Análisis Experimental de Árboles de Pivotes Fijos *

Ricardo Baeza-Yates
Iván Rivera Jofré
Depto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
E-mail: {rbaeza,irivera}@dcc.uchile.cl

Resumen

Un árbol de pivotes fijos (o Fixed-Queries Tree) es una estructura de datos que permite realizar búsquedas de elementos similares a una consulta dada, usando una función de distancia métrica entre elementos. En este trabajo presentamos una optimización de los árboles de pivotes fijos, y de los algoritmos de búsqueda asociados, la cual se logró mediante el análisis del comportamiento experimental de búsquedas aproximadas usando un espacio métrico sobre las palabras de un diccionario. Se presentan valores para la altura óptima del árbol para así obtener eficientes resultados en forma rápida y con buen manejo de la memoria. Se entregan detalles de los experimentos y cómo se puede extender a otros espacios métricos, así como aspectos desconocidos que pueden aplicarse a otras estructuras que utilicen la distancia de edición.

1 Introducción

La búsqueda aproximada es un problema que puede tener aplicaciones muy importantes en diversas áreas, en particular, encontrar objetos parecidos a uno dado, incluyendo impresiones digitales, archivos, grafos, etc. El caso particular de búsqueda aproximada de palabras en un vocabulario permite a un sistema poder realizar corrección ortográfica u otro tipo de búsqueda en un procesador de textos. La eficiencia en rapidez de respuesta, manejo de memoria y respuesta adecuada hace necesario la existencia de estructuras de datos especializadas que incluyan estos aspectos. Numerosos intentos han sido desarrollados por lograr una estructura y algoritmos eficientes, sin embargo, el desempeño no es absolutamente aceptable sobre todo para sistemas en línea, los cuales necesitan respuestas en muy poco tiempo y realizar varias búsquedas en forma simultánea (se piensa por ejemplo en un sistema conectado en Internet que permita realizar búsquedas aproximadas en el Web). Este trabajo intenta optimizar una de las estructuras más eficientes en este tipo de aplicaciones, el árbol de pivotes fijos ó *Fixed-Queries Tree* (FQT) de altura fija (FHQT), de manera experimental logrando obtener valores de los parámetros de construcción de la estructura que permiten revisar menos de un 1% de un diccionario para buscar una palabra con 1 error. Este trabajo se realizó exclusivamente para espacios métricos de palabras o strings utilizando la distancia de edición.

Para lograr la optimización de la estructura se estudian los tres aspectos más relevantes: espacio utilizado en memoria, características de la distancia de edición y número total de comparaciones realizadas en un proceso de búsqueda de proximidad. El trabajo comienza con una breve explicación

*Este trabajo fue financiado parcialmente por el proyecto Fondecyt 1990627.

de los trabajos previos sobre la estructura, para luego seguir con los aspectos teóricos necesarios para la comprensión de los experimentos que siguen en la sección siguiente. Finalmente se entregan conclusiones y las recomendaciones para los valores de los parámetros que optimizan los FHQT.

2 Trabajo Previo

La búsqueda aproximada de elementos en un espacio cualquiera es un problema que ha recibido numerosos intentos por ser resuelto en forma eficiente [7]. Se han creado diversas estructuras que intentan particionar el espacio en pequeñas partes indexadas de alguna manera con el objetivo de reducir el número de comparaciones necesarias para determinar la solución. En el caso de búsqueda aproximada de elementos en un espacio métrico (que utiliza una función de distancia que cumple con la desigualdad triangular) se han creado estructuras que permiten revisar solo un pequeño porcentaje de los elementos del espacio permitiendo una respuesta rápida y correcta.

Para búsquedas de tipo en línea (*on-line*), donde se puede realizar un preproceso al patrón de búsqueda y además a los elementos del espacio, se utiliza normalmente estructuras complicadas y que por lo general ocupan un gran espacio en memoria. Estas estructuras permiten entregar una solución de manera mucho más rápida que una búsqueda de tipo retardado (*off-line*), donde no se necesita entregar una respuesta tan rápida, por ejemplo en un sistema mono-usuario. Algunas estructuras utilizadas son los árboles BK (en honor a sus autores Burkhard y Keller [4]), árboles de pivotes fijos o FQT con y sin la variante de altura fija [2], y muchas otras variadas estructuras [7]. Existen algunas otras estructuras más recientes que obtienen la solución correcta utilizando menos memoria pero no logran reducir el número de comparaciones a una cantidad razonable. Para mayor información en estructuras de datos para espacios vectoriales ver [6] y para espacios métricos, incluyendo una comparación experimental [5].

Este trabajo se centra en la estructura árbol de pivotes fijos ó Fixed-Queries Tree (FQT) desarrollado en [2]. Esta estructura se utiliza en búsquedas on-line sobre espacios métricos y se obtienen excelentes resultados sobre todo en la búsqueda aproximada de palabras o strings utilizando la distancia de edición o Levenshtein. En el trabajo inicial de esta estructura [2] se muestran diversos ejemplos del funcionamiento del FQT y se abre el problema de encontrar los valores adecuados de los parámetros para obtener un rendimiento óptimo. En aquel trabajo los resultados son buenos pero no lo suficiente como para destacarse sobre otras estructuras. En un trabajo posterior [1], se realiza una comparación experimental entre los árboles BK y FQT obteniendo resultados bastante buenos y en muchos casos mejor que los árboles BK. En ese trabajo se desarrolla más en detalle la variante de los FQT de altura fija y se llega a encontrar una fórmula para optimizar la altura del FHFQT.

3 Conceptos Básicos

3.1 Búsqueda en Espacios Métricos

El concepto de búsqueda aproximada de elementos tiene varias interpretaciones, o varios tipos de búsqueda. En general existen 2 tipos de búsqueda a partir de una consulta q:

a) Recuperar todos los elementos a una distancia D de q.

b) Recuperar los M elementos más cercanos a q.

La distancia a la cual se alude es aquella función que permite que el espacio se considere un espacio métrico, es decir, una función que cumple las siguientes 3 condiciones:

$$d(x,y) = 0 \iff x = y$$
$$d(x,y) = d(y,x)$$
$$d(x,z) \leq d(x,y) + d(y,z)$$

Esta última condición es la llamada desigualdad triangular.

Las condiciones impuestas para la función de distancia son válidas para muchas funciones entre las cuales destaca la distancia de edición en el espacio de palabras, que es aquella distancia que calcula la cantidad mínima de caracteres que hay que agregar, cambiar y/o eliminar a una palabra para obtener otra palabra. Esta distancia es discreta, lo cual facilita su uso en la estructura que se analiza. Calcular la distancia de edición necesita tiempo proporcional al producto del largo de las dos palabras (es decir, tiempo cuadrático) usando un algoritmo basado en programación dinámica. Por esta razón, dado el alto costo de cada comparación entre dos palabras, queremos minimizar el número de comparaciones.

Esta distancia, a pesar de ser simple a primera vista, posee ciertas características que se pueden aprovechar en la optimización de cualquier estructura que la utilice y que se comprenderán más adelante al observar los resultados experimentales. Esta característica de la distancia de edición se complementa con las particularidades del vocabulario y de su alfabeto produciendo ciertas distribuciones en las distancias entregadas que no son intuitivas y que pueden llegar a ser deducidas analíticamente.

3.2 Árboles de Pivotes Fijos

La idea del FQT es básicamente particionar el espacio en pequeñas divisiones que posean una cantidad máxima de b elementos. El árbol se define de la siguiente manera: para cada nodo interno a una profundidad h del árbol se asociará un elemento fijo del espacio (pivote), y todos los elementos que se encuentren bajo el hijo i del nodo estarán a distancia i del pivote fijo. Cuando los elementos de un subárbol sean b elementos o menos, entonces estos se almacenarán en un bucket de tamaño máximo b. La Figura 1 muestra un ejemplo de esta estructura con buckets de tamaño 2 y palabras de largo 4 sobre un alfabeto binario, siendo los q_i los pivotes fijos. En este caso, la distancia usada es Hamming (es decir, el número de caracteres distintos en la misma ubicación).

De esta manera para responder consultas del tipo "todos los elementos a distancia D de q", se busca a partir de la raíz del árbol. La raíz del árbol (nodo interno de altura 0) tendrá asociada un pivote fijo. Si la distancia entre la consulta q y el pivote fijo es j, entonces los elementos a retornar como solución son aquellos que están entre las hijos $j - D$ y $j + D$. Se aplica lo mismo recursivamente para cada hijo de ese rango. Al momento de llegar a un bucket, simplemente se compara uno por uno cada elemento del bucket.

La ventaja de este árbol es que se pueden realizar las comparaciones con todos los pivotes fijos de una sola vez y posteriormente se puede recorrer el árbol de cualquier manera sin necesidad de calcular una nueva distancia, cuya evaluación se supone que es un proceso caro.

La variante de altura fija consiste en que todos los buckets están a la misma altura h y no poseen capacidad máxima (aunque en promedio tienen un sólo elemento para la altura óptima). En

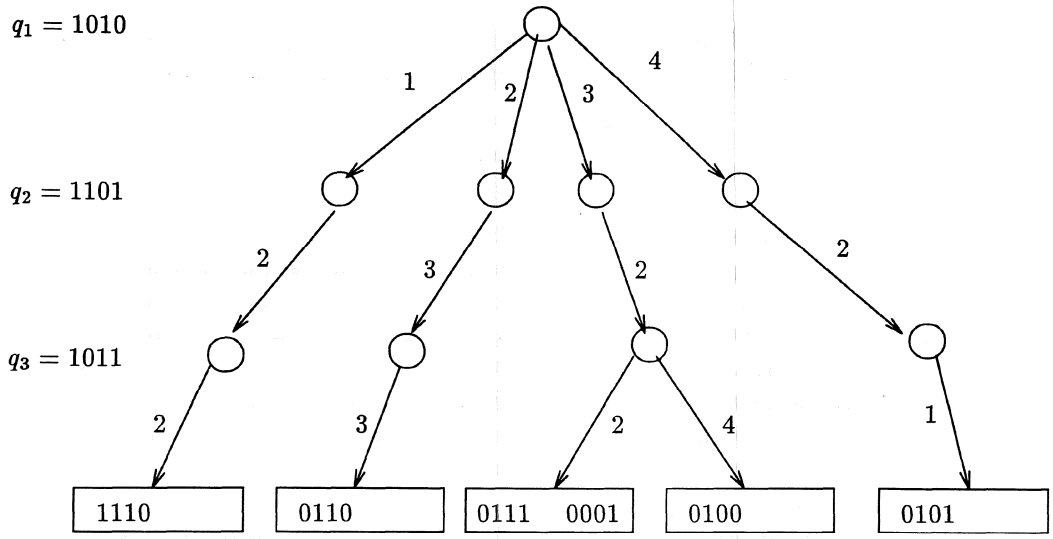

Figura 1: FQT de altura fija 3 y buckets de tamaño 2.

algunos casos se tendrá que algunos buckets estarán con muy pocos elementos y a una profundidad innecesaria, pero esto se aprovechará al realizar búsquedas puesto que se pueden descartar varias ramas de este tipo debido a la desigualdad triangular. La Figura 1 muestra un árbol de este tipo usando altura fija tres. Como se puede observar, algunos caminos unarios podrían eliminarse, si no forzamos a los buckets a tener todos la misma altura.

Los pivotes fijos del FQT (o del FHFQT) son elementos del espacio métrico y no son necesariamente elementos que estarán en el FHFQT. Pueden ser escogidos al azar en el espacio o en el conjunto de elementos del FQT.

Para un espacio métrico de palabras o strings, se debe poseer un vocabulario o diccionario sobre el cual se realizan las búsquedas y que corresponde a todas las palabras que se ingresarán en el FQT. Este vocabulario posee un alfabeto de σ caracteres y se debe tener en cuenta la distancia máxima que pueda haber entre dos palabras cualesquiera. Esto corresponderá a la dimensión del espacio o largo máximo de una palabra.

4 Análisis Experimental

4.1 Metodología

La optimización experimental del FHFQT se realizó mediante la búsqueda aproximada de palabras utilizando la distancia de edición. Para los experimentos se utilizó un diccionario en inglés de 69069 palabras con un alfabeto de 26 letras. La palabra más larga tiene 21 letras. El vocabulario tiene 8.4 letras por palabra en promedio. La implementación se realizó en GNU C++ sobre SunOS 5.6 en un procesador UltraSparc.

Los experimentos realizados en busca de una optimización de la estructura corresponden a los tres aspectos más importantes: memoria utilizada, características del vocabulario y búsquedas. Una vez terminadas estas tres etapas, se procedió a buscar valores óptimos mediante índices de eficiencia simples. Finalmente se realiza una comparación con trabajos anteriores sobre esta estructura.

4.2 Elección de los Pivotes

La elección de los pivotes fijos en un FQT (y en general, en todas las estructuras que utilizan elementos en el espacio como pivotes) sigue siendo un problema abierto. Debido a ello, se decidió para los experimentos utilizar 2 tipos de pivotes:

a) Palabras aleatorias escogidas del vocabulario.

b) Palabras aleatorias de largo fijo.

En el caso de pivotes fijos del tipo (b) las palabras no tienen sentido necesariamente y pueden no semejar palabras del idioma (por ejemplo, `hfkjsdhjhf` puede escogerse como palabra fija de largo 10. Las palabras fijas del tipo (a) serán notadas en las figuras como **dict** y las del tipo (b) de largo ℓ serán notadas como $||\mathbf{fixed}|| = \ell$. Al final se podrá determinar que tipo de pivotes fijos entrega un funcionamiento mejor.

4.3 Memoria utilizada

El espacio en memoria utilizado por la estructura será necesariamente mayor al tamaño del vocabulario. A medida que crece el tamaño del vocabulario, crece también el espacio extra ocupado por el FQT, ya sea normal o de altura fija. Por lo tanto, no existe una altura óptima considerando sólo la memoria ocupada por el FHFQT dado su uso de espacio monótonamente creciente. De hecho, en el límite, con cada aumento de altura, el espacio crece en $O(n)$. Si la altura del FQT es h y la altura del FHFQT es H, entonces $Espacio(FHFQT) - Espacio(FQT) = O((H-h)n)$. La Figura 2 muestra el espacio extra utilizado por la estructura (es decir sin considerar el espacio ocupado por el vocabulario almacenado en el FQT) para distintas alturas fijas del FHFQT. El espacio se mide como un porcentaje con respecto al tamaño del vocabulario.

Se puede observar el comportamiento creciente de las curvas y notar que el espacio extra es bastante grande (por ejemplo, el FHFQT de altura 16 con elementos fijos del vocabulario ocupa 10 veces el tamaño del vocabulario). Para pivotes fijos aleatorios de largo fijo, el espacio ocupado es siempre menor que con palabras fijas del vocabulario. Esto se produce debido a que cuando las palabras ingresadas en el FQT se distribuyen en un rango mayor en cada nodo, existe mayor cantidad de nodos internos y por lo tanto ocupará más memoria. En cambio cuando las palabras ingresadas no se distribuyen tanto en cada nodo interno, se tendrán las palabras agrupadas o acumuladas en torno a un centroide y que producirá buckets con mayor cantidad de elementos, por lo tanto el espacio ocupado es menor, debido a que hay menos nodos internos. Cuando los pivotes fijos aleatorios crecen en longitud, aumenta la memoria lo que hace suponer una mejor distribución. Para longitud 8 de las palabras fijas se produce una curva de memoria inferior a la de longitud 7, o podría decirse que la curva de longitud 7 es mayor a la esperada por la secuencia. Para longitudes mayores, la memoria ocupada se acerca monótonamente a la curva **dict** hasta ser casi la misma para longitud 16.

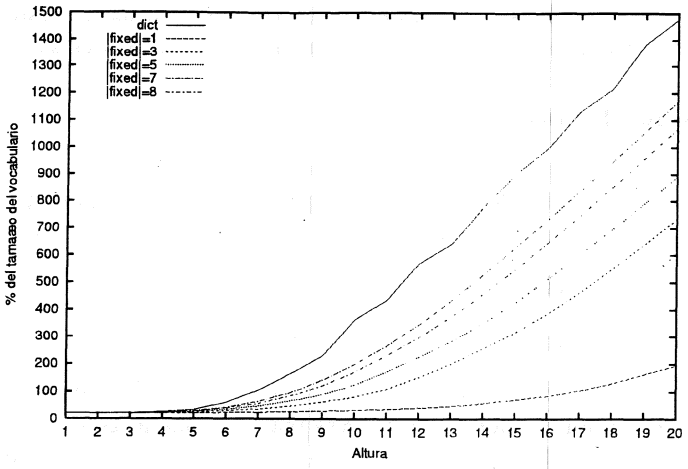

Figura 2: Espacio extra utilizado medido con respecto al tamaño del vocabulario.

En la Figura 3 se muestra la utilización de los buckets (palabras por buckets) para distintas alturas. La linea horizontal corresponde a la utilización de los buckets cuando las palabras fijas son del vocabulario. La idea de estos gráficos es ayudar al lector a comprender lo que sucede en un FHFQT para distintos valores de los parámetros. La utilización de los buckets se podría deducir directamente del gráfico de memoria utilizada o viceversa.

4.4 Distancia de Edición y Vocabulario

La distancia de edición se comporta de una manera no fácil de explicar pero si muy útil. Para un sistema de procesamiento de textos es útil poseer un corrector ortográfico que permita encontrar la palabra correcta cuando el usuario la escribe con errores. Sea una palabra del vocabulario w_0 y esta misma palabra con k errores (ya sea de cambio de letra, ausencia de una letra, o letra sobrante) w_k. Se desea entonces encontrar w_0 a partir de w_k. Utilizando un FQT se comparará w_k con las palabras fijas del FQT y se buscará todas las palabras a distancia k de w_k. Entre todas las palabras retornadas por el algoritmo de búsqueda, debe estar necesariamente w_0. Ahora, este conjunto retornado puede tener muchas palabras o muy pocas, pero siempre retornará las mismas para cualquier FQT si se utiliza el mismo vocabulario. Lo que interesa saber es si existe alguna manera de saber a priori en que hijos estarán esas palabras al momento de llegar a un nodo del árbol. Si los k errores son aleatorios, se puede pensar que esas palabras están uniformemente distribuidas en el rango de hijos que el algoritmo debe revisar (es decir, $[j-k, j+k]$ donde j es la distancia al pivote fijo), pero la distribución de las palabras es distinta.

Los histogramas de la Figura 4 muestran la distribución de la ubicación de la palabra w_0 con respecto a w_k al ser comparadas ambas con palabras aleatorias del diccionario. El gráfico superior izquierdo corresponde a palabras con 1 error (es decir, w_0 vs. w_1), el superior derecho a w_0 vs. w_2

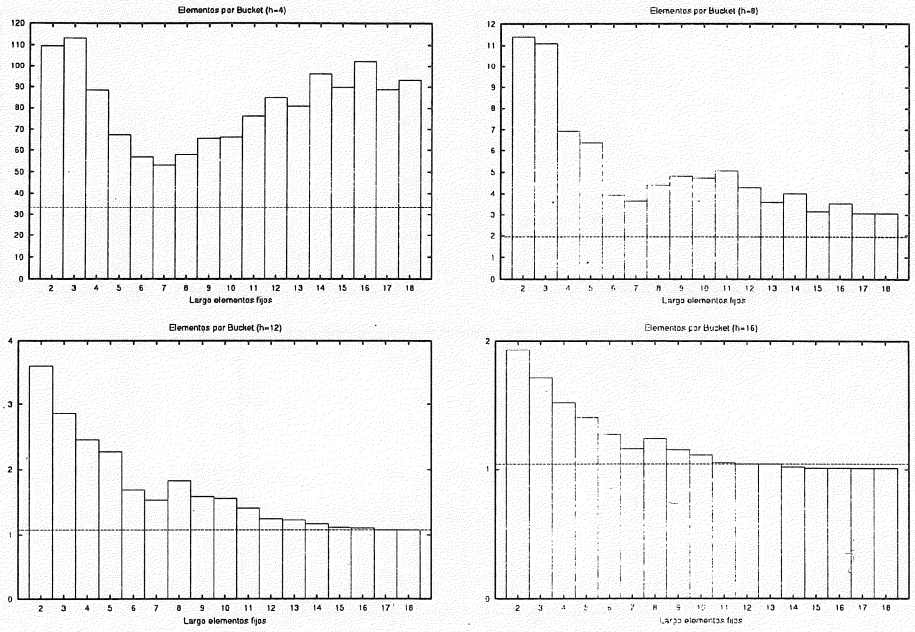

Figura 3: Utilización de los buckets para distintas alturas.

y el inferior a w_0 vs. w_3. En los tres histogramas donde d es la distancia de w_k al pivote fijo, se observa una tendencia a que la palabra w_0 aparezca en los hijos a la izquierda del hijo j.

Para una búsqueda del tipo (a), es decir de todos las palabras a distancia D de la consulta q, necesariamente debo buscar en todas las ramas del rango $[j-D, j+D]$ con lo cual esta característica sólo serviría para reportar más rápido las palabras buscadas, pero para el caso de una búsqueda del elemento más cercano, conviene utilizar esta característica pues de esta manera se hace más probable descartar ramas tempranamente lo que deriva en un número mucho menor de comparaciones o distancias evaluadas. En otras palabras, hay un orden óptimo para explorar las ramas : se recomienda buscar primero en la rama j y luego en las ramas $j-1$, $j+1$, $j-2$, etc. Una posible explicación de esta asimetría es que el número de palabras a distancia $j-1$ y $j+1$ es aproximadamente similar, pero el espacio de palabras a distancia $j-1$ es menor en volumen a las de distancia $j+1$, así que suponiendo una densidad uniforme de palabras similares en cada caso, la fracción recuperada en la rama $j-1$ es mayor (de hecho, sólo la desigualdad triangular indica que el espacio es más compacto, porque la distancia más grande dentro de un conjunto de elementos a distancia j de otro es a lo más $2j$).

En los histogramas anteriores, las palabras w_0 y w_k se comparaban con palabras aleatorias del diccionario. Las figuras 5 y 6 muestran el comportamiento de la distancia de edición con respecto al largo de los pivotes fijos para un o dos errores, donde se grafican las curvas de los histogramas correspondientes con respecto al largo.

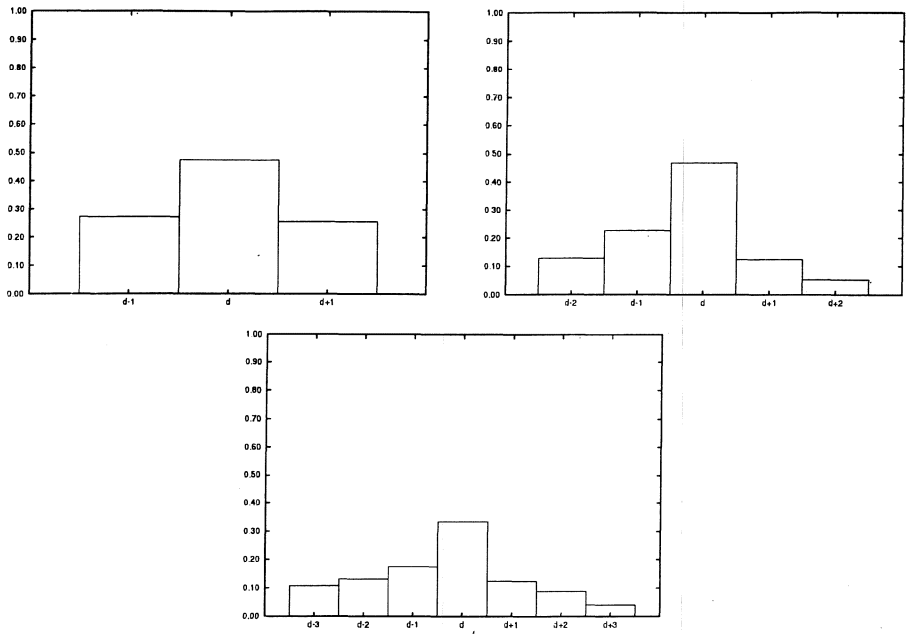

Figura 4: Histogramas de la ubicación de la palabra buscada con respecto al pivote.

4.5 Comportamiento de la Distancia de Edición

Para el tipo de búsqueda (a), puede no convenir que las palabras buscadas no estén distribuidas uniformemente, puesto que esto necesariamente implica que se harán comparaciones inútiles al buscar en los hijos donde es poco probable encontrar alguna palabra útil. La idea es aprovechar todas las comparaciones que se hagan. Entonces sería útil encontrar el largo de los pivotes fijos aleatorios que me permita aprovechar esas comparaciones, o mejor dicho, aquel largo que haga que la distribución de las palabras sea más uniforme en todo el rango de hijos, lo que está implícito o explícito en muchos trabajos previos de búsqueda en espacios métricos. En otras palabras, en los puntos donde las curvas de las figuras 5 y 6 están más cercanas. Para encontrar aquella longitud óptima se puede definir un criterio para calcular un índice sobre la cercanía de estas curvas. Utilizando el criterio de minimizar $\max(P) - \min(P)$ donde P es el conjunto de los valores de las probabilidades (P tiene 3 valores en el caso de $k = 1$) de las distancias y el criterio de minimizar la desviación estándar de P, se obtienen las longitudes óptimas para $k = 1$ dadas en la Tabla 1. En esta misma tabla se compara el valor obtenido con el del caso de usar elementos del diccionario como pivotes, observando que para ambos criterios el valor obtenido es mejor.

Se debe tener en cuenta que este criterio no quiere decir que con palabras fijas de longitud 4 ó 5 se tendrán menos comparaciones, pues estos criterios son solo teóricos y experimentalmente pueden variar un poco.

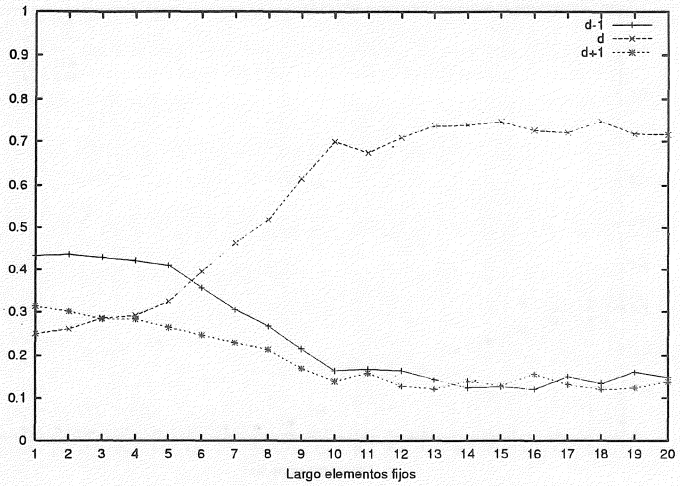

Figura 5: Distribución de la distancia de edición para 1 error.

Criterio	Longitud	Valor	Caso **dict**
$\max(P) - \min(P)$	4	0.15	0.22
σ^2	5	0.011	0.029

Tabla 1: Longitud óptima de los pivotes fijos y su comparación con **dict**.

4.6 Búsquedas Aproximadas

La existencia de una altura óptima esta teóricamente probado [3, 1]. En aquel documento se presenta una fórmula que permite encontrar la altura óptima de un FHFQT. Esta fórmula se basa en la distribución de frecuencias de la distancia de edición para el vocabulario utilizado. Es decir, se necesita saber la probabilidad que una palabra cualquiera del vocabulario esté en el subárbol bajo el hijo i de un nodo cualquiera. Lamentablemente la distribución de las palabras del vocabulario depende mucho del pivote fijo de cada nivel o altura, entonces la fórmula no calculará exactamente el valor de la altura óptima sino que entregará aquella altura óptima para el caso ideal en que todos los pivotes fijos, cualesquiera que sean, harán que las palabras se distribuyan siempre igual o cercano a la distribución promedio. En otras palabras, la fórmula utiliza la misma distribución para cada nivel y cada nodo, siendo que en la práctica los elementos bajo cada nodo de un mismo nivel no son los mismos (y además son cercanos). La altura óptima teórica esta dada por:

$$h_k = \frac{\log(n(1-B_k))}{\log(1/B_k)}, \qquad B_k = \sum_{i \geq 0} p_i \sum_{j=i-k}^{i+k} p_j$$

donde B_k es la probabilidad de no filtrar un elemento al hacer una búsqueda de todos los elementos a distancia k de la consulta y p_i es la probabilidad de que una palabra esté a distancia i del pivote fijo.

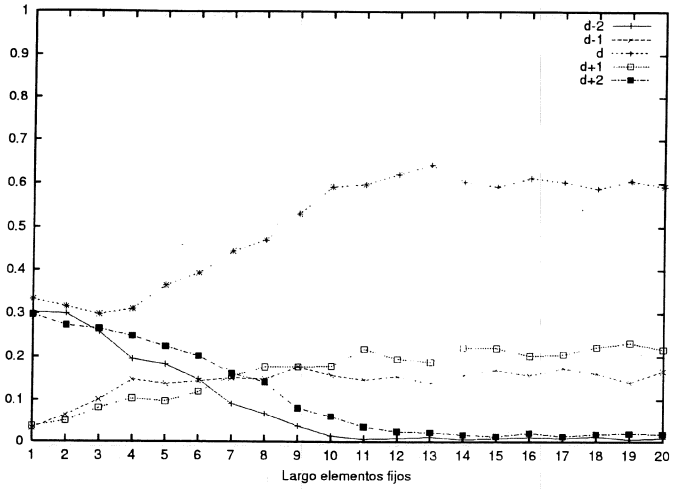

Figura 6: Distribución de la distancia de edición para 2 errores.

Aplicando esta fórmula al vocabulario de inglés de 69069 palabras, se necesita p_i con $i = 0 \ldots 21$. Para encontrar esta distribución de forma aproximada, se escogieron 300 palabras al azar del vocabulario y se compararon contra todo el vocabulario. Los resultados se muestran en la Tabla 2 (izquierda), utilizando el valor entero más cercano.

k	1	2	3
h_k	11	20	35

h	15	18	21
%	1.7	1.0	0.5

Tabla 2: Izquierda: Altura óptima teórica del árbol dependiendo del número de errores permitidos. Derecha: Porcentaje del vocabulario examinado durante la búsqueda con distintas alturas para $k = 1$ y pivotes del diccionario (**dict**).

Para el experimento de búsqueda se promedió para cada altura el numero de comparaciones para 100 búsquedas de todas las palabras a distancia k de la consulta (palabra aleatoria del vocabulario con k errores aleatorios). Se realizó el experimento para pivotes fijos del vocabulario (**dict**) y de palabras aleatorias de varias longitudes para $k = 1$ y $k = 2$. Sin embargo, en ninguno de estos casos se encontró un óptimo menor a altura 22 (ver Figura 7). Por la tanto, la altura óptima en la práctica es mayor a la predicha teóricamente. Es decir, la distribución real parece estar más sesgada. Para alturas mayores se necesita una implementación que maneje árboles en memoria secundaria, lo que es parte de un futuro estudio, pues es necesario minimizar la pérdida de espacio al usar páginas en disco.

La Tabla 2 (derecha) muestra el caso usando palabras fijas del diccionario (**dict**) y $k = 1$. En teoría, el porcentaje a revisar en el óptimo debería ser en este caso aproximadamente 0.04% y para altura 21 es aún 10 veces mayor. Esto quiere decir que la distribución de los datos afecta significativamente la altura óptima (es más del doble a lo esperado teóricamente) y que por ende

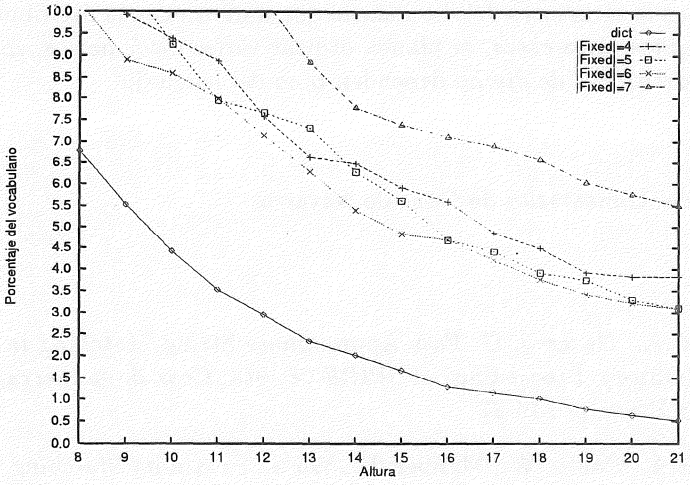

Figura 7: Número de comparaciones para búsqueda con 1 error.

no es razonable pues el espacio necesario es altísimo (más de 20 veces el tamaño original del diccionario).

Por otra parte se debe destacar que la cantidad de comparaciones realizadas para altura 20 y $k = 1$ es menor al 1% del total del diccionario (para **dict**), lo cual es una cantidad menor a la realizada por otras estructuras. En [1], para un diccionario de 500 mil palabras, usando árboles BK, FQT o FHQT de altura 15, se obtuvo un porcentaje de alrededor de entre el 2.2% y el 1.9%, en ese orden.

5 Conclusiones

La optimización del FHFQT debe permitir encontrar valores a los parámetros de manera que el funcionamiento y la eficiencia sean los mejores. Debido a que la altura óptima considerando el mínimo de comparaciones es bastante alta y el espacio ocupado por la estructura para esas alturas es bastante alto, aún cuando es razonable, es factible introducir un índice que permita optimizar simultáneamente el espacio ocupado y el número de comparaciones. En todo caso, una de las conclusiones más importantes de este trabajo, es que es mejor utilizar palabras del diccionario como pivotes que palabras aleatorias de largo fijo, aunque no es claro cual es la altura que debemos usar.

Este trabajo realiza otros aportes nuevos tales como el análisis del comportamiento de la distancia de edición, el cual entrega resultados no conocidos anteriormente y que puede aplicarse para la optimización de otras aplicaciones y/o estructuras de datos. Estos resultados también pueden usarse para heurísticas para búsqueda aproximada (por ejemplo en el caso en que no interesa el elemento más cercano, sino uno de los más cercanos) en esta estructura y en otras. Por ejemplo, sólo recorrer las ramas d y $d-1$ podría bastar para encontrar un elemento bastante cercano usando posiblemente sólo la mitad de las comparaciones.

Trabajo futuro incluye optimizar la estructura de datos para reducir el uso de espacio, incluyendo métodos de almacenamiento en memoria secundaria, y optimizar la selección de los pivotes a usar (por ejemplo, escogiendo elementos del diccionario que uniformicen la distribución de probabilidad de distancias). En todos estos casos, se planea utilizar varios diccionarios, en distintos lenguajes, para verificar si las decisiones de diseño dependen o no del lenguaje.

Agradecimientos

Agradecemos los útiles comentarios de Gonzalo Navarro.

Bibliografía

[1] Baeza-Yates, R.A., Navarro, G. Fast Approximate String Matching in a Dictionary. In B. Ribeiro-Neto (editor), Proceedings of SPIRE'98, Sta. Cruz de la Sierra, Bolivia, September 1998. IEEE CS Press, pp. 14-22

[2] Baeza-Yates. R.A., Cunto, W., Manber, U., Wu S., Proximity Matching Using Fixed-Queries Trees. 5th Symp. on Combinatorial Pattern Matching, Springer Verlag LNCS 807, Asilomar, CA, June 1994, pp. 198-212.

[3] Baeza-Yates, R.A. Searching: An Algorithmic Tour. In Allen Kent and James G. Williams, editors, Encyclopedia of Computer Science and Technology, volume 37, Marcel Dekker, Inc., Allen Kent and James G. William (Editors), pp. 331-359, 1997.

[4] Burkhard, W., Keller, R., Some approaches to best-match file searching. *Communications of the ACM* 16(4):230-236, 1973.

[5] Chávez, E., Navarro, G., Baeza-Yates, R. and Marroquín, J. Searching in Metric Spaces (Survey), Dept. of Computer Science, Univ. of Chile, TR/DCC-99-3, 1999.

[6] Gaede, V. and Günter, O. Multidimensional Access Methods, *ACM Computing Surveys* 30(2):170-231, 1998.

[7] Navarro, Gonzalo. Approximate Text Searching. PhD thesis, Dept. of Computer Science, Universidad de Chile, TR/DCC-98-14, 1998.