Árbol Binario de Búsqueda Óptimo.

November 4, 2021

Supongamos que estamos diseñando un programa para traducir texto del inglés al francés. Para cada aparición de cada palabra en inglés en el texto, necesitamos buscar su equivalente en francés. Podríamos realizar estas operaciones de búsqueda construyendo un árbol de búsqueda binario con n palabras en inglés como claves y sus equivalentes en francés como datos.

Dado que buscaremos en el árbol cada palabra del texto, queremos que el tiempo total dedicado a la búsqueda sea lo más bajo posible. Podríamos asegurar un tiempo de búsqueda $O(\log n)$ por palabra usando un árbol de búsqueda binario balanceado. Sin embargo, las palabras aparecen con diferentes frecuencias y una palabra de uso frecuente como *el* puede aparecer lejos de la raíz, mientras que una palabra de uso poco frecuente como *matacán* aparece cerca de la raíz. Esta organización ralentizaría la traducción, ya que el número de nodos visitados al buscar una clave en un árbol de búsqueda binaria es igual a uno más la profundidad del nodo que contiene la clave.

Queremos que las palabras que aparecen con frecuencia en el texto se coloquen más cerca de la raíz. Además, es posible que algunas palabras del texto no tengan traducción al francés y no aparezcan en el árbol de búsqueda binaria en absoluto.

¿Cómo organizamos un árbol de búsqueda binario para minimizar el número de nodos visitados en todas las búsquedas, dado que sabemos con qué frecuencia aparece cada palabra?

Lo que necesitamos se conoce como un árbol de búsqueda binario óptimo. Formalmente, se nos da una secuencia $K = \langle k_1, k_2, \ldots, k_n \rangle$ con n claves distintas ordenadas (de modo que $k_1 < k_2 < \cdots < k_n$), y deseamos construir un árbol de búsqueda binario a partir de estas claves. Para cada clave k_i, tenemos una probabilidad p_i de que la búsqueda sea por k_i. Algunas búsquedas pueden ser de valores que no están en K, por lo que también tenemos $n + 1$ "claves falsas" $d_0, d_1, d_2, \ldots, d_n$ que representan valores que no están en K. En particular, d_0 representa todos los valores menores que k_1, d_n representa todos los valores mayores que k_n, y para $i = 1, 2, \ldots, n-1$, la clave falsa d_i representa todos los valores entre k_i y k_{i+1}. Para cada clave ficticia d_i, tenemos una probabilidad q_i de que una búsqueda corresponda a d_i.