries_almacenadas,recorrido);
    if(recorrido[0]!=0){
      rele_total_Promedio=0;
      for (int i = 0; i < conta_query[0]; ++i)
      {

rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Promedio+contad
or_documentos[k];
      }
```

```
      aux_recorrido=recorrido[0];
      aux_relevantes=rele_total_Promedio;
      cuenta_coseno[0]=0;
      sumador[0]=0;
      cantidad_relevancia_coseno[0]=0;
      relevantes_no_query[0]=0;
      recorrido[0]=0;
      no_relevante[0]=0;

buscar_Coseno_All_Average(Cluster_Coseno_Average,todo_Doc,Documentos,contador_
documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,c
onta_query[0],queries_almacenadas,recorrido,no_relevante);
      if(recorrido[0]!=0){
        recorrido_coseno_Promedio=recorrido_coseno_Promedio+recorrido[0];

resumen_relevantes_coseno_Promedio=resumen_relevantes_coseno_Promedio+cantida
d_relevancia_coseno[0];
        resumen_relevantes_Promedio=resumen_relevantes_Promedio+aux_relevantes;
        recorrido_Promedio=recorrido_Promedio+aux_recorrido;
        ++total_Promedio_queries;
      }
    }
  }
/*  Comprobacion clustering coseno-Last*/
  resumen_relevantes_Last=0;resumen_relevantes_coseno_Last=0;total_Last_queries=0;
  recorrido_Last=0;recorrido_coseno_Last=0;aux_recorrido=0;aux_relevantes=0;
  k=0;
  for (k= 0; k < total_query; ++k)
  {
    for (int i = 0; i < conta_query[0]; ++i)
    {
      queries_almacenadas[i]=0;
    }
```

```
    queries_almacenadas[0]=0;
    conta_query[0]=0;
    sumador[0]=0;
    recorrido[0]=0;
    no_relevante[0]=0;
    cuenta_Last[0]=0;

buscar_QDSM_Last_Average(Cluster_Last_Average,cuenta_Last,sumador,conta_query,k
,query_final,queries_almacenadas,recorrido);
    if(recorrido[0]!=0){
      rele_total_Promedio=0;
      for (int i = 0; i < conta_query[0]; ++i)
      {
        rele_total_Promedio=rele_total_Promedio+Documentos[k][0].relevancia_Last;
      }
      aux_recorrido=recorrido[0];
      aux_relevantes=rele_total_Promedio;
      cuenta_coseno[0]=0;
      sumador[0]=0;
      cantidad_relevancia_coseno[0]=0;
      relevantes_no_query[0]=0;
      recorrido[0]=0;
      no_relevante[0]=0;

buscar_Coseno_Last_Average(Cluster_Coseno_Average,todo_Doc,Documentos,contador
_documentos,sumador,cuenta_coseno,cantidad_relevancia_coseno,relevantes_no_query,
conta_query[0],queries_almacenadas,recorrido,no_relevante);
      if(recorrido[0]!=0){
        recorrido_coseno_Last=recorrido_coseno_Last+recorrido[0];

resumen_relevantes_coseno_Last=resumen_relevantes_coseno_Last+cantidad_relevanci
a_coseno[0];
        resumen_relevantes_Last=resumen_relevantes_Last+aux_relevantes;
        recorrido_Last=recorrido_Last+aux_recorrido;
```

```
        ++total_Last_queries;
      }
     }
   }
/*RESUMENES*/
  printf("\n");
  printf("Metodo Average Link\n");
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes All Average:%d\n",resumen_relevantes_Promedio);
  printf("Total recorrido All:%d\n",recorrido_Promedio );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_Promedio);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Promedio );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes AllParcial Average:%d\n",resumen_relevantes_All);
  printf("Total recorrido AllParcial:%d\n",recorrido_All );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_All);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_All );
  printf("\n");
  printf("Total de consultas:%d\n",total_query);
  printf("Total relevantes Last Average:%d\n",resumen_relevantes_Last);
  printf("Total recorrido Last:%d\n",recorrido_Last );
  printf("Total relevantes Coseno:%d\n", resumen_relevantes_coseno_Last);
  printf("Total recorrido Coseno:%d\n",recorrido_coseno_Last );
}


/* Retorna el largo del LCS */
int lcs( char *X, char *Y, int m, int n )
{
```

```
  int L[m+1][n+1];
  int i, j;
  for (i=0; i<=m; i++)
  {
    for (j=0; j<=n; j++)
    {
      if (i == 0 || j == 0)
        L[i][j] = 0;
      else  if (X[i-1] == Y[j-1])
              L[i][j] = L[i-1][j-1] + 1;
            else
              L[i][j] = max(L[i-1][j], L[i][j-1]);
    }
  }
  return L[m][n];
}

/* Funcion que entrega el máimo entre dos enteros */
int max(int a, int b)
{
  return (a > b)? a : b;
}

float max2(float a, float b)
{
  return (a > b)? a : b;
}


int main(int argc, char *argv[]){
```

```
  char
*id[5000],*timer[5000],*item_ranks[5000],aux[300],stop_words[5000],*stops[5000],*all_Doc
umentos[6000],*Documentos_no_repetidos[6000];

  int i=0,contador,k=0,largo_stop,cantidad_url=1;

  int cont;

  char *queries[5000];

  char query_aux[5000],time_aux[5000];

  int con[2];

  QUERY query=NULL;

  QUERY_ID query_id=NULL;

  QUERY_URLS query_url2=NULL;

  struct Matriz_Docu **Documentos;

  struct Matriz_relevancia_Docu **Documentos_relevancia;

  srand (time(NULL));

  char
aux2_i[300],aux3_i[300],aux4_i[300],aux5_i[300],aux6_i[300],aux7_i[300],aux8_i[300],aux
9_i[300],aux10_i[300];

  FILE *archivo,*archivo2;

/* el fopen lee el archivo que contiene el dataset leyendolo todo, por lo que hay archivos
con dataset con menos o mas lineas de información*/

/* por lo que existen varios archivos a leer con las cantidades de consultas diferentes de
forma ascendente, por ejemplo: query1, query2, query3, entre otros */

  archivo=fopen("query.txt","r");

  if(archivo!=NULL){

    while(!feof(archivo)){

      fgets(aux,300,archivo);


query=insertar(aux,query,aux2_i,aux3_i,aux4_i,aux5_i,aux6_i,aux7_i,aux8_i,aux9_i,aux10
_i);


    }

  }

  fclose(archivo);

  int c=0;
```

```
/* Se lee el archivo de las stopwords qu contienen las palabras que se sacarán de las
consultas */
  archivo2=fopen("stopwords_en.txt","r");
  if(archivo2!=NULL){
    while(!feof(archivo2)){
      fgets(stop_words,300,archivo2);
      largo_stop=strlen(stop_words);
      stop_words[largo_stop-2]='\0';
      sacar_stop_words(query_url2,stop_words);
      stops[c]=(char *)malloc(sizeof(strlen(stop_words)));
      strcpy(stops[c],stop_words);
      ++c;
    }
  }
  fclose(archivo2);
  cont=0;
/* Función que considera solo a las consultas que contengan a una URL */
  solo_validos(query,queries,all_Documentos,item_ranks,id,timer,i,all_Documentos);
  i=0;
  while(queries[k]!='\0'){
    k++;
    contador++;
  }
  int conta_Doc[0],total_Doc;
  total_Doc=k;
  conta_Doc[0]=0;
  /*Función que saca los documentos repetidos de all_Documentos */
  sacar_Documentos_repetidos(all_Documentos,Documentos_no_repetidos,k,conta_Doc);
  i=k;
  --i;
  con[0]=0;
  Documentos=(struct Matriz_Docu **)malloc(5000*sizeof(struct Matriz_Docu *));
```

```
  for(i=0;i<total_Doc;i++)
    Documentos[i]=(struct Matriz_Docu *)malloc((total_Doc+1)*sizeof(struct Matriz_Docu));
  int contador_documentos[5000],posicion[0];
  posicion[0]=0;
  char auxiliar_id[5000],auxiliar2_id[5000],*auxiliar3_id[5000];
  while(con[0]<i){
/* Función que separa la estructura de las consultas, como consulta,id de
usuario,fecha,rank e url */

query_id=urls_con_id(query_id,queries,all_Documentos,item_ranks,id,timer,i,cont,query_a
ux,time_aux,cantidad_url,con,Documentos,contador_documentos,posicion,auxiliar_id,auxil
iar2_id,auxiliar3_id);
    ++con[0];
    ++cont;
  }
  --con[0];
  ++posicion[0];
  char
auxiliar_w[2000],auxiliar2_w[2000],*auxiliar3_w[2000],aux3_w[2000],*temp_w[200],*unir_
w=NULL;
  int v=0,total[2];
  total[0]=0;
  while(stops[v]!=NULL){
/* Función que saca las palabras que estan en las stopwords y reconstruye denuevo la
consulta */

sacar_stop_words21(query_id,stops[v],total,auxiliar_w,auxiliar2_w,auxiliar3_w,aux3_w,te
mp_w,unir_w);
    ++v;
  }
  int total_terminos[1];
  total_terminos[0]=0;
  char *terminos_sacados[3000];
/* Función que saca los terminos d la consulta */
  sacar_terminos(query_id,terminos_sacados,total_terminos);
```

```
  char *terminos_final[3000],*query_final[3000];
  int total_terminos_final[1],total_query_final[1];
  total_terminos_final[0]=0,total_query_final[0]=0;
/* Función que saca los terminos repetidos */

sacar_terminos_repetidos(terminos_sacados,terminos_final,total_terminos[0],total_termin
os_final);
/* Función que saca todas las consultas y las almacena en un vector */
  sacar_query(query_id,query_final,total_query_final);
  int cantidad_query,cantidad_termino;
  cantidad_query=total_query_final[0];
  cantidad_termino=total_terminos_final[0];
  Documentos_relevancia=(struct Matriz_relevancia_Docu **)malloc(3000*sizeof(struct
Matriz_relevancia_Docu *));
  for(i=0;i<total_Doc;i++)
    Documentos_relevancia[i]=(struct Matriz_relevancia_Docu
*)malloc((total_Doc+1)*sizeof(struct Matriz_relevancia_Docu));
  int paso[0];
  paso[0]=0;
/* Función que define si un documentos es relevante o no relevante */
  probabilidad_relevancia(query_id,Documentos,paso);
/* Función que alamcena todos los documentos en un vector */

poner_todos_documentos(Documentos,Documentos_relevancia,contador_documentos,ca
ntidad_query);
  int conta=0,cant_url[2000];
  char
matriz_All[cantidad_query+1][300],vector_Last[cantidad_query+1],vector_promedioAll[cant
idad_query+1];
  int cant_All[cantidad_query+1],cant_Promedio[cantidad_query+1];
/* Función que almacena la relevancia de los documentos */

guardar_relevancia(query_id,matriz_All,vector_Last,vector_promedioAll,cant_All,cant_Pro
medio,cantidad_query+1,conta,cant_url);
/* Función donde se realizan los experimentos*/
```

```
experimentos(query_final,terminos_final,cantidad_query,cantidad_termino,matriz_All,vecto
r_Last,vector_promedioAll,cant_url,cant_All,cant_Promedio,total_Doc,conta_Doc[0],all_Do
cumentos,Documentos_no_repetidos,Documentos,Documentos_relevancia,contador_doc
umentos);

  return 0;

}
```