Capítulo 3
Ranking y Filtro

3.1. Recuperación de Información

En los sistemas de motores de búsqueda de la Web, así como en los sistemas de recuperación de información clásicos, se define un documento como una "unidad de recuperación", la cual puede ser un párrafo, una sección, un capítulo, una página Web, un artículo o un libro completo [7].

En general los documentos de texto no están estructurados para proveer fácilmente la información deseada. Se puede buscar en los documentos de texto por cadenas que están presentes o no, pero en muchos casos se busca en ellos por conceptos semánticos de interés. Este problema se resuelve básicamente buscando documentos que son similares a una consulta dada.

Existe una gran cantidad de trabajos realizados con respecto a la recuperación de información [73]. Sin embargo la mayoría de las investigaciones realizadas en sistemas de recuperación de información son sobre colecciones homogéneas bien controladas tales como colecciones de publicaciones científicas. De echo, la principal base de datos para recuperación de información, *Text Retrieval Conference, TREC* [55] utiliza una colección bien controlada de documentos y las cosas que funcionan bien en esta colección, generalmente no dan buenos resultados en la Web. Esto se debe a que la Web es una vasta colección de documentos heterogéneos completamente incontrolables, ya que estos documentos varían en

sus lenguajes, en sus vocabularios, tipo o formato, y hasta pueden haber sido generados por una máquina. Además en la Web no se controla lo que las personas colocan allí.

Otro problema relacionado a la recuperación de textos, es cómo se escriben las palabras. Debido a que en la Web han surgido enormes bases de datos sin control de calidad donde es común encontrar errores de tipeo en los documentos y en las consultas, no se pueden recuperar documentos que contienen palabras mal escritas realizando una consulta correctamente escrita. Existen modelos de similitud entre palabras que capturan muy bien ese tipo de errores. En este caso se da una palabra y se intenta obtener todas las palabras cercanas a ella [59].

Estos modelos de similitud son utilizados en los motores de búsqueda en la Web, para predecir qué documentos son relevantes y cuales no lo son para una consulta del usuario. Esta decisión generalmente depende del algoritmo de ranking que intenta establecer un orden de los documentos recuperados. Los documentos que aparecen en la parte superior del resultado entregado, son considerados los mas relevantes. Por lo tanto el algoritmo de ranking es la operación central de los sistemas de búsqueda en textos.

3.2. Modelos de Ranking

Existen tres modelos clásicos para realizar la operación de ranking: el modelo booleano, el vectorial y el probabilístico [7]. En el modelo booleano, los documentos y consultas son representadas como conjuntos de términos del índice. En el modelo vectorial, los documentos y las consultas son representadas como vectores en un espacio t-dimensional. En el modelo probabilístico, el marco para representar los documentos y las consultas está basado en teoría de probabilidades.

La mayoría de las máquinas Web utiliza variantes del modelo booleano o

vectorial, debido a que el texto no suele estar disponible al momento de la consulta. Una diferencia importante entre realizar la operación de indexación de la Web y la recuperación de información normal [7], está dada por los $links$, debido a que las páginas que reciben muchos $links$ o las páginas apuntadas por otras, pueden ser consideradas muy populares, y las páginas muy conectadas entre sí o referenciadas desde la misma página origen, podrían contener información similar.

3.2.1. Modelo Booleano

El modelo booleano es un modelo de recuperación simple basado en una teoría de conjuntos y en álgebra booleana. Como el concepto de conjunto es bastante intuitivo, el modelo booleano provee un marco sencillo para el trabajo del usuario. Además, las consultas son expresadas como expresiones booleanas las cuales tienen una semántica precisa. Debido a su simplicidad inherente y a su formalismo, el modelo booleano recibió gran atención en los años pasados y fue utilizado por los primeros sistemas bibliográficos comerciales.

Desafortunadamente el modelo booleano sufre de grandes desventajas. Primero, su estrategia de recuperación está basada en criterios de decisión binarias (ej: un documento es relevante o no), sin ninguna escala gradual. Por lo tanto, el modelo booleano es mas un modelo de recuperación de datos que de información. Segundo, mientras que las expresiones booleanas tienen una semántica precisa, frecuentemente no es posible traducir una necesidad de información en una de estas expresiones. De hecho, para muchos usuarios es difícil la tarea de expresar sus requerimientos en términos de expresiones booleanas. A pesar de estas desventajas, el modelo booleano sigue siendo dominante en los sistemas de base de datos comerciales de documentos.

El modelo booleano considera que los términos de índice están o no están presentes en un documento. Como resultado de esto, los pesos de los términos del

índice son binarios, ej. $w_{i,j}\epsilon\{0,1\}$. Una consulta q está compuesta por términos del índice conectada por tres conectivas: and, not, or.

Definición: 1. *Para el modelo booleano las variables de peso para los términos del índice son binarios ej.* $w_{i,j}\epsilon\{0,1\}$. *Una consulta* q *es una expresión convencional booleana. Sea* $\vec{q}_{dnf}$ *la forma normal disyuntiva de una consulta* q. *Sea* $\vec{q}_{cc}$ *cualquier componente conjuntiva de* $\vec{q}_{dnf}$. *La similitud de un documento* d_j *a una consulta* q *es definida como:*

$$sim(d_j, q) = \begin{cases} 1 & \textit{si } \exists \vec{q}_{cc} \mid (\vec{q}_{cc}\epsilon\vec{q}_{dnf}) \wedge (\forall k_i, g_i(\vec{d_j} = g_i(\vec{q}_{cc}))) \\ 0 & \textit{en otro caso} \end{cases}$$

Donde k_i es un término del índice, d_j es un documento que tiene asociado un vector de términos del índice $\vec{d_j} = \{w_{1,j}, w_{2,j}, ..., w_{3,j}\}$, y g_i es la función que retorna el peso asociado al término del índice k_i en cualquier vector t-dimensional.

Si $Sim(d_j, q) = 1$ el modelo booleano predice que el documento d_j es relevante para la consulta q. En otro caso, la predicción es que el documento no es relevante.

¿Por qué es malo?

- No discrimina entre documentos mas o menos relevantes.
- Da lo mismo que una consulta contenga una o cien veces las palabras de la consulta.
- No considera una coincidencia parcial de un documento (ej: que cumpla con casi todas las cláusulas de un AND).
- No permite ordenar los resultados.
- El usuario promedio no entiende el concepto de consultas booleanas.

¿Por qué es popular?

- Es de las primeras ideas que se utilizaron.
- Muchos sistemas se basaron en él.
- Es simple de formalizar y eficiente de implementar.

3.2.2. Modelo Vectorial

La Web, como conjunto de páginas html relacionadas mediante enlaces o hypervínculos, puede ser representada como un grafo, en el que cada página constituye un nodo, y cada enlace puede considerarse un arco. Dado que los enlaces o hypervínculos parten de páginas o nodos concretos y apuntan a otra página concreta, podemos hablar de grafo dirigido, de manera que los arcos tienen un sentido o dirección determinada, independientemente de que los navegadores puedan tener utilidades de vuelta atrás o mecanismos similares [22].

Aunque las páginas html contienen, desde luego, mas cosas además de los enlaces o arcos, éstos constituyen elementos informativos de importancia, que pueden ayudar, a caracterizar dichas páginas. En efecto, los hypervínculos establecen una relación entre unas páginas y otras. De una manera intuitiva, podemos pensar que dos páginas que reciben enlaces desde los mismos nodos deben tratar acerca de los mismos temas o parecidos. Páginas que apuntan o enlazan con los mismos nodos podrían ser mas o menos similares en su temática o contenido.

Por otro lado, los sistemas mas habituales de recuperación de la información se basan en conseguir una representación procesable y homogénea de documentos y consultas, y en el cálculo subsiguiente de alguna función de similitud entre la representación de una consulta dada y los documentos de una colección. Aquellos documentos con un índice de similitud mas alto, respecto de

una consulta dada, son los que, presuntamente, se ajustan mejor a las necesidades informativas expresadas en la consulta. La bondad de un sistema de recuperación (referente a su efectividad) descansa fundamentalmente sobre el sistema elegido para representar o modelar documentos y consultas. Depende también, obviamente, de la función de similitud utilizada, pero éste es un aspecto directamente dependiente del sistema de representación, de manera que éste impone una serie de limitaciones y de pautas a las cuales deben ajustarse los cálculos de similitud. Dicho de otro modo, estas funciones sólo pueden operar con los elementos elegidos para representar documentos y consultas.

Precisamente, uno de los modelos de recuperación de la información mas conocido es el modelo vectorial, o del espacio vectorial, propuesto por G. Salton a principios de los años 70 [63, 61] y ampliamente difundido desde entonces, tanto a nivel experimental como en implementaciones operativas. La base de dicho modelo consiste en la representación de los documentos a través de vectores, en los que cada elemento sería una característica observable en los documentos. Una visión elemental del concepto "característica", es la equivalente a palabra, y de hecho esto es con lo que con mas frecuencia se ha trabajado, aunque podrían contemplarse características de otro tipo. De manera que, según este esquema, cada palabra posible sería una característica, y en consecuencia, un elemento de los vectores de los documentos.

Un documento concreto, naturalmente, contiene algunas palabras y otras no. De manera que ese documento podría representarse mediante un vector que, en los elementos correspondientes a las palabras que forman dicho documento, contenga un valor determinado; mientras que para los elementos correspondientes a las palabras que no forman parte de dicho documento, se asigne otro valor distinto que represente esta circunstancia. En un esquema binario (el mas simple, pero también el mas ineficaz) podríamos asignar a los elementos del vector un 1 si la palabra forma parte de documento y un 0 en caso contrario.

De la misma manera, una consulta o interrogación a la colección de documentos puede expresarse en lenguaje natural, mediante una frase, o varias, o incluso varios párrafos si se desea. De esta manera, la consulta puede ser tratada de la misma forma que un documento, y representada también mediante un vector.

Reducidos los documentos y las consultas al mismo tipo de representación, pueden ser comparados fácilmente recurriendo a alguna de las muchas funciones existentes para comparar vectores. Un ejemplo elemental, cuando se trabaja con vectores binarios podría ser el mero producto de vectores.

Sin embargo, los valores binarios no resultan lo suficientemente expresivos, pues parece claro que unas palabras pueden resultar mas significativas que otras. Intuitivamente, podemos pensar que no se debe representar de la misma manera el contenido de una palabra que sólo aparece una vez en un documento, que otra que aparece varias veces en el mismo documento. Así pues, se suele utilizar algún tipo de coeficiente o peso que intente expresar el valor o importancia de cada palabra en cada uno de los documentos. Este coeficiente puede calcularse de muchas formas [62], y en función de diversos parámetros y criterios. Uno de los modos mas habituales es hacerlo teniendo en cuenta la frecuencia de aparición de la palabra. En líneas generales, suele partirse de la idea de que el peso de un término en un documento dado es inversamente proporcional a su frecuencia en la colección de documentos, y directamente proporcional a su frecuencia en el documento en cuestión. Hay una buena cantidad de fórmulas propuestas para calcular el peso, basadas en estas ideas.

El modelo vectorial reconoce que el uso de pesos binarios es muy limitado, y propone un marco en el cual las coincidencias parciales son posibles. Esto lo logra asignando pesos no binarios a los términos de las consultas y de los documentos. Estos pesos son finalmente utilizados para calcular el grado

de similitud entre cada documento almacenado en el sistema y la consulta del usuario. Ordenando los documentos recuperados en orden decreciente respecto de este grado de similitud, el modelo vectorial considera los documentos que coinciden parcialmente con los términos de la consulta. El principal efecto resultante es que el conjunto de respuesta de documentos rankeados es mas preciso que el conjunto de respuestas obtenido por el modelo booleano.

El peso de los términos se puede calcular de varias maneras. El trabajo realizado por Salton y McGill [63] analiza varias técnicas de pesos para términos.

En este trabajo de tesis se utilizó el modelo vectorial para realizar la operación de ranking durante el procesamiento de las consultas. De las fórmulas existentes para este modelo se seleccionaron las presentadas en [7], donde:

- Se selecciona un conjunto de palabras útiles para discriminar (términos).
- En los sistemas modernos toda palabra existente es un término, excepto posiblemente por las palabras vacías o $stopwords$.
- Sea $\{t_1...t_n\}$ el conjunto de términos y $\{d_1...d_n\}$ el conjunto de documentos, un documento d_i se modeliza como un vector:

  $$d_i \rightarrow \vec{d_i} = (w(t_1, d_i), ..., w(t_k, d_i))$$

  donde $w(t_r, d_i)$ es el peso del término t_r en el documento d_i.
- En particular una consulta puede verse como un documento (formada por esas palabras),z y por lo tanto como un vector.
- La similitud entre la consulta q y el documento d esta dada por:

  $$0 <= sim(d, q) = \sum_t (w_{q,t} * w_{d,t})/W_d <= 1.$$

Se calcula la similitud entre la consulta q y el documento d como la diferencia coseno, que geométricamente corresponde al coseno del ángulo entre los dos vectores (ver Figura 3.1). La similitud es un valor entre 0 y 1. Notar que los documentos iguales tienen similitud 1, y los ortogonales (si no comparten términos) tienen similitud 0. Por lo tanto esta fórmula permite calcular la relevancia del documento d para la consulta q.

- El peso de un término para un documento es:

$$0 <= w_{d,t} = f_{d,t}/max_k * idf_t <= 1.$$

En esta fórmula se refleja el peso del término t en el documento d (es decir qué tan importante es este término para el documento).

- $f_{d,t}/max_k$ es la frecuencia normalizada. $f_{d,t}$ es la cantidad de veces que aparece el término t en el documento d. Si un término aparece muchas veces en un documento, se supone que es importante para ese documento, por lo tanto $f_{d,t}$ crece. max_k es la frecuencia del término mas repetido en el documento d, o dicho de otra forma, la frecuencia mas alta de cualquier término del documento d. En esta fórmula se divide por max_k para normalizar el vector y evitar favorecer a los documentos mas largos.

- $idf_t = log10(N/nt)$, donde N es la cantidad de documentos de la colección, y nt es el número de documentos donde aparece t. Esta fórmula refleja la importancia del término t en la colección de documentos. Le da mayor peso a los términos que aparecen en una cantidad pequeña de documentos. Si un término aparece en muchos documentos, no es útil para distinguir ningún documento de otro (idf_t decrece). Lo que se intenta medir es cuanto ayuda ese término a distinguir ese documento de los demás. Esta función asigna pesos altos a términos que son encontrados en un número pequeño de documentos de la colección. Se supone que los términos raros tienen un alto valor de discriminación, y la presencia de dicho término tanto en un

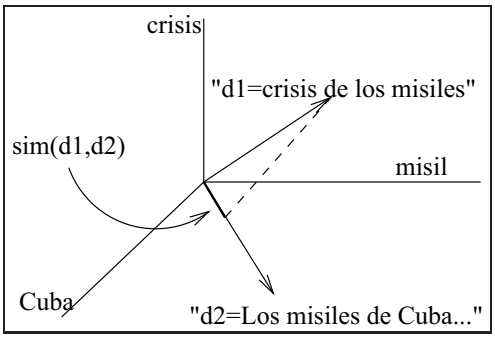

Figura 3.1: Coseno del ángulo entre dos vectores.

documento como en una consulta, es un buen indicador de que el documento es relevante para la consulta [57].

- $W_d = (\sum(w_{d,t}^2))^{1/2}$. Es utilizado como factor de normalización. Es el peso del documento d en la colección de documentos. Este valor es precalculado y almacenado durante la construcción de los índices, para reducir las operaciones realizadas durante el procesamiento de las consultas.

- $w_{q,t} = (f_{q,t}/max_k) * idf_t$, donde $f_{q,t}$ es la frecuencia del término t en la consulta q y max_k es la frecuencia del término mas repetido en la consulta q, o dicho de otra forma, es la frecuencia mas alta de cualquier término en q. Proporciona el peso del término t para la consulta q.

Una versión simple para este algoritmo se muestra en la Figura 3.2. Este modelo es mas general y permite: (1) que la consulta sea un documento, (2) hacer clustering de documentos similares y (3) retroalimentación de relevancia ("mas como estos").

Las principales ventajas del modelo vectorial son:

- Su esquema de pesos de términos mejora la performance de recuperación.

- Su estrategia de coincidencia parcial permite recuperar documentos que se aproximan a las condiciones de la consulta.

1: **for** cada documento d en la colección **do**
2: $A_d \leftarrow 0$
3: **end for**
4: **for** cada término t en la consulta **do**
5: recuperar la lista invertida para t
6: **for** cada entrada del término $< d, f_{d,t} >$ en la lista invertida **do**
7: $A_d \leftarrow A_d + sim_{q,d,t}$
8: **end for**
9: **end for**
10: Dividir cada acumulador A_d por el factor de normalización W_d
11: Identificar los K acumuladores de mayor valor (donde K es el número de resultados a ser presentados al usuario), y recuperar los correspondientes documentos

Figura 3.2: Algoritmo básico para realizar la operación de ranking a través del modelo vectorial.

- Su fórmula de ranking de coseno ordena los documentos de acuerdo a sus grados de similitud con la consulta.
- Es simple y rápido.

Por estas razones, el modelo vectorial es el mas popular en los sistemas de motores de búsqueda hoy en día.

3.2.3. Modelo Probabilístico

El modelo probabilístico fue introducido en 1976 por Roberston y Sparck Jones [60], que posteriormente se conoció como el modelo de recuperación de independencia binaria (BIR). La idea fundamental de este modelo es que dada una consulta existe un conjunto de documentos que contiene exactamente los documentos relevantes y ningún otro. Este conjunto se conoce como el conjunto de respuestas ideal. Por lo tanto se puede pensar en el proceso de consultas como un proceso que especifica las propiedades del conjunto de respuesta ideal. El problema está en que no se conoce cuáles son exactamente esas propiedades. Todo lo que se conoce es que existen términos indexados cuya semántica debería ser utilizada para caracterizar estas propiedades. Como estas propiedades no se

conocen en tiempo de consulta, debe hacerse un esfuerzo para suponer lo que ellas podrían ser inicialmente. Esta suposición permite generar una descripción probabilística preliminar del conjunto de respuesta ideal, que es utilizado para devolver el primer conjunto de respuestas. Luego se inicializa una interacción con el usuario con el propósito de mejorar las descripciones probabilísticas del conjunto de respuesta ideal.

Por lo tanto el modelo probabilístico presupone que existe exactamente un subconjunto de documentos que son relevantes para una consulta dada, y para cada documento se intenta evaluar la probabilidad de que el usuario lo considere relevante. Para ello, la relevancia de un documento d se calcula como:

$$\frac{P(d\, relevate\, para\, q)}{P(d\, no\, relevante\, para\, q)}$$

Esto se traduce en la siguiente fórmula:

$$sim(d_i, q) \;=\; \frac{P(R \mid d_i)}{P(\overline{R} \mid d_i)} \;=\; \frac{P(d_i \mid R) * P(R)}{P(d_i \mid \overline{R}) * P(\overline{R})} \;\sim\; \frac{P(d_i \mid R)}{P(d_i \mid \overline{R})}$$

donde $P(R \mid d_i)$ es la probabilidad de que d_i sea relevante, $P(R)$ la probabilidad de que un documento cualquiera sea relevante, y $P(d_i \mid R)$ la probabilidad de elegir aleatoriamente d_i entre los documentos relevantes.

Luego de bastante manejo algebraico, suponiendo independencia de los términos y tomando logaritmos, se llega a:

$$sim(d_i, q) \;\sim\; \sum_{r=1}^{k} w_{r,i} * w_{r,q} * \left(log \frac{P(t_r \mid R)}{1 - P(t_r \mid R)} + log \frac{1 - P(t_r \mid \overline{R})}{P(t_r \mid \overline{R})} \right)$$

$$= \sum_{t_r \epsilon q \cap d_i} \left(log \frac{P(t_r \mid R)}{1 - P(t_r \mid R)} + log \frac{1 - P(t_r \mid \overline{R})}{P(t_r \mid \overline{R})} \right)$$

donde $P(t_r \mid R)$ es la probabilidad de que t_r aparezca en un documento relevante, y $w_{r,i}$ es 1 si t_r aparece en d_i, y cero en otro caso.

Inicialmente se supone $P(t_r, \mid R) = 0{,}5$ y $P(t_r, \mid \overline{R} = n_r/N$. Luego de una iteración se recuperan V documentos; sea v_r el número de documentos recuperados que contienen el término t_r, se calcula $P(t_r \mid R) = v_r/V$ y $P(t_r \mid \overline{R}) = (n_r - v_r)/(N - V)$.

La mayor ventaja de este modelo, es que los documentos son rankeados en orden decreciente respecto de sus probabilidades de ser relevantes. Por otro lado, las desventajas de este modelo incluyen:

- La necesidad de suponer inicialmente la separación de documentos relevantes y no relevantes, es decir que se comienza adivinando y luego se refina esa apuesta iterativamente.
- El método no considera la frecuencia con la que un término indexado aparece en un documento, sino que ve cada documento como un conjunto de términos.
- Necesita presuponer que los términos son independientes.

Sin embargo tiene una base teórica que es distinta al del modelo vectorial y permite algunas extensiones que sí son populares [7].

3.2.4. Comparación de los Modelos Clásicos

En general el modelo booleano es considerado el método clásico mas débil. Su principal problema es su imposibilidad para reconocer coincidencias parciales lo cual frecuentemente lleva a una performance pobre. Existe cierta controversia respecto si el modelo probabilístico es mejor que el modelo vectorial. Croft [18, 19] realizó ciertos experimentos y sugirió que el modelo probabilístico provee una

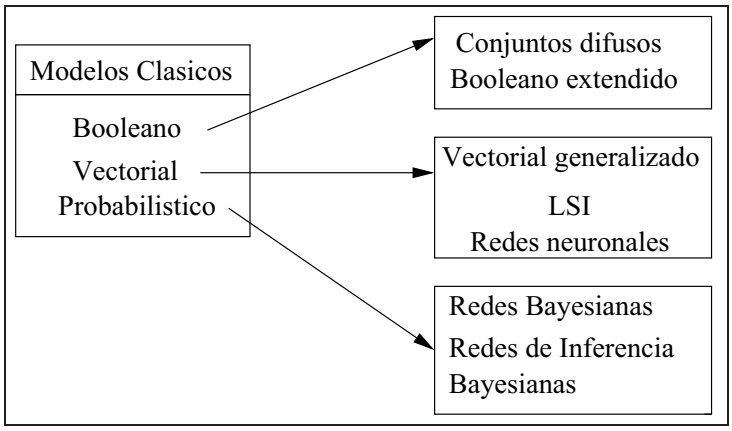

Figura 3.3: Modelos Alternativos.

mejor performance en la recuperación. Sin embargo, experimentos realizados después por Salton y Buckley [62] mostraron que se espera que el modelo vectorial mejore el comportamiento del modelo probabilístico con colecciones generales. Esto también parece ser el pensamiento dominante entre los investigadores, practicantes, y la comunidad Web, donde la popularidad del modelo vectorial crece rápidamente [7].

3.2.5. Modelos Alternativos

En la bibliografía se pueden encontrar otros modelos alternativos a los presentados anteriormente, pero en general son costosos de implementar, y no siempre dan grandes resultados. Por ello son en general poco populares, con la excepción del LSI, las redes neuronales y Bayesianas.

Las extensiones del modelo booleano son: (1) booleano extendido y (2) conjuntos Difusos. Las extensiones del modelo vectorial son: (1) vectorial generalizado, (2) LSI: Latent Semantic Indexing, y (3) redes neuronales. Finalmente las extensiones del modelo probabilístico son las redes Bayesianas y las redes de inferencias Bayesianas (ver Figura 3.3) [72, 33].

3.3. Listas Invertidas Secuenciales

Los motores de búsqueda de la Web usualmente adoptan el sistema de indexación a través de alguna estructura de datos (listas invertidas, arreglos de sufijos, etc.) para indexar y recuperar los documentos. En un sentido restrictivo un término del índice es una palabra clave, y en una forma mas general es una palabra que aparece en el texto de una colección de documentos.

Los índices invertidos o listas invertidas son una estructura de datos popular, frecuentemente utilizadas como índices a bases de datos de textos. Su propósito es acelerar las operaciones de consultas sobre grandes colecciones de textos. Actualmente su aplicación mas importante es en los motores de búsqueda en la Web, como en Google [51, 50]. Vale la pena construir y mantener un índice cuando la colección de texto es grande y semi-estática. Las colecciones semi-estáticas pueden ser actualizadas razonablemente en intervalos regulares (ej: diariamente).

La razón por la que los índices invertidos son los mas populares, es porque el espacio requerido para mantener el vocabulario es mas bien pequeño, su construcción y mantenimiento son tareas con costos relativamente bajos. En principio, un índice invertido puede ser construido en tiempo de O(n) sobre un texto de n caracteres. Existen otras estructuras de datos como los árboles y arreglos de sufijos o los archivos de firma (*signature files*). Los árboles y arreglos de sufijos son rápidos para la búsqueda de frases y otras consultas menos comunes, pero son complicadas de construir y mantener actualizadas. Por otro lado, los archivos de firma que son estructuras de índice basadas en hashing tienen una complejidad de búsqueda lineal (en lugar de sublineal como los casos anteriores), son mas bien lentos, lo cual hace que la técnica sea adecuada para textos cortos.

Un índice invertido es un mecanismo orientado a palabras para indexar una colección de texto para acelerar la tarea de búsqueda. La estructura del índice invertido tiene dos componentes: la tabla de vocabulario y las listas asociadas, como la que se muestra en la Figura 3.4, que utiliza los documentos y términos de la Tabla 3.1. La tabla de vocabulario es un conjunto de todas las palabras diferentes que aparecen en el texto. Por cada palabra, una lista con todas las posiciones del documento donde la palabra aparece es almacenada [5].

Es decir que las listas asociadas consisten de los identificadores de los documentos que contienen al término, y opcionalmente pueden incluir la posición del término, así como la cantidad de veces que éste aparece en el documento que está siendo analizado.

Por lo tanto, se puede observar que para construir una lista invertida es necesario procesar cada documento para extraer las palabras o términos de importancia, registrando su posición y la cantidad de veces que éste se repite.

Una vez que se obtiene el término, con su información correspondiente, se almacena en la lista invertida para luego ser ordenada lexicográficamente.

Documentos	*Términos*
Doc1	casa, rojo, casa
Doc2	casa, azul
Doc3	azul, perro

Tabla 3.1: Información en la Base de Datos.

Originalmente se había pensado en sistemas de base de datos centralizados, donde todos los componentes del sistema residen en una sola computadora o sitio, pero con el paso del tiempo surgieron ciertos inconvenientes (espacio de almacenamiento, acceso simultáneo, etc.), que no eran posibles solucionar. Estos problemas impulsaron la creación de almacenamiento distribuido, el cual hoy en día provee características indispensables en el manejo de información; es de-

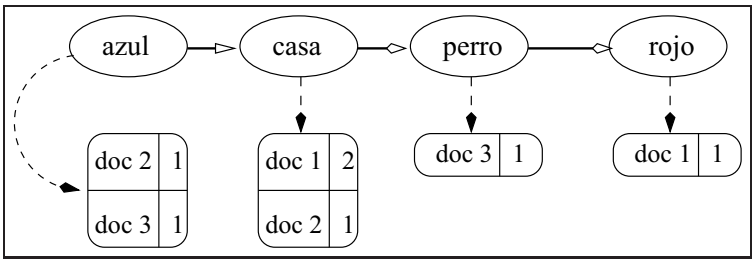

Figura 3.4: Ejemplo de una lista invertida.

cir, introdujeron la combinación de las redes de comunicación y las bases de datos.

Cuando se habla de una base de datos distribuida, se hace referencia al manejo de datos almacenados en facilidades de cómputo localizadas en muchos sitios conectados a través de una red de comunicaciones [2]. La clave del éxito al trabajar sobre bases de datos distribuidas, es poder reunir un conjunto de información almacenada en adecuadas estructuras de datos, de manera tal que el acceso a las mismas resulte simple y eficiente, y a la vez permita la resolución en paralelo de consultas.

3.4. Técnica de Filtro Centralizada

Las técnicas de ranking son efectivas para encontrar respuestas en las colecciones de documentos, pero pueden ser costosas de evaluar. La utilización de filtros [56], es una técnica de evaluación que permite reconocer en forma temprana qué documentos son probables de ser rankeados, para lograr de esta manera reducir el volumen de memoria principal requerida y el tiempo de evaluación de la consulta, sin perjudicar la efectividad de los resultados.

En las implementaciones de la medida de similitud, tal como en la medida del coseno, los documentos de la base de datos tienen un índice invertido que contiene, por cada término de la base de datos, una lista invertida de los identificadores de los documentos que contienen a dicho término. Los costos de

la evaluación de las consultas rankeadas en este tipo de índices son: memoria, para almacenar los valores de similitud, usualmente requieren un acumulador por documento de la base de datos; tráfico de disco, para transferir las listas invertidas para cada término de la consulta desde el disco hacia la memoria para ser procesada; y el tiempo de CPU, para procesar esta información indexada.

Para grandes base de datos de documentos, el costo de evaluación para la medida del coseno puede ser bastante alto, porque las consultas rankeadas son expresadas generalmente en lenguaje natural, y por lo tanto pueden contener un número grande de términos, alguno de los cuales pueden aparecer en una gran porción de los documentos de la base de datos, y porque las técnicas de ranking asignan un valor de similitud para cada documento que contiene cualquiera de los términos de la consulta. Como consecuencia, la mayoría de los documentos de la base de datos tendrán un valor de similitud distinto de cero, y por lo tanto serán candidatos a ser presentados al usuario. Por esta razón sólo los mejores documentos rankeados son recuperados - la mayoría de los documentos candidatos son descartados.

La técnica de filtrado permite una reducción importante en el volumen de memoria requerida. El efecto del filtro es que el acumulador del documento es actualizado sólo si la combinación de la frecuencia del término en el documento, y la importancia del término es suficientemente grande para tener un impacto en el ordenamiento final de documentos. Por lo tanto los términos comunes de la lista invertida pueden ser procesados, pero sólo se actualizará el acumulador de aquellos documentos en los cuales el término es frecuente.

Utilizando un umbral S_{add} se pueden ignorar las entradas a la lista invertida que poseen valores de similitud pequeñas, reduciendo de esta manera el tiempo de cpu. De esta misma manera, el umbral S_{ins} permite ignorar algunos documentos ahorrando espacio en memoria. En otras palabras, los umbrales

proveen mecanismos para cambiar la carga del sistema. Los umbrales han sido utilizados previamente para decidir cuando procesar o rechazar una lista invertida completa [34], pero no para decidir cuando rechazar o procesar documentos individuales [57].

Los valores para ambos umbrales para un término t son determinados como función de la similitud parcial acumulada S_{max} de los documentos mas relevantes. Esto supone que si el documento mas relevante tiene un peso alto, luego no es necesario procesar un documento que tiene un valor de similitud mas pequeño para la consulta.

Los valores de los umbrales son determinados según la Ecuación (3.1) y la Ecuación 3.2.

$$S_{ins} = C_{ins} * S_{max} \tag{3.1}$$

$$S_{add} = C_{add} * S_{max} \tag{3.2}$$

Donde $0 <= C_{add} <= C_{ins}$ son constantes. El efecto es que a medida que los términos de la consulta son procesados y el valor del acumulador de similitud de documentos en el conjunto de respuestas crece, se hace mas difícil actualizar o agregar nuevos acumuladores. Se procesa la entrada $< d, f_{d,t} >$ para el término de la lista invertida, si la similitud parcial $Sim(d_i, q)$ de un documento d y una consulta q es mayor que el valor del umbral corriente s, donde s es S_{ins} o S_{add}. Si se sustituye la definición de $w_{q,t}$ y $w_{d,t}$ en la definición de $Sim(d_i, q)$ se obtiene:

$$s <= f_{d,t} * wt * f_{q,t} * wt \tag{3.3}$$

La condición final es:

$$s/f_{q,t} * wt^2 <= f_{d,t} \tag{3.4}$$

expresando así, la decisión de cuando procesar una entrada de término $< d, f_{d,t} >$ como una condición de $f_{d,t}$. Los umbrales ahora pueden ser expresados

directamente en términos de frecuencias:

$$f_{ins} = C_{ins} * S_{max}/f_{q,t} * w_{d,t}^2 \tag{3.5}$$

$$f_{add} = C_{add} * S_{max}/f_{q,t} * w_{d,t}^2 \tag{3.6}$$

Estos umbrales son constantes durante el procesamiento de la lista invertida, por lo tanto la decisión de cuando usar una entrada de un término requiere sólo una simple comparación de enteros.

El uso de umbrales, provee una suave transición desde la aceptación al rechazo de entradas de los términos en las listas invertidas, a medida que es progresivamente mas difícil agregar o incrementar acumuladores. Para el primer término procesado, el valor de S_{max} es bajo y los valores de $w_{d,t}$ son altos. A medida que S_{max} se incrementa y $w_{d,t}$ decrece, los umbrales crecen hasta que en el límite, todos los valores $f_{d,t}$ son menores que f_{add}, de manera tal que procesar la lista invertida no tiene efectos sobre los valores de los acumuladores.

Las constantes C_{ins} y C_{add} son usadas para controlar los recursos requeridos por el algoritmo. Incrementando C_{add}, se reduce la cantidad de entradas de términos inspeccionadas y acumuladas por el algoritmo (y correspondientemente se reduce el número de similitudes parciales de documentos y una consulta), y por lo tanto decrece el tiempo de CPU. Incrementando la constante C_{ins} se reduce el número de documentos que pueden ser candidatos y por lo tanto se reduce la memoria usada. Las constantes deben ser seleccionadas de modo tal que la información descartada tenga un impacto mínimo en el resultado final. En un sistema de producción, las constantes pueden ser simplemente ajustadas a cada consulta basándose en observaciones de carga del sistema, u ocasionalmente las consultas pueden ser ejecutadas para varios valores de cada constante y los mejores valores son seleccionados de acuerdo a la distorsión introducida en el conjunto de respuesta.

Un potencial punto débil de la técnica de filtrado es la vulnerabilidad a la presencia de documentos con un gran número de ocurrencias de un término raro. Dichos documentos tienen pesos altos y pueden teóricamente hacer que los valores de los filtros sean tan grandes que ningún otro documento pueda ser considerado. Si este documento contiene el primer término (mas raro) de una consulta, luego el conjunto de respuestas a la consulta consistirá sólo de los documentos que contienen a dicho término.

En la Figura 3.5 se muestra el pseudo-código para la técnica de filtro, donde los términos de las consultas son ordenados de acuerdo a sus frecuencias ($f_{q,t}$), de forma tal que los términos mas importantes son procesados primero. Antes de que cada término t sea procesado, se calculan los dos umbrales S_{ins} y S_{add}. A medida que la lista invertida para el término t es procesada, se compara contra estos umbrales la similitud parcial $sim_{q,d,t}$ de la consulta q y cada documento d. Si la similitud es mayor a S_{ins}, el documento es insertado en el conjunto de documentos candidatos para la respuesta final y se suma el valor del acumulador del documento. Sino, si $S_{add} <= sim_{q,d,t}$, aunque el documento d no es tan importante para ser un candidato, su similitud puede afectar el ranking; por lo tanto si tiene un acumulador creado, se le suma el valor de similitud calculado. Si ninguno de estos dos casos se cumple, la similitud parcial es descartada y el procesamiento continúa con el próximo término.

3.5. Técnica de Filtro Distribuida

La técnica de filtro distribuida es utilizada para realizar la operación de ranking localmente en cada procesador, y reducir la cantidad de información comunicada al momento de enviar los resultados. Es decir, que los procesadores no envían todas las soluciones encontradas, sino las mas impoertantes. Este técnica es implementada en las estrategias de $K - Buckets$ (ver Sección 4.9).

1: $Ad \leftarrow \emptyset$ ▷ Crear una estructura vacía de acumuladores
2: Ordenar en forma decreciente los términos de la consulta de acuerdo a $f_{q,t}$
3: $S_{max} \leftarrow 0$
4: **for** $(t \epsilon q)$ **do**
5: Calcular f_{ins} y f_{add}
6: **for** (cada entrada $< d, f_{d,t} >$ para el término) **do**
7: **if** $(f_{d,t} >= f_{ins})$ **then**
8: Crear un acumulador Ad si es necesario
9: $Ad \leftarrow Ad + Sim(d_i, q)$
10: **else**
11: **if** ($f_{d,t} >= f_{add}$ y Ad existe en el conjunto de acumuladores) **then**
12: $Ad \leftarrow Ad + Sim(d_i, q)$
13: **end if**
14: **end if**
15: $S_{max} = max(S_{max}, Ad)$
16: **end for**
17: **end for**
18: $Ad \leftarrow Ad/W_d$
19: Identificar los k valores mas altos y devolver los documentos correspondientes

Figura 3.5: Algoritmo de la técnica de filtro.

El acumulador S_{max} tiene una importante influencia sobre la eficiencia de la técnica de filtro, debido a que S_{max} crece progresivamente a medida que aumenta el valor de la similitud acumulada de los documentos. El crecimiento de esta variable en el algoritmo distribuido tiene un comportamiento diferente del algoritmo secuencial [3].

La operación de ranking distribuida en una organizavción de índice local, tiene el problema de que los procesadores mantienen sólo algunos documentos de importancia, y por lo tanto el crecimiento de S_{max} es pequeño; y concecuentemente la cantidad de recursos filtrados es menor que en algoritmo secuencial.

En el algoritmo distribuido de una organización de índice global, el problema surge cuando los procesadores reciben sólo algunos términos de la consulta, porque el valor de S_{max} es una fracción del valor obtenido en el algoritmo secuencial. De esta manera la performance del algoritmo puede ser perjudicada.

El trabajo presentado en [8] propone una solución a este problema, previendo el valor de S_{max} antes de procesar las consultas. También se muestra que el crecimiento de S_{max} sigue al crecimiento de la similitud parcial de los documentos mas relevantes, que generalmente son los documentos que contienen a la mayoría de los términos de la consulta.

La similitud parcial puede ser calculada de forma global, si se conocen los valores de N, nt, $f_{d,t}$ y $f_{q,t}$. Para estimar el valor de $f_{d,t}$, se adopta la máxima frecuencia del término t en dentro de los documentos (f_{max_t}). De esta manera, los puntos de crecimiento de S_{max} pueden ser previamente claculados y distribuidos a los procesadores, junto con las consultas. El cálculo del punto de crecimiento de S_{max} está dado por la Ecuación (3.7):

$$PGS_{max_i} = f_{q,t} * log(N/nt) * f_{max_t} * log(N/nt) \tag{3.7}$$

1: $Ad \leftarrow \emptyset$ ▷ Crear una estructura vacía de acumuladores
2: Ordenar en forma decreciente los términos de la consulta de acuerdo a $f_{q,t}$
3: $S_{max} \leftarrow 0$
4: **for** $(t \epsilon q)$ **do**
5: $S_{max} \leftarrow S_{max} + PGS_{max_t}$
6: Calcular f_{ins} y f_{add}
7: **for** (cada entrada $< d, f_{d,t} >$ para el término) **do**
8: **if** $(f_{d,t} >= f_{ins})$ **then**
9: Crear un acumulador Ad si es necesario
10: $Ad \leftarrow Ad + Sim(d_i, q)$
11: **else**
12: **if** ($f_{d,t} >= f_{add}$ y Ad existe en el conjunto de acumuladores) **then**
13: $Ad \leftarrow Ad + Sim(d_i, q)$
14: **end if**
15: **end if**
16: $S_{max} = max(S_{max}$,Ad)
17: **end for**
18: **end for**
19: $Ad \leftarrow Ad/W_d$
20: Identificar los k valores mas altos y devolver los documentos correspondientes

Figura 3.6: Proceso de filtro distribuido.

Además de proponer una técnica de filtro distribuida, en el trabajo prensentado en [8] se obtienen mejores resultados que la técnica de filtro propuesta para el algoritmo secuencial en [56, 57]. En la Figura 3.6 se muestra el algoritmo para la técnica de filtro distribuida, donde sólo difiere del secuencial en la línea 5.

3.6. Discusión

Los modelos y técnicas presentadas en este capítulo, permiten mejorar las búsquedas de los algoritmos, y optimizar la utilización de los recursos disponibles en el sistema. Los modelos de ranking siven para seleccionar los documentos que mejor se ajustan a las consultas realizadas por los usuarios. Baeza-Yates ha realizafo una clasificación taxonómica de los distintos modelos de recuperación de información [7]. Este autor divide a los modelos basados en la recuperación

en dos grupos: clásicos y estructurados. En el primero de ellos incluye a los modelos booleano, espacio vectorial y probabilístico. Los modelos estructurados corresponden a listas de términos sin solapamiento y a nodos próximos (son modelos escasamente difundidos).

El problema del modelo booleano es que no puede reconocer coincidencias parciales, lo cual frecuentemente lleva a una performance pobre, debido a que sólo puede decir si el documento coincide con la consulta en su totalidad o no. Actualmente, el modelo mas popular es el modelo del coseno, o modelo vectorial propuesto por G. Salton a principios de los años 70 [63, 61], donde los documentos y las consultas son representadas como vectores en un espacio n-dimensional. La similitud entre un documento y una consulta, se expresa a través del coseno, cuanto mayor es este coseno, mayor es la similitud. Este modelo, exige mas información que el procesamiento de consultas booleanas, y se deben tomar desiciones importantes para realizar el procesamiento de consultas en forma eficiente.

La evaluación del modelo vectorial, para grandes colecciones, es potencialmente caro en tiempo de CPU, demanda de memoria y transferencia de disco. Para solucionar estos problemas, se puede limitar el número de documentos candidatos a ser rankeados, se ordena de acuerdo a la frecuencia de los términos en el documento, y se debe obtener una implementación eficiente del peso del documento W_d. El objetivo es no usar mas memonia ni tiempo de CPU necesarios para el procesamiento de una consulta. Esto último se logra aplicanto la técnica de filtro [56] al algoritmo de ranking.

A partir del modelo de ranking y la técnica de filtro, el próximo paso es analizar las estrategias de indexación existentes para poder afrontar los problemas que éstas presentan y así proponer nuevas soluciones.