Búsqueda por semejanza

Rafael Camps Paré

1. INTRODUCCION

En múltiples aplicaciones informáticas se presenta la necesidad de determinar si dos palabras son o no semejantes. Uno de los casos más frecuentes es la búsqueda de personas de un fichero, dados sus apellidos. Buscando, por ejemplo, a GIMENEZ VELASCO podrá ocurrir que no encontramos a nadie que figure así en el fichero, porque figura como XIMENIS VASCO. Nos interesará pues detectar que XIMENIS VASCOS y GIMENEZ VELASCO son semejantes.

Otro ejemplo de aplicación en la que interesa reconocer la semejanza, podría ser la «Enseñanza con ayuda de ordenador». Si a la pregunta ¿Cuál es la capital de España? el alumno responde MADRIZ, el sistema podrá considerar que se trata de una respuesta aceptable, advirtiendo sin embargo al alumno que la respuesta correcta es MADRID.

En general, necesitaremos el reconocimiento de la semejanza en aquellas situaciones en las que es conveniente aceptar un cierto grado de variabilidad o incorrección en la información suministrada al sistema informático.

Es obvio que la solución al problema depende enormemente de lo que se entienda por *semejante*. En nuestro caso nos interesa aceptar como semejantes dos palabras, si con cierta probabilidad una de ellas puede ser producida por errores en la transmision oral o en la escritura (manual o por teclado) de la otra. Ej.: Árbol y Abrol, Txyki y Chiqui.

Este problema ha sido abordado desde los primeros tiempos de la informática [GLA-57]. Se han ido proponiendo métodos concretos para casos concretos y desde muy diversos puntos de vista. Recientemente han aparecido dos trabajos [CAM-81] y [Hall-80] que presentan con detalle la situación actual del tema.

Aquí nos limitaremos a presentar algunos métodos de interés práctico (haciendo un resumen de [CAM-81]) orientando a los ejemplos el caso concreto de búsqueda de apellidos.

2. CODIFICACION Y COMPARACION

Existen dos formas de enfocar el problema que nos ocupa, y que llevan a dos «tipos» o «familias» de soluciones.

El primer enfoque considera el problema siguiente: Dado un conjunto (o fichero) F de palabras fi (entenderemos por palabra una tira de caracteres), se trata de obtener el subconjunto F'cF de todas las palabras semejantes (en el sentido citado anteriormente) a una palabra dada f).

Una de las soluciones más extendidas (especialmente para el caso de que fi sean apellidos) es la llamada *codificación fonética*, consistente en hacer corresponder un código a cada palabra, procurando que todas las palabras semejantes tengan asociado el mismo código. Como ejemplo, consideremos un método de codificación fonética muy simple, que forme el código de una determinada palabra a base de concatenar las tres primeras consonantes de la palabra. Así, GARCÍA, GRACIA, GARCES y GROC serían considerados semejantes (código GRC), pero MARTI (código MRT) no lo sería. Si tenemos agrupadas las palabra fi por códigos, para encontrar las semejantes a una f determinada bastará calcular el código de f y extraer el subconjunto f con ese código.

Un enfoque alternativo considera el siguiente problema: Dadas dos palabras f y g, determinar si son o no semejantes. Este planteo supone, de hecho, la existencia de algún algoritmo o función de semejanza que permita comparar f y g. En este sentido, un algoritmo de comparación muy elemental puede ser el siguiente: dos palabras se considerarán semejantes cuando el número de posiciones en las que no coincidan los caracteres sea menor que la mitad de la longitud de la palabra más larga. Así, GARCIA y BARCIA se considerarán semejantes porque sólo difieren en posición, siendo

$$1 < \frac{5}{2}$$

Sin embargo GARCIA y GROC no se considerarán semejantes porque difieren en 4 posiciones y

$$4 < \frac{5}{2}$$

A este enfoque le denominaremos de *comparación*

3. EL ESPACIO DE LAS PALABRAS

Es útil imaginar las palabras como puntos de un *espacio* en el que los puntos están más (o menos) distantes unos de otros según su menor (o mayor) semejanza (Figura 1). Así, GARCIA estará cerca de GRACIA pero lejos de GIRASOL.

Esta conceptualización supone, de hecho, que medimos tal semejanza (o mejor dicho desemejanza) de algún modo como una *distancia.* Esta semejanza «real», o distancia, dependerá de las aplicaciones que se consideren en cada momento.

La codificación fonética divide este espacio en partes, una por cada código. Como no todas las palabras semejantes a una dada estarán en la misma parte, los métodos de codificación fonética no siempre identifican como semejantes palabras que lo son «realmente» *(subidentificación)* y además, identifican como semejantes palabras que no lo son *(sobreidentificación).* Ver [CAM-81].

En la figura 1 (que supone el método de las tres primeras consonantes) al buscar GARCIA se produce la sobreidentificación de GIRACOL, GRIACOL, AGUARACANO y GROC y la subidentificación de GARSIA y GARSA.

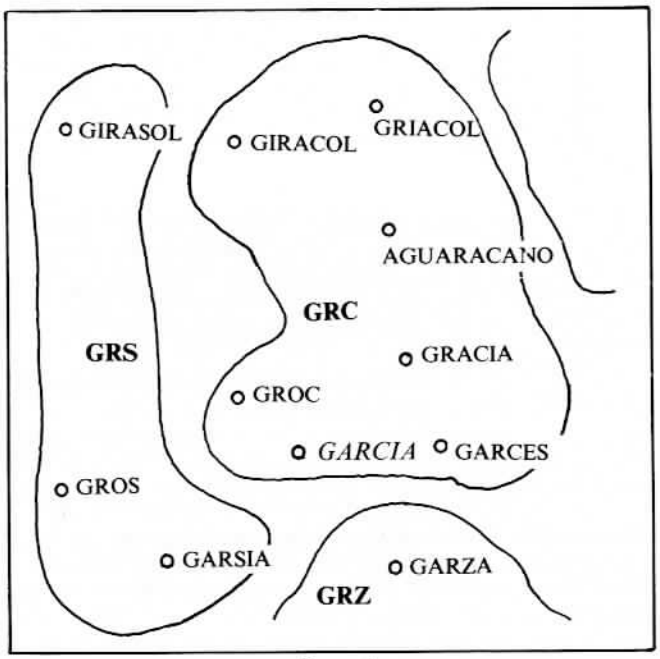

Figura 1

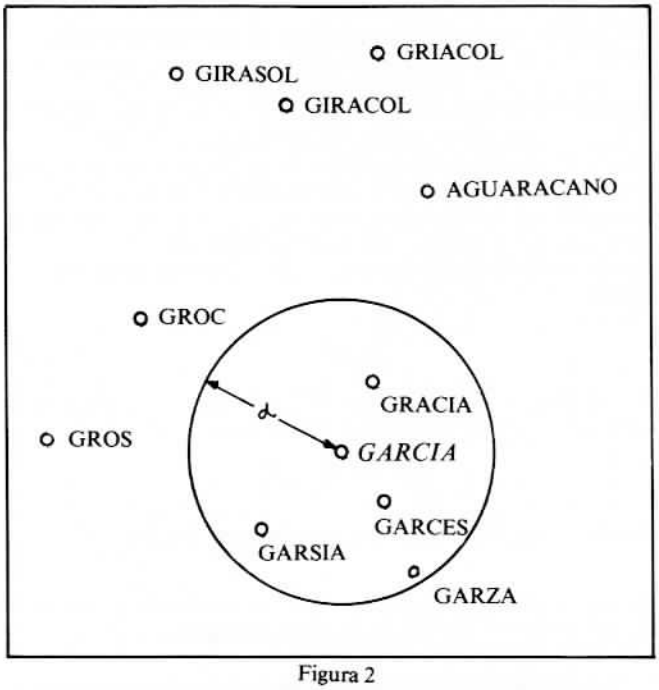

Figura 2

Algunos métodos de comparación calculan una medida de la semejanza o distancia. Entonces consideran semejantes a la palabra f todas aquellas palabras cuya distancia a ella es menor que un valor dado d. Figura 2.

El problema está en encontrar una función de distancia que de unos resultados parecidos a la «realidad» (quizá subjetiva). Ocurre con frecuencia que a y b nos parecen mucho más semejantes que a y c y sin embargo en las distancias calculadas resulta que $d(a, b) \geqslant d(a, c)$.

Esta discrepancia entre la distancia calculada y la «real» provoca, naturalmente, *subidentificación* y *sobreidentificación.*

Los métodos de cálculo de distancias dan, generalmente, mejores resultados que los de codificación en lo que se refiere a sobreidenficación. Sin embargo tiene el inconveniente (como veremos más adelante) de que resultan mucho más costosos en las búsquedas en ficheros.

4. METODOS DE CODIFICACION

Hemos visto ya en qué consisten, en líneas generales, los métodos de codificación fonética y hemos puesto como ejemplo el de las tres primeras consonantes.

En un artículo anterior [CAM-75] aparecido en Novática (del cual éste puede considerarse complementario) hablábamos sobre la codificación fonética y presentábamos el métodos SONS para apellidos españoles. (Al final del presente artículo se da una fe de erratas de aquel otro). En [CAM-81] pueden encontrarse variantes del SONS. Otros métodos de codificación se pueden encontrar en [ACH-68], [BLA-60], [DAV-62], y [DOL-70].

Aunque estos métodos están orientados básicamente a resolver problemas de *semejanza fonética,* al mismo tiempo resuelven problemas de *asemejanza ortográfica,* pues dos palabras semejantes ortográficamente a menudo lo son también fonéticamente. La mayor parte de los errores ortográficos más frecuentes (B en lugar de V, etc.) son «correctos» fonéticamete. Los errores ortográficos menos frecuentes suelen tener menor semejanza fonética [ALB-67].

La mayor parte de errores cometidos en la escritura de textos (especialmente con teclado) son: supresión o inclusión de un carácter; sustitución de un carácter por otro y transposición de dos caracteres.

Para abordar problemas de este tipo existen métodos de comparación sencillos muy válidos (ver 7.). Pero en ficheros con apellidos, sobretodo si hay diversidad de orígenes lingüísticos, esos métodos no son suficientes. Piénsese por ejemplo en un sistema de reserva de plazas de hotel o avión, en el que un apellido polaco es comunicado por teléfono por un norteamericano a un empleado canario. Se suele recurrir entonces a los métodos de codificación fonética. Precisamente la mayoría de estos métodos se han diseñado especialmente para apellidos (y eventualmente onomásticos y toponímicos).

5. TECNICAS DE BUSQUEDA PARA METODOS DE CODIFICACION

Dado que la codificación fonética divide el conjunto o fichero en partes y que cada parte tiene asociado un código (figura 1) podemos usar esos códigos para acceder a cada parte (o grupo de palabras) deseada, sin necesidad de acceder a las otras partes y usando técnicas clásicas o búsqueda (índices, «hashing», etc.). [BUR-76]. [KNU-75], [ROD-77], [SEV-74].

En la práctica, el código fonético suele formar parte de una clave más completa, más identificadora del objeto buscado. Veamos a continuación un ejemplo de realización de un esquema de este tipo.

El fichero principal, al que llamaremos PERSONAS, contiene información personal de varios cientos de miles de personas y consta de un registro por persona. Se utiliza desde un terminal de teclado haciendo consultas individualizadas. Los datos a partir de los que se plantea la búsqueda son normalmente los siguientes: apellidos, sexo, mes y año de nacimiento. Con frecuencia se desconocen algunos de estos datos, aunque se exige especificar siempre el primer apellido. Además pueden haber errores en los apellidos. Por otra parte se desea también poder acceder por DNI y por NSS (n.º de seguridad social).

Para resolver el problema de los errores en los apellidos se utiliza la codificación fonética SONS ([CAM-75]) que usa dos letras por apellido. Al fichero PERSONAS se accede por «Hashing» (cálculo de dirección) usando como identificador un código, CIDEN, con el siguiente formato: XXYYSAAMMNN, donde:

XX es el código fonético SONS del primer apellido
YY es el código fonético SONS del segundo apellido

S es el sexo (H, V)
AA es el año de nacimiento
MM es el mes de nacimiento
NN es el número de secuencia

El número de secuencia tiene como objeto diferenciar entre sí las personas cuyos códigos fonéticos, sexo y mes y año de nacimiento coinciden. Tal código es generado, manipulado y conocido exclusivamente por el sistema, siendo inaccesible desde el exterior por un usuario.

Se dispone de un fichero auxiliar, AUXCIDEN (ver figura 3), que contiene el código CIDEN de cada persona, uno por registro. Este fichero se organiza como secuencial-indexado, por la clave CIDEN, y tiene posibilidad de acceso por claves parciales (clave parcial = cualquier parte izquierda de la clave; esta posibilidad existe en casi todos los secuenciales-indexado del mercado).

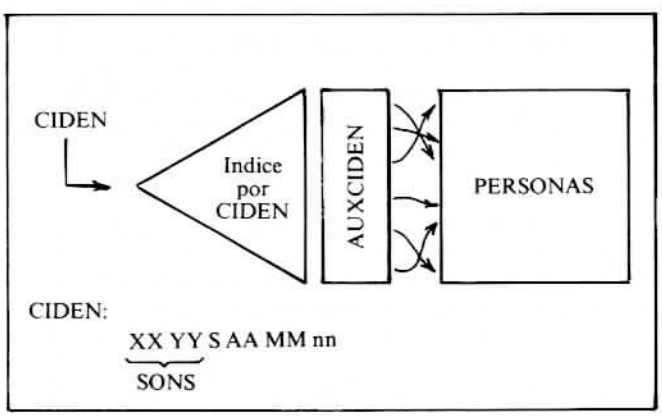

Figura 3

Con esta organización la búsqueda, por ejemplo, de «VAQUER DIAS, año de nacimiento = 1936» se haría del modo siguiente: tras calcular los códigos fonéticos SONS de VAQUER y de DIAS (que son BG y TX respectivamente), se accederá «directamente», por medio del índice de CIDEN, al primer registro de AUXCIDEN que empiece por BGTX. A partir de este registro se irá avanzando secuencialmente por AUXIDEN buscando (rápidamente pues caben varios centenares de registros por bloque) aquellos CIDEN cuyas posiciones 6.ª y 7.ª contengan 36. Con aquellos CIDEN que cumplan la condición se accederá directamente a PERSONAS. La búsqueda acabará al encontrar en AUXCIDEN un CIDEN cuyos siete primeros caracteres sean posteriores a BGTXV36. En el caso de un fichero enormemente grande, sería mejor fraccionar la búsqueda en tres, usando como claves parciales BGTXD, BGTXH y BGTXV.

El índice de CIDEN es también útil para la asignación de nuevos CIDEN. Al insertar una nueva persona en PERSONAS, se buscará por medio del índice el CIDEN más pequeño cuyas primeras nueve posiciones (sin número de secuencia) coincidan con los dos de la nueva persona. Si no se encuentra ninguno, se inserta un nuevo CIDEN con número de secuencia = 99. En caso contrario, se inserta un nuevo CIDEN con número de secuencia una unidad inferior al del encontrado.

Para las búsquedas por DNI se dispone de un fichero auxiliar al cual se accede por «hashing». Sus registros, uno por persona, contienen simplemente las parejas CIDEN/DNI. Para el acceso por NSS se usa una técnica parecida.

Como ya hemos hecho notar anteriormente, los métodos de codificación producen bastante sub y sobreidentificación. La sobreidentificación puede ser indeseable ya sea porque produce listas de respuesta excesivamente largas o porque tiene efectos psicológicos desfavorables en los usuarios. La sobreidentificación se puede reducir haciendo pasar el grupo de palabras obtenidas como respuestas a la búsqueda, por un filtro adicional, que puede consistir en una función de distancia y un umbral ajustable por el usuario del sistema. Fig. 4 (Ver 6).

En el caso de ficheros PERSONAS, se usa como filtro el método de comparación FERM (ver 8) que aunque no deja pasar algunas palabras semejantes identificadas como tales por el SONS, elimina casi todas las palabras no semejantes que el SONS sobreidentifica. Cada vez que el proceso de búsqueda obtiene una persona de PERSONAS, pasan sus apellidos por el filtro FERM y se decide si deben incluirse en la respuesta o no.

6. METODOS DE COMPARACION

En 2 hemos introducido el concepto de método de comparación de función de semejanza o distancia y hemos dado como ejemplo un método muy simple de diferencias por posición.

La mayor parte de métodos de comparación explotan la coincidencia de caracteres entre las dos palabras que se comparan. Por lo tanto son métodos que tienen en cuenta esencialmente los aspectos físicos, por lo que parecen más

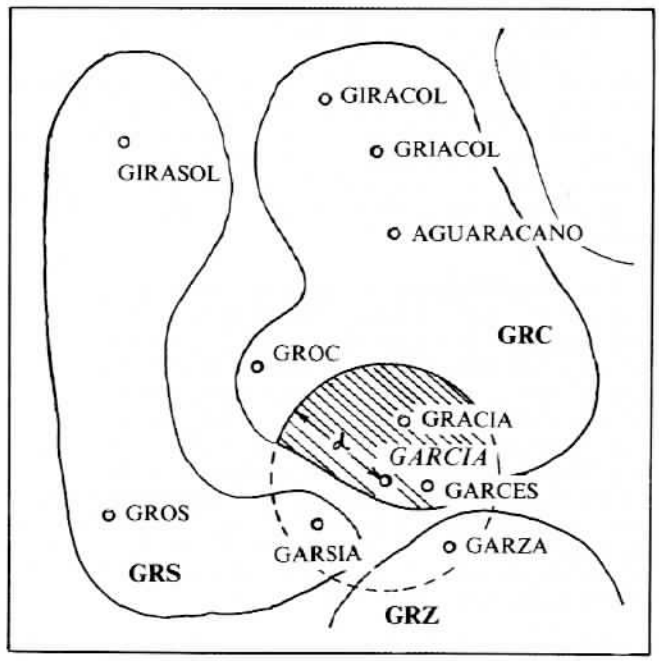

Figura 4

ortográficos que fonéticos. Pero ya hemos hecho notar en 4 la fuerte correlación existente entre semejanza fonética y ortográfica.

Los algoritmos de comparación de palabras pueden clasificarse en dos grandes grupos: *cualitativos* y *cuantitativos*. Llamamos cualitativos a los que producen un resultado binario, determinando simplemente si dos palabras pueden considerarse semejantes o no. Llamamos cuantitativos a los que generan un valor numérico como medida de la semejanza.

Los métodos cuantitativos tratan de medir la semejanza entre dos palabras, a base de evaluar una *distancia* entre ellas.

En la práctica se suelen presentar situaciones en las que el concepto de distancia es muy útil. Por ejemplo: para la corrección automática de textos, se acostumbra a consultar un diccionario de palabras correctas; cuando determinada palabra de un texto no se localiza en el diccionario, se busca en él la palabra «más parecida», palabra que se incluye en el texto en sustitución (siempre que sea suficientemente parecida) de la anterior. La noción de «la más parecida» exige un método de comparación cuantitavivo, una medida de la distancia.

Otro ejemplo es la búsqueda de personas en un fichero a partir de un solo apellido en el que pueden haber errores. En un terminal se acostumbra a mostrar otros datos de las personas encontradas, que permitirán al operador identificar a la persona buscada. En este caso y si el número de personas «respuesta» resulta ser muy grande, se puede tratar de reducir el tiempo total de operador, mostrando a las personas por orden decreciente de semejanza respecto al apellido dado. Este «orden decreciente» exige de nuevo la existencia de una «distancia».

En cualquier caso, es interesante notar que todo un método cuantitativo puede pasar a ser cualitativo fijando un *umbral* a la distancia. Los métodos cualitativos basados en el umbral tienen la ventaja de que alterando éste puede regularse la severidad del criterio de semejanza.

Nótese que estamos usando aquí el término distancia en un sentido amplio. No se trata necesariamente de una distancia métrica, pues muchas de las distancias propuestas no cumplen la ley de desigualdad triangular:

$$d(a, b) + d(d, c) \geqslant d(a, c)$$

Generalmente los métodos de comparación someten a *transformaciones* previas las tiras de caracteres a comparar. En primer lugar se procura ponerlas en un formato estandarizado o *normalizado* que, en principio, dependerá de la aplicación, de los procedimientos de entrada de datos y del método de comparación utilizado. Ejemplos: eliminación de grupos y caracteres no significativos para la búsqueda (partículas DE, Y,..., SA, AVDA., Número y piso, Caracteres especiales, etc.); en el nombre de una calle eliminar todas las palabras menos la última, etc.

Tras la normalización se efectúa a menudo alguna *transformación fonética*, como por ejemplo V→B, preparando las palabras para que cuando más adelante se comparen dos letras (o grupos de letras) fonéticamente equivalentes, el resultado sea la igualdad. Eso reducirá la subidentificación. Algunos métodos tienen una fase de transformación fonética muy importante. Así por ejemplo, en el sistema de reserva de plazas una cierta compañía aérea se someten los dos apellidos a comparar, a transformaciones fonéticas que convierten cada carácter o grupo de caracteres en un código de dos cifras (del 01 al 11). Después se comparan las dos tiras de códigos y se aceptan como semejantes si tienen como mínimo tres códigos coincidentes.

Presentamos a continuación algunos métodos de comparación.

7. ALGORITMO DE MORGAN-DAMERAU

Un buen ejemplo de método cualitativo es el de *Morgan-Damerau* [MOR-70]

Se propuso en conexión con los errores de escritura de programas. Damerau [DAM-64] señaló y Morgan confirmó, que el 80 % de los errores en los textos de programas son de uno de los cuatro tipos siguientes: Falta de un carácter, sobra de un caracter, sustitución de un caracter por otro y transposición de dos caracteres adyacentes. (Son los errores típicos que se producen al usar un teclado).

Morgan propone un algoritmo de comparación (cualitativo) muy rápido, (ver figura 5. Es básicamente el ya propuesto por Damerau) que da como semejantes aquellos pares de palabras que se diferencian en uno (y sólo uno) de esos cuatro errores. En 10 discutiremos el algoritmo (cuantitativo) de Wagner que es mucho más general que el de Morgan-Damerau.

```
Las dos palabras a comparar, A y B, están encuadradas a la izquierda y sus longitudes
son L_A y L_B.
R ← A xor B
Si R = 0
entonces son semejantes (pues son iguales)
si no
    si L_A = L_B
    entonces si en R hay un solo caracter no nulo
             entonces son semejantes (pues es una sustitución)
             si no
                  si en R hay más de dos caracteres no nulos
                  entonces no son semejantes
                  si no
                       si los dos caracteres de R, no nulos, son adyacentes e
                       iguales
                       entonces si hay transposición en esas posiciones de A y
                                 B
                                 entonces son semejantes (pues es una transpo-
                                 sición)
                                 si no
                                      no son semejantes
                                 finsi
                       si no
                            no son semejantes
                       finsi
                  finsi
             finsi
    si no
         si |L_A - L_B| = 1
         entonces encuadrar A y B a la derecha
                  S ← A xor B
                  si el primer caracter por la izquierda no nulo de R, ocupa la
                  misma posición que el primer caracter por la derecha no nulo
                  de S
                  entonces son semejantes (pues falta o sobra un caracter)
                  si no
                       no son semejantes
                  finsi
         si no
              no son semejantes
         finsi
    finsi
finsi
```

Figura 5

8. EL METODO FERM

En este apartado presentamos un método de comparación, al que llamamos *FERM*, pensado para ser usado como filtro del método SONS (Ver [CAM-75] y [CAM-81]) pero que puede usarse también en solitario.

Este método supone que tras el proceso de estandarización o normalización, las palabras a comparar no contienen espacios en blanco ni enmedio ni a su izquierda y que el final de palabra se reconoce por un espacio.

Además de esta normalización, y dependiendo de la aplicación podrán hacerse también transformaciones fonéticas, pero se recomienda que en ellas sólo se hagan equivalentes los caracteres que siempre suenan de forma idéntica (o casi). Por ejemplo, en una aplicaciónm en la que se trate con apellidos, en un contexto mixto castellano-catalán, se podrín hacer las siguientes transformaciones: V→B, Z→SS, Q→G, Ñ→N.

Si llamamos K y C las dos palabras a comparar, y adoptamos la convención de indicar por K (i) y C (i) el caracter iésimo de K y C respectivamente, la idea básica del algoritmo puede expresarse como sigue:

–Si C(i) es igual a K(i), se decrementa el nivel de un marcador» (que inicialmente contiene cero) en una cantidad prefijada *DECRE.*

–Si C(i) no es igual a K(i), se comprueba si es igual a K(i-1) o a K(i+1) o a K(i+2). Si se cumple alguna de estas tres igualdades, no se altera el marcador y se desplaza la palabra K para que coincidan los caracteres iguales en la misma posición.

–Si C(i) no es igual a ninguno de esos cuatro (K(i-1) hasta K (i+2) se añaden unos puntos *INCRE,* al marcador.

Ejemplo: Sean BLAS y VAZ las palabras a comparar. Tras las transformaciones fonéticas que convierten VAZ en BAS, tienen lugar las comparaciones siguientes:

	1.º	2.º	3.º	4.º	5.º
BLAS	B	L	B	A	S
BAS	B	A	A	A	S
	=	≠	≠	=	=
	↓				↓
	restar				restar

Al final de ese proceso se aceptan K y C como semejantes si la puntuación, *PUNTOS,* del marcador no llega a un cierto valor, *LIMITE.*

Es evidente que el procedimiento no es simétrico. Si en un primer paso se ve que las palabras K y C no son semejantes, habrá que intercambiar las mismas (K↔C) y hacer un segundo paso.Dos palabras se definen como *no semejantes,* solamente cuando este resultado se ha obtenido *en los dos* pasos.

La figura 6 contiene un organigrama del algoritmo (en realidad, de sólo un paso). El algoritmo precisa que las palabras estén encuadradas a la izquierda, con espacios a la derecha, en campos que tengan como mínimo la longitud de la más larga más dos.

Por supuesto, los parámetros LIMITE, INCRE y DECRE se pueden ajustar según las necesidades. Se han obtenido buenos resultados prácticos, en la comparación de apellidos, con los siguientes valores: INCRE = 3, DECRE = 1 y LIMITE entre 3 y 5. Veamos dos ejemplos de comparación con esos valores de los parámetros (LIMITE = 4).

K–GARCIA	G	A	R	C	R	I	A	I	C	A	b
	G	A	R	S	S	S	S	A	A	A	b
C–GARSA	=	=	=	≠	≠	≠	≠	≠	≠	=	=
	-1	-1	-1				+3				

PUNTOS = -1 -1 -1 +3 = 0 < 4 => Semejantes

9. METODO ALBERGA-25

Alberga [ALB-67] hizo un estudio de 65 métodos de comparación, casi todos cuantativos y midió la eficacia de los mismos tomando como base un estudio sobre errores cometidos por colegiales del estado de Iowa (USA). Aunque no es evidente que sus conclusiones sean aplicables a España, presentamos a continuación el método que a él le da mejores resultados, al que denominamos método Alberga-25.

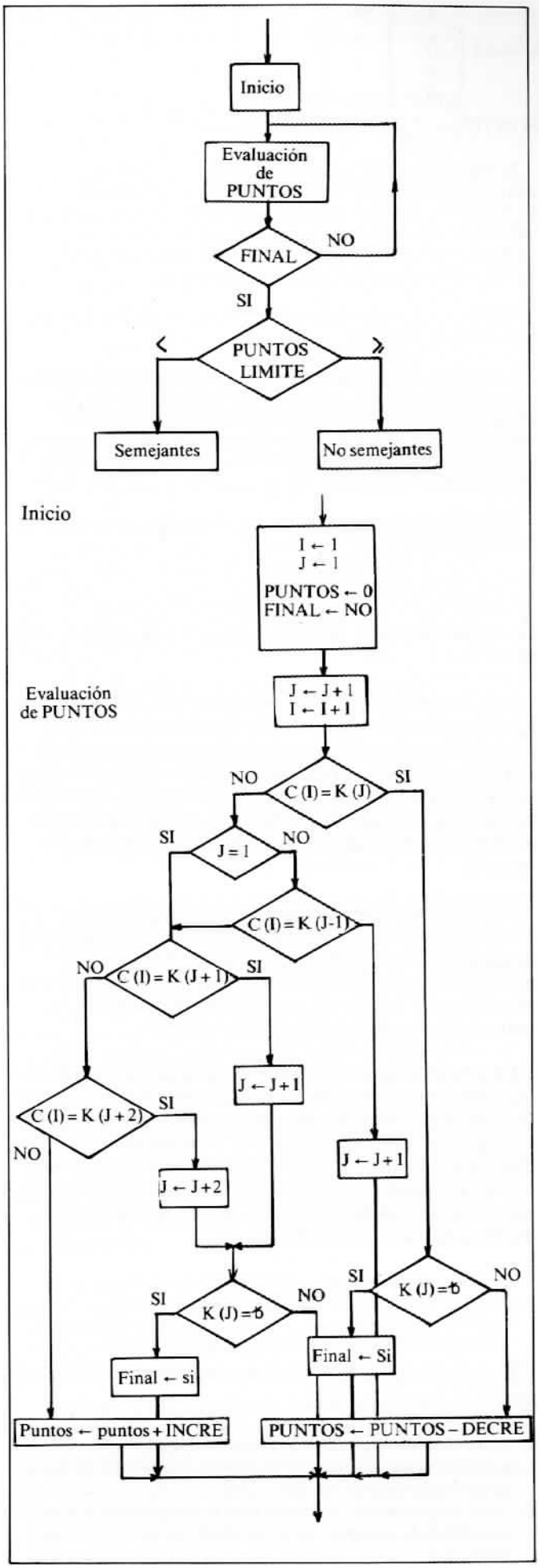

Figura 6

K—A	A		$\not\sigma_1$		A		$\not\sigma_2$		$\not\sigma_3$		$\not\sigma_2$
C—AB	A		B		B		B		B		$\not\sigma$
	=		$\neq$		$\neq$		$\neq$		$\neq$		=
	−1								+3		

PUNTOS = − 1 + 3 = 2 < 4 ⇒ Semejantes

El algoritmo empleado puede describirse como un conjunto de operaciones a realizar sobre una *matriz de coincidencia*. En una matriz de coincidencia el elemento (i, j) vale uno o cero según que el carácter i de la 1.ª palabra coincida o no con el caracter j de la 2.ª. Ver por ejemplo la figura 7.ª (en ella se han omitido los ceros).

El algoritmo consta de tres fases: Ponderación, Selección y Cálculo de la distancia.

La división del algoritmo en tres fases de operaciones sobre una matriz de coincidencia no es privativa de Alberga-25, sino que es el esquema que Alberga utilizó para describir varios métodos de comparación. Son muchos los algoritmos que pueden adaptarse a tal esquema. Las tres fases llevan a cabo las tareas siguientes:

En la fase de *Ponderación* se reemplazan los valores (i, j) = 1 de la matriz por

$$1 - \left| \frac{i-j}{m-1} - \frac{j-1}{n-1} \right|$$

siendo m la longitud de la 1.ª palabra y n la de la 2.ª. Ver figura 7b.

En la fase de *Selección* se transforma la matriz de modo que no haya columnas ni filas con más de un elemento no nulo. Para ello se empieza seleccionando el mayor elemento de la 1.ª fila y se suprimen el resto de elementos correspondientes a su fila y columna. A continuación se selecciona el mayor de los elementos que quedan en la fila 2.ª y se suprimen el resto de elementos de su fila y columna, y así sucesivamente. Ver figura 7c.

Hacemos notar que esta fase es asimétrica (fig. 7c). La asimetría puede ser muy notable. Por ejemplo, comparando RARREROTI con RARRERO da un distancia de 0,843 mientras que al revés la distancia resultante es 0,429. Alberga propone que se tome como la 1.ª palabra la «correcta», lo cual sólo tiene sentido en algunas aplicaciones.

En la ultima fase se efectúa el *Cálculo de la distancia* de la siguiente forma (figura 7d). Para cada conjunto de elementos no nulos diagonalmente consecutivos se hace una suma acumulativa. Así, en un conjunto de n elementos diagonalmente consecutivos, el 1.º (el de menores i j) se suma n veces, el segundo n-1 veces, etc. Si llamamos S a la suma de esas sumas y la longitud de la mayor de las dos palabras, la distancia normalizada se define como:

$$d = 1 - \frac{2S}{l^2 + l}$$

De la forma que se ha concebido este algoritmo se consigue que:

a) A las coincidencias de dos caracteres (entre las dos palabras se les de menor importancia cuanto más se alejan de posición (fase de Ponderación).
b) Para cada caracter de una palabra se tenga en cuenta su coincidencia con otro, como máximo, de la otra (fase de Selección).
c) Los caracteres coincidentes, formando grupos, cuenten tanto más cuanto al principio del grupo estén, primándose al mismo tiempo la longitud de los grupos (fase de Cálculo de l distancia).

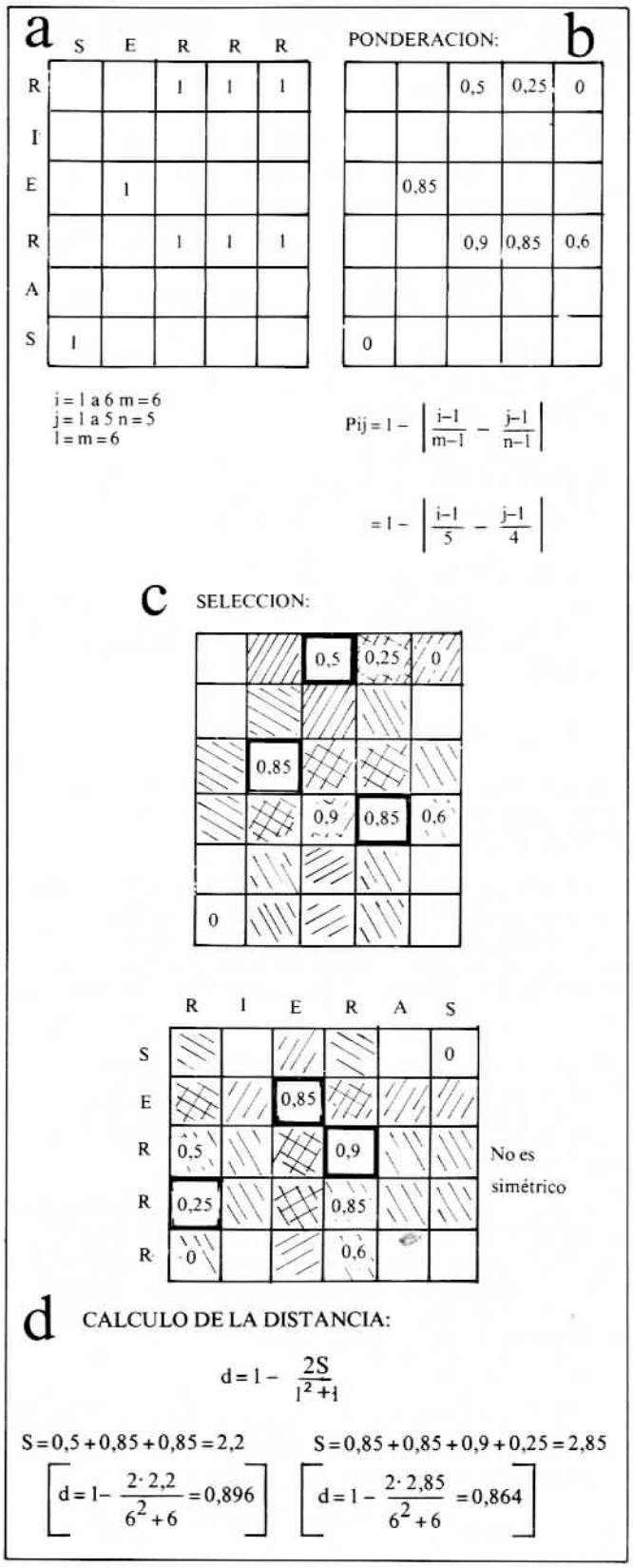

Figura 7

Convendría incluir una fase previa de transformaciones en las que se haga: V→B, Z→S, J→G.

Como Alberga-25 es asimétrico y en la mayoría de

aplicaciones no hay una palabra que pueda considerarse como «correcta», se pueden calcular las dos distancias y escoger la más pequeña.

Para disponer de un criterio cualitativo se puede fijar un umbral, que la experiencia aconseja situar entre 0,750 y 0,820.

10. DISTANCIA DE WAGNER

Hemos visto en 7 el algoritmo de Morgan-Damerau para detectar si dos palabras son semejantes según el criterio de Damerau, o sea si se diferencian en una y sólo una de las operaciones siguientes:

–Inserción de un carácter
–Supresión de un carácter
–Sustitución (o cambio) de un caracter
–Transposición de dos caracteres adyacentes.

Wagner y Fischer [WAG-74] llamaron a esas cuatro operaciones, *operaciones «de edición»* (y definieron una distancia muy general basándose en las mismas. Al mismo tiempo propusieron un algoritmo para el cálculo de tal distancia.

A cada operación de edición se le asocia un *coste*, un número real no negativo, que puede ser función de los símbolos (caracteres) involucrados en la operación. Por ejemplo: la sustitución de la letra M por N podrá tener un coste inferior a la sustitución de M por R.

Para transformar una palabra A cualquiera en otra B, hará falta una secuencia de operaciones de edición, pero habrá muchas secuencias válidas para una misma transformación A → B. A cada secuencia puede asociarsele un coste, suma de los costes de las operaciones que la componen. Llamaremos *distancia de Wagner* entre A y B, al coste *mínimo* de los asociados con todas las secuencias posibles que transforman A en B.

En el algoritmo propuesto en [WAG-74] las únicas operaciones de edición consideradas fueron la *inserción*, la *supresión*, la *sustitución*.

En [SHI-78] se extiende el algoritmo (en un contexto de «reconocimiento de formas») de forma que el coste de la operación de inserción de un determinado carácter dependa del carácter delante del cual se inserta.

En [LOW-75] se extiende el algoritmo incluyendo la operación de *transposición* (pero se supone constante el coste de cada operación).

A continuación damos una versión del algoritmo inicial de Wagner, que no considera la operación de transposición. En la descripción del mismo usamos la siguiente notación:

- Las palabras a comparar son A y B y sus longitudes respectivas son L_A y L_B
- A(I) y B(J) son, respectivamente, el carácter I-ésimo de la palabra A y el carácter J-ésimo de la palabra B.
- SUPER (x) es el coste de la supresión del signo o carácter x.
- INSE (x) es el coste de la inserción del signo o carácter x.
- SUST (x_1, x_2) es el coste de la sustitución del signo o carácter x_1 por el x_2.

El algoritmo utiliza como área de trabajo, una matriz COST ($L_A + 1$, $L_B + 1$).

Algoritmo de Wagner

1) COST (1, 1) ← 0
2) Ejecutar para I ← 1 a L_A:
   COST (I + 1, 1) ← COST (I, 1) + SUPR (A (I))
3) Ejecutar para J ← 1a L_B:
   COST (1, J + 1) ← COST (1, J) + INSE (B (J))
4) Ejecutar para I ← 1 a L_A:
   Ejecutar para J ← 1 a L_B:

   WI ← COST (I, J) + SUST (A (I), B (J))
   W2 ← COST (I, J + 1) + SUPR (A (I))
   W3 ← COST (I + 1, J) + INSE (B (J))
   COST (I + 1, J + 1) ← Min (W1, W2, W3)
5) DISTANCIA ← COST ($L_A + 1$, $L_B + 1$)

Evidentemente, una cuestión importante es qué valores adoptar para los costes SUP, INSE y SUST.

Para determinar valores adecuados convendría disponer de datos sobre frecuencia de errores de cada uno de los tres tipos. Los costes deberían ser inversamente proporcionales a las frecuencias. Esas frecuencias podrán ser diferentes según las aplicaciones.

Como en la tabla de costes podemos asociar un coste nulo (o casi) al cambio de caracteres casi idénticos como por ejemplo B y V, no nos serán necesarias las transformaciones previas de esos caracteres. De todas maneras puede seguir interesándonos hacer transformaciones de grupos de caracteres, por ejemplo CH → X.

Una simplificación del algoritmo consiste en atribuir un costo 1 a cualquier operación de edición. Entonces la distancia se hace igual al número mínimo de operaciones de edición necesarias para pasar de una palabra a la otra. En este caso, antes de la evaluación del costo convendrá efectuar las transformaciones siguientes: B → V, J → G, Z → S. Es una distancia métrica. Si a esa distancia le fijamos un umbral de 2, estaremos aceptando las mismas diferencias que Damerau y Morgan (Ver 7).

11. TECNICAS DE BUSQUEDA PARA LOS METODOS DE COMPARACION

Para buscar, en un fichero determinado, todas las palabras semejantes a una dada usando un método de comparación parece que, en principio, tendremos que recorrer una a una todas las palabras del fichero. Esto sólo es aceptable en contadísimas aplicaciones, como por ejemplo en la reserva de plazas de avión u hotel, en donde la búsqueda suele abarcar sólo algunos centenares de personas. En un fichero con 100.000 personas, la consulta total es impracticable. Si tenemos en cuenta que en ese fichero el número de apellidos diferentes será de unos 13.000, podríamos estructurar el fichero con un índice de apellidos, pero aun en ese caso habría que leerse y comparar 13.000 apellidos lo cual sigue siendo excesivo.

En el caso de la codificación fonética, basta tener estructurado el fichero en partes, en las partes que el propio método determina; e ir directamente a la (única) parte que contiene las palabras semejantes (todas las palabras de esa parte lo son).

En el caso de los métodos de comparación podemos pensar en tener también el fichero estructurado en partes y en el momento de la búsqueda empezar por una fase de preselección en la que se descartan aquellas partes en las que no es posible que existan palabras semejantes a la dada. Entonces la comparación la haremos solamente con las palabras de aquellas partes preseleccionadas. Pero esta estructura en partes sólo es fácil encontrarla en casos muy simples. Veamos un ejemplo.

Si adoptamos el criterio de semejanza de Morgan-Damerau (7), (sólo una supresión/inserción/sustitución/transposición) se puede tener el fichero dividido en partes según las longitudes y los dos primeros caracteres. A cada parte se le puede asociar una clave compuesta así: $1 c_1 c_2$. Entonces puede usarse un «índice de partes» que para cada clave nos lleve a la parte correspondiente.

Habrá $8 \times 26 \times 27 = 5.616$ partes (aunque gran número de ellas estarán vacías).

Por ejemplo, la clave 6BL nos llevaría a la parte en la que figuran los apellidos de longitud 6 con el primer carácter igual a B y el segundo igual a L.

Si buscáramos los apellidos semejantes a BLASCO, deberíamos preseleccionar las siguientes partes que son las únicas que los pueden contener:

5BL, 6BL, 7BL
5Bx, 6Bx, 7Bx } Para todos los valores posibles de x.
5Lx, 6xL }
6LB, 7LB

En total hay que buscar en 131 partes (muchísimas estarán vacías) o sea 1/43 del número total de partes.

En el fichero de 100.000 personas citado anteriormente, para buscar los semejantes a BLASCO hay que consultar $1/43 \times 13.000 = 300$ apell. aprox. lo cual es ya aceptable.

En [CAM-81] se exponen técnicas de búsqueda para los casos en que la distancia cumple la ley triangular.

12. METODOS COMPUESTOS

Para conseguir reducir la sobreidentificación o la subidentificación, se pueden combinar varios métodos por medio de combinaciones lógicas.

La condición Y (*a* y *b* se aceptan como semejantes sólo en el caso en que sean considerados semejantes por los métodos M1 y M2) reduce la sobreidentificación aunque aumenta la subidentificación. La condición 0 (*a* y *b* se aceptan como semejantes en el caso en que sean considerados semejantes por los métodos M1 o M2) reduce la subidentificación aunque aumenta la sobreidentificación.

De hecho, ya hemos visto en 5 un ejemplo en el que se usa como filtro adicional para reducir la sobreidentificación del método SONS, el método FERM. En este caso se combina en condición Y un método de orientación «fonética» con otro de orientación «ortográfica».

13. DICCIONARIO

En muchas aplicaciones (Ejemplo: callejeros), se desea reconocer también como semejantes variantes que no son ortográficas o fonéticas, sino sinónimos. Así por ejemplo: GRAN VIA / JOSE ANTONIO, PEPE / JOSE, RUSIA / UNION SOVIETICA / URSS, etc. En estos casos se puede introducir en el método la consulta de un diccionario de sinónimos en el que figuren todas las variantes conocidas.

De hecho, existen algunos sistemas en los que se usa un diccionario no sólo para los sinónimos sino para cualquier tipo de variante o error. Las grandes ventajas de esta solución son la flexibilidad y la rapidez (el diccionario puede ser de acceso por cálculo de dirección o por índice).

Estos diccionarios se suelen ir ampliando (dinámicamente por ejemplo, de forma conversacional) a medida que se van detectando variantes o errores aún no existentes en el diccionario.

Finalmente, es interesante notar que se puede aprovechar también la técnica del diccionario, como auxiliar de un método de comparación para errores conocidos pero no resueltos por el algoritmo utilizado.

Rafael Camps

APENDICE

Fe de erratas

Aunque con muchísimo retraso, aprovecho la ocasión para corregir algunos errores importantes que se deslizaron en el artículo anterior [CAM-75].

identificación	*Debe decir*
regla E1	Si sólo hay una letra o ninguna o sólo hay vocales o se trata de una o varias vocales seguidas tan sólo de una consonante => XX Ej.: T => XX AOT => XX
pág. 8 col 1 línea 25	74
pág. 8 col 1 línea 27	$74 \times 74 = 5.476$

BIBLIOGRAFIA

ACH - 68. ACHESON, E. D. *Record linkage in medecine.* Ed: E & Livingstone Ltd. 1968.

ALB - 67. ALBERGA C. N. «String similarity and mispellings». *Comm. ACM 10, 5* (mayo 1967).

BLA - 60- BLAIR, C. P. «A program for correcting spelling errors». *Infor. Control 3* (1960).

BUR - 76. BURKHARD, W. A. «Hashing and trie algorithms for partial match retrieval». *TODBS-ACM 1, 2* (junio 1976).

CAM - 75. CAMPS, R. y CASAS, R. «Codificación fonética de apellidos españoles». *Novática 6* (Nov. Dic. 1975).

CAM - 81. CAMPS, R. *Búsqueda por semejanza ortográfica o fonética.* Facultad de Informática de Barcelona. (octubre 1981).

DAM - 64. DAMERAU, F. J. «A technique for computer detection and correction of spelling errors». *Comm. ACM 7, 3* (marzo 1964).

DAV - 62 DAVIDSON, L. «Retrieval of mispelled names in an airlines passenger record system». *Comm. ACM 5, 3* (marzo 1962).

DOL - 70. DOLBY, J. L. «An algorithm for variable-length proper-name compression». *Journ. Library Autom. 3, 3* (setiembre 1970).

GLA - 57. GLANTZ, H. T. «On the recognition of information with a digital computer». *J. ACM 4* (1957).

HALL - 80. HALL, P. A. V. & DOWLING, G. R. «Approximate String Matching». *Compunting Surveys. ACM 12, 4* (diciembre 1980).

KNU - 75. KNUTH, D. E. *The art of computer programming. Vol 3 Sorting and Searching.* Ed. Addison Wesley. (2.ª edición 1975).

LOW - 75. LOWRANCE, R. & WAGNER, R. A. «An Extension of the String-to-String Correction Problem». *Journ. ACM 22, 2* (1975).

MOR - 70. MORGAN, H. L. «Spelling Correction in Systems Programs». *Comm. ACM 13, 2* (febrero 1970).

ROD - 77. RODRIGUEZ, H. «Técnicas de búsqueda». *Novática 17* (Set. oct. 1977).

SEV - 74. SEVERANCE, D. G. «Identifier search mechanisms: A survey an generalized method» *ACM Comp. Surveys 6, 3* (set. 1974).

SHI - 78. SHIN-YEE LU & KING SUN FU = «A sentence-to-sentence Clustering Procedure for Pattern Analysis». *IEEE Transactions on Sys. Man and Cybern, Vol SMC-8 n.º 5* (Mayo 1978).

WAG - 74. WAGNER, R. A. & FISCHER, M. J. «The String-to-String Correction Problem». *Journ. ACM 21, 1* (1974).