UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Estrategias de planificación para motores de búsqueda verticales

Danilo Fernando Bustos Pérez

Profesor Guía: Dra. Carolina Bonacic Castro
Profesor Co-guía: Dr. Mauricio Marín Caihuán

Trabajo de Titulacíon presentado en conformidad a los requisitos para obtener el Título de Ingeniero Civil Informático

SANTIAGO DE CHILE
2013

documento.

AGRADECIMIENTOS

Dedicado a ... .

RESUMEN

Resumen en Castellano

Palabras Claves: keyword1, keyword2 .

ABSTRACT

Resumen en Inglés

Keywords: keyword1, keyword2 .

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

1.2 DESCRIPCIÓN DEL PROBLEMA

1.3 OBJETIVOS Y SOLUCIÓN PROPUESTA

1.3.1 Objetivo General

1.3.2 Objetivos Específicos

1.3.3 Alcances

CHAPTER 2. MARCO TEÓRICO

En este capítulo se exponen los conceptos teóricos del presente trabajo de tesis. Rellenar al final

2.1 MOTORES DE BÚSQUEDA VERTICALES

A medida que pasa el tiempo y la Web sigue creciendo, los motores de búsqueda se convierten en una herramienta cada vez más importante para los usuarios. Estas máquinas ayudan a los usuarios a buscar contenido dentro de la Web, puesto que conocen en qué páginas de la Web aparecen qué palabras. Sin un buscador los usuarios estarían obligados a conocer los localizadores de recursos uniformes (URL) de cada uno de los sitios a visitar. Además, los motores de búsquedas en cierto modo conectan la Web, ya que existe un gran número de páginas Web que no tienen referencia desde otras páginas, siendo el único modo de acceder a ellas a través de un motor de búsqueda.

Un motor de búsqueda está construído por diversos componentes, y su arquitectura típica la podemos ver en la Figura 2.1. Existe un proceso denominado *crawling*, éste posee una tabla con las páginas Web iniciales en las que se extrae el contenido de cada una de ellas. A medida que el *crawler* comienza a encontrar enlaces a otras páginas Web, la tabla de páginas a visitar crece. El contenido que se extrae en el procedimiento de *crawling* es enviado al proceso de indexamiento, este se encarga de crear un índice de las páginas visitadas por el *crawler*.

Dado el volúmen de datos involucrado en el procesamiento, se debe tener una estructura

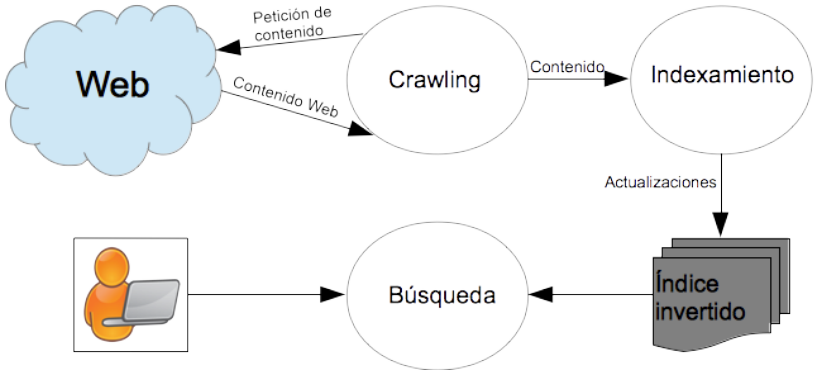

Figure 2.1: Arquitectura típica de un motor de búsqueda

de datos que permita encontrar cuáles páginas contienen las palabras presentes en la búsqueda, todo esto dentro de un período de tiempo aceptable. El índice invertido es una estructura de datos que contiene una lista con todas las palabras que el proceso de *crawling* ha visto, y asociada a cada palabra se tiene una lista de todas las páginas Web donde ésta palabra aparece mencionada. El motor de búsqueda construye esta estructura con el objetivo de acelerar el proceso de las búsquedas que llegan al sistema. El proceso de búsqueda es el encargado de recibir la consulta (query), generar un *ranking* de las páginas Web que contienen las palabras de la consulta y finalmente generar una respuesta. Las diversas formas de calcular la relevancia de una página Web será explicado en secciones posteriores.

En un motor de búsqueda se pueden encontrar diversos servicios tales como (a) cálculo de las mejores páginas Web para una cierta consulta; (b) construcción de la página Web con los resultados de la consulta; (c) publicidad relacionada con las *queries*; (e) sugerencia de *queries*; entre muchos otros servicios.

Lo que se hace hoy en día es agrupar computadores para procesar una consulta y producir la respuesta de ésta. Este conjunto de computadores recibe el nombre de *cluster*.

La diferencia entre un motor de búsqueda vertical y uno general, es que el primero se centra solo en un contenido específico de la Web. El *crawler* también debe extraer solo el contenido de aquellas páginas Web que están dentro del dominio permitido. Al ser un dominio acotado,

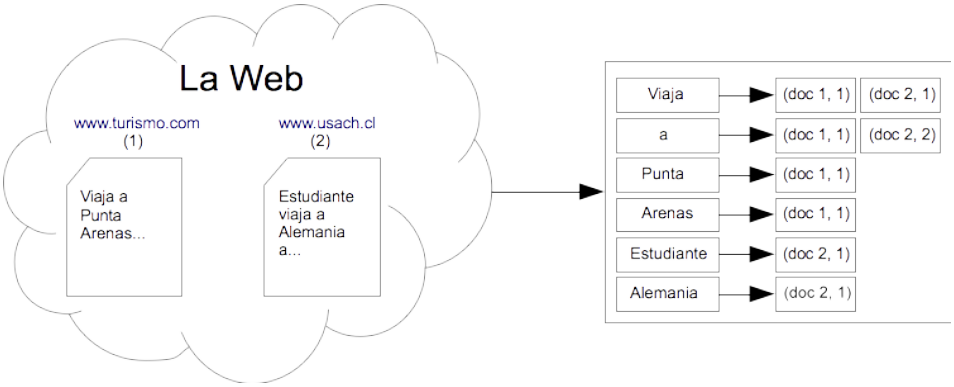

Figure 2.2: Índice invertido

las páginas Web a procesar serán menos y por tanto la lista de los términos del índice invertido serán eventualmente de menor tamaño. Sin embargo, en un motor de búsqueda vertical las actualizaciones al índice invertido ocurren con mayor frecuencia.

2.2 ÍNDICE INVERTIDO

Es una estructura de datos que contiene todos los términos (palabras) encontrados. A cada uno de los términos, está asociado una lista de punteros a los documentos (páginas Web) que contienen dicho término. Además se almacena información que permita realizar el *ranking* de las respuestas a las consultas (queries) que llegan al sistema, por ejemplo, el número de veces que aparece el término en el documento. Esta lista recibe el nombre de lista invertida.

Para construir un índice invertido se debe procesar cada palabra que existe en un documento, registrando su posición y la cantidad de veces que éste se repite. Cuando se procesa el término con la información asociada correspondiente, se almacena en el índice invertido (ver Figura 2.2).

El tamaño del índice invertido crece rápido y eventualmente la memoria RAM se agotará antes de procesar toda la colección de documentos. Cuando la memoria RAM se agota, se almacena en disco el índice parcial hasta aquel momento, se libera la memoria y se continúa con el proceso. Además, se debe hacer un *merge* de los índices parciales uniéndo las listas invertidas de cada uno de los términos involucrados.

2.3 ESTRATEGIAS DE EVALUACIÓN DE QUERIES

Existen dos principales estrategias para encontrar los documentos y calcular sus respectivos puntajes de una determinada *query*. Estas son (a) *term-at-a-time* (TAAT) y (b) *document-at-a-time* (DAAT).

2.3.1 TAAT

Este tipo de estrategia procesa los términos de las *queries* uno a uno y acumula el puntaje parcial de los documentos. Las listas invertidas asociadas a un término son procesadas secuencialmente, esto significa que los documentos presente en la lista invertida del término t_i, obtienen un puntaje parcial antes de comenzar el procesamiento del término t_{i+1}. La secuencialidad en este caso es con respecto a los términos contenidos en la *query*.

2.3.2 DAAT

En este tipo de estrategias se valúa la contribución de todos los términos de la query con respecto a un documento antes de evaluar el siguiente documento. Las listas invertidas de cada término de la *query* son procesadas en paralelo, de modo que el puntaje del documento d_j se calcula considerando todos los términos de la query al mismo tiempo. Una vez que se obtiene el puntaje del documento d_j para la *query* completa, se procede al procesamiento del documento d_{j+1}.

2.3.3 Consideraciones

Las estrategias TAAT son mayormente usadas en sistemas de recuperación de información (IR), como son los motores de búsqueda. Cuando se tiene un índice invertido pequeño, las estrategias TAAT rinden adecuadamente, sin embargo cuando los índices invertidos son de gran tamaño las estrategias TAAT poseen dos grandes ventajas: (a) Requieren menor cantidad de memoria para su ejecución, ya que el puntaje parcial por documento no necesita ser guardado y (b) Explotan el paralismo de entrada y salida (I/O) más eficientemente procesando las listas invertidas en diferentes discos simultáneamente.

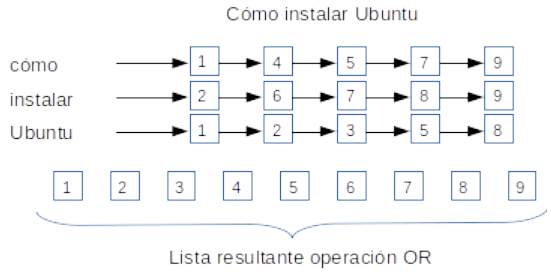

Figure 2.3: Operación OR

2.4 OPERACIONES SOBRE LISTAS INVERTIDAS

Cuando una *query* llega al motor de búsqueda, cada término tiene asociado una lista con todos los documentos en los cuales aparece dicho término. El sistema debe decidir qué documentos se analizarán para obtener la respuesa con el conjunto de los K mejores. A continuación se presentan las diferentes formas de operar los documentos pertenecientes a las listas invertidas de una *query*.

2.4.1 OR

Este operador toma las listas invertidas de cada uno de los términos de la *query* y ejecuta la disyunción entre ellas. El resultado de este operador es una lista invertida con todos los documentos que contengan al menos un término de la query. Finalmente, esta lista invertida se ocupará para obtener los mejores K documentos. Un ejemplo sencillo se muestra en la FIGURA 2.3.

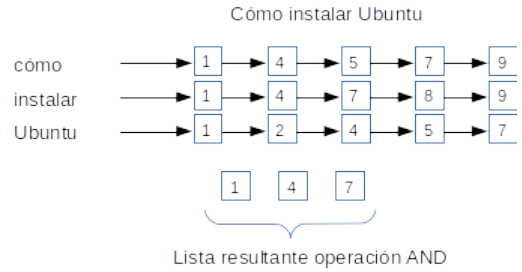

Figure 2.4: Operación AND

2.4.2 AND

Este operador ejecuta la conjunción entre las listas invertidas de los términos de una *query*. Se obtiene una lista invertida con los documentos que contengan todos los términos de la *query*. Se debe notar que aquí se obtiene una lista resultante de menor que la obtenida en el operador OR (Ver FIGURA 2.4).

2.4.3 WAND

Método de evaluación de *queries* para obtener eficientemente el conjunto de los mejores *top* K documentos que mejor satisfacen una *query* dada. *WAND* (**?**) (*Weak AND*) es un proceso menos estricto que el método *AND* y es basado en dos niveles. Dentro del proceso de evaluación de una *query*, uno de los procesos más costoso en términos de tiempo es el proceso de *scoring*. Este proceso corresponde a entregarle a cada uno de los documentos analizados, un puntaje que representa la relevancia del documento para una *query* dada, esto se denomina evaluación completa o cálculo cálculo del puntaje exacto del documento.

El objetivo de WAND es minimizar la cantidad de evaluaciones completas de los

documentos ejecutando un proceso de dos niveles. En el primer nivel se intenta omitir rápidamente grandes porciones de las listas invertida, lo que se traduce en ignorar el cálculo del puntaje exacto de grandes cantidades de documentos, ya que éste cálculo en motores de búsqueda a gran escala es una tarea que requiere de mucho tiempo para llevarse a cabo y depende de factores como la cantidad de ocurrencia del término dentro del documento, el tamaño del documento, entre otros.

En el primer nivel se itera sobre los documentos del índice invertido de cada término y se identifica candidatos usando una evaluación aproximada. En el segundo nivel, aquellos documentos candidatos son completamente evaluados y su puntaje exacto es calculado. De esta forma se obtiene el conjunto final de documentos.

2.5 RANKING

Los sistemas de recuperación de información como los motores de búsqueda deben ejecutar un proceso el cual asigna un puntaje a documentos con respecto a una determinada *query*, este proceso se denomina *ranking*. Como se puede ver en la Figura 2.5, este proceso toma como entrada las representaciones de *queries* y documentos, y asigna un número real (puntaje) a un documento d_j dada una query q_i.

Un motor de búsqueda guarda billones de documentos que están formados por términos o palabras, los cuales no todos poseen la misma utilidad para describir el contenido del documento. Determinar la importancia de un término en un documento no es tarea sencilla. Para ello se asocia un peso positivo $w_{i,j}$ a cada término k_i del documento d_j. De esta forma, para un término k_i que no aparezca en el documento d_j se tendrá $w_{i,j} = 0$. La asignación de pesos a los

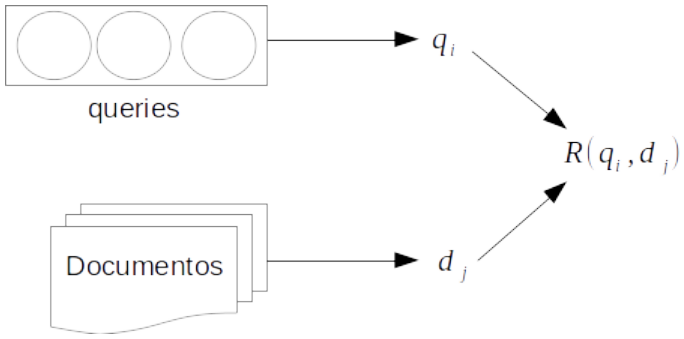

Figure 2.5: Operación AND

términos permite generar un *ranking* numérico para cada documento en la colección.

2.5.1 TF-IDF

El tf-idf (term frequency - inverse document frequency) es un estadístico que tiene por objetivo reflejar cuán importante es una palabra para un documento en una colleción o corpus. Este estadístico se divide en dos términos, el primero corresponde a la frecuencia del término en un documento (tf) y que en su versión más sencilla se utiliza la frecuencia bruta del término t en el documento d: f(t,d). El segundo término corresponde a la frecuencia inversa de documento (idf) y se utiliza para observar si es que el término es común en el corpus. El idf obtiene calculando el logaritmo entre el número total de documentos del corpus y el número de documentos que contienen el término. De esta forma se tiene:

$$tf(t,d) = \frac{f(t,d)}{maxf(w,d) : w \in d}$$

$$idf(t,D) = log\frac{|D|}{1 + |d \in D : t \in d|}$$

Por lo que el estadístico *TF-IDF* queda de la forma

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

y éste incrementa proporcionalmente al número de veces que la palabra aparece en el documento, sin embargo es compensado por la frecuencia de la palabra en la colección completa de documentos o corpus. Notar que la compensación ayuda a controlar el hecho de que algunas palabras son generalmente más comunes que otras.

2.5.2 BM25

CHAPTER 3. CONCLUSIONES