RECUPERACIÓN DE INFORMACIÓN EN DICCIONARIOS

Octavio Santana Suarez
Margarita Díaz Roca
Antonio Ballester Monzón
Francisco J. Santana Pérez

0) Introducción

La localización de palabras en texto libre constituye una parte fundamental de un problema práctico de gran importancia: la organización y utilización de información procedente de fuentes heterogéneas.

Existen dos aproximaciones a este problema. En la primera, se efectúa la búsqueda directamente sobre el documento sin ningún tipo de preparación previa; si se trata con actas de longitud mediana o las consultas son algo complejas, esta manera de proceder puede resultar muy costosa en cuanto a tiempo de respuesta. Desde la segunda perspectiva, se someten anteriormente los escritos a un tratamiento; la intención es generar un índice que haga viable los accesos posteriores al ejemplar cuando sea necesario.

Siguiendo este último enfoque, se utiliza como índice la estructura DITE [SD87,89,DS90], construida a partir del conjunto de las palabras diferentes que se obtienen del documento al descartar las cadenas consideradas *vacías*. El corpus que se va a indizar para llevar a cabo las localizaciones textuales es un Diccionario de Medicina [JV87].

De las 993.753 palabras de que consta el Diccionario de Medicina 424.689 se consideran vacías _589 es la cardinalidad de este conjunto_ y de las 569.064 restantes, solamente 53.334 son distintas. En la figura 1, se muestra la distribución de frecuencias por longitudes de este conjunto de palabras diferentes.

Se modifican los nodos-SIT[2] añadiendo un puntero a cada sinónimo-DIT, de forma que señale a una lista de posiciones en las que aparece esa palabra en el diccionario, como se observa en la figura 2.

Sobre la estructura DITE así construida se permiten los siguientes tipos de búsquedas:

- Exacta
- Más similares
- Con operadores booleanos: *o*, *y*, *y-no*.
- Máscaras
- Truncamientos: a la derecha, a la izquierda, a ambos lados.
- Cercanía
- Antecedencia
- Párrafos
- Sentencias
- Frases
- Compleja

[2] Véase el apéndice.

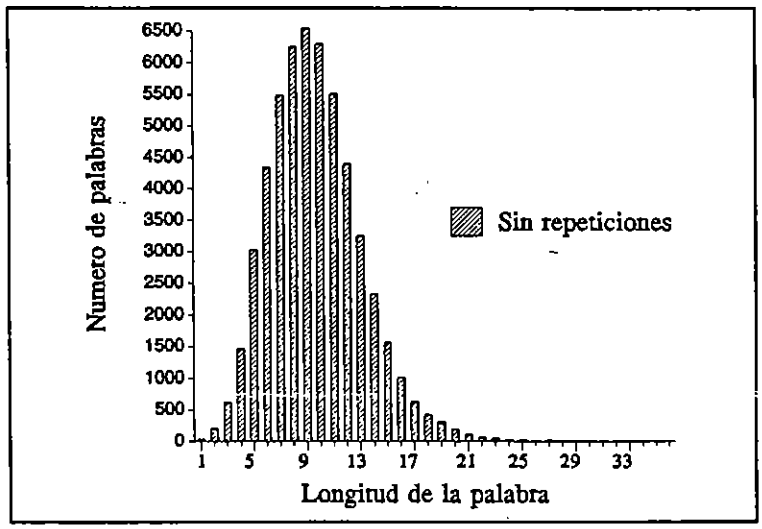

Figura 1

Un Diccionario está dividido de forma natural en *bloques de texto* donde cada uno se compone de una palabra y su significado. La lista de posiciones asociada a cada sinónimo-DIT indica los distintos bloques en los que está presente. Si una palabra se repite dentro de un mismo bloque, su lista de posiciones hace referencia una sola vez a éste bloque.

Las búsquedas de: **cercanía, antecedencia, párrafos, sentencias y frases,** así como las complejas que incluyan a alguna de las citadas, requieren el tratamiento de cada bloque candidato al conjunto respuesta con el fin de verificar las restricciones especificadas en la petición en cuanto a la situación de las palabras.

Cuando se solicita una búsqueda del tipo: **más similares, máscaras** o **truncamientos,** que tienen respuesta múltiple se muestran en la pantalla las palabras que satisfacen la petición. Se permite que el usuario seleccione un subconjunto de ellas para ojear los bloques asociados.**1) Búsqueda Exacta**

La búsqueda más elemental consiste en determinar todas las localizaciones de una palabra en el documento. La petición se expresa simplemente indicando cuál es la que se solicita. Por ejemplo: *sida*, aparece en ocho bloques.

2) Búsqueda de las palabras más similares

Proporcionada una métrica en el espacio de las palabras, DD [LE66,UK83,85,LV85,86], las más similares [GG86,SD87,DS90] a una cadena dada (esta puede pertenecer o no al diccionario) son todas las que se encuentran en el diccionario a distancia mínima de ella.

El formato de la petición es: *+cadena*. Por ejemplo, al solicitar *+rida* se obtienen: *ria, vida, rica, risa, sida, oida, brida.*

3) Búsquedas con operadores booleanos

Cuando se quieren realizar búsquedas a partir de varias palabras se pueden utilizar como conectores los operadores lógicos: *o, y, y-no.*

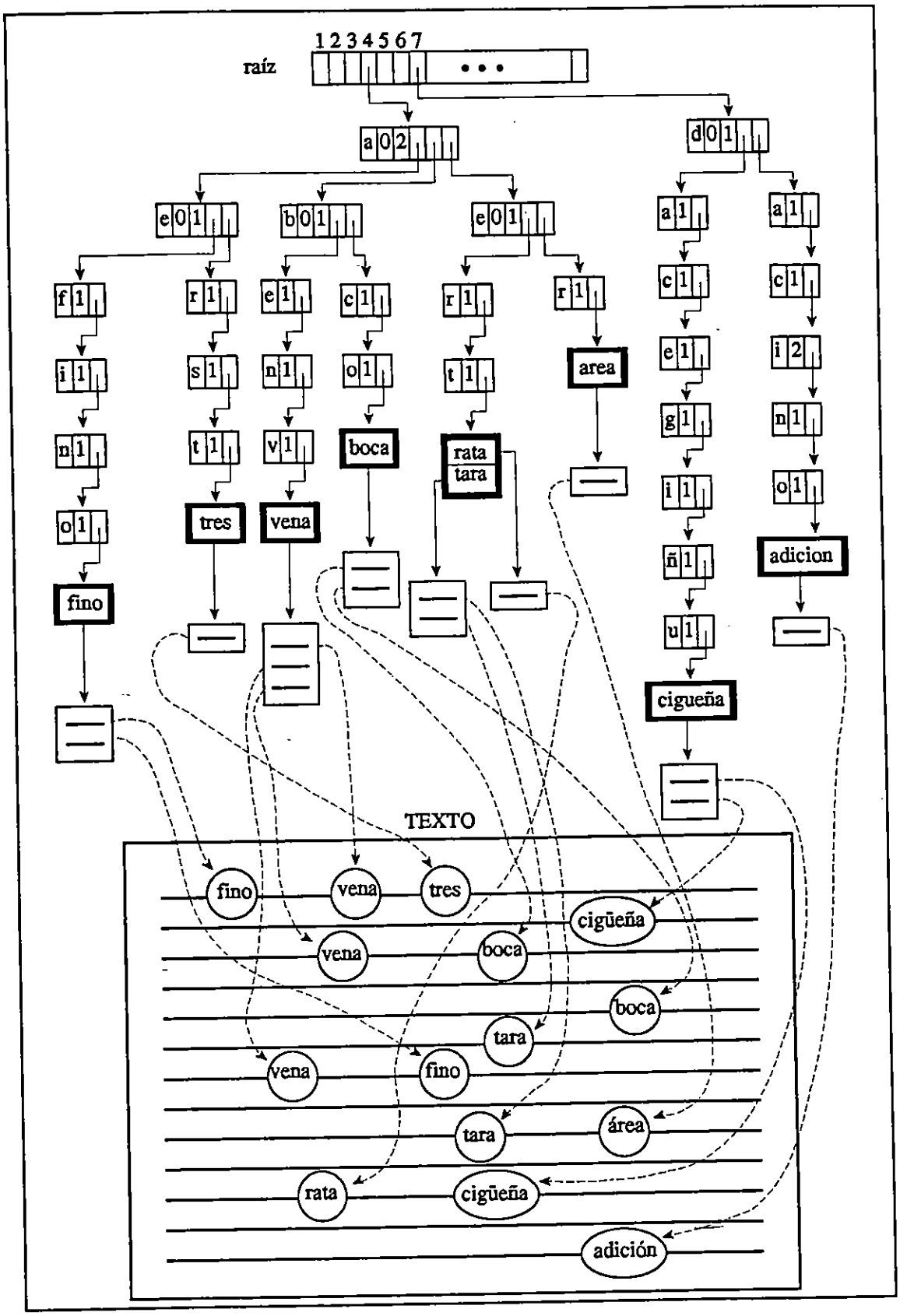
raíz
1 2 3 4 5 6 7
a 0 2
d 0 1
e 0 1
b 0 1
e 0 1
a 1
a 1
f 1
r 1
e 1
c 1
r 1
r 1
c 1
c 1
i 1
s 1
n 1
o 1
t 1
area
e 1
i 2
n 1
n 1
t 1
v 1
boca
rata
tara
g 1
o 1
o 1
tres
vena
i 1
adicion
fino
ñ 1
u 1
cigueña
TEXTO
fino
vena
tres
cigüeña
vena
boca
boca
tara
vena
fino
tara
área
rata
cigüeña
adición

Figura 2

3.1) Búsqueda disyuntiva

Una búsqueda disyuntiva proporciona todos los bloques en los que aparece al menos una de las palabras especificadas en la petición. Se solicita de la siguiente forma: *cadena1 o cadena2*. Por ejemplo: *cáncer o tumor.*

3.2) Búsqueda conjuntiva

La respuesta de una petición conjuntiva está formada por aquellos bloques que contienen a todas las palabras especificadas. Esta petición se formula como: *cadena1 y cadena2*. Por ejemplo: *cáncer y tumor.*

3.3) Búsqueda con operador *y-no*

En una búsqueda de una petición de la forma *cadena1* ***y-no*** *cadena2* se obtienen aquellos bloques en los que se encuentra la palabra situada a la izquierda del operador y que a su vez no contienen a la especificada a la derecha. Por ejemplo: *tos* ***y-no*** *fiebre.*

4) Búsqueda con máscaras

Si se desea hacer una búsqueda de una palabra en la que se desconocen caracteres en determinadas posiciones, se puede formular una petición en la que se sustituyan los caracteres desconocidos por *comodines* (*).

En una búsqueda con máscaras, se recuperan todos los bloques en los que se encuentra una palabra de longitud igual a la especificada y que difiera solamente en los caracteres indicados por los asteriscos. Por ejemplo para *t*m*r* se recuperan los bloques en los que aparecen palabras como *tumor, temor, tomar, timor.*

5) Búsquedas con truncamientos

Las búsquedas con truncamientos a la derecha, a la izquierda y a ambos lados permiten recuperar bloques que contienen palabras que empiezan, terminan o presentan alguna coincidencia central con la petición.

Cada tipo de truncamiento se expresa de la siguiente forma:

a la derecha:	*cadena!*
a la izquierda:	*!cadena*
a ambos lados:	*!cadena!*

Por ejemplo:

a la derecha: *tos!*,	se obtienen: *tos, tose, tosa, toser, tosco, tosca, toscos, toscas, tossach, tostada.*
a la izquierda: *!tipo*,	se recuperan: *subtipo, ojotipo, idiotipo, optotipo, fenotipo, genotipo, serotipo, fagotipo, prototipo, cariotipo, inmunotipo, somatotipo.*
a ambos lados: *!cubo!*,	se localizan: *cubo, cuboide, cuboides, cuboideo, cuboidal, cuboidea, cuboideum, cuboideos, cuboideas, calcaneocuboidea, calcaneocuboideo, calcaneocuboideas.*

6) Búsquedas de cercanía y antecedencia

Cercanía

La búsqueda de cercanía se caracteriza por limitar el número máximo de palabras que pueden existir entre las dos especificadas en la petición. Las peticiones son del tipo: *cadena1* ***c/n*** *cadena2*. Por ejemplo: *fiebre* ***c/5*** *aguda*, donde si se encuentra *fiebre*, *aguda* ha de estar como mucho en la quinta palabra contando a partir de la siguiente a *fiebre*, o si se localiza primero *aguda*, *fiebre* no ha de estar más allá de la quinta palabra después.

Antecedencia

La antecedencia entre palabras presenta una restricción más con respecto a la búsqueda anterior ya que establece el orden de aparición de las mismas. Las peticiones son del tipo: *cadena1* ***a/n*** *cadena2*. Por ejemplo: *nervio* ***a/3*** *facial*, lo que indica que primero se ha de encontrar *nervio* y como mucho la tercera palabra, después ha de ser *facial*.

7) Búsquedas en párrafos y sentencias

Párrafos

La búsqueda en párrafos es una alternativa a la de cercanía entre palabras en la que se requiere que las palabras especificadas se encuentren en el mismo párrafo. Se solicita de la siguiente forma: *cadena1* ***p/*** *cadena2*. Por ejemplo: *pies* ***p/*** *manos*.

Sentencias

Es análoga a la búsqueda en párrafos pero restringida a una sentencia. La petición se expresa: *cadena1* ***s/*** *cadena2*. Por ejemplo: *pies* ***s/*** *manos*.

8) Búsqueda de frases

La búsqueda de frases se puede ver como una búsqueda de cercanía entre palabras en las que la máxima separación entre las especificadas en la petición es uno, es decir, *c/1*. La frase se denota entre comillas. Por ejemplo: *«de pies y manos»*.

9) Búsqueda compleja

Cualquier combinación de los anteriores tipos de búsqueda en una expresión utilizando los conectores lógicos (*y*, *o*, *y-no*) es a lo que se denomina una petición compleja. Por ejemplo: *s**a* ***y*** *(can! o tumor)* ***y-no*** *sana*.

Cuando interviene algún tipo de búsqueda con respuesta múltiple se realiza una fusión de las listas de posiciones asociadas que haya seleccionado el usuario.

El orden de prioridad de los conectores es de izquierda a derecha. Si se desea alterar esta prioridad, se deben utilizar paréntesis y en el caso de que existan paréntesis anidados, los más interiores son los que se resolverán en primer lugar.

La obtención de la respuesta de una búsqueda compleja se realiza de forma recursiva para cada par de abre-paréntesis y cierra-paréntesis. A partir del abre-paréntesis, se actúa de izquierda a derecha ejecutando cada búsqueda no-booleana y resolviendo los operadores lógicos una vez que se alcanza el cierra-paréntesis. Por defecto, se considera que la petición compleja está encerrada entre paréntesis.

Búsqueda en la que intervienen peticiones anteriores

Tras echar una ojeada a las peticiones previas, es posible modificar una ya existente para confeccionar otra más compleja. El formato *@n* indica que la n-ésima consulta realizada forma parte de la nueva petición. Supóngase que se desea añadir a la 5ª búsqueda, cuya expresión de contenido es: *cáncer y tumor*, un nuevo término: *y-no pulmón*, la nueva petición se escribe de la siguiente forma: *@5 y-no pulmón*.

También se pueden hacer combinaciones de peticiones previas sin añadir ningún término de búsqueda nuevo, utilizando los operadores booleanos. Por ejemplo: *@2 y (@4 o @6)*, combina los resultados de las consultas 2, 4 y 6.

9.1) Analizador sintáctico de la petición

El analizador sintáctico cumple un doble cometido: diferenciar las componentes _cada tipo de subpetición_ y los operadores lógicos que las conectan y, a la vez, determinar la correctitud sintáctica de la petición.

Se consideran erróneas las siguientes situaciones:

- Si alguna petición no-booleana presenta un formato incorrecto, bien porque se especifica una búsqueda no permitida, por ejemplo: *t*m!*, o bien porque le falta alguna condición, por ejemplo: si se detecta la presencia de *'/'* precedida de *a* o *c* y no le sigue un valor entero o se especifica una frase y falta cerrar las comillas, etc.

- Si se utilizan conectores no permitidos.

- Si después de un operador lógico no existe una búsqueda no-booleana o un abre-paréntesis.

- Si se especifica la búsqueda exacta de una palabra considerada como vacía.

- Si existe un abre-paréntesis para el que no se encuentra el correspondiente cierra-paréntesis.

En caso de que se detecte un error tal, se informará al usuario de ello y se señalará en la pantalla la posición en la que se ha encontrado dicho fallo.

Para analizar la petición _formada por una cadena de 100 caracteres de longitud máxima_ se recorre de izquierda a derecha identificando las diversas búsquedas no-booleanas, operadores lógicos y paréntesis, comprobándose que no ocurre ninguna de las situaciones indicadas anteriormente.

Una vez que se ha verificado que la petición es gramaticalmente correcta e identificado sus componentes, se pasa a la fase de ejecución de la búsqueda.

Bibliografía:

[DS90] DIAZ, M.; SANTANA, O.; RODRIGUEZ, J.C.:»Búsqueda de las Cadenas Mas Similares: Esquema Decreciente con Radio de Búsqueda Ascendente, Esquema Creciente». XVI Conferencia Latinoamericana de Informática (1990), Vol I, 90/97.

[GG86] GALIL, Z.; GIANCARLO, R.:»Improved String Matching with k Mismatches», SIGACT News, 17, 4, 52/54, (1986).

[JV87] JOVEN, J.; VILLABONA, C.; JULIA, G.; GONZALEZ-HUIX, F.:»Diccionario de Medicina». Tercera edición (1987). Editorial MARIN, S.A.

[LE66] LEVENSHTEIN, V. I.: «Binary Codes Capable of Correcting, Insertions and Reversals». Soviet Phys. Dokl. 10 (1966), 707/710.

[LV85a] LANDAU, G. M.; VISHKIN, U.: «Efficient String Matching in the Presence of Errors». Proc. 26th IEEE FOSC, 126/136, (1985).

[LV85b] LANDAU, G. M.; VISHKIN, U.: «Efficient String Matching with k Differences». TR_36/85, Departament of Computer Science, Tel Aviv Univ., Submitted for Journal Publication, (1985).

[LV86a] LANDAU, G. M.; VISHKIN, U.: «Efficient String Matching with k Mismatches». Theoretical Computer Science, 43, 239/249, (1986).

[LV86b] LANDAU, G. M.; VISHKIN, U.; NUSSINOV, R.: «An Efficient String Matching Algorithm with k Differences for Nucleotide and Amino Acid Sequences». Nucleic Acid Research 14 (1), 31/46, (1986).

[SD87] SANTANA, O.; DIAZ, M.; MAYOR, O.; REYES, J.: «Esquemas y Estructura para la Búsqueda de las Palabras Más Similares a una Dada». XIII Conferencia Latinoamericana de Informatica (1987), Vol. II, 1169/1189.

[SD89] SANTANA, O.; DIAZ, M.; DUQUE, J.D.; RODRIGUEZ, G.:»Estructuración de las Componentes de la Distancia Invariante Trasposicional, DIT, con Compartición de la Zona No_Discriminante en la Búsqueda de las Cadenas Mas Similares». XV Conferencia Latinoamericana de Informática (1989), Vol. II, 335/341.

[SP89] SANTANA. O.; PEREZ, J.; RODRIGUEZ, J. C.: «Increasing Radius Search Schemes for the Most Similar Strings on the Burkhard_Keller Tree». International Workshop on Computer Aided Systems Theory, EUROCAST'89, (1989).

[SR90] SANTANA, O.; RODRIGUEZ, J. C.; DIAZ, M.: «Búsqueda de las Cadenas Más Similares: Incidencia de la Subsecuencia Común Más Larga en los Esquemas Decreciente y Creciente». XVI Conferencia Latinoamericana de Informática (1990), Vol I, 98/104.

[UK83] UKKONEN,E.:»On Approximate String Matching». Proc. Int. Conf. Found. Comp. Theor., Lecture Notes in Computer Science 158, Springer-Verlag, (1983), 487/495.

[UK85] UKKONEN, E.:»Finding Approximate Pattern in Strings». J. of Algorithms, 6, 132/137, (1985).

APÉNDICE

DD -. Se llama **distancia direccional** de edición, DD(X,Y), entre las cadenas *X* e *Y*, al mínimo costo de todas las secuencias de edición que transformen *X* en *Y*.

DIT -. La **distancia invariante trasposicional** entre dos cadenas *X* e *Y* se calcula como la suma acumulada del valor absoluto de las diferencias de frecuencias de los caracteres de *X* e *Y* incrementada por la diferencia de longitudes.

DITE -. El árbol tiene un nodo raíz que discrimina por longitudes de cadenas. El encadenamiento asociado a cada longitud señala a una parte-árbol constituida exclusivamente por nodos discriminantes que contienen un carácter discriminante, _, y varios enlaces e_i, donde cada e_i es un enlace que apunta a un subconjunto de cadenas D^i, tal que la frecuencia del carácter _ en cada una de las cadenas de D^i, es *i*. Cuando una rama de la parte-árbol ya no discrimina se pasa a la parte-cadena que son listas encadenadas formadas por los restantes pares *_f*, donde _ es un carácter, *f* es la frecuencia con que aparece _ y el enlace *e* de la lista direcciona a un nodo del mismo tipo o a un nodo, **nodo-SIT**, de **sinónimos-DIT**, es decir, cadenas formadas por los mismos caracteres.

Debido a que se considera que la no aparición de la tilde y la diéresis supone un costo cero de edición a efectos de evaluación de DIT y DD, es por lo que no se han tenido en cuenta estas puntuaciones en la construcción del índice. Ello no es óbice para localizar en el texto las palabras correctamente escritas.