LA ESTRUCTURA DE BURKHARD-KELLER EN LA BUSQUEDA DE LAS CADENAS MAS SIMILARES A UNA DADA

AUTORES: **O. SANTANA, J. PEREZ, G. LOPEZ, G. RODRIGUEZ.**
Dpto. de Informática y Sistemas; Universidad Politécnica de Canarias.
Aptdo. 550, Las Palmas de Gran Canaria, España.

Se aborda el siguiente problema: búsqueda de las cadenas más similares a una dada. El concepto de similitud es en el sentido de distancia direccional, **DD**, de Levenshtein [LE66].

Se incorpora una distancia invariante a trasposiciones [SD87], **DIT**, cuyo costo computacional es inferior al de **DD**, usándose como filtro adaptivo.

Se realiza un estudio comparativo del esquema de búsqueda **DITE+DD** [SD87] con los siguientes esquemas de búsqueda de las cadenas más similares en el árbol de Burkhard-Keller [BK73] y [NK82]:

a) El árbol se construye usando **DD**.

b) El árbol se construye usando **DIT**. Se implanta el filtro adaptivo **DIT+DD**.

c) Utilizar **DIT** para hacer una búsqueda previa a fin de limitar el radio de búsqueda en **(b)**.

d) Utilizar **DIT** conjuntamente con **DD** para calcular previamente el radio final de búsqueda en **(b)**.

e) Combinar los esquemas **(c)** y **(d)** para obtener primero una acotación del radio de búsqueda sólo con **DIT**, luego hacer un cálculo rápido del radio final y por último entrar en la búsqueda **(b)**.

Se analiza la bondad de los distintos esquemas explicando el comportamiento de los diferentes tiempos promedios experimentales.

LA ESTRUCTURA DE BURKHARD-KELLER EN LA BUSQUEDA DE LAS CADENAS MAS SIMILARES A UNA DADA

O. SANTANA, J. PEREZ, G. LOPEZ, G. RODRIGUEZ.
Dpto. de Informática y Sistemas; Universidad Politécnica de Canarias.
Aptdo. 550, Las Palmas de Gran Canaria, España.

RESUMEN:

En este trabajo se aborda el problema de la búsqueda de las cadenas más similares, en el sentido de distancia direccional, **DD**, de Levenshtein [LE66], a una dada. Se incorpora una distancia invariante frente a trasposiciones, **DIT**, que tiene un costo computacional inferior a la **DD** y que se usa como filtro adaptivo para realizar las búsquedas. Se estudian, comparativamente, los esquemas de búsqueda **DITE+DD** [SD87] y Burkhard-Keller [BK73] y [NK82], construidos con **DD** y con **DIT**; en los últimos se llevan a cabo las búsquedas en una, dos y tres etapas.

0.- INTRODUCCION.

El problema que se tratará es el siguiente. Dado un diccionario de cadenas, una distancia en el espacio de las cadenas y dada una nueva cadena de búsqueda (que puede encontrarse o no en el diccionario) encontrar todos las cadenas del diccionario que se encuentran a distancia mínima de la de búsqueda.

ALBERGA [AL67] usó matrices binarias para evaluar medidas de aproximación entre dos cadenas. SZANSER [SZ73] desarrolló un proceso matemático de coincidencia elástica que era efectivo en un 95%. MORGAN [MO70] realizaba un o-exclusivo entre dos cadenas para determinar si había ocurrido un único error simple. WAGNER y FISCHER [WF74] desarrollaron un algoritmo de programación dinámica que puede determinar la distancia entre dos cadenas medida como el mínimo coste de la secuencia de operaciones de edición sustitución, inserción y extracción. UKKONEN [UK83] desarrolló una forma de cálculo alternativa de la matriz de diferencias de LEVENSHTEIN [LE66] usando sus diagonales.

En la sección 1 se establecen los tipos de errores de edición que puede sufrir una cadena, para llegar a la sección 2 a definir y calcular la distancia direccional, **DD**, (distancia de Wagner y Fischer [WF74]). En la sección 3 se estudia una alternativa para el cálculo de **DD** que es menos costosa [UK83] y [LA87]. En la sección 4 se introduce un tipo de distancia, **DIT** [SD87], invariante a las trasposiciones, cuyo valor es siempre menor o igual al de la distancia direccional, **DD**. En la sección 5 se expone la estructura y el esquema de búsqueda **DITE+DD** [SD87] a fin de poder comparar sus resultados experimentales con los de este trabajo. En la sección 6 se compara experimentalmente la influencia de la optimización del cálculo de **DD** en los esquemas de búsqueda secuencial y **DITE+DD**. En la sección 7 se introduce el árbol de búsqueda de Burkhard- Keller [BK73] y [NK82], **BK**, y se plantea el algoritmo de búsqueda de los más similares. En la sección 8 se articula el árbol **BK** con la **DD**, **BK_DD**. En la sección 9 se construye el árbol **BK** con la **DIT** y se plantea el algoritmo de búsqueda usando **DIT** como filtro adaptivo para eludir cálculos de **DD**, **BK_DIT+DD**. En la sección 10 se introduce un filtro de la búsqueda que consiste en usar solamente **DIT** para encontrar las más similares y a partir de ahí tener una acotación del radio de búsqueda

en **BK_DIT+DD**, **[FB(BK_DIT)] + [BK_DIT+DD]**. En la sección 11 se implanta un filtro de la búsqueda que utiliza **DIT** y **DD** para calcular el radio final (mínimo) de búsqueda antes de entrar en **BK_DIT+DD**, **[FB(BK_DIT+DD)1] + [BK_DIT+DD]**. En la sección 12 se hace una modificación del filtro de la búsqueda que se antepone en el esquema de la sección 11, **[FB(BK_DIT+DD)A] + [BK_DIT+DD]**. En las secciones 13 y 14 se antepone la primera fase del esquema utilizado en la sección 10 con los esquemas de las secciones 12 y 11, respectivamente, a fin de acotar el radio, usando sólo **DIT**, para luego hacer el cálculo del radio final (mínimo), usando **DIT** y **DD**, y por último entrar en **BK_DIT+DD**, **[FB(BK_DIT)] + [FB(BK_DIT+DD)X] + [BK_DIT+DD]**. En la sección 15 se presentan los estudios experimentales comparativos de la ocupación y los tiempos promedios de búsqueda, así como las conclusiones de este trabajo.

1.- DISTANCIA DIRECCIONAL.

1.1.- Errores posibles en una palabra.

Wagner considera tres tipos posibles de errores básicos que pueden darse en una cadena:

1.- Sustitución de un carácter por otro.
2.- Extracción de un carácter.
3.- Inserción de un carácter.

Se considera que todas las cadenas que se mencionan en lo sucesivo están compuestas por una secuencia de caracteres extraídos de un alfabeto $\sqrt{\alpha}$**1,...,**α**m²**.

Sea **X** una cadena, sea **X<i>** el carácter que está en la i-ésima posición de la cadena **X**; **X<i:j>** la secuencia de caracteres que va de **X<i>** a **X<j>** ambos inclusive, si **i>j** entonces **X<i:j>=**μ, la hilera nula. **|X|** denota la longitud de **X**.

1.2.- Operaciones de edición.

Una operación de edición es el par **(**θ**,**Ω**)=,/(**μ**,**μ**)** donde θ y Ω son cadenas de longitud menor o igual que uno, es decir, ó son μ ó son un único carácter. La cadena **Y** resulta de aplicar **(**θ**,**Ω**)** a la cadena **X**, que se escribe **X->Y**, si **X=**$\sigma\theta\tau$ e **Y=**$\sigma\Omega\tau$.

Sea **(**θ**,**Ω**)** una operación de sustitución si θ**=,/**μ, Ω**=,/**μ y θ**=,/**Ω, una operación de extracción si Ω**=**μ y una operación de inserción si θ**=**μ.

Sea **S** una secuencia **s1**, **s2**, ..., **sn** de operaciones de edición. Una derivación-S de **X** hasta **Y** es una secuencia de cadenas **X0**, **X1**, ..., **Xn** tal que **X=X0**, **Y=Xn** y **Xj-1->Xj** vía **sj** para todo **j=1,...,n**.

Se dice que **S** convierte **X** en **Y** si hay una derivación-S de **X** hasta **Y**.

1.3.- Función de coste.

Sea Γ una función arbitraria de coste que asigne a cada operación de edición (θ,Ω) un número real no negativo $\Gamma(\theta,\Omega)$. Se puede extender Γ a la secuencia **S** definiendo:

$$\Gamma(S) = \sum_{j=1}^{n} \Gamma(sj) \quad \text{si} \quad n \geq 1 \quad \text{y} \quad \Gamma(S)=0 \quad \text{si} \quad n=0$$

Se llama distancia direccional de edición al mínimo coste de todas las secuencias de edición que transformen **X** en **Y**. Formalmente $\sigma(X,Y)=\min\sqrt{\Gamma(S)^2}$.

2.- CALCULO DE LA DISTANCIA DIRECCIONAL.

Si **X** e **Y** son dos cadenas cualesquiera, se define **X(i)=X<1:i>**, **Y(j)=Y<1:j>** y **DD(i,j)=σ(X(i),Y(j))**.

$$DD(0,0) = 0,$$

$$DD(i,0) = \sum_{r=1}^{i} \Gamma(X<r>,\mu), \qquad DD(0,j) = \sum_{r=1}^{j} \Gamma(\mu,Y<r>)$$

Es decir, el coste de convertir μ en sí mismo es cero, el coste de reducir la cadena **X** a μ es la suma de los costes de extraer todos los caracteres de **X** y el coste de obtener **Y** a partir de μ es el la suma de los costes de añadir cada uno de los caracteres de **Y**.

Para calcular **DD(i,j)**, **i=1,...,|X|**, **j=1,...,|Y|** hay que tener en cuenta tres casos.

1.- Convertir **X(i-1)** en **Y(j-1)** y **X<i>** en **Y<j>**.
2.- Convertir **X(i-1)** en **Y(j)** y extraer **X<i>**.
3.- Convertir **X(i)** en **Y(j-1)** y añadir **Y<j>**.

De estas tres posibilidades se escoge aquella cuyo coste sea mínimo.

Por lo tanto:

$$DD(i,j)=\min\sqrt{\begin{matrix} DD(i-1,j-1)+\Gamma(X<i>,Y<j>), \\ DD(i-1,j)+\Gamma(X<i>,\mu), \\ DD(i,j-1)+\Gamma(\mu,Y<j>)^2 \end{matrix}}$$

Y la distancia direccional entre **X** e **Y** viene dada por **DD(|X|,|Y|)**.

A lo largo de este trabajo se supondrá que el coste de cualquier operación de edición (inserción, sustitución o extracción) es uno.

3.- UNA ALTERNATIVA AL CALCULO DE LA DISTANCIA DIRECCIONAL.

Aquí se desarrolla la forma de cálculo de la distancia direccional basada en una idea de Ukkonen [UK83] y Landau [LA87].

Este cálculo se hará utilizando las diagonales de la matriz **DD** de distancia direccional de Wagner definida en el apartado anterior. Una diagonal **d** de la matriz consiste en todos los **DD(i,j)** tales que **i-j=d**.

Dada una distancia **e** y una diagonal **d**, se define:

$$\mathbf{L(d,e) = max\ \sqrt{i}\ /\ DD(i,j)=e\ con\ i-j=d^2}$$

Esto implica que:

e = DD(L(d,e),L(d,e)-d) y **X<L(d,e)+1>=,/ Y<L(d,e)-d+1>**

Además si $|\mathbf{X}| \leq |\mathbf{Y}|$, **df = |X|-|Y|** y $\mathbf{L(df,e)} \geq |\mathbf{X}|$ entonces **e = DD(|X|,|Y|)**.

A continuación se estudia como se puede calcular, dados **e** y **d**, el valor de **L(d,e)**.

Supóngase que para todo **x<e** y para todo **y** ya está calculado **L(y,x)**.

Si L(d,e)=i, (o sea, DD(i,j)=e y i-j=d) es porque ocurre alguna de las siguientes posibilidades:

(a) DD(i-1,j-1) = e-1 y X<i> =,/ Y<j>
ó DD(i,j-1) = e-1
ó DD(i-1,j) = e-1.

(b) DD(i-1,j-1) = e y X<i> = Y<j>.

Esto implica que se puede empezar en **DD(i,j)** y seguir los predecesores en la diagonal **d** (**d=i-j**) mediante la posibilidad **(b)** mientras no ocurra la posibilidad **(a)**.

El siguiente esquema invierte esta descripción para calcular los **L(d,e)**. Será necesario conocer **L(d-1,e-1)**, **L(d,e-1)** y **L(d+1,e-1)** para inicializar una variable, **fila**, que se irá incrementando hasta alcanzar el valor de **L(d,e)** de la siguiente forma:

Sea **fila = max $\sqrt{}$L(d-1,e-1)+1, L(d,e-1)+1, L(d+1,e-1)²**. Mientras sea **X<fila+1>=Y<fila-d+1>** se incrementa **fila**, cuando **fila** no pueda seguir siendo incrementada será **L(d,e)=fila**.

El cálculo de los valores de **L(d,e)** se hará de forma que la fila **df** sea la que primero se calcule, dentro de lo posible, ya que la condición para calcular **DD** es que un valor de **L(df,e)** supere la longitud de la cadena más corta **|X|**.

Habrá que inicializar los elementos de la matriz **L** para poder realizar los cálculos necesarios.

El siguiente algoritmo resume como se llevan a cabo los cálculos que hemos descrito:

INICIALIZACION:
Supóngase que $|X| \leq |Y|$

```
        L(0,-1)=-1,
        para d = 1 hasta |X| hacer
             L(d,d-2)=-1; L(d,d-1)=d-1
        fin para
        para d = 1 hasta |Y| hacer
             L(-d,d-2)=-1; L(-d,d-1)=-1
        fin para
        df=|X|-|Y|

     procedimiento
        para k = 0 hasta |X|/2 hacer
             para td = -1 hasta 0 hacer
                  e=k-df+td
                  para d = df-k hasta df-1 hacer
                       e=e+1
                       fila=max √L(d-1,e-1)+1, L(d,e-1)+1, L(d+1,e-1)²
                       mientras X<fila+1> = Y<fila-d+1> hacer
                                fila=fila+1
                       fin mientras
                       L(d,e)=fila
                  fin para
                  e=k+td
                  para d = k hasta df paso = -1 hacer
                       e=e+1
                       fila=max √L(d-1,e-1)+1, L(d,e-1)+1, L(d+1,e-1)²
                       mientras X<fila+1> = Y<fila-d+1> hacer
                                fila=fila+1
                       fin mientras
                       L(d,e)=fila
                  fin para
                  si L(df,e) ≥ |X| entonces
                     DD(|X|,|Y|)=e; retornar
                  fin si
             fin para
        fin para
     retornar
```

La principal aportación de esta optimización en el cálculo de **DD** es que se evitan calcular todos los elementos de la matriz de distancias descrita en la sección anterior, con el consiguiente ahorro de tiempo de cálculo.

4.- DISTANCIA INVARIANTE TRASPOSICIONAL.

4.1.- Definición.

Sean **X** e **Y** dos cadenas, se definen **xαi** e **yαi** como el número de veces que **αi** esté contenido en **X** y en **Y** respectivamente. Se observa que:

$$|X| = \sum_{i=1}^{m} x\alpha i$$

Donde **m** es el número de caracteres que componen el alfabeto.

Se define la distancia invariante trasposicional **DIT** como:

$$DIT(X,Y) = \frac{1}{2}\left[\sum_{i=1}^{m} abs(x\alpha i - y\alpha i) + abs(|X|-|Y|)\right]$$

4.2.- <u>Comportamiento de DIT respecto a DD</u>.

Obsérvese como se comporta la distancia **DIT** ante sustituciones, inserciones y extracciones.

Sea **X** una cadena que se convierte en **Y** mediante Φ1 inserciones, Φ2 extracciones, Φ3 sustituciones. Sea **C** el subconjunto de los caracteres de **X** e **Y** que no han sido afectados por ninguna operación de edición, **C1** el subconjunto de caracteres afectados por las inserciones, **C2** el subconjunto de caracteres afectados por extracciones y **C3** el subconjunto de caracteres afectados por sustituciones. Supóngase por simplicidad que:

$$C1 \cap C2 = C1 \cap C3 = C2 \cap C3 = \phi$$

$$\sum_{\alpha h \varepsilon C} abs(x\alpha h - y\alpha h) = 0 \qquad \sum_{\alpha i \varepsilon C1} abs(x\alpha i - y\alpha i) = \Phi 1$$

$$\sum_{\alpha j \varepsilon C2} abs(x\alpha j - y\alpha j) = \Phi 2 \qquad \sum_{\alpha k \varepsilon C3} abs(x\alpha k - y\alpha k) = 2*\Phi 3$$

$$abs(|X|-|Y|) = abs(\Phi 1 - \Phi 2) \leq \Phi 1 + \Phi 2$$

$$2*DIT = \sum_{\alpha h \varepsilon C} abs(x\alpha h - y\alpha h) + \sum_{\alpha i \varepsilon C1} abs(x\alpha i - y\alpha i) +$$

$$+ \sum_{\alpha j \varepsilon C2} abs(x\alpha j - y\alpha j) + \sum_{\alpha k \varepsilon C3} abs(x\alpha k - y\alpha k) +$$

$$+ abs(|X|-|Y|) \leq 0 + \Phi 1 + \Phi 2 + 2*\Phi 3 + \Phi 1 + \Phi 2 = 2*(\Phi 1 + \Phi 2 + \Phi 3)$$

Y por tanto **DIT ≤ Φ1+Φ2+Φ3**.

En este caso **DD(X,Y)=Φ1+Φ2+Φ3** por lo que:

DD(X,Y) ≥ DIT(X,Y).

La **DIT** mayor se obtiene al imponer la condición de disyunción. Como en el caso, ya estudiado, de múltiples sustituciones, si la disyunción entre los conjuntos **Ci** no se cumple, puede suceder que la interacción entre errores de distinto tipo que afecten al mismo carácter haga que estos influyan en la **DIT** en menor medida que la suma de cada uno de ellos por separado, por lo tanto:

$$2*DIT = \sum_{i=1}^{m} abs(x\alpha i - y\alpha i) + abs(|X|-|Y|) \leq$$

$$i=1$$

$$\leq \Phi1+\Phi2+2*\Phi3+abs(\Phi1-\Phi2) \leq \Phi1+\Phi2+2*\Phi3+\Phi1+\Phi2 = 2*\Phi1+2*\Phi2+2*\Phi3$$

En este caso **DD(X,Y)=Φ1+Φ2+Φ3** por lo que, en general:

DD(X,Y) ≥ **DIT(X,Y)**.

5.- ESTRUCTURA Y ESQUEMA DE BUSQUEDA DITE+DD.

5.1.- Estructura.

El diccionario se encuentra estructurado como un árbol [SD87] en el cual se estructuran las componentes que intervienen en el cálculo de **DIT**. Tiene un nodo-raíz que discrimina por longitudes de cadenas. A partir de ahí tiene una parte-árbol en la que cada nodo contiene un carácter **αi** y un conjunto de enlaces **Ek** que señalan al subdiccionario donde se encuentran las cadenas en las que la frecuencia de aparición de **αi** es **k**. Cuando una rama de la parte-árbol ya no discrimina se pasa a la parte-cadena que son listas encadenadas formadas por nodos donde se encuentran los caracteres no utilizados en la parte-árbol, con su frecuencia de aparición. Pendiendo de la parte-cadena se encuentra la cadena-SIT que es una lista de sinónimos **DIT**.

5.2.- Esquema de búsqueda.

El esquema de búsqueda presenta cuatro recorridos perfectamente diferenciados: por el nodo-raíz, por la parte-árbol, por la parte-cadena y por la cadena-SIT.

En el recorrido del nodo-raíz se empieza por la posición correspondiente a la longitud de la cadena de búsqueda y se van tomando alternativas por proximidad de longitud mientras la diferencia de longitud no supere el radio de búsqueda **DDM** (la **DD** más pequeña encontrada). Al mismo tiempo se inicializa el valor de **DIT** a la diferencia de longitudes.

En el recorrido por la parte-árbol se van calculando las componentes de **DIT** a la vez que se recorre la estructura empezando por el ramal del nodo actual correspondiente a la frecuencia de aparición del carácter en la cadena de búsqueda y continuando por las frecuencias más próximas. Si la **DIT** en ese punto supera **DDM** ya no se exploran más alternativas.

El recorrido de la parte-cadena es secuencial. Si en algún punto la **DIT** supera **DDM** se suspende el recorrido y se vuelve al último nodo visitado en la parte-árbol.

En el caso de que **DIT** no supere **DDM** se pasa a calcular las **DD** de la cadena de búsqueda con las cadenas que se encuentran en la cadena-SIT y se almacenan las palabras a distancia mínima ó se ajusta **DDM** según proceda. Nótese que la **DD** ha de ser evaluada para todas las cadenas de la cadena-SIT ya que todas tienen el mismo valor de **DIT** y éste es inferior a la **DD** de cualquiera de ellas.

6.- INFLUENCIA DE LA OPTIMIZACION DEL CALCULO DE DD.

La figura 1 representa la mejora relativa en el tiempo promedio de búsqueda a diferentes distorsiones en los esquemas de búsqueda secuencial, **DDS**, y **DITE+DD**, como consecuencia de la optimización en el cálculo de **DD**.

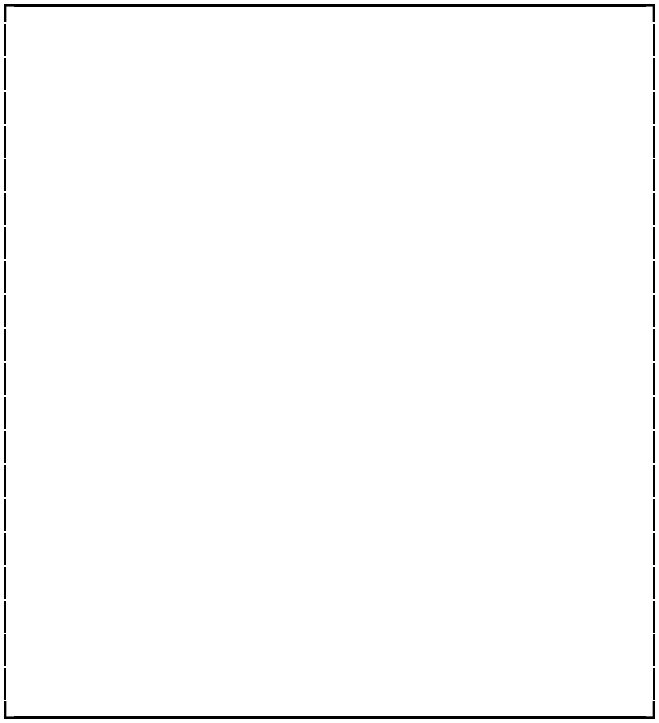


Figura 1

Ambas mejoras decaen al aumentar la distorsión, aunque en el esquema **DITE+DD** lo hace más lentamente y siendo siempre superior a la del **DDS**, debido a que sólo se calculan **DD** entre cadenas que superen el filtro **DIT**, lo que aumenta la tendencia a calcular **DD** entre cadenas más parecidas, respecto al esquema **DDS** en el que se computan **DD** con todas las cadenas del diccionario.

En adelante se evaluará **DD** siguiendo esta optimización en su cálculo.

7.- EL ARBOL DE BUSQUEDA DE BURKHARD-KELLER.

7.1.- Distancia y estructura.

Sea **S** un conjunto de **n** puntos pertenecientes a un espacio multidimensional, en el cual se tiene definida una distancia, **D**, que es una métrica.

La estructura de **Burkhard-Keller** [BK73] y [NK82], en adelante **BK**, se construye en función de la distancia y con el objeto de buscar cuales son los puntos de **S** más próximos, en términos de dicha distancia, a uno dado (que puede ó no pertenecer a **S**).

Los puntos de **S** serán almacenados en un árbol **T**, tal que en el nivel de la raíz se encuentra un punto **X°** de forma que del ramal **i** cuelgan todos los puntos que se encuentran a distancia **i** de **X°**, es decir, **D(X,X°)=i** para todos

los **X** del subárbol **T(X°,i)**. La misma acción se repite recursivamente hasta que el tamaño de un subárbol sea menor o igual que una cantidad prefijada **tmc** (tamaño máximo de la celda). En este trabajo se considerará **tmc=1**.

7.2.- Búsqueda de los más similares.

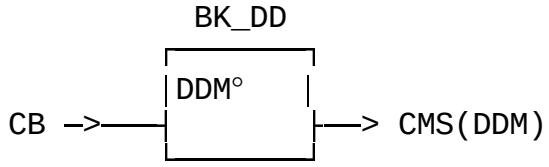
El problema que se plantea es el siguiente: dado un conjunto **S** de puntos (estructurado como un árbol **BK**), un punto de búsqueda **q** (que puede pertenecer o no **S**) y una distancia **D** en el espacio de puntos, encontrar el subconjunto de **S** formado por los puntos que se encuentran a distancia mínima de **q**.

Se utiliza el esquema clásico de búsqueda en el **BK**, con el criterio de unión y corte siguiente: Si se denomina δ a la distancia máxima a la cual interesa buscar puntos y **X°** al nodo actual entonces no es necesario explorar el ramal **k** si se verifica que:

$$| k - D(q,X°) | > \delta$$

8.- ESQUEMA BK_DD.

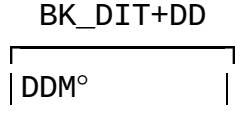
En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DD** y seleccionando al azar la cadena discriminante en cada nodo.

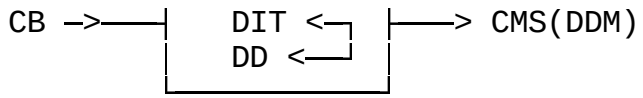


El procedimiento consiste en posicionar la cadena de búsqueda, **CB**, en el ramal del nodo actual según su **DD** a la cadena de ese nodo y proseguir en todos los ramales que se encuentren a una separación de dicha posición inferior o igual a **DDM**. El valor **DDM** comienza siendo **DDM°** (una cota superior de la **DD** mínima entre **CB** y las cadenas del diccionario), y cada vez que se encuentra un valor **DD** menor que **DDM** se actualiza, **DDM=DD**, de forma que al final, **DDM** será igual a la mínima **DD** entre la cadena de búsqueda y las del diccionario. **CMS(DDM)** es el conjunto de cadenas del diccionario que se encuentran a **DD=DDM** de **CB**.

9.- ESQUEMA BK_DIT+DD.

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando la **DIT** como distancia para su construcción y como filtro adaptivo para evitar cálculos de **DD** y seleccionando al azar la cadena discriminante en cada nodo.

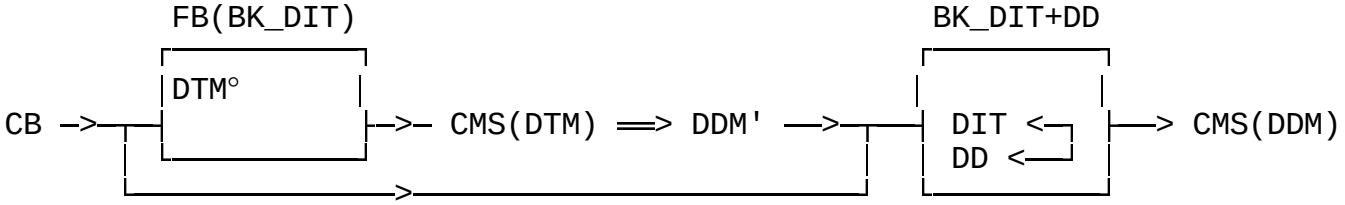




El procedimiento consiste en posicionar la cadena de búsqueda, **CB**, en el ramal del nodo actual según su **DIT** a la cadena de ese nodo y proseguir en todos los ramales que se encuentren a una separación de dicha posición inferior o igual a **DDM**. El valor **DDM** comienza siendo **DDM°**, cada vez que se encuentra un valor **DIT** menor o igual que **DDM** se evalúa **DD**, si **DD** es menor que **DDM** se actualiza **DDM**, **DDM=DD**, de forma que al final, **DDM** será igual a la mínima **DD** entre la cadena de búsqueda y las del diccionario. **CMS(DDM)** es el conjunto de cadenas del diccionario que se encuentran a **DD=DDM** de **CB**.

10.- ESQUEMA [FB(BK_DIT)] + [BK_DIT+DD].

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DIT** y seleccionando al azar la cadena discriminante en cada nodo. Para reducir el radio de búsqueda inicial en el esquema **BK_DIT+DD**, se antepone un filtro de la búsqueda usando sólo **DIT** que proporciona la menor **DD** de entre las cadenas del diccionario con menor **DIT**.



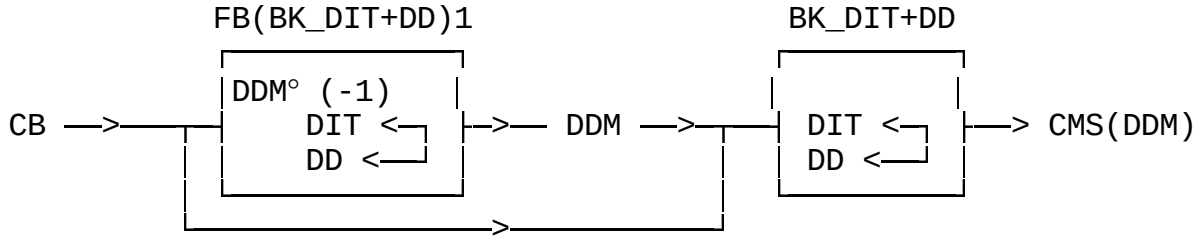
El procedimiento consta de dos etapas: la segunda etapa, denominada **BK_DIT+DD**, es la descrita en la sección 9 con la salvedad de que el valor inicial de **DDM** es **DDM'**, evaluado antes de comenzar esta fase. **DDM'** estará más cerca del valor final de **DDM** que **DDM°**. La primera parte consiste en posicionar la cadena de búsqueda, **CB**, en el ramal del nodo actual según su **DIT** a la cadena de ese nodo y proseguir en todos los ramales que se encuentren a una separación de dicha posición inferior o igual a **DTM**. El valor **DTM** comienza siendo **DTM°** (una cota superior de la **DIT** mínima entre **CB** y las cadenas del diccionario), cada vez que se encuentra un valor **DIT** menor que **DTM** actualiza **DTM**, **DTM=DIT**, de forma que al final, **DTM** será igual a la mínima **DIT** entre la cadena de búsqueda y las del diccionario. **CMS(DTM)** es el conjunto de cadenas del diccionario que se encuentran a **DIT=DTM** de **CB**. **DDM'** es la mínima **DD** entre **CB** y **CMS(DTM)**.

Durante la primera etapa, al abandonar el nodo actual se alojan en él, la **DIT** evaluada en ese nodo y el valor más reciente de **DTM**; de tal manera que si se accede a ese nodo en la segunda etapa no sea necesario volver a calcular la **DIT**.

11.- ESQUEMA [FB(BK_DIT+DD)1] + [BK_DIT+DD].

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DIT** y seleccionando al

azar la cadena discriminante en cada nodo. La idea que se plasma en este esquema, a lo largo de su primera fase, es encontrar la **DD** mínima entre la cadena de búsqueda y las cadenas del diccionario, usando el mismo procedimiento de búsqueda **BK_DIT+DD**, con la salvedad de que, al no buscar todas las cadenas más parecidas sino sólo la **DD** mínima, cuando se alcanza un valor de **DD** se pasa a buscar con el siguiente, **DD-1**.

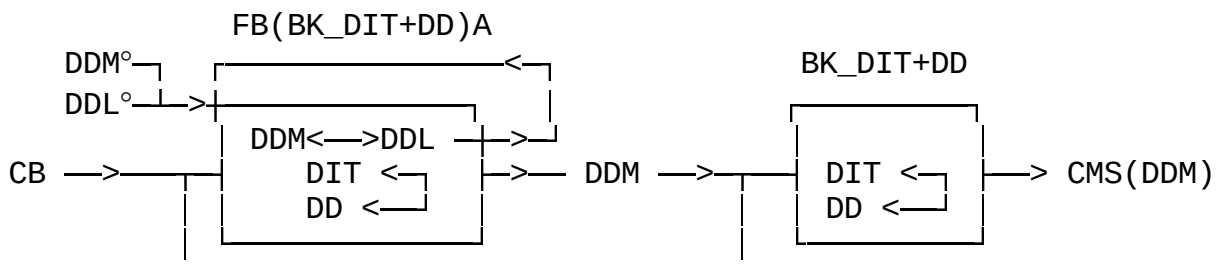


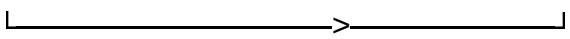
El procedimiento consta de dos etapas: la segunda etapa, denominada **BK_DIT+DD**, es la descrita en la sección 9 con la salvedad de que el valor inicial de **DDM**, evaluado en la primera fase, es precisamente su valor final. La primera parte consiste en posicionar la cadena de búsqueda, **CB**, en el ramal del nodo actual según su **DIT** a la cadena de ese nodo y proseguir en todos los ramales que se encuentren a una separación de dicha posición inferior o igual a **DDM-1**, donde **DDM** es inicialmente **DDM°**. Cada vez que **DIT** es menor o igual que **DDM-1** se calcula la **DD**, si ésta es menor o igual que **DDM-1** se actualiza **DDM**, **DDM=DD**. Cuando **DDM=0**, ó no existe ninguna **DD** menor o igual que **DDM-1**, esta fase concluye siendo **DDM** el valor inicial para la segunda fase.

En la primera etapa, cuando se visita un nodo por primera vez, al abandonarlo se alojan en él: la **DIT** evaluada en ese nodo, el valor de **DDM-1** con que se accedió a ese nodo, **DDC**, los valores de los extremos inferior y superior del conjunto de ramales explorados, **DTCi** y **DTCs**, y la **DD** si se hubiera evaluado; de esta forma en la segunda etapa no es necesario calcular las **DIT** en los nodos ya visitados en la primera etapa ni las **DD** ya calculadas previamente.

12.- ESQUEMA [FB(BK_DIT+DD)A] + [BK_DIT+DD].

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DIT** y seleccionando al azar la cadena discriminante en cada nodo. Se introduce una modificación del filtro de búsqueda descrito en el apartado anterior con el objeto de alcanzar antes, sobre todo en el caso de rangos muy grandes, el radio final mínimo de búsqueda. En lugar de intentar hallar la **DD** siguiente a la obtenida, se busca la **DD** que se encuentra en el punto medio entre la obtenida y el valor más grande por debajo del cual se sabe que no es posible hallar valores de **DD**.



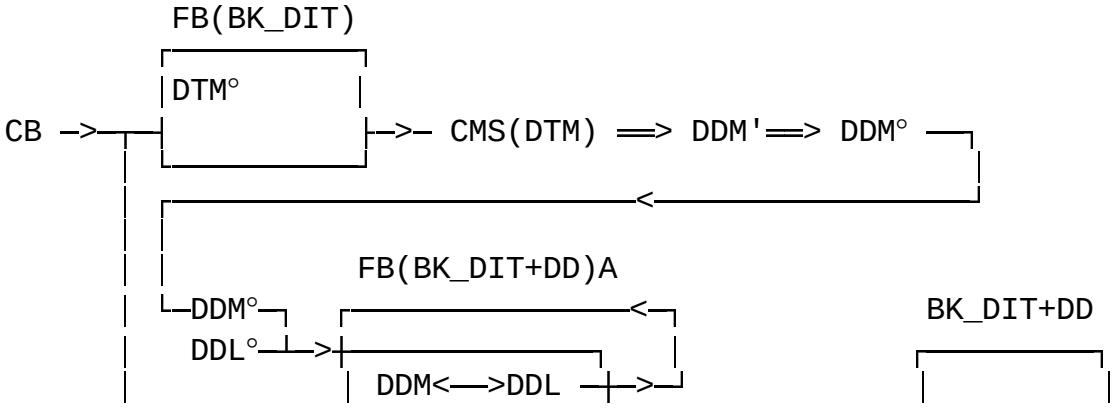


El procedimiento consta de dos etapas: la segunda etapa, denominada **BK_DIT+DD**, es la descrita en la sección 9 con la salvedad de que el valor inicial de **DDM**, evaluado en la primera fase, es precisamente su valor final. La primera parte consiste en posicionar la cadena de búsqueda, **CB**, en el ramal del nodo actual según su **DIT** a la cadena de ese nodo y proseguir en todos los ramales que se encuentren a una separación de dicha posición inferior o igual a **DD½**. El valor **DD½** es la semisuma de **DDM** y **DDL**, donde **DDM** es inicialmente **DDM°** y **DDL** es la cota superior de las **DD** que no son posibles de encontrar entre **CB** y las cadenas del diccionario, se inicializa a **-1**. Cada vez que **DIT** es menor o igual que **DD½** se calcula la **DD**, si ésta es inferior que **DDM** se actualiza **DDM**, **DDM=DD**, recomenzando esta etapa si es **DDM≤DD½**. Si no existe ninguna **DD** menor o igual que **DD½** se actualiza **DDL**, **DDL=DD½** reanudándose esta parte. Cuando la diferencia entre **DDM** y **DDL** sea indivisible, **DDM** es el valor inicial para la segunda fase.

En la primera etapa, cuando se visita un nodo por primera vez, al abandonarlo se alojan en él: la **DIT** evaluada en ese nodo, el valor de **DD½** con que se accedió a ese nodo, **DDC**, los valores de los extremos inferior y superior del conjunto de ramales explorados, **DTCi** y **DTCs**, y la **DD** si se hubiera evaluado; de tal manera que si se accede a ese nodo posteriormente no sea necesario volver a calcular la **DIT**, y no sea necesario calcular **DD** si ya se ha hecho. En esta etapa cuando un nodo es revisitado no es necesario calcular **DIT**, al abandonar el nodo si se han explorado ramales que anteriormente no lo hubieran sido han de actualizarse **DTCi** y **DTCs**; si es necesario calcular **DD** se almacena su valor y el valor de **DD½** con que se accedió al nodo se guarda en **DDC**. De esta forma en la segunda etapa no es necesario calcular las **DIT** en los nodos ya visitados en la primera etapa ni las **DD** ya calculadas previamente.

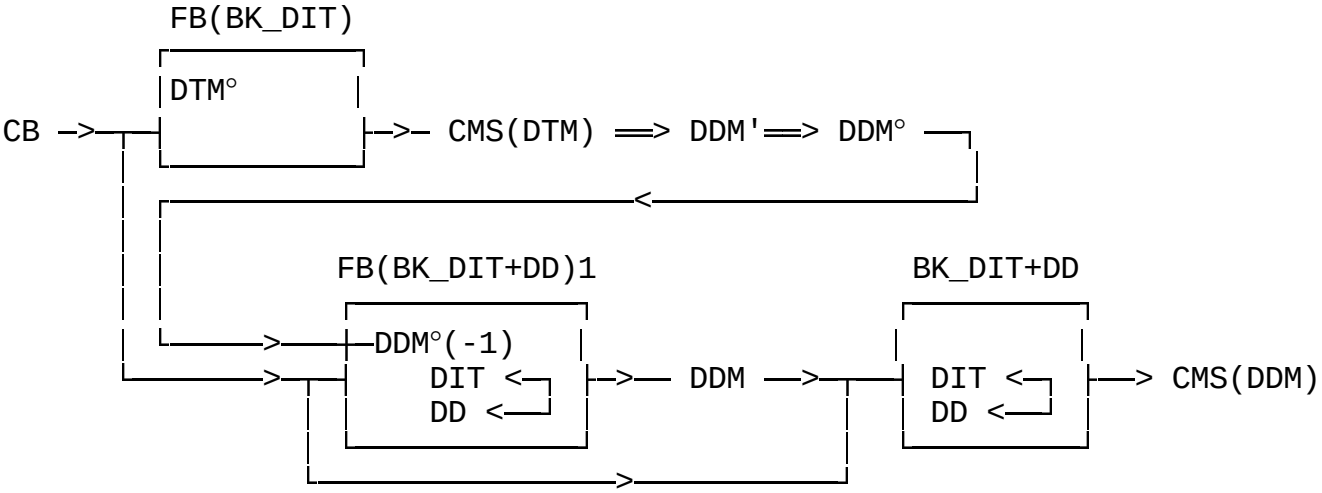
13.- ESQUEMA [FB(BK_DIT)] + [FB(BK_DIT+DD)A] + [BK_DIT+DD].

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DIT** y seleccionando al azar la cadena discriminante en cada nodo. El procedimiento consta de tres etapas: la primera, **FB(BK_DIT)**, coincide con la primera fase comentada en la sección 10. La segunda y tercera etapas son análogas a las descritas en la sección 12, **[FB(BK_DIT+DD)A] + [BK_DIT+DD]**, con la salvedad de que el valor inicial, para la segunda fase, de **DDM°** es **DDM'** que resulta de la primera etapa.



14.- ESQUEMA [FB(BK_DIT)] + [FB(BK_DIT+DD)1] + [BK_DIT+DD].

En este esquema el diccionario se encuentra estructurado como un árbol **BK** utilizando como distancia para su construcción la **DIT** y seleccionando al azar la cadena discriminante en cada nodo. El procedimiento consta de tres etapas: la primera, **FB(BK_DIT)**, coincide con la primera fase comentada en la sección 10. La segunda y tercera etapas son análogas a las descritas en la sección 11, **[FB(BK_DIT+DD)] + [BK_DIT+DD]**, con la salvedad de que el valor inicial, para la segunda fase, de **DDM°** es **DDM'** que resulta de la primera etapa.



15.- RESULTADOS EXPERIMENTALES Y CONCLUSIONES.

A continuación se analizan los resultados de los esquemas de búsqueda llevando a cabo una interpretación de los mismos y una evaluación de los esquemas.

Para los estudios experimentales se ha elegido un diccionario con las 2089 palabras de uso más frecuente en castellano. Se han realizado búsquedas con dichas palabras y con las cadenas que resultan de distorsionar las mismas en diferentes grados.

Respecto a los requerimientos de memoria se establece un resumen en la tabla 1. En la que se expresa, para cada uno de los esquemas, la ocupación total dividida entre el número de palabras del diccionario, donde se ha supuesto que los punteros ocupan 4 Bytes.

ESTRUCTURA (CRITERIO CONSTRUCCION: ALEATORIO)	BYTE/PALABRA
DICCIONARIO	7.07
DDS	11.07
DITE+DD	35.80

BK_DD	12.52
BK_DIT+DD	47.79
[FB(BK_DIT)] + [BK_DIT+DD]	49.35
[FB(BK_DIT+DD)1] + [BK_DIT+DD]	51.92
[FB(BK_DIT+DD)A] + [BK_DIT+DD]	51.92
[FB(BK_DIT)] + [FB(BK_DIT+DD)A] + [BK_DIT+DD]	51.92
[FB(BK_DIT)] + [FB(BK_DIT+DD)1] + [BK_DIT+DD]	51.92

TABLA 1

Se observa una escasa diferencia entre la ocupación de la estructura **BK_DD** y la **DDS**. Esto es debido a que, aunque en la **BK_DD** los nodos no-hojas tienen un promedio de 2.6 punteros, algo más del 50% son nodos hojas, sin punteros; mientras que en la **DDS** cada nodo contiene un puntero.

El incremento en la ocupación de la **BK_DIT+DD** respecto a la **BK_DD** se debe atribuir casi exclusivamente a la conveniencia de disponer, en cada nodo, de las frecuencias de los distintos caracteres que componen cada una de las palabras, en forma de lista encadenada.

El aumento en la ocupación de la **DITE+DD** frente a la **DDS** se debe a que la propia estructura **DITE+DD** está formada por los caracteres y sus frecuencias de manera encadenada. Tal aumento es inferior al del párrafo anterior debido a que esta estructura comparte información carácter-frecuencia.

Todos los esquemas definidos a lo largo de este trabajo necesitan un valor inicial que sirve como acotación del radio de búsqueda. Dicho valor, referenciado como **DDM°** (**DTM°** en el caso del **FB(BK_DIT)**), es una cota superior de la distancia entre la cadena de búsqueda, **CB**, y las cadenas del diccionario. Como ya se discutió, **[SD87]**, $\mathbf{DDM^\circ = |CB| * \pi/(1-\pi)}$, siendo π la distorsión máxima permitida para la cadena **CB**. Dado que **DIT≤DD**, es posible establecer **DTM°=DDM°**.

El esquema **BK_DD** como puede apreciarse claramente en la figura 2 queda ampliamente mejorado por el **BK_DIT+DD**, aunque éste aún no llega a superar la realización del **DITE+DD**.

Debido a que los aciertos se mantienen razonablemente altos hasta una distorsión del **30%** y luego caen rápidamente [SD87], parece lógico situar los esquemas de búsqueda para distorsiones permitidas en el intervalo $\mathbf{0 \leq \pi \leq 1/3}$. Esta consideración implica que $\mathbf{DDM^\circ = |CB|/2}$.

Con $\mathbf{DDM^\circ = |CB|/2}$ no es posible garantizar que se encuentre respuesta para aquellas cadenas de búsqueda, **CB**, tales que la distorsión que han sufrido sea superior al **33.33%**, pero en todo caso, de hallarse respuesta, ésta será el conjunto de palabras más similares según **DD** a **CB**. Para evitar la posibilidad de que para palabras distorsionadas por encima de un **33.33%** no exista respuesta, tendrá que inicializarse **DDM** a un valor superior, si $\mathbf{DDM^\circ = \infty}$ los esquemas siempre obtienen respuesta.

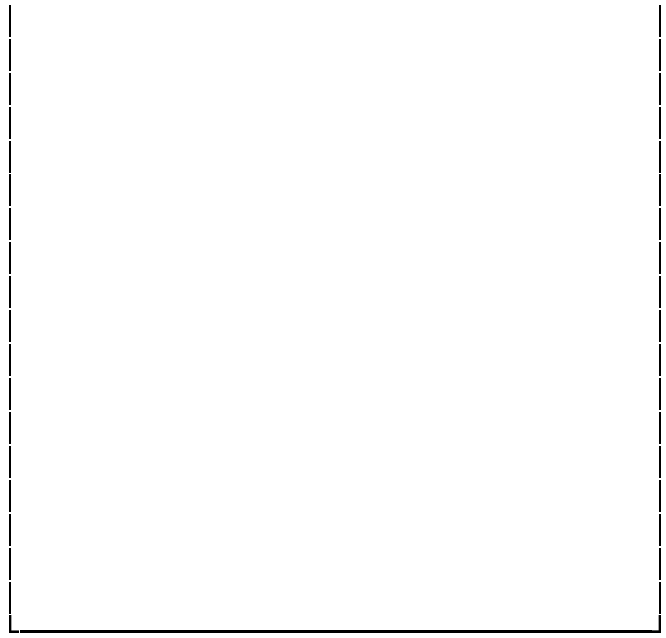


Figura 2

Figura 3

En la figura 3 se observa que para distorsiones bajas todos los esquemas con filtro de búsqueda, salvo **[FB(BK_DIT+DD)1] + [BK_DIT+DD]**, se comportan mejor que el **DITE+DD**. El **[FB(BK_DIT+DD)A] + [BK_DIT+DD]** es el esquema que da una mejor realización siendo su conducta mejor que la de **DITE+DD** hasta distorsiones entorno al 20%. El resto de los esquemas se comportan de manera muy similar entre si y son mejores que el **DITE+DD** hasta distorsiones en torno al 17%.

La optimización de los esquemas de búsqueda con filtros de búsqueda respecto a la **BK_DIT+DD** se debe atribuir principalmente al descenso del número de cadenas **DD** exploradas. Ya que, si bien se produce una disminución del número de cadenas a las que se calcula **DIT** a distorsiones bajas, al aumentar la distorsión dicha diferencia disminuye. No obstante debe observarse que para las distorsiones más altas se produce una mejor realización en el **BK_DIT+DD** respecto a los esquemas con filtros, esta mejora no es debida a un menor cálculo ni de **DIT** ni de **DD** sino a una menor complicación del esquema **BK_DIT+DD** respecto a los otros.

El número de componentes de **DIT** evaluadas en estos esquemas es siempre superior al de **DITE+DD**, aumentando con la distorsión. Esta es la razón por la que, en distorsiones mayores, figura 4, este efecto sobrepasa al ahorro en el cálculo de **DD**.

En los esquemas con filtro de búsqueda, así como en el **DITE+DD** el número de componentes de **DIT** evaluado crece con la distorsión aunque en aquellos lo hace más rápidamente, figura 4, y es por lo que a partir de una cierta distorsión, como puede verse en la figura 5, la pérdida de realización por este efecto no queda compensada por el ahorro en cálculo de **DD**. Comparando la figura 5 con la figura 3 se puede observar una diferente sensibilidad a la variación de la inicialización de **DDM**, sobre todo entre los esquemas de dos etapas con filtro de búsqueda **FB(BK_DIT+DD)** con y sin aproximación adaptiva.

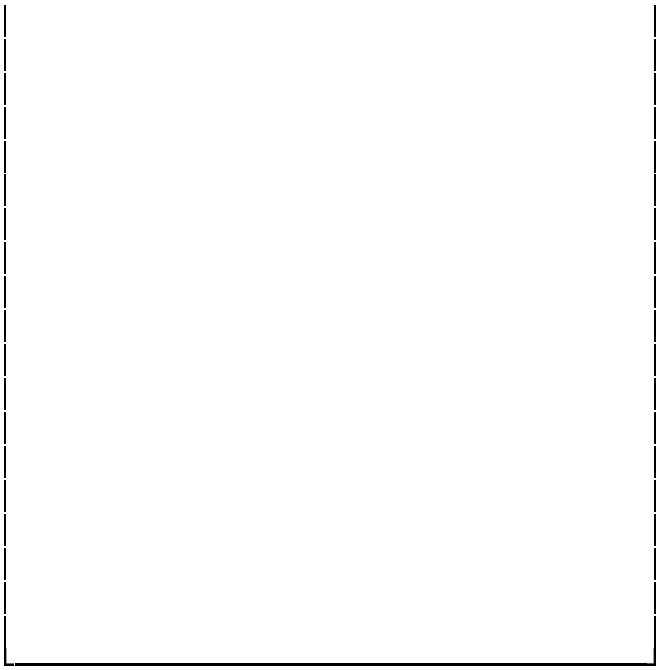


Figura 4

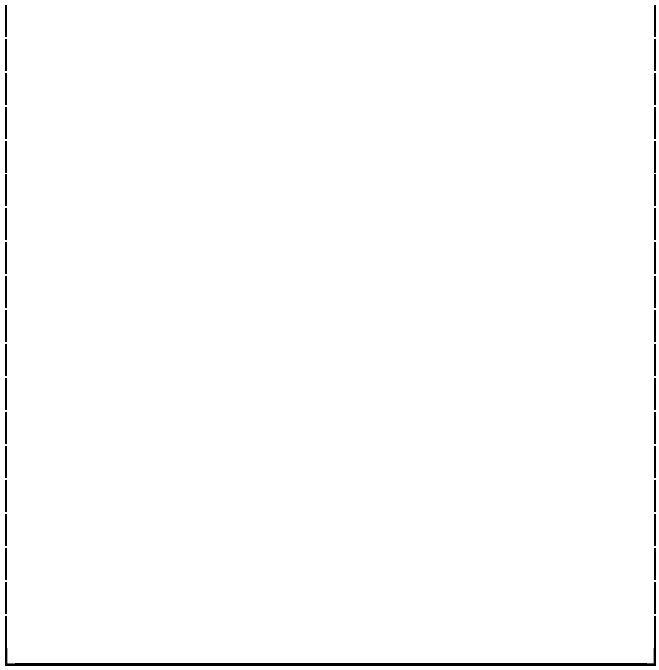


Figura 5

La línea de investigación que actualmente se sigue apunta hacia la reducción del cálculo de **DIT**, usando los criterios siguientes:

a) Disminuir el número de sus componentes que se han de evaluar, mediante la estructuración adecuada de las mismas.

b) Hacer evolucionar la estructura **BK** a fin de que puedan reflejarse estructuralmente las distancias **DIT** y **DD**, con el objeto de que no sólo sean utilizadas las menores **DD** (**DDM**) para discriminar cadenas sino todas las **DD** que se calculen.

BIBLIOGRAFIA:

[AL67] ALBERGA, C.N.: String Similarity and Misspellings. CACM 1967, 10 (5),302-313.

[BK73] BURKHARD, W.A.; KELLER, R.M.: Some Approches to Best-Mach File Searching. Comm. ACM 16, 4 (1973).

[LA87] LANDAU, G. M.: String matching in erroneous input. Thesis submitted for degree Ph Doctor Tel-Aviv University. Technical Report 57/87.

[LE66] V. I. LEVENSHTEIN, Binary codes capable of correcting, insertions and reversals. Soviet Phys. Dokl. 10 (1966), 707-710.

[MO70] MORGAN, H.L.: Spelling Correction in System Programs. CACM 1970, 13 (2), 90-94

[NK82] NEVALAINEN, O.; KATAJAINEN J.: Experiments with a Closet Point Algorithm in Hamming Space. Angewandte Informatik 5/82, 277-281.

[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 Informática (1987), Vol. II, 1169-1189.

[SZ73] SZANSER, A.J.M.: Automatic Error Correction of Natural Text. Part I. Computer Science N. 46 National Physics Laboratory, 1973.

[UK83] UKKONEN, E.: On approximate string matching. Proc. Int. Conf. Found. Comp. Theor., Lecture Notes in Computer Science 158, Springer-Verlag, (1983), 487-495.

[WF74] WAGNER, R.A.; FISCHER, M.J.: The String-to-String Correction Problem. JACM 1974, 21 (1), 168-173.