Estructuración del Índice de un Buscador Utilizando la Frecuencia de Consultas*

Ricardo Baeza-Yates Felipe Saint-Jean †

Departamento de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile

Resumen

Las consultas realizadas por los usuarios a un buscador Web siguen una distribución muy poco uniforme, motivo por el cual es buena idea adaptar el índice según frecuencia de consulta. Se muestra en este estudio una forma de organizar el índice de un buscador de manera de aprovechar la distribución de las palabras consultadas por los usuarios, logrando un buscador más eficiente en tiempo y espacio y por ende más escalable.

1 Introducción

Dado el tamaño de la Web, la cantidad de recursos necesarios para mantener un índice rápido y eficaz es muy grande. Es necesario dar una respuesta muy rápido en forma concurrente y sobre un conjunto muy grande para satisfacer al usuario de un buscador, por lo tanto es importante lograr la mayor eficiencia posible del uso de los recursos. En este contexto el hecho de que las consultas realizadas por los usuarios siguen una distribución muy sesgada (de $Zipf$), nos permite dar distinto trato a las palabras más consultadas, de manera que esas consultas más comunes tengan respuestas más rápidas. Este es un ejemplo de análisis de bitácoras (*logs*) Web para mejorar una aplicación específica, como un buscador. Hay pocos ejemplos publicados de este tipo, ya que los buscadores protegen este tipo de información (una excepción es [3]). En este trabajo se muestra una estructuración del índice de un buscador basado en las consultas realizadas por los usuarios al buscador TodoCL [4], por lo que se muestra la distribución de las consultas y resultados experimentales en base a un modelo analítico, mejorando el uso de memoria y el tiempo de respuesta.

La organización de este artículo es la que sigue. En la sección 2 entregamos los conceptos básicos relacionados con la Web y buscadores. En la sección 3 describimos los datos usados y el análisis de los mismos. En la sección 4 presentamos el análisis para un índice en dos niveles de memoria mientras que en la sección 5 agregamos respuestas precalculadas. En la última sección entregamos nuestras conclusiones y trabajo a futuro.

*Parcialmente financiado por Proyecto Fondecyt 1020803, Chile.

†{rbaeza,fsaint}@dcc.uchile.cl

2 Conceptos Básicos

2.1 Ley de Zipf

La ley de Zipf lleva el nombre del profesor de linguística de Harvard, George Kingsley Zipf (1902-1950). Es básicamente una distribución en la cual, si definimos F_i como la frecuencia de ocurrencia del i-ésimo evento más frecuente, tendremos que

$$F_i \sim \frac{1}{i^a}$$

donde a es una constante cercana a 1, que llamaremos parámetro de la distribución de Zipf. La ley de Zipf es una distribución caracterizada por presentar eventos poco frecuentes y eventos muy frecuentes. Por ejemplo, si contamos las palabras en un texto veremos que hay pocas palabras que aparecen muchas veces y muchas que aparecen pocas veces. En este caso i es la i-ésima palabra más frecuente y F_i su frecuencia. Análogamente, si contamos la población en ciudades, y llamamos i a la i-ésima ciudad más poblada y F_i a su población, veremos que hay pocas ciudades con mucha población y muchas con poca población. Los dos ejemplos anteriores fueron los mencionados por Zipf en [5].

Al ser la ley de Zipf una función exponencial, al graficar F_i en escala log-log veremos una línea recta, cuya pendiente será el inverso aditivo del exponente o parámetro de Zipf.

Debido a está distribución sesgada, el número de palabras distintas en los datos no crece linealmente con el texto, sino que es sublineal. En la práctica, el vocabulario crece aproximadamente en forma proporcional a la raíz cuadrada del número total de palabras. Por esta razón es válido suponer que el vocabulario siempre puede ser almacenado en memoria principal [1].

2.2 Índices invertidos

Las componentes fundamentales de un índice son el vocabulario y la lista de posiciones (o de *posteo* usando un anglicismo). En el vocabulario están todas las palabras contenidas en documentos que han sido indexados por el buscador. Por cada palabra hay una lista (de posteo) con los documentos (y opcionalmente, en que posición del documento) en los que aparece, más alguna información adicional para calcular el ranking (normalmente el peso de la palabra en el documento). Con esos elementos básicos podemos responder consultas.

Al hablar del índice del buscador en lo que sigue, nos estaremos refiriendo al modelo recién descrito. En la figura 1 se ejemplifica el índice de un motor de búsqueda, a la izquierda está el vocabulario, y por cada palabra hay un puntero a una lista de documentos que son relevantes a la palabra. Naturalmente estas listas son de distinto largo.

La clave para la rapidez del motor de búsqueda es almacenar la mayor cantidad de información en memoria principal (RAM). Lo que no se almacena en RAM será almacenado en disco, el que tiene tiempo de acceso un par de órdenes de magnitud más lento que la memoria principal.

Un buscador de tamaño mediano tiene un vocabulario del orden de millones de palabras y similar orden de documentos, por lo que el tamaño del índice puede llegar a ser extremadamente grande. Al ser, en general, una restricción más fuerte para un computador la capacidad de memoria RAM que el tamaño de su disco, será deseable utilizar la mínima cantidad de memoria RAM que

nos permita un buen desempeño, ya que esto nos permite mayor escalabilidad en una Web siempre creciente.

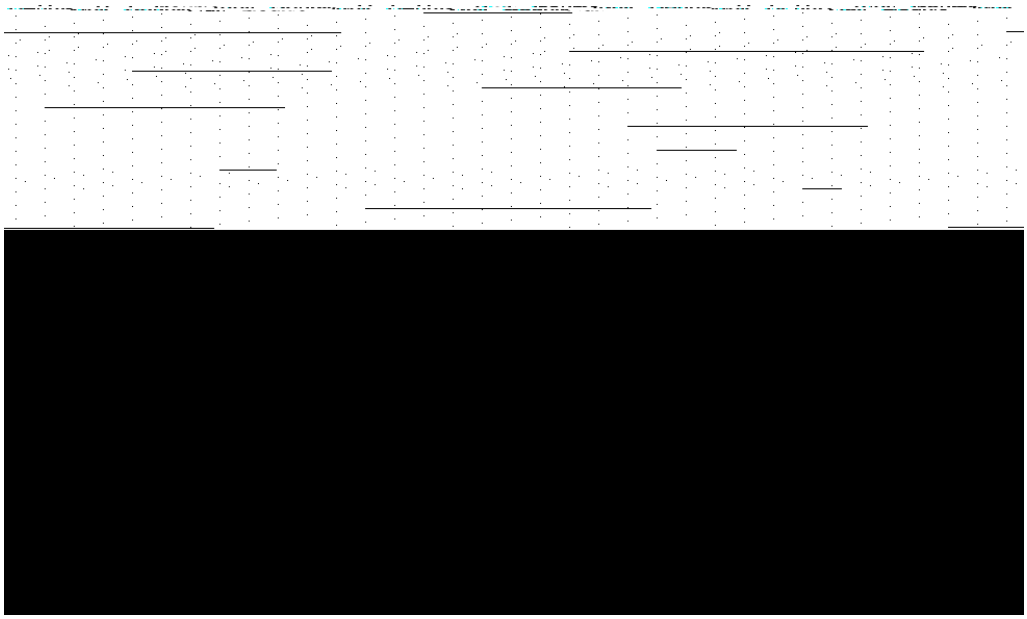

Figura 1: Estructura del índice de un motor de búsqueda.

3 Distribuciones Empíricas

En este trabajo ocupamos alrededor de 800 mil páginas (documentos) de la Web Chilena proporcionadas por el buscador TodoCL [4]. En esas páginas encontramos $T = 151.173.460$ palabras, de las cuáles $N = 2.067.040$ eran distintas. El largo promedio de una palabra en el vocabulario es de $\bar{x} = 8.46$. La cantidad de documentos en las que aparece una palabra sigue una distribución de Zipf, como se ve en la figura 2.

Además se consideraron consultas de un périodo de tres meses del año 2001 que incluían 738.390 palabras en consultas simples, con un vocabulario único de $C = 465.021$ palabras. En la figura 3 se muestra la frecuencia de las palabras consultadas. Se observa claramente que también siguen una distribución tipo Zipf. El parámetro de la distribución es aproximadamente 1.42. Eso indica que pocas palabras son consultadas muchas veces, y muchas palabras pocas veces. El mismo análisis en un buscador de España, Buscopio [2], arroja una distribución con parámetro similar (1.31).

La correlación estándar entre la frecuencia de una palabra en los documentos y la consulta es de 0.15, bastante baja, lo que se puede ver en la figura 4.

4 Zipf es Nuestro Amigo

En esta sección se expondrá un modelo simple de buscador y una forma de organizarlo internamente, de manera de hacer uso eficiente de recursos; mejorando la escalabilidad en una Web que crece

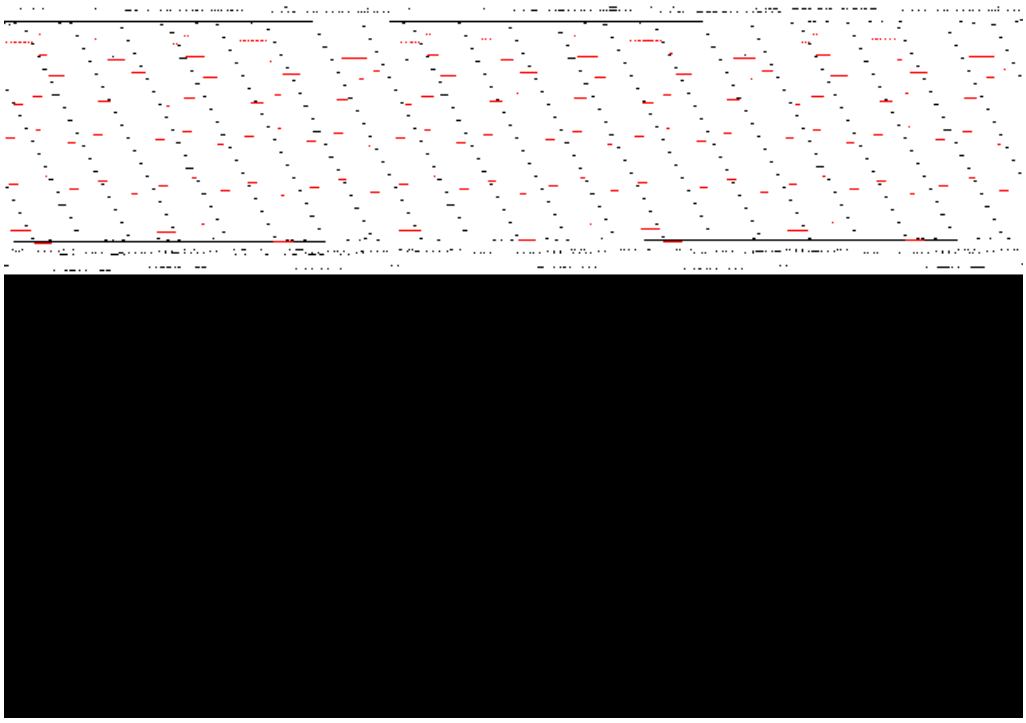

Figura 2: Palabras en el índice v/s cantidad de documentos en los que está la palabra.

aceleradamente.

Como hemos visto en secciones anteriores, tanto las palabras consultadas como las palabras que aparecen en los documentos, siguen distribuciones tipo Zipf en lo que respecta a su frecuencia. Esto nos indica que pocas palabras son consultadas muchas veces y pocas palabras aparecen muchas veces en los documentos. Este hecho puede ser utilizado a favor del buscador, ya que deja de ser importante cuánto crece el vocabulario, y nos comienza a interesar cuánto crece el segmento más consultado, quedándonos sólo con éste en memoria principal y el resto en memoria secundaria, como se ilustra en la figura 5, donde el primer bloque de vocabulario tiene su lista de posteo en memoria principal, ahí estan las palabras más consultadas. En el segundo bloque (de más abajo) se encuentran las palabras menos consultadas, las palabras mismas están en memoria principal pero sus listas de posteo son llevadas a disco.

Sea N la cantidad de palabras en el vocabulario, L_i la cantidad de documentos en los que aparece la $i-$ésima palabra del vocabulario, donde las palabras $1..N$ están ordenadas por frecuencia en las consultas en forma decreciente. Por cada palabra del vocabulario se necesita el espacio de la palabra, más un puntero a la lista (4 bytes), y por cada elemento de la lista se requiere un entero para identificar el documento (puede ser a la vez un puntero a su ubicación en disco) y otro entero para almacenar la frecuencia de la palabra en ese documento (tf). Entonces el espacio promedio, E, utilizado por la estructura ya mencionada, en bytes, es:

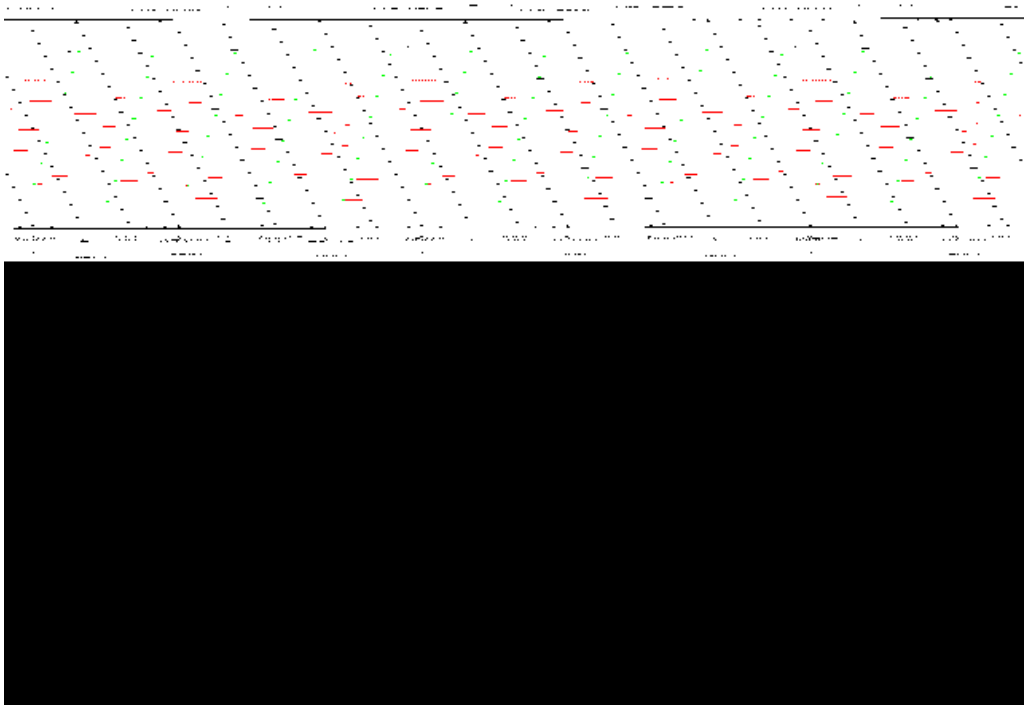

Figura 3: Frecuencia de las palabras consultadas en TodoCL en escala log-log.

$$E = \sum_{k=1}^{N}(4 + \overline{x} + 8L_k) = \underbrace{(4 + \overline{x})N}_{V} + 8\sum_{k=1}^{N} L_k$$

donde $\overline{x}$ es el largo promedio de las palabras en el vocabulario y V el tamaño del vocabulario. Notemos que al usar $\overline{x}$ estamos asumiendo independencia entre el largo de las palabras y la frecuencia de consulta, lo cual es cierto para el caso de TodoCL, donde la correlación es muy cercana a cero (-0.02).

Si el computador que almacena este buscador tiene memoria de tamaño M, mientras $E < M$ podemos almacenar el índice completo en memoria. El momento interesante es cuando $E > M$, caso en el que debemos almacenar parte de la estructura en disco. Sería posible confiar en el paginamiento de disco, pero en general es poco controlable, dado el tamaño de las páginas de disco, por lo que no hay garantías de optimalidad y, en el peor caso, obtenemos un sistema extremadamente lento.

Analicemos cómo vamos a distribuir la información, entre memoria y disco, cuando $E > M$. Mantendremos siempre todo el vocabulario en memoria principal (supondremos $M \geq V$). Ahora, tenemos que distribuir las listas de las palabras entre memoria principal y disco. Formalmente el problema a resolver es encontrar los índices $i_1, ..., i_p$ tal que maximizan la suma de la frecuencia de consulta de las palabras $i_1, ..., i_p$ con la restricción que $V + 8\sum_{j=1}^{p} L_{i_j} \leq M$.

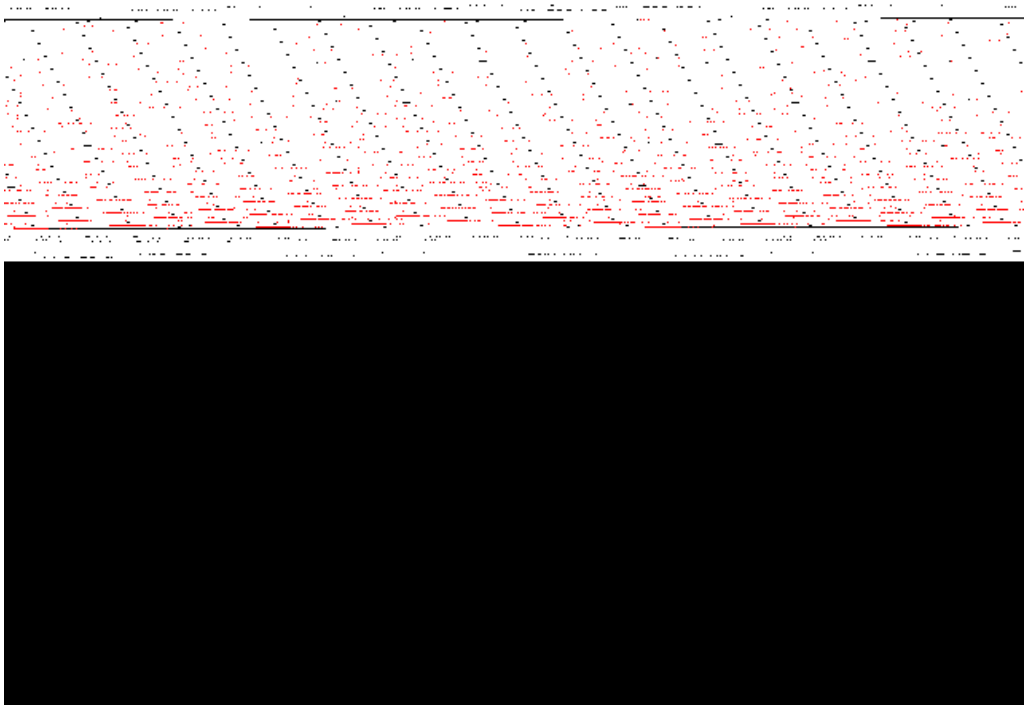

Figura 4: Cantidad de consultas v/s documentos relevantes para las palabras.

Como la distribución es tan sesgada, si numeramos las palabras desde la de mayor frecuencia de consulta a la de menor frecuencia, una buena heurística es poner en memoria principal las listas asociadas a las palabras más consultadas. Definamos p como la mayor cantidad de palabras del vocabulario tal que

$$V + 8 \sum_{k=1}^{p} L_k \leq M$$

Luego las p palabras más consultadas tendrán las listas en memoria principal. Esto no es óptimo ya que en algunos casos será más eficiente reemplazar una palabra de alta frecuencia de consulta, y lista de posteo larga, por dos palabras que en suma tengan mayor frecuencia y menor tamaño de lista de posteo. Pero, como veremos en la próxima sección, la heurística ya es suficientemente buena, dada la distribución tan sesgada de los datos.

Recordemos que el acceso a las palabras sigue una ley de $Zipf$, es decir, si F_i es la frecuencia de consulta de la palabra i, ésta es tal que

$$F_i = \frac{R}{i^\alpha}$$

donde α es el parámetro de Zipf y R la constante de proporcionalidad. Entonces la cantidad de consultas para las que no se hará acceso a disco será:

$$R \sum_{i=1}^{p} \frac{1}{i^\alpha} \leq R \int_1^{p+1} \frac{1}{x^\alpha} dx = R \frac{(p+1)^{1-\alpha} - 1}{1 - \alpha}$$

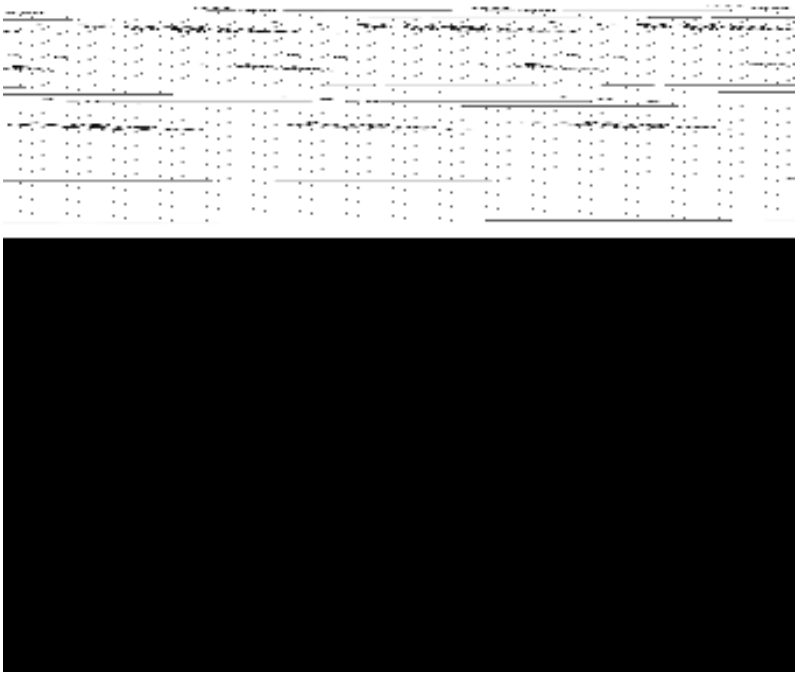

Figura 5: Orden para el índice de motor de búsqueda.

Recordemos que, como ordenamos las palabras del vocabulario según su frecuencia de consulta, las palabras cuyas listas quedan en memoria principal son las más consultadas. En la figura 5 se observa lo descrito.

A continuación veremos un caso numérico para TodoCL, en el cual calculamos los tamaños de las listas de posteo para las palabras en el índice. Este cálculo se hizo siguiendo las estimaciones de la sección anterior. En el gráfico de la figura 6 se ve la diferencia entre un orden simple (curva que va por abajo), por palabras más consultadas, y un orden aleatorio de la lista (es similar a un caso promedio, en el que todas las palabras contribuyen a la misma utilización del índice e igual frecuencia de consulta). En el eje x está la porción de palabras consultadas que en el vocabulario tienen su lista en memoria principal. Notemos que no son necesariamente distintas, ya que una palabra puede ser consultada varias veces. El eje y indica la porción del tamaño total de índice que se debe tener en memoria para alcanzar esa porción de consultas accesando sólo memoria. Directamente del gráfico se pueden extraer algunas conclusiones. Por ejemplo,

- Para alcanzar el 100% de consultas a memoria basta con el 80% del espacio total, esto se debe a que hay un 20% del índice contiene las palabras que nunca se consultan (que son más del 75% del total).
- Para alcanzar un 50% de consultas basta el 20% del índice en memoria.
- Con el 50% del índice en memoria tendremos un 80% de accesos a memoria.

Así, este ordenamiento simple mejora el uso de recursos del buscador, ahorrando uso de memoria

primaria en al menos un 20%. El tiempo de respuesta promedio, pese a usar menos memoria, se mantiene.

Observemos la brecha que aparece en la curva del orden aleatorio. Esta brecha aparece por que en ese punto se encuentran dos de las palabras más consultadas, su aporte al tamaño del índice es bajo comparado con la alta frecuencia que aportan a los accesos a memoria. En la curva optimizada esas palabra saparecen en el primer punto (los puntos se grafican cada 100 palabras).

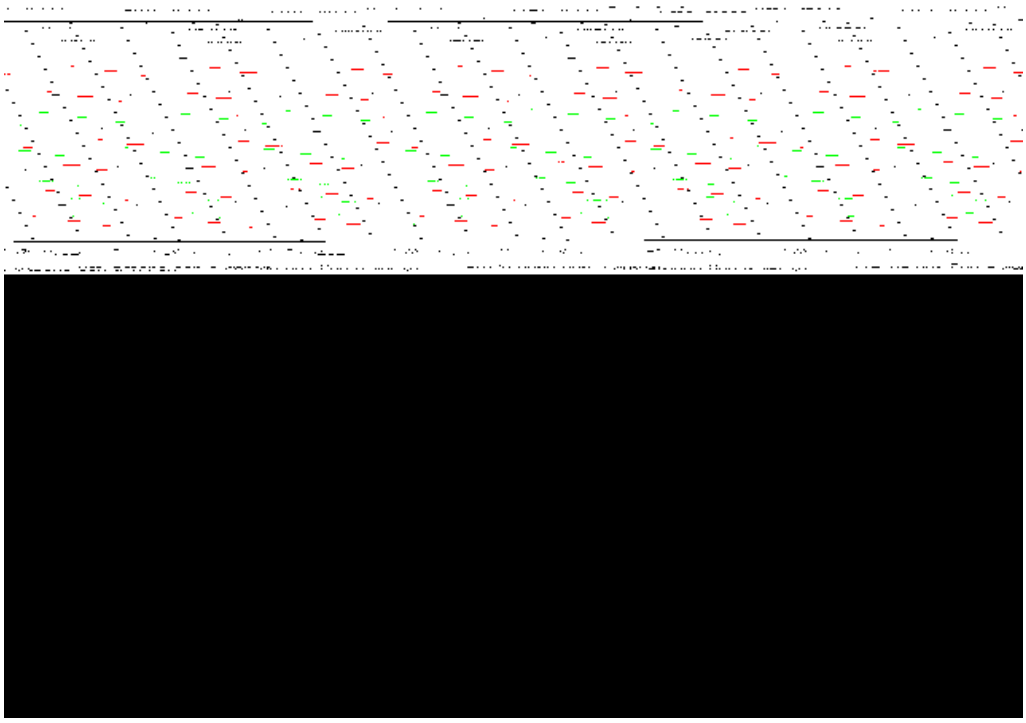

Figura 6: Uso del índice para caso optimizado y aleatorio.

5 Optimizando el Uso de Memoria: Respuestas Precalculadas

Las consultas siguen una distribución similar a las palabras, por lo que hay pocas consultas que se hacen muchas veces. Para esas consultas se pueden tener precalculados los resultados en un cache, liberando al buscador de carga al retornar esos resultados sin acceder al índice.

En la figura 7 el eje x representa la cantidad de consultas para las que se han precalculado respuestas. El eje y es la porción de consultas realizadas que se responden del cache de respuestas precalculadas. Vemos que la cantidad de respuestas precalculadas necesarias para responder al primer 20% de las consultas es muy baja, es decir, es barato mantener las respuestas precalculadas para el primer 20% de las consultas: con 2000 respuestas precalculadas atendemos del cache el 20% de las consultas hechas al buscador.

Para que las respuestas precalculadas se utilicen rápido, deben estar en memoria principal compitiendo por recursos con el índice, por lo que al tener más respuestas precalculadas posiblemente

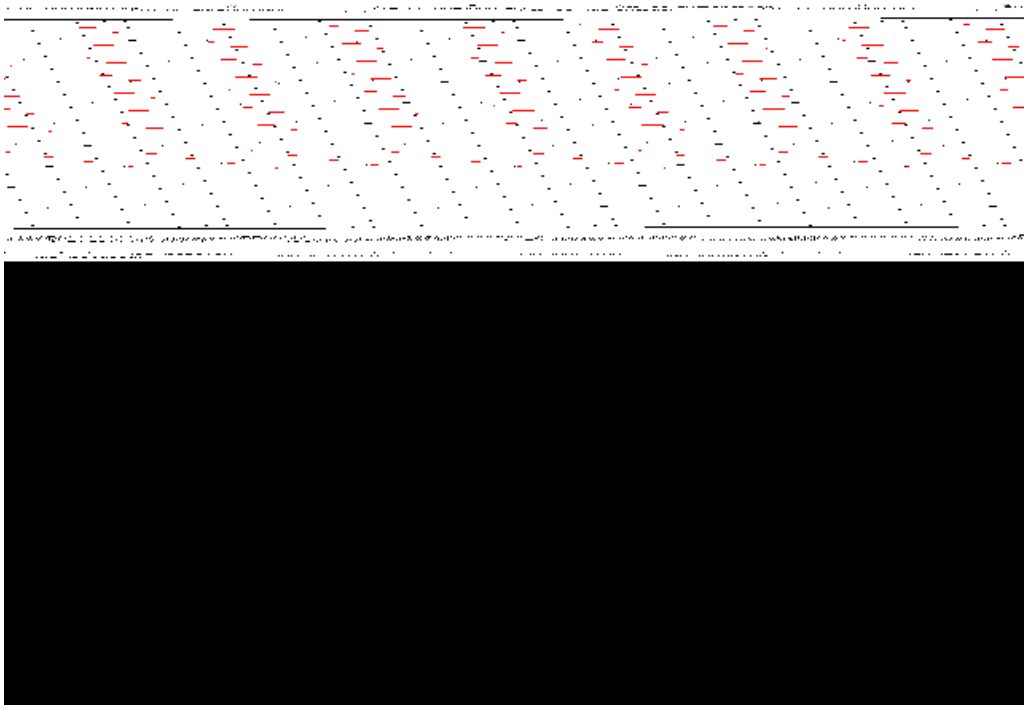

Figura 7: Uso del cache de respuestas precalculadas.

tenemos que llevar listas de posteo de más palabras a disco.

Supongamos que las k palabras más frecuentemente consultadas tienen su respuesta precalculada y cada respuesta utiliza W bytes de memoria. Luego se debe cumplir que

$$kW + V + 8 \sum_{i=k+1}^{p+k} L_i \leq M$$

donde de $k+1$ a $p+k$ son las p siguientes palabras más consultadas cuyas listas de posteo están en memoria y M la cantidad de memoria principal disponible. Queremos obtener los valores de k y p que minimicen el tiempo de respuesta, los que están relacionados para una cantidad de memoria fija. Ahora, notemos que al haber respuestas en el cache, cambia el conjunto de las palabras más consultadas que están en memoria, por lo que debe recalcularse la suma para cada cantidad posible de respuestas precalculadas (k).

Bajo el supuesto que el largo de la lista de posteo L_i es independiente de la frecuencia de consulta (como vimos antes tienen una correlación baja), es posible estimar el largo de las listas por:

$$E(L_i) = \frac{T}{N} = \gamma$$

donde recordemos que $T = \sum_{i=0}^{N} L_i$ es el número total de palabras en los documentos y N es la cantidad de palabras en el vocabulario. Por lo tanto se tiene aproximadamente:

$$M = kW + V + 8p\gamma$$

Despejando p obtenemos:

$$p = \frac{(M - kW - V)}{8\gamma} = \frac{(M - V)}{8\gamma} - \frac{W}{8\gamma}k$$

Claramente $k \leq (M - V)/W$, pues p es no negativo.

De esta forma, hay tres secciones en el motor de búsqueda de donde podemos responder consultas. Estas son el cache, índice en memoria principal (RAM) e índice en memoria secundaria (disco). En la figura 8 se observa cómo cambian las frecuencias de uso de las distintas secciones de respuesta, al tener en cache distintas cantidades de respuestas precalculadas.

Podemos luego calcular el tiempo de respuesta promedio del motor de búsqueda según la cantidad de respuestas precalculadas, donde el tiempo de respuesta es:

$$t = \frac{\beta_1 \sum_{i=1}^{k} F_i + \beta_2 \sum_{i=k+1}^{p+k} F_i + \beta_3 \sum_{i=p+k+1}^{C} F_i}{\sum_{i=1}^{C} F_i}$$

donde C es el vocabulario de las consultas y β_1, β_2 y β_3 constantes que representan los tiempos de acceso relativos de los distintos medios de almacenamiento que son cache, índice en RAM e índice en disco, respectivamente. Para los cálculos se utilizó $M = 400Mb$, $W = 40Kb$, $\beta_1 = 1$, $\beta_2 = 5$ y $\beta_3 = 50$. Usando los valores que ya teníamos obtenemos $\gamma = 73.1$ y $V = 24.56Mb$. La figura 9 muestra el tiempo de respuesta en función de las respuestas precalculadas, en la cual observamos que el óptimo está en 1338 respuestas precalculadas para los datos considerados, mejorando el tiempo de respuesta en aproximadamente un 7%.

6 Conclusiones

Al ordenar el vocabulario de un motor de búsqueda según la frecuencia de consulta de las palabras, podemos mejorar el uso de recursos del buscador haciendolo más eficiente y escalable. Además es buena idea, para mejorar el tiempo de respuesta, mantener un cache de respuestas precalculadas. Los dos métodos anteriores son combinables en un índice que mantiene tres secciones de información: en cache, memoria principal y memoria secundaria. La utilización combinada da como resultado un índice eficiente en el uso de los recursos.

Los resultados anteriores fueron obtenidos a partir de la simulación de un motor de búsqueda optimizado sometido a las consultas que recibió TodoCL. En caso de implementarse la metodología expuesta es necesario utilizar algoritmos en línea para el ordenamiento del índice. Una alternativa sería aplicar alguna estrategia como trasposición (intercambiar dos elementos de la lista de palabras más consultadas) para la división del índice entre memoria principal y secundaria. Además el modelo de uso de memoria es más complicado en la práctica al igual que el modelo de tiempo de respuesta, pero similares a los presentados en este artículo.

También queda por analizar nuestra heurística para determinar su factor de aproximación (es decir, cuán lejos del óptimo se encuentra).

Referencias

[1] R. Baeza-Yates y B. Ribeiro-Neto. *Modern Information Retrieval*, ACM Press/Addison-Wesley, England, 513 pages, 1999. URL: `http://sunsite.dcc.uchile.cl/irbook/`.

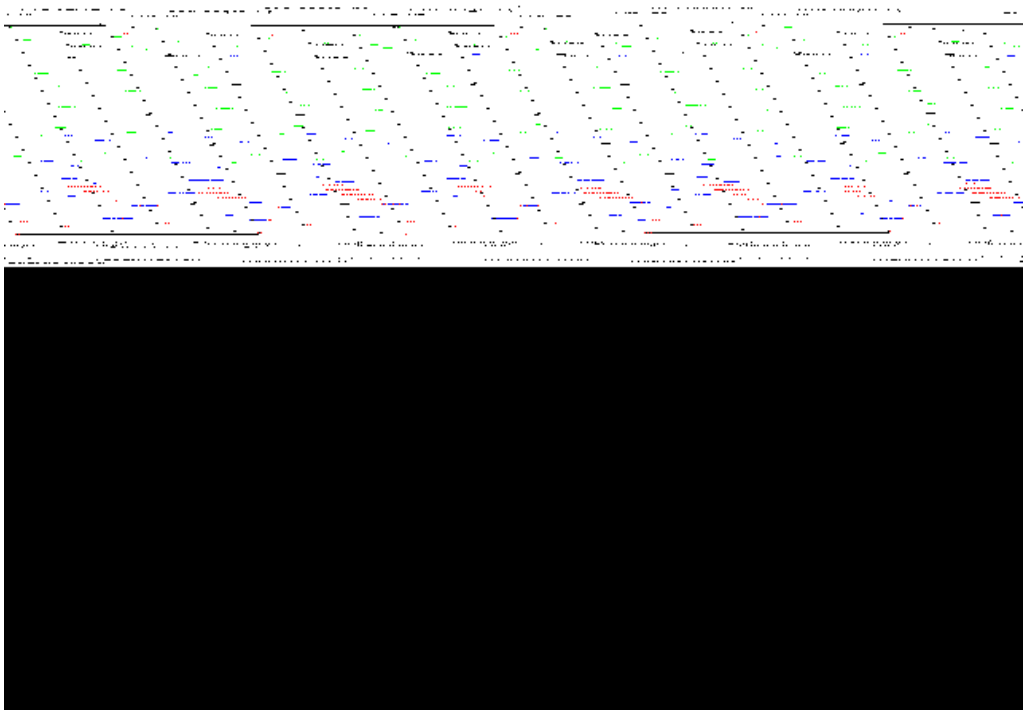

Figura 8: Frecuencia de uso de las distintas secciones de respuesta.

[2] Buscopio: Página principal `http://www.buscopio.com`, 2001.

[3] Doug Beeferman y Adam Berger. Agglomerative clustering of a search engine query log, Proceedings on the 2000 Conference on Knowledge Discovery and Data Mining (Boston,MA), pages 407-416, Aug. 2000

[4] TodoCL: Página principal `http://www.todocl.cl`, 2000.

[5] George Zipf. Selective Studies and the Principle of Relative Frequency in Language, Harvard University Press, Cambridge, MA, 1932.

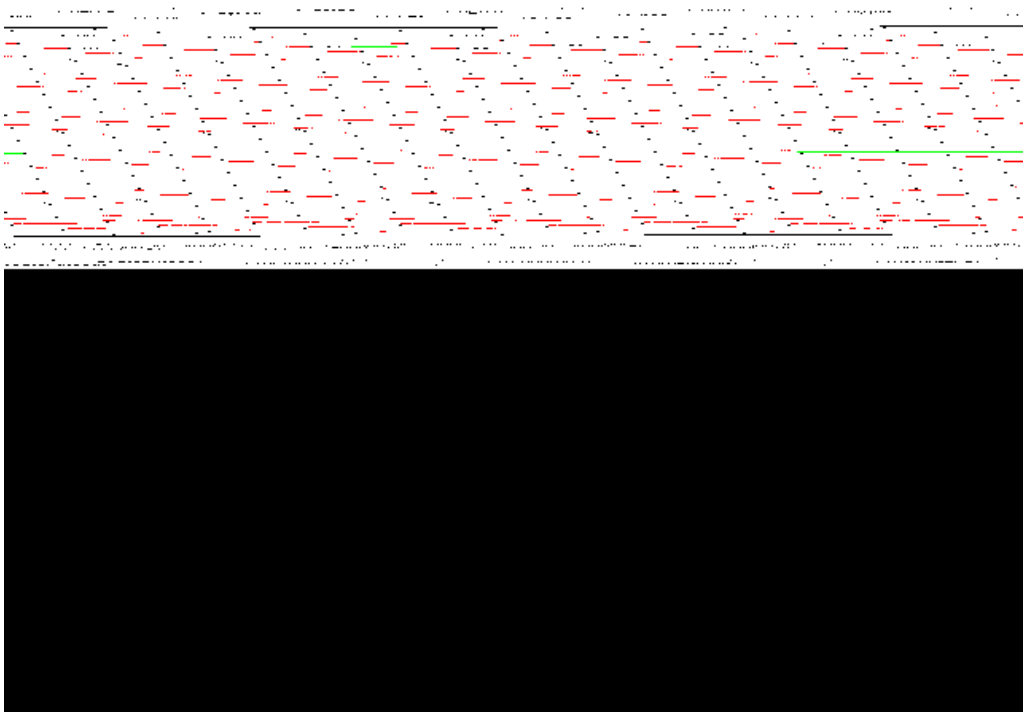

Figura 9: Tiempo de respuesta en función de la cantidad de respuestas precalculadas.