Universitat d'Alacant
Universidad de Alicante
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Detección de regularidades en contornos 2D, cálculo aproximado de medianas y su aplicación en tareas de clasificación.

José Ignacio Abreu Salas

Memoria para optar al grado de Doctor en Informática bajo la dirección de:

Dr. Juan Ramón Rico Juan

Alicante, 2012

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Detección de regularidades en contornos 2D, cálculo aproximado de medianas y su aplicación en tareas de clasificación.

José Ignacio Abreu Salas

Memoria para optar al grado de Doctor en Informática bajo la dirección de:

Dr. Juan Ramón Rico Juan

Alicante, 2012

Este trabajo ha sido desarrollado bajo el auspicio del CICYT, España, dentro del proyecto DPI2006-15542-C04-01, el MCINN a través del proyecto TIN2009-14205-CO4-01 y por el programa Consolider Ingenio 2010: MIPRCV (CSD2007-00018).

Resumen

El trabajo se enfoca en dos problemas principales: la identificación de similitudes y la obtención del promedio para contornos representados mediante cadenas de Freeman. Estos problemas son abordados empleando información obtenida en el cálculo de la distancia de Levenshtein entre las cadenas que codifican a los contornos.

En la tesis se presenta un nuevo método para cuantificar la regularidad entre los contornos y establecer una comparación en términos de un criterio de similitud. El criterio empleado permite encontrar subsecuencias de la cadena de operaciones de edición con coste mínimo que especifican un alineamiento entre pares de segmentos, uno en cada contorno, que se consideran semejantes de acuerdo a dos parámetros definidos externamente. La información sobre las partes similares en cada contorno queda codificada por una cadena que representa el "promedio" de los segmentos. A partir de la identificación de las similitudes, se define un procedimiento para construir un prototipo que representa a los contornos. Para evaluar que tan representativa es esta instancia se sustituyen grupos de contornos por su prototipo en el conjunto de entrenamiento de un clasificador K-NN. De este modo se logra reducir la talla del conjunto de entrenamiento sin afectar sensiblemente su poder de representación. Resultados experimentales muestran que este procedimiento es capaz de alcanzar reducciones del conjunto de entrenamiento cercanas al $80\,\%$ mientras el error en la clasificación solo aumenta en un $0.45\,\%$ en una de las bases de datos estudiadas.

Por otra parte, se propone un nuevo algoritmo para el cálculo rápido de una aproximación a la media entre dos cadenas representando un con-

torno 2D, así como una implementación voraz que permite reducir significativamente el tiempo necesario para construir la aproximación al contorno medio. Éste se combina junto a un nuevo método de edición que suaviza las restricciones impuestas por el algoritmo de Wilson para desechar una instancia. En la práctica, no todas las instancias mal clasificadas por sus vecinos son eliminadas. En su lugar, una instancia artificial es añadida al conjunto esperando que de este modo la instancia se clasifique correctamente en el futuro. La instancia artificial es la media entre la instancia mal clasificada y su vecino más próximo de igual clase. Los experimentos desarrollados empleando tres conocidos conjuntos de contornos denotan que los algoritmos propuestos mejoran los resultados de otros métodos descritos en la literatura para el cálculo del promedio de dos contornos. El bajo coste computacional de la implementación voraz la hace muy adecuada para cadenas de Freeman de gran longitud. Resultados empíricos demuestran además que mediante el procedimiento de edición descrito es posible reducir el error en la clasificación en un 83 % de los casos, independientemente del algoritmo empleando para obtener el contorno promedio.

Finalmente, se describe un algoritmo para construir una aproximación a la media de un conjunto de cadenas, la que se obtiene a través de sucesivas mejoras a una solución parcial. En cada iteración se calcula la distancia de la solución parcial a cada cadena del conjunto llevando cuenta de la frecuencia de las operaciones de edición en cada posición de la media aproximada. Esta información permite calcular un *índice de calidad* al multiplicar la frecuencia por el coste de la operación. Cada operación es evaluada, comenzando por aquella con mejor calidad, para valorar si su aplicación conduce a una mejora. En caso afirmativo, se comienza una nueva iteración a partir de la nueva solución. El algoritmo concluye después de examinar todas las operaciones sin que se logre una mejora. Los experimentos comparativos utilizando cadenas de Freeman muestran que se pude obtener aproximaciones equivalentes a otros enfoques pero en menor tiempo.

Summary

In this work, we address two main problems: identifying the regularities and the construction of average contours from contours encoded by Freeman chain codes. Solutions for those problems are proposed which relies in the information gathered from the Levenshtein edit distance computation.

We describe a new method for quantifying the regularity of contours and comparing them, when encoded by Freeman chain codes, in terms of a similarity criterion. The criterion used allows subsequences to be found from the minimal cost edit sequence that specifies an alignment of contour segments which are similar. Two external parameters adjust the similarity criterion. The information about each similar part is encoded by strings that represent an average contour region. An explanation of how to construct a prototype based on the identified regularities is also reviewed. The reliability of the prototypes is evaluated by replacing contour groups, samples, by new prototypes used as the training set in a classification task. This way, the size of the data set can be reduced without sensibly affecting its representational power for classification purposes. Experimental results show that this scheme achieves a reduction in the size of the training data set of about $80\,\%$ while the classification error only increases by $0.45\,\%$ in one of the three data sets studied.

Also this thesis presents a new fast algorithm for computing an approximation to the mean between two strings of characters representing a 2D shape and its application to a new Wilson-based editing procedure. The approximate mean is built by including some symbols from the two original strings. Besides, a greedy approach to this algorithm

is studied which allows to reduce the time required to computed an approximate mean. The new dataset editing scheme relaxes the criterion for deleting instances proposed by the Wilson editing procedure. In practice, not all instances misclassified by their near neighbors are pruned. Instead, an artificial instance is added to the dataset in the hope of successfully classifying the instance in the future. The new artificial instance is the approximated mean of the misclassified sample and its same-class nearest neighbor. Experiments carried over three widely known databases of contours show the proposed algorithms performs very well in computing the mean of two strings, outperforming methods proposed by other authors. Particularly the low computational time required by the heuristic approach make it very suitable when dealing with long length strings. Results also shows the proposed preprocessing scheme can reduce the classification error in about $83\,\%$ of trials. There is empirical evidence that using the greedy approximation to compute the approximated mean does not affect the editing procedure performance.

Finally, a new algorithm with which to compute an approximation to the mean of a set of strings is presented. The approximated mean is computed through the successive improvements of a partial solution. In each iteration, the edit distance from the partial solution to all the strings in the set are computed, thus accounting for the frequency of each of the edit operations in every position of the approximated mean. A goodness index for edit operations is later computed by multiplying their frequency by the cost. Each operation is tested, starting from that with the highest index, in order to verify whether applying it to the partial solution leads to an improvement. If successful, a new iteration begins from the new approximated mean. The algorithm finishes after all the operations have been examined without a better solution being found. Comparative experiments involving Freeman chain codes encoding 2D shapes show that the quality of the approximated mean string is similar to other approaches but achieves a much faster convergence.

Índice

Índice de tablas

Índice de figuras

Parte I

Introducción

Capítulo 1

Introducción

1.1. Motivación

Dentro del reconocimiento de formas, la clasificación y recuperación de patrones visuales - entre otras tareas - constituyen áreas en constante expansión, como evidencian recientes trabajos al respecto (Han-fa *et al.*, 2010; Salleh *et al.*, 2011; Ozkan *et al.*, 2011). Este hecho no es muy sorprendente si se tiene en cuenta la enorme cantidad de materiales visuales disponibles en Internet o la importancia de este tema en el desarrollo de la robótica - fundamentalmente en la visión artificial - o en el etiquetado e interpretación automáticos de imágenes por solo mencionar algunos ejemplos.

Concretamente, el reconocimiento de objetos basado en su figura encierra particular interés, por ser un atributo de la imagen muy importante en la percepción humana como apuntan Zhang *et al.* (2006) y Kontschieder *et al.* (2010). La literatura recoge una copiosa cantidad de artículos y libros donde se detallan técnicas que realizan la representación y comparación de los objetos basados en el contorno, a modo de ejemplo baste mencionar a van Otterloo (1988), Ghosh & Deguchi (2008) o Bai *et al.* (2008)

Al igual que en muchos otros problemas de reconocimiento de formas, una cuestión fundamental a resolver es la creación o selección

del formalismo con el que se representarán computacionalmente los patrones tratados, contornos en este caso. Específicamente en las tareas de clasificación, otro aspecto vital es el diseño del clasificador, proceso en gran medida dependiente de la codificación empleada.

Para resolver el primer punto, se han propuesto buena cantidad de métodos que persiguen describir los contornos con mayor o menor detalle, o lograr robustez ante deformaciones elásticas y geométricas entre otros aspectos. Atendiendo a su diseño estos pueden clasificarse en dos grandes campos, las representaciones globales como la Transformada de Fourier o la de Wavelet y las estructurales (Zhang & Lu, 2004). Dentro del segundo grupo, las cadenas de Freeman (Freeman, 1974) o *chain codes* son una técnica muy empleada directamente o como base para otro tipo de representaciones, como muestran los trabajos de Zhou & Zahir (2006) y Nallaperumal *et al.* (2006). Esta técnica tiene varias ventajas como permitir una codificación compacta de los contornos, al mismo tiempo de ser una representación completa, es decir, es posible obtener cualquier rasgo del contorno a partir de la cadena (Jusoh & Zain, 2011).

La otra cuestión, referida al método de clasificación empleado, ha sido también un tema profusamente abordado desde los inicios mismos de la ciencia de la computación. La literatura describe una buena cantidad de algoritmos o modelos diferentes, pasando por las Redes de Neuronas Artificiales, las Máquinas de Soporte Vectorial, Métodos Bayesianos, Gramáticas y la regla de los K-vecinos más cercanos o K-NN (Duda & Hart, 2001). Sobre esta última técnica debe acotarse que aun cuando tiene un planteamiento bien sencillo puede mostrar un excelente desempeño; razón por la que continúa siendo ampliamente estudiada, sirviendo de base al desarrollo de nuevos métodos de clasificación (Moreno-Seco *et al.*, 2004; Angiulli, 2005). Específicamente en el contexto de la clasificación y recuperación de imágenes, ha sido empleada con resultados muy satisfactorios como muestran los experimentos realizados por Michie *et al.* (1994).

Una vez definidos estos dos puntos, nos encontramos con varias cuestiones no menos importantes que inciden notablemente en la calidad

de la solución dada al problema. En este sentido puede mencionarse la conveniencia de contar con un número de ejemplos de entrenamiento lo más reducido posible, pero que describa adecuadamente las clases de objetos que se desea aprender (Wilson, 1972; Angiulli, 2007). Por otra parte, en varios contextos la problemática no se limita a obtener buen desempeño en tareas de clasificación, sino que es conveniente tener algún criterio *interpretable* - es decir comprensible para los humanos (Wang & Fu, 1998) - que justifique los resultados, como sería que la clasificación se realiza teniendo en cuenta los elementos comunes a las instancias de una clase de contornos (Duta *et al.*, 1999). En el primer caso, y como se ilustra en los tempranos trabajos de Wilson (1972), se evidencia que el desempeño de un clasificador resulta afectado no solo en el tiempo de respuesta sino en la disminución del error cometido.

Ciertamente, más allá de su amplia influencia en problemas de clasificación, estas interrogantes han sido abordadas desde otras aristas, como los trabajos de Li *et al.* (1999) para la diferenciación de regiones similares en cadenas de ADN o por Duta *et al.* (2001) para construir modelos de varias estructuras del cerebro estudiadas a través de su imagen obtenida por Resonancia Magnética. Trabajos más recientes (Cárdenas, 2004; Marzal *et al.*, 2006; Bai *et al.*, 2008) ilustran el actual interés de la comunidad científica en responder a las cuestiones planteadas. Un motivo fundamental es que el método a desarrollar está en buena medida determinado por las características de la representación computacional empleada para describir al objeto en cuestión. En el caso de las representaciones vectoriales una alternativa consiste en sustituir un grupo de instancias por el vector centroide. El grupo de objetos a sustituir puede ser definido de antemano o mediante algún método de agrupamiento como el algoritmo K-Medias. Sin embargo, para las representaciones estructurales es habitual que las operaciones aritméticas carezcan de sentido. En este caso, es común encontrar enfoques lingüísticos, es decir, mediante técnicas de inferencia se trata de construir un modelo capaz de reconocer el lenguaje formado por las representaciones correspondientes a elementos de una misma clase. Esta gramática o autómata, entre otras alternativas, serviría para describir todo un grupo de objetos.

Particularmente cuando los elementos quedan codificados mediante cadenas una posibilidad consiste en encontrar una instancia - pudiendo esta pertenecer o no al conjunto original - tal que sea razonable esperar que el conjunto es caracterizado por este elemento. En esta línea, el cálculo de la llamada *cadena media* constituye una alternativa estudiada desde hace bastante tiempo. Esta es, de todas las cadenas posibles, aquella con la menor distancia acumulada al resto de los objetos en el conjunto. Sin embargo se ha demostrado que en general este es un problema NP-Completo (Casacuberta, 2000), por lo que se han descrito diferentes aproximaciones, entre las cuales la más simple es la llamada *cadena mediana* que no es más que la cadena perteneciente al conjunto con la menor distancia promedio al resto de los elementos. Para obtener aproximaciones más precisas se han propuesto diferentes heurísticas como las descritas por Martínez-Hinarejos *et al.* (2003) y Jiang *et al.* (2011) entre otros.

En el ámbito de este trabajo, los problemas de identificación de elementos similares entre contornos y el cálculo de un contorno prototipo son sustituidos por los equivalentes: identificación de secuencias similares entre dos cadenas y cálculo de la cadena media.

La propia definición de cadena media requiere establecer la noción de distancia o *similitud* entre los elementos, una función análoga a la operación de resta aritmética. Para este propósito se han descrito diferentes propuestas, siendo un enfoque bastante extendido calcularla en términos de ciertas operaciones que permiten transformar una cadena en otra (Levenshtein, 1966; Winkler, 1999; Arslan, 2006). De manera general, cada operación conlleva un coste o penalización por lo que dada una secuencia de operaciones a esta se podrá asociar también un coste. El valor de la distancia quedará fijado por el coste de la secuencia de operaciones con mínima penalización. Comúnmente los algoritmos para el cálculo de la distancia permiten no solo obtener el valor numérico correspondiente sino también la secuencia de operaciones con coste mínimo.

En general los métodos descritos para obtener la cadena media limitan el uso de la distancia a la fase de evaluar la calidad de la aproxima-

ción obtenida. Sin embargo creemos que la información que aporta el cálculo de la distancia puede ayudar a diseñar soluciones a los problemas anteriores. En la tesis se recogen diferentes experimentos que sustentan esta hipótesis.

1.2. Objetivos.

En resumen, la tesis se plantea como cuestión principal desarrollar algoritmos para la detección de similitudes y el cálculo de aproximaciones a la media entre contornos representados mediante cadenas de Freeman. Específicamente, los objetivos de la tesis se pueden sintetizar en:

- Definir algoritmos que permitan identificar similitudes entre dos contornos representados mediante cadenas de Freeman.
- Proponer algoritmos que proporcionen una buena aproximación a la media entre dos contornos representados mediante cadenas de Freeman.
- Describir algoritmos que proporcionen una buena aproximación a la media entre varios contornos representados mediante cadenas de Freeman.
- Definir algoritmos iterativos que permitan obtener una aproximación a la media de un conjunto de cadenas.
- Aplicar las aproximaciones a la media entre dos contornos en algoritmos para reducir la talla y filtrar instancias ruidosas del conjunto de entrenamiento de un clasificador K-NN.
- Comparar la calidad de la aproximación a la media entre dos contornos con otros enfoques descritos en la literatura al respecto.

- Evaluar la efectividad de los algoritmos de reducción y filtrado de instancias ruidosas del conjunto de entrenamiento de un clasificador K-NN.

- Comparar la calidad de la aproximación a la cadena media obtenida con los resultados que alcanzan otros algoritmos descritos en la literatura.

1.3. Metodología

Para conseguir los objetivos propuestos se propone cumplir las etapas que a continuación se describen:

- Estudio previo: Se analizarán las principales propuestas referidas a los problemas tratados, más concretamente:

  – Algoritmos para el cálculo de la cadena media: Se estudiarán las propuestas recogidas en la literatura atendiendo al coste computacional, capacidad de funcionamiento incremental, aplicación al cálculo del contorno promedio y métodos de validación empleados.

  – Algoritmos para la detección de similitudes entre contornos: Se hará énfasis en aquellos que utilizan representaciones estructurales.

  – Métodos de selección de instancias: Se clasificarán atendiendo a si eliminan o no instancias ruidosas, analizándose las diferencias, las ventajas e inconvenientes y aplicación a instancias representadas mediante cadenas.

- Diseño e implementación de algoritmos que permitan identificar similitudes entre dos contornos representados mediante cadenas de Freeman a partir del análisis de la información que aporta el cálculo de la distancia entre las cadenas.

- Diseño e implementación de algoritmos para la obtención de aproximaciones a la media entre dos contornos representados mediante cadenas de Freeman. En esta etapa se propondrán nuevos algoritmos que utilicen la información que aporta el cálculo de la distancia para mejorar la calidad de la aproximación y disminuir el tiempo de cómputo necesario.

- Diseño e implementación de algoritmos iterativos para la obtención de aproximaciones a la cadena media. En esta fase serán descritos algoritmos que utilicen información estadística obtenida en el cálculo de la distancia de la solución parcial a cada elemento en el conjunto para mejorar la calidad de la solución.

- Diseño e implementación de algoritmos para la reducción y filtrado de instancias ruidosas del conjunto de entrenamiento de un clasificador. En esta fase se propondrán nuevos algoritmos que apliquen los esquemas propuestos para el cálculo de la cadena media y la identificación de similitudes.

- Realización de experimentos para validar y analizar el comportamiento de los algoritmos propuestos.

1.4. Estructura de la memoria.

El trabajo queda entonces estructurado de la siguiente manera. En las restantes secciones del presente capítulo se introducen algunas técnicas y conceptos fundamentales para la compresión del resto de la tesis. El capítulo (2) recoge una revisión de los principales métodos descritos en la literatura sobre la construcción de prototipos para un conjunto de contornos, particularmente aquellos donde se emplea alguna representación estructural. Además se repasan diferentes propuestas para el cálculo de la cadena media. En el apartado (3) se aborda el estado actual de los métodos para la selección de prototipos basados en un clasificador K-NN. Los algoritmos propuestos para la identificación de similitudes entre dos contornos representados mediante ca-

denas de Freeman, la obtención de aproximaciones al contorno y a la cadena media así como diferentes aplicaciones de estos son expuestos en los capítulos (4), (5) y (6) respectivamente.

Las conclusiones finales de la investigación se exponen en el capítulo (7) además de presentar algunas líneas a donde puede encaminarse el trabajo futuro.

1.5. Conceptos generales.

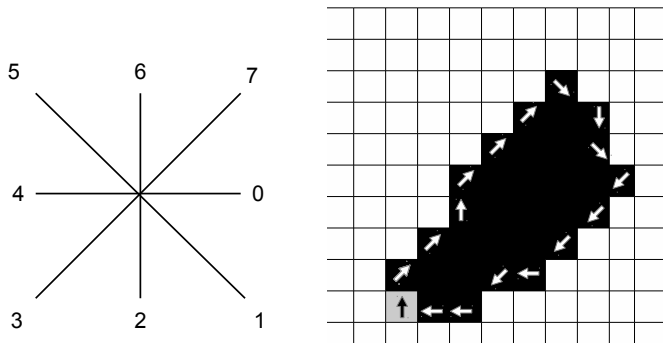
En esta sección se repasarán algunos conceptos o técnicas indispensables para una correcta exposición de los métodos propuestos en el resto de la memoria.

1.5.1. Cadenas de Freeman

Los *códigos de cadena* son una técnica ampliamente utilizada para describir el contorno de un objeto. La línea del contorno es aproximada mediante una secuencia de segmentos de dirección y longitud determinada, donde cada dirección queda codificada por un número como muestra la figura (1.1a).

La forma genérica para describir un contorno utilizando cadenas de Freeman (Freeman, 1974) consiste en dividir la imagen en cuadrados iguales de dimensión arbitraria determinada según el grado de exactitud con que se desee representar los contornos. En este caso, cada píxel de la imagen constituye un elemento de la cuadrícula. A partir de una celda C_1 específica por donde pase la línea del contorno, se busca entre los cuadros vecinos uno C_2 por donde también atraviese la línea del contorno. Suponiendo que se localice en la dirección d_1, se repite el proceso ahora desde C_2 y así sucesivamente hasta llegar nuevamente a C_1. Se asume que la región no se intercepta con los bordes de la imagen. De esta forma se obtiene la secuencia $\{d_1, d_2, ..., d_n\}$

interpretada como vectores unitarios (Zhang & Lu, 2004) dados en coordenadas polares donde la dirección es d_i, que no es más que el código de cadena que representa al contorno. El proceso se ilustra en la figura (1.1b), comenzando por el punto sombreado se obtiene la cadena $\{6, 7, 7, 6, 7, 7, 7, 1, 2, 1, 3, 3, 3, 4, 3, 4, 4\}$.



(a) (b)

Figura 1.1: Conjunto de direcciones principales para construir el código de cadena (1.1a). Construcción de la cadena de Freeman (1.1b).

A pesar de las ventajas mencionadas por Jusoh & Zain (2011) como permitir una codificación compacta y completa de los contornos, las cadenas de Freeman presentan varias limitaciones. Por ejemplo, dependiendo de las dimensiones de los elementos de la cuadrícula o de la frecuencia de muestreo sobre los puntos del contorno este tipo de representación será más o menos sensible al ruido. Por otra parte, no son robustas a transformaciones como la rotación. Derivadas de esta técnica se han descrito otras variantes que resuelven algunos de los inconvenientes, como las introducidas por Liu & Žalik (2005) o Zhou & Zahir (2006) para lograr invariancia ante la transformación de rotación.

En el trabajo, se empleará una extensión a la definición clásica de las cadenas de Freeman consistente en tomar un conjunto infinito de

direcciones en $\mathbb{R}$, tales que $0 \leqslant d < 8$, además del símbolo "?" para denotar una dirección no determinada.

1.5.2. Distancia de edición de Levenshtein

Sea $\sum$ un alfabeto que incluye el carácter nulo ε y S_1, S_2 dos cadenas sobre $\sum$; se define la distancia de edición de Levenshtein (Levenshtein, 1966) entre las cadenas en términos de operaciones de edición elementales necesarias para transformar S_1 en S_2, en lo sucesivo se denotará por $D(S_1, S_2)$. Usualmente se consideran las siguientes alternativas:

- *sustitución* de un símbolo $a \in S_1$ por un símbolo $b \in S_2$, denotado como $w(a, b)$
- *inserción* de un símbolo $b \in S_2$ en S_1, denotado como $w(\varepsilon, b)$
- *borrado* de un símbolo $a \in S_1$, denotado como $w(a, \varepsilon)$.

Sea $Q = \{q_1, q_2, ..., q_k\}$ una secuencia de operaciones de edición que convierte S_1 en S_2. Si cada operación tiene coste $e(q_i)$ el coste de Q es $E_Q = \sum_{i=1}^{k} e(q_i)$ y la distancia de edición es definida como:

$$D(S_1, S_2) = argmin_Q\{\ E_Q\}\ .$$

Dado que las cadenas utilizadas son códigos de Freeman los costes de sustitución están dados por:

$$w(a, b) = \begin{cases} min\{|a-b|, 8-|a-b|\} & \text{si} \quad a, b \neq \text{"?"} \\ 2 & \quad\ \ eoc \end{cases}$$

En el caso de las inserciones y borrados se considerará que $e(w(\varepsilon, b)) = e(w(a, \varepsilon)) = 2$. Este valor es la mitad del máximo peso de una sustitución; esta elección coincide con la utilizada por Rico-Juan & Micó

(2003). Para simplificar la notación se empleará siempre que sea posible las expresiones $w(a, \varepsilon)$, $w(\varepsilon, b)$ y $w(a, b)$ tanto para denotar la correspondiente operación como el coste asociado.

Si L_{S1} y L_{S2} son las longitudes respectivas de S_1 y S_2, el algoritmo descrito por Wagner & Fischer (1974) permite calcular la distancia de edición definida en un tiempo proporcional a $\mathcal{O}(L^2)$, donde $L = max\{L_{S1}, L_{S2}\}$, de la manera que se explica a continuación. Se denotará como $S_1^{1,i}$ una subsecuencia de S_1 que comienza en el primer símbolo y termina en el i-ésimo y simplemente como $S_1[i]$ al símbolo en la posición i (para S_2 se establece una nomenclatura semejante) entonces la distancia entre dos prefijos de S_1 y S_2 se denotará como $d(i, j) = D(S_1^{1,i}, S_2^{1,j})$ donde $i \leqslant L_{S1}$ y $j \leqslant L_{S2}$. La distancia está dada por la recurrencia de la fórmula en (e1.1) en la que $d(i, 0)$ contiene la suma de los costes correspondientes a borrar los primeros i símbolos de S_1 y de igual forma $d(0, j)$ indica la suma de costes de inserción de los j primeros símbolos de S_2.

$$d(i,j) = min \begin{cases} d(i-1,j) + w(S_1[i], \varepsilon) & i > 0 \\ d(i,j-1) + w(\varepsilon, S_2[j]) & j > 0 \\ d(i-1,j-1) + w(S_1[i], S_2[j]) & i,j > 0 \\ 0 & i,j = 0 \end{cases} \quad \text{(e1.1)}$$

Una implementación usual, mediante programación dinámica (Wagner & Fischer, 1974), consiste en tener una matriz M de $L_{S1} \times L_{S2}$ donde se almacenan los diferentes $d(i, j)$ calculados en el procedimiento. En la tabla (1.1) se muestra un ejemplo para las cadenas $S_1 = \{1, 1, 1, 1.5, 1.5, 2, 2\}$ y $S_2 = \{0.3, 0.3, 7, 1, 1.5, 1.6, 2, 3\}$.

De la matriz resultante puede obtenerse no solamente el coste de transformación de S_1 en S_2, o distancia de edición, sino que es posible extraer la secuencia de operaciones realizadas , para el ejemplo mostrado esta sería $\{w(1, 0.3), w(1, 0.3), w(1, 7), w(1.5, 1.5), w(1.5, 1.6), w(5, \varepsilon), w(2, 2), w(2, 3)\}$ (Fernández & Díaz, 2006).

Tabla 1.1: Obtención de los diferentes $d(i,j)$ en el cálculo de la distancia de edición entre dos cadenas, $D(S_1, S_2) = 6.5 = M[8,7]$.

		0.3	0.3	7	1.5	1.6	2	3
	0	2	4	6	8	10	12	14
1	2	0.7	2.7	4.7	6.5	8.5	10.5	12.5
1	4	2.7	1.4	3.4	5.2	7.1	9.1	11.1
1	6	4.7	3.4	3.4	3.9	5.8	7.8	9.8
1.5	8	6.7	5.4	5.4	3.4	4	6	8
1.5	10	8.7	7.4	7.4	5.4	3.5	4.5	6.5
5	12	10.7	9.4	9.4	7.4	5.5	6.5	6.5
2	14	12.7	11.4	11.4	9.4	7.5	5.5	7.5
2	16	14.7	13.4	13.4	11.4	9.5	7.5	6.5

1.5.3. Cadena media

Nuevamente sean $\sum$ y $\sum^*$ un alfabeto y el conjunto de todas las cadenas de longitud finita posibles sobre este respectivamente. Dada una distancia definida sobre $\sum^*$ y un conjunto $S = \{S_1, S_2, ..., S_N\} \in \sum^*$ la *cadena media* R^* de S es aquella que minimiza la expresión (e1.2). En la literatura esta cadena recibe diferentes denominaciones, como *cadena media generalizada* pero en el trabajo seguiremos la nomenclatura adoptada por Martínez-Hinarejos *et al.* (2003). Por otra parte, entenderemos por *cadena mediana* aquella $R^M \in S$ para la que (e1.2) toma valor mínimo, es decir para la mediana el conjunto de posibles soluciones se limita solamente a S.

$$SOD(R^*, S) = \sum D(R^*, S_j) | S_j \in S \tag{e1.2}$$

Puede verse que la definición de cadena media es análoga al concepto de *vector medio* sobre $\mathbb{R}^n$. Este no es un problema tan reciente, ya desde los 80s Kruskal (1983) propuso una solución exacta utilizando la distancia de Levenshtein. El algoritmo presenta un coste computacional proporcional a $\mathcal{O}(L^N)$ siendo L la longitud de la mayor cadena en S por lo que en la mayoría de las aplicaciones prácticas no es adecuado. Como han apuntado Casacuberta & Antoni (1997) y Nicolas & Rivals (2005), diferentes formulaciones de este problema pueden clasificarse como NP-Completos. Por esta razón es común emplear una

solución aproximada, obtenida mediante alguna heurística enfocada a reducir la talla del espacio de búsqueda. Un ejemplo es el empleo de la cadena mediana que puede obtenerse en tiempo $\mathcal{O}(N^2 \times D)$ donde D es el coste computacional del cálculo de la distancia entre dos cadenas.

1.5.4. Regla de clasificación según el vecino más cercano

La regla del *vecino más cercano* establece un criterio de decisión muy simple que permite, dado un conjunto de instancias previamente etiquetadas con un determinado número de categorías y una instancia cuya clase se desconoce, clasificar esta última según la clase a la que pertenece la instancia más cercana a ella dentro del conjunto.

Esto es, siendo $M = \{\theta_1, \theta_2, ..., \theta_m\}$ un conjunto de categorías, $S = \{S_1, S_2, ..., S_N\}$ un grupo de instancias dentro de un espacio en el que se ha establecido cierta métrica $D(S_i, S_j)$ etiquetadas por alguna clase en M; la clasificación de una instancia $S_h \ni S$ se realiza asignándole la clase θ asociada a la instancia $S_l \in S$ que minimiza la expresión $D(S_h, S_l)$.

En este caso no se crea una aproximación a la función objetivo que describa completamente el espacio formado por lo que sería el conjunto de entrenamiento, sino que ante cada nueva instancia a clasificar se estima de manera local (Mitchell, 1997), creando una superficie de decisión formada por polígonos - en el caso 2D - cada uno definido por $S_i \in S$ y los puntos del espacio cuyo vecino más próximo es S_i.

En realidad este método puede presentarse como la formulación más sencilla de la familia de clasificadores basado los K-vecinos más cercanos o K-NN en cuyo caso, θ_k se toma como la clase mayoritaria dentro del grupo de las K instancias más cercanas a S_h. Sin embargo, a pesar de lo simple de su formulación, el empleo de esta variante brinda buenos resultados como se demuestra en los tempranos trabajos de Cover & Hart (1967) o como sugieren Theodoridis & Koutroumbas

(2006) y Duda & Hart (2001) donde se muestra que cuando la cantidad de ejemplos es grande la probabilidad de cometer un error en la clasificación queda acotada por el doble del error bayesiano óptimo. Las potencialidades de esta técnica también se observan en los experimentos desarrollados por Michie *et al.* (1994) donde este tipo de clasificador supera a otros algoritmos en una tarea de clasificación sobre bases de datos de imágenes.

1.5.5. Descripción de los datos empleados en los experimentos

En este epígrafe se describen las principales características de los datos empleados en los diferentes ejemplos y experimentos realizados. Con el objetivo de lograr una buena representatividad se han construido tres corpus independientes procedentes de las conocidas bases de datos *NIST Special Databse 19* (NIST-19) que unifica a las anteriores NIST-3 y NIST-7, *US Postal Service Database* (USPS) y *MPEG-7 Database* (MPEG-7). Éstas han sido estudiadas en diferentes trabajos como (Jain & Zongker, 1997; García-Díez *et al.*, 2011) y (Rico-Juan & Iñesta, 2012) entre otros.

Para construir las cadenas de Freeman correspondientes a cada contorno se verificó la normalización respecto a las rotaciones, para luego establecer como punto de partida el píxel perteneciente al contorno situado más arriba y a la izquierda buscando el siguiente en el sentido del reloj. La tabla (1.2) incluye una descripción de la muestra seleccionada en cada caso mientras que las figuras (1.2), (1.3) y (1.4) ilustran con mayor detalle la longitud promedio de las cadenas de Freeman codificando las instancias de cada clase.

Tabla 1.2: Características de las muestras seleccionadas de cada corpus.

Corpus	Tipo de contorno	Cantidad de Clases	Instancias x Clase	Longitud de las cadenas		
				Promedio	Max.	Min.
NIST-19	Letras	26	80	200±60	477	11
USPS	Dígitos	10	80	160±40	271	78
MPEG-7	Contornos varios	11	20	1300±900	5199	185

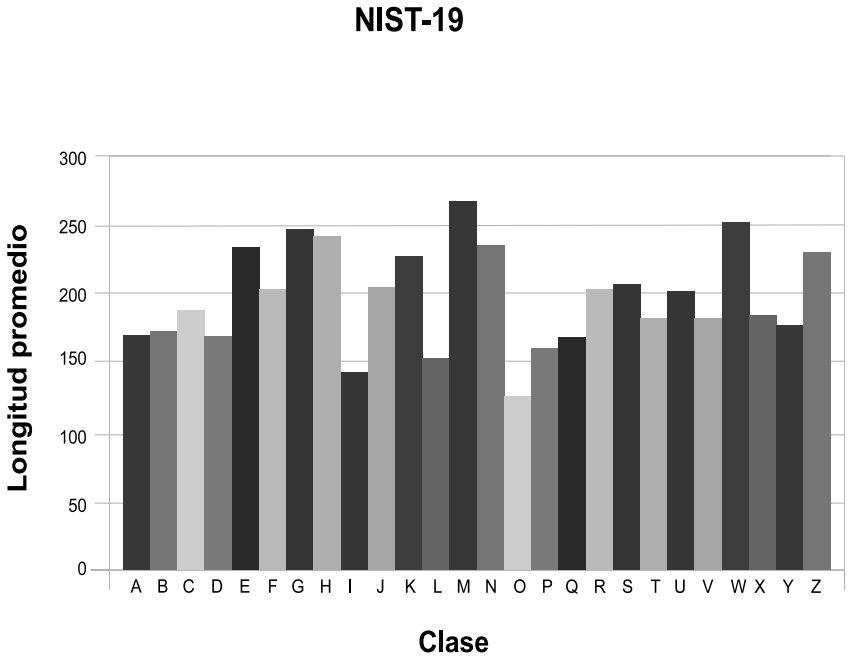


Figura 1.2: Longitud promedio de las cadenas de Freeman (NIST-19).

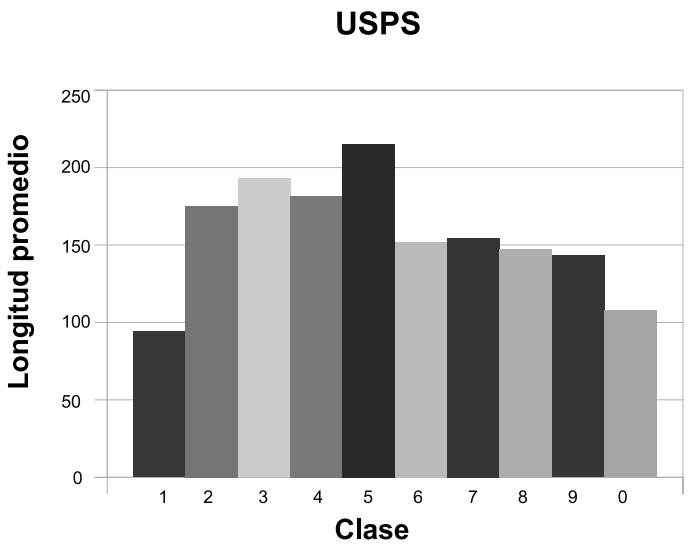


Figura 1.3: Longitud promedio de las cadenas de Freeman (USPS).

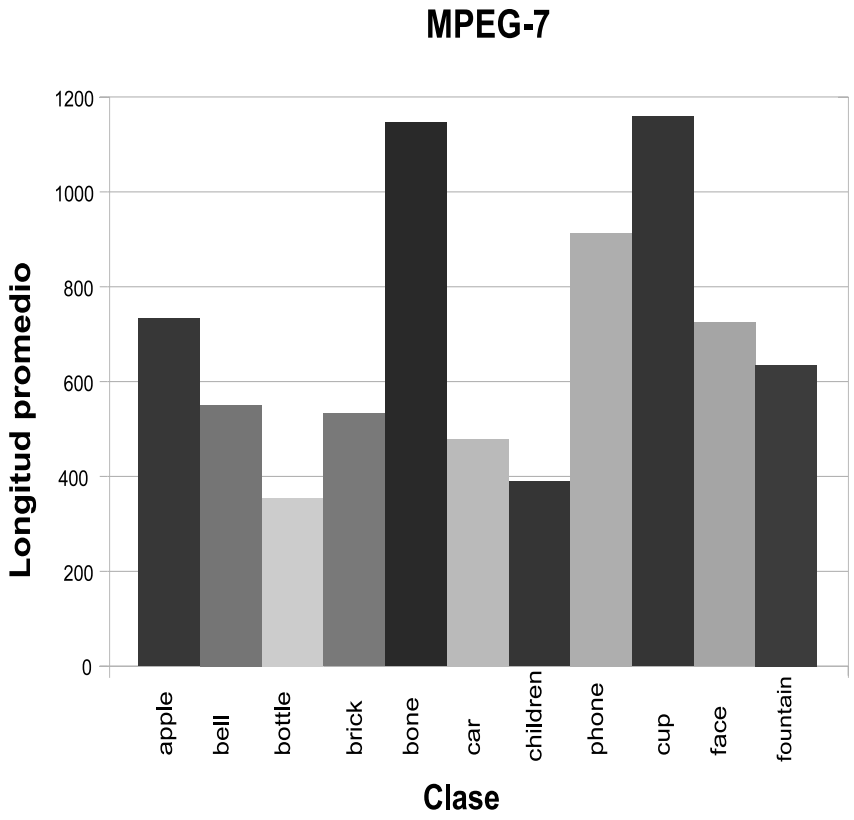


Figura 1.4: Longitud promedio de las cadenas de Freeman (MPEG-7).

Capítulo 2

Métodos para la detección de similitudes y obtención de la media entre dos contornos representados mediante cadenas de Freeman.

En el capítulo se estudian las principales propuestas concernientes a algunos de los problemas abordados en la tesis. Primeramente serán examinados diferentes enfoques que tratan la obtención de la cadena media u aproximaciones, poniendo especial interés en aquellos donde las cadenas han sido empleadas para representar contornos. En una segunda parte, se revisan varios trabajos relativos a la construcción de prototipos que describan a un conjunto de contornos representados mediante cadenas. Como quiera que el cúmulo de trabajos en esta área pudiera organizarse en diferentes categorías, se hará distinción entre aquellos en los que se emplea la cadena mediana, media o alguna aproximación de esta y aquellos enfoques donde los prototipos son construidos por otras vías.

2.1. Algoritmos para el cálculo de la cadena media

Extender el concepto de "media" a representaciones estructurales como las cadenas ha planteado un gran desafío a los especialistas en Reconocimiento de Formas. Este problema se presenta en varias aplicaciones como la representación de figuras 2D y la construcción de

prototipos (Jiang *et al.*, 2000; Bunke *et al.*, 2002), agrupamiento de cadenas (Lourenço & Fred, 2005), construcción de Mapas Autoorganizados de Cadenas (Kohonen, 1998; Fischer & Zell, 2000) o en la combinación de traductores automáticos (González-Rubio & Casacuberta, 2010).

Varias aproximaciones han sido propuestas desde el trabajo de Kruskal (1983). Una estrategia general es construir la media aproximada letra a letra a partir de una cadena inicial vacía. Para decidir cuál es el próximo símbolo que será añadido a la cadena es preciso definir una función de calidad. Este es el enfoque que sigue el procedimiento voraz descrito por Casacuberta & Antoni (1997). Definiendo $R^{t-1} = \{s_1, s_2, ..., s_{t-1}\}$ como la aproximación a la media en el paso $t-1$, el algoritmo evalúa cada símbolo α en $\sum$ como candidato para ser añadido a R^{t-1}. Para una cadena particular $S_k \in S$, sea S_k^{1,m_k} la subcadena con menor distancia a $R^t = \{s_1, s_2, ..., s_{t-1}, \alpha\}$, entonces el mejor símbolo s_t será aquel que minimice $\sum D(R^t, S_k^{1,m_k}) | S_k \in S$. Se agregan caracteres a la cadena mientras se logre una mejor aproximación o R^t alcance la longitud de la cadena más larga en S. Kruzslicz (1999) describe una mejora al algoritmo anterior estableciendo un criterio más refinado para seleccionar el siguiente símbolo.

Otro enfoque que ha sido estudiado por diferentes autores consiste en construir la media aproximada a través de sucesivas perturbaciones de una cadena inicial, como la mediana o la aproximación voraz obtenida por Casacuberta & Antoni (1997). Dos aspectos importantes de este tipo de algoritmo se refieren a cómo seleccionar la perturbación que conduce a una mejora y como lograr una rápida convergencia sin deteriorar los resultados. Otro tópico muy interesante apunta al estudio del efecto que tiene efectuar las modificaciones una a una o varias simultáneamente. Kohonen (1985) comienza por la mediana modificándola sistemáticamente aplicando inserciones, borrados y sustituciones en todas las posiciones de la cadena. Un cambio es aceptado si conduce a una disminución de (e1.2). En Martínez-Hinarejos *et al.* (2003) y Martínez-Hinarejos (2003) se propone un orden específico para efectuar las operaciones de edición. Se estudia además el efecto de variar la cadena inicial y la posibilidad de extenderlo a otras métri-

cas no monotónicas como la distancia de edición normalizada, donde no es posible asumir que los cálculos previos son válidos al ir añadiendo símbolos. Las operaciones de edición son aplicadas de acuerdo a dos posibles esquemas. En primer lugar, realizando sustituciones en cada posición de R^{t-1} y probando con cada símbolo del alfabeto $\sum$ para construir nuevas cadenas tomando como siguiente aproximación R^t aquella que minimice (e1.2). Posteriormente se realizan borrados del i-ésimo símbolo tratando de encontrar alguna cadena que satisfaga las restricciones antes enunciadas, de ser posible, se vuelve a probar con sustituciones para finalmente practicar inserciones en cada posición repitiendo todo el proceso en caso de haberse logrado mejoras.

Una segunda alternativa aparece al variar el orden en que se realizan las perturbaciones en la i-ésima posición de R^{t-1}. Así se generan tres conjuntos de cadenas candidatas al borrar, sustituir o insertar, probando cada $\alpha \in \sum$ en el caso de las sustituciones e inserciones, de entre los cuales se toma como nueva solución a la cadena que cumpla las mismas condiciones que el caso anterior - o se mantiene R^{t-1} en caso de no existir tal cadena - repitiendo ahora para la $(i + 1)$-ésima posición. Este procedimiento se lleva a cabo tantas veces como se obtengan mejoras. En una comparación con la cadena mediana se muestra que ambas variantes logran una mejor aproximación. Es preciso señalar que los experimentos no fueron realizados con cadenas que representaran contornos sino utilizando el corpus de cromosomas *Copenhagen*. Resultados teóricos indican que mediante este enfoque se pueden obtener muy buenas aproximaciones a la media verdadera.

Debe notarse que en general estos métodos realizan una especie de búsqueda "ciega" ya que no incluyen alguna información a priori acerca de qué tan buena pudiera ser una modificación. Habitualmente es preciso calcular una cierta cantidad de distancias para evaluar la calidad de la solución parcial, de ahí que seleccionar apropiadamente la solución candidata pueda evitar desperdiciar una cantidad considerable de tiempo en evaluaciones innecesarias de (e1.2). Información heurística que pudiera ayudar a evaluar qué tan prometedora será una solución sin calcular (e1.2) es empleada por Fischer & Zell (2000) y Cárdenas (2004). Para tratar de acelerar la convergencia de la búsque-

da Fischer & Zell (2000) aplican varias modificaciones simultáneamente.

En este caso, se calcula la distancia desde la solución parcial R^t a cada elemento en S. Cada distancia especifica un conjunto de operaciones de edición, incluyendo "no hacer nada", una sustitución de un símbolo por el mismo, relativas a cada posición en R^t. Por ejemplo, $D(R^t, S_1)$ puede especificar que $R^t[1]$ debe borrarse, $R^t[2]$ no debe modificarse y así sucesivamente. Luego de computar la distancia a cada cadena, se aplica la operación más frecuente en cada posición. Este procedimiento se basa en la idea de que, a medida que la solución se acerca a la media verdadera, la operación "no hacer nada" será la que más a menudo ocurra en cada posición. El proceso se repite mientras se encuentre una solución mejor.

Para el caso particular en que hay solamente dos cadenas en S el trabajo de Bunke *et al.* (2002) describe la forma de calcular una cadena R^t, llamada "media ponderada" que satisface $D(S_1, R^t) = \alpha$ y $D(S_1, S_2) = \alpha + D(R^t, S_2)$. Una vez calculada la distancia entre las cadenas, R^t se construye aplicando a S_1 un subconjunto con coste α de la secuencia de operaciones de edición con coste mínimo. Como quiera, para un conjunto de pesos arbitrarios seleccionar dicho subconjunto parece equivalente a resolver el problema de la *suma de los subconjuntos* o *subset-sum problem*, un conocido ejemplo de la clase de problemas NP-Completos.

El algoritmo propuesto por Bunke *et al.* (2002) es utilizado por Jiang *et al.* (2003) en un algoritmo incremental que permite calcular una aproximación a la media de N cadenas a partir de la media previamente computada R^{N-1} de $N-1$ elementos y la cadena recién incorporada al conjunto. Cada vez que una cadena S_n se agrega, se calcula la media ponderada entre R^{N-1} y S_n. Los autores prueban diferentes valores de α para seleccionar el que conduzca a un menor valor de (e1.2). En este caso los autores reportan varios experimentos donde se comparan el método propuesto, la aproximación mediante la mediana y el algoritmo descrito por Casacuberta & Antoni (1997) de acuerdo a la distancia acumulada de la media obtenida a cada elemento. Los

contornos, un subconjunto de la base de datos UNIPEN incluyendo las clases correspondientes a los caracteres “1”, “2”, “3” y “6”, se codifican mediante cadenas de coordenadas (x, y). En estas pruebas la propuesta descrita alcanzó mejores resultados aunque no se ofrecen otros criterios sobre el poder de representación de los prototipos obtenidos.

El concepto de media ponderada también es empleado por Jiang *et al.* (2011). En lugar de buscar la media de N elementos en $\sum^*$ las cadenas se embeben en un espacio vectorial V de modo que cada cadena es un punto en $\mathbb{R}^m$. Primero, un subconjunto P de m elementos es seleccionado en S. Para una cadena dada S_k se calculan las distancias a cada elemento en P resultando m distancias $d_1, d_2, ..., d_m$ que son ordenadas en un vector que representa a S_k. En un segundo paso se calcula la media de los N vectores en V. Finalmente, es necesario aplicar la transformación inversa, es decir, encontrar la cadena asociada al vector medio. Los autores combinan varias heurísticas basadas en la media ponderada. Por ejemplo, seleccionando los dos vectores más cercanos al vector medio in V, la media aproximada puede ser obtenida como la media ponderada de las cadenas asociadas a dichos vectores.

Más recientemente otros autores (Rodríguez & Carratalá, 2008) han estudiado el cálculo de aproximaciones a la cadena media no solamente empleando la distancia de Levenshtein sino otras métricas como la distancia de edición estocástica (Ristad & Yianilos, 1998).

2.2. Creación de prototipos a partir de un conjunto de contornos.

La obtención de un pequeño grupo de prototipos que describan adecuadamente una clase de contornos ha sido un problema que, aunque ampliamente tratado, sigue siendo de gran interés como lo demuestran recientes publicaciones al respecto (Ong & Seghouane, 2008). En este sentido, un enfoque bastante común es buscar los prototipos dentro

del mismo conjunto. Usualmente estos algoritmos muestran un menor coste computacional, tal es el caso de la cadena mediana, que puede ser calculada en tiempo polinómico.

Aun así, un grupo no menos considerable de autores ha propuesto algoritmos donde los prototipos no pertenecen necesariamente al conjunto original, en este caso se obtienen mediante algún procedimiento diseñado para lograr nuevas instancias que, por construcción o definición, caractericen al conjunto de objetos dados. El diseño y éxito de estos métodos está influido por factores como el tipo de codificación con el que se tratan los objetos, así como el criterio empleado para medir qué tan representativo es un prototipo.

2.2.1. Métodos para la construcción de prototipos basados en la cadena media.

Cuando los contornos son codificados mediante algún tipo de cadena, ya sean códigos de Freeman, cadenas cíclicas o vectores ordenados entre otros formalismos, encontrar el contorno prototipo puede transformarse en el problema equivalente de obtener la cadena media u alguna aproximación.

Un enfoque muy simple consiste en calcular la cadena mediana, método que aunque de bajo coste computacional no brinda en todos los casos resultados satisfactorios (Cárdenas, 2004; Jiang *et al.*, 2000). Naturalmente, para una mejor aproximación puede emplearse cualquiera de los métodos descritos en la sección (2.1) u otros enfoques presentados no como procedimientos generales para el cálculo de la cadena media pero que parten de algún tipo de cadena. En tal sentido pueden encontrarse otros trabajos como Jiang *et al.* (2000), que explora la aplicación particular del algoritmo descrito por Casacuberta & Antoni (1997) a la obtención de la media entre contornos. Estos se representan muestreándolos de modo que la distancia entre dos puntos sucesivos p_k y p_{k+1} sea un valor constante, de donde resulta una lista ordenada de vectores de p_k a p_{k+1}. Se reportan experimentos con

varios contornos mostrando la tolerancia a la presencia de instancias ruidosas.

El trabajo de Sánchez *et al.* (2002) presenta un método para calcular la media entre dos contornos codificados mediante cadenas cíclicas, donde cada símbolo representa un segmento caracterizado por los atributos longitud y dirección. A partir de la secuencia de edición óptima obtenida de la distancia entre las cadenas se estable una relación entre pares de subcadenas, una en cada contorno. La media de estas es calculada tratándolas como si fuesen funciones lineales por tramos *(piecewise linear functions)*. Una limitante es que el algoritmo está diseñado para calcular la media R^t de solo dos cadenas. En otro caso se aplica un procedimiento progresivo cuyo esquema ilustra la figura (2.1) para tres cadenas. El algoritmo con coste computacional $\mathcal{O}(N \times T^2 \log T)$, donde N es el número de polígonos y T el de aristas, se aplica en un experimento para calcular la media a varios pares de contornos, aunque es validado visualmente por lo que no ofrece un patrón con el que comparar futuros trabajos.

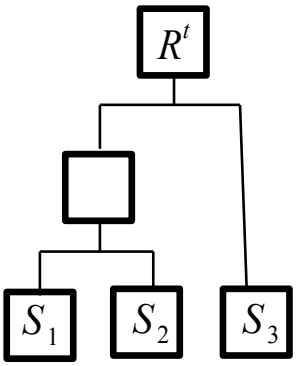


Figura 2.1: Procedimiento progresivo para calcular la media de más de dos contornos (Sánchez *et al.*, 2002).

2.2.2. Métodos para la construcción de prototipos que no se basan en la cadena media.

Según se apuntó en la parte introductoria del capítulo, se han distinguido dos grandes grupos de trabajos en lo que respecta a la creación de contornos prototipos cuando estos son representados mediante algún tipo de cadenas. Aquellos que toman como base a la cadena media y que debido a la imposibilidad de calcular ésta eficientemente proponen diversas aproximaciones, y aquellos donde los prototipos son construidos o seleccionados mediante algún enfoque no sustentado en este concepto - o al menos no directamente - es decir, aplicando un procedimiento diseñado de forma tal que permita esperar un buen poder de representación. En esta categoría encontramos el trabajo de Duta *et al.* (2001) donde se aplica un proceso de *agrupamiento* a las instancias, descartando aquellas consideradas atípicas, para luego calcular el prototipo de cada clúster. Para cada instancia S_k del conjunto se calcula una aproximación poligonal $S_k^{'}$ que se intenta relacionar con cada $S_j \in S$ entendiendo que esto se logra si se puede hacer corresponder un conjunto de puntos $\{(x_i^{ck\prime}, y_i^{ck\prime})\}|i = 1...n$ en $S_k^{'}$ con otro $\{(x_i^{cj}, y_i^{cj})\}$ en S_j donde el valor definido en (e2.1) se mantiene dentro de cierto rango. Los conjuntos deben tener una cantidad de puntos mínima fijada previamente.

$$MAE(S_k^{'}, S_j) = \sum_{j=1}^{n} \frac{[(x_i^{ck\prime} - x_i^{cj})^2 + (y_i^{ck\prime} - y_i^{cj})^2]}{n} | i = 1...n \quad \text{(e2.1)}$$

Luego se construye un clúster (C) seleccionando la aproximación $S_k^{'}$ que minimice (e2.1) y las L instancias relacionadas con $S_k^{'}$, haciendo $S = S \sim S_h | S_h \in C$, para repetir nuevamente cada paso mientras $S \neq \varnothing$. Este procedimiento conduce a que los puntos $\{(x_i^{ck\prime}, y_i^{ck\prime})\}$ de la aproximación poligonal seleccionada tengan un correspondiente en cada $S_h \in C$ y definiendo esta relación como transitiva se tiene que si los puntos $\{(x_i^{ch}, y_i^{ch})\}|S_h \in C$ se relacionaron con $\{(x_i^{ck\prime}, y_i^{ck\prime})\}$ entonces lo hacen entre ellos, es decir cada punto de $S_k^{'}$ define un conjunto de puntos en el resto de los contornos del clúster de cuyo

promedio se obtiene las coordenadas del punto correspondiente en el prototipo que los representará. El algoritmo, cuyo coste computacional depende de varios términos como la potencia cúbica de la cantidad de puntos en S'_k y del número N de instancias en un factor cuadrático, fue probado en la obtención de prototipos para un conjunto de contornos describiendo diferentes estructuras del cerebro los cuales se emplearon para segmentar automáticamente varias imágenes con resultados equivalentes al proceso realizado de forma manual.

Como se ha podido observar - y en correspondencia con el amplio espectro que cubre el empleo de cadenas como representación de objetos - varios autores han descrito métodos fuera del contexto del reconocimiento de contornos, como Cárdenas (2004) que presenta un método potencialmente aplicable para construir prototipos tanto en cadenas de ADN como en contornos según muestran varios experimentos. La idea básica plantea tomar varios prototipos candidatos (R^{mi}) como representantes de las distintas clases y modificarlos, en término de operaciones de edición, de forma que se eliminen las diferencias frecuentes con las instancias de la clase que representan y a su vez se repitan poco en cadenas de diferente categoría. Previamente se fija un umbral para decidir cuándo una operación se considera frecuente y cuándo no.

Este procedimiento no se aplica con todas las instancias disponibles, sino que a partir de cada prototipo se construye un clúster C_i - donde pueden aparecer instancias de distinta clase - tomando aquellas más próximas de acuerdo a una distancia de edición D específica. Posteriormente basándose en $D(R^{mi}, S_k)$ se calcula la secuencia de edición que permite transformar R^{mi} en S_k para todo $S_k \in C_i$. Después de este paso, a cada operación de edición se le calcula un coeficiente de frecuencia contando 1 a cada edición que aparece en $D(R^{mi}, S_k)$ si S_k es de la misma clase que R^{mi} y -1 si no lo es. De este modo los valores más altos indican operaciones de edición que son comunes para transformar R^{mi} en un S_k de igual clase, es decir, denotan diferencias comunes entre R^{mi} y las instancias de su mismo tipo dentro de C_i mientras que son ediciones no frecuentes en la transformación de R^{mi}

en un S_k de diferente clase. Por último se modifica R^{mi} realizando las operaciones de edición cuya frecuencia sobrepasen un umbral dado.

Respecto a la validación experimental, los autores reportan pruebas sobre secuencias de ADN y contornos de caracteres numéricos manuscritos representados mediante cadenas. Se tomaron 100 instancias por clase para un total de 1000, representándolos por cadenas de 150 símbolos de longitud promedio y utilizando la distancia de Levenshtein. Las 1000 instancias se dividieron en 5 grupos para realizar una validación cruzada; luego aplicando el algoritmo descrito se construyeron un promedio de 30 prototipos por clases, siendo del $1.0\,\%$ el error al clasificar con estos mediante el vecino más cercano, mientras que al hacerlo con las 800 instancias el error fue de $0.7\,\%$. Adicionalmente se realizó otro experimento consistente en dividir las 1000 instancias en dos particiones de 500 cuidando distribuir las clases de manera uniforme, de los primeros se extrajeron 5 conjuntos independientes de 50 prototipos, 10 por clase, clasificando los restantes 500 cometiendo un error de $3.8\,\%$. Luego se aplicó el algoritmo a cada uno de los 5 conjuntos - para mejorar los prototipos mediante el método propuesto - y clasificando nuevamente con los prototipos así generados, se reportó un error de $1.8\,\%$ y una reducción cercana al $70\,\%$ del conjunto de entrenamiento; experimentos y resultados que por su diseño pudieran ser empleados en futuras comparaciones.

Finalmente, como evidencia de la variedad de enfoques con que se ha abordado la problemática tratada tenemos el trabajo de Bontempi & Marcelli (1995) donde los contornos son representados mediante una cadena de bits, para emplearlos como individuos en una búsqueda utilizando algoritmos genéticos. En realidad los bits codifican una representación jerárquica de relaciones entre dos tipos de componentes - arcos y segmentos - de los cuales se emplean 4 para aproximar los contornos tomados consecutivamente en una relación de orden cíclica. La cadena que representa cada instancia consta de 32 bits repartidos de la siguiente forma: 3 bits codificando dirección del componente, 2 la curvatura, 2 para la relación espacial - arriba/abajo, derecha/izquierda - entre el componente y su sucesor y 1 para su longitud relativa dando un total de 8 bits por cada uno de los 4 componentes utilizados para

representar el contorno. En este punto se realiza una búsqueda tomando como función de adaptación la cantidad de contornos del conjunto de entrenamiento que son cubiertos por el prototipo, notar que al simplificarse información puede ocurrir que contornos con cierto grado de diferencia sean representados por individuos iguales. Un aspecto distintivo es que no se restringe la cantidad de individuos en la población pues se reporta un mejor desempeño que al emplear un algoritmo genético clásico con tamaño de la población fijo, además de demostrar que esta mantiene una magnitud manejable. Sin embargo, no se explican detalladamente qué tipos de experimentos se realizaron para evaluar los prototipos encontrados.

En algunos artículos como Platonov & Langer (2007) y Ong & Seghouane (2008) se ha tratado el tema de la construcción de prototipos para representar un grupo de contornos explorando otros tipos de representaciones. La mayor parte de los trabajos examinados se basan en alguna métrica de similitud, fundamentalmente la distancia de Levenshtein cuando se trata de cadenas. Sin embargo, otras distancias como *Dynamic Time Warping* (DTW) han sido estudiadas (Marzal *et al.*, 2006; Yu *et al.*, 2007; Tak & Hwang, 2007). Estos trabajos reportan interesantes resultados cuando se emplea esta medida aplicada a la comparación y recuperación de imágenes. Cuando las cadenas a tratar no son propiamente cadenas de Freeman sino secuencias de puntos o polígonos, la distancia de Levenshtein puede tener algunos inconvenientes tal como apunta Marzal *et al.* (2006). Esta distancia es muy sensible a la frecuencia de muestreo dado que no permite correspondencias uno-a-muchos entre los símbolos de las cadenas.

2.3. Conclusiones parciales.

Después de examinar un grupo de los principales trabajos en el área pueden distinguirse varios aspectos coincidentes, que además marcarán el camino a seguir en la investigación. En primer lugar, se puede señalar la ausencia de un marco unificado para los experimentos

de validación, por lo que resulta engorroso o prácticamente imposible realizar comparaciones entre los diversos métodos; empleando en algunos casos criterios bastante subjetivos como la similitud visual. Un segundo punto en común consiste en que los algoritmos propuestos presentan órdenes cuando menos cuadráticos respecto a la cantidad de instancias o superiores en otros casos. Como otro aspecto que se deriva del análisis realizado es pertinente señalar que en los casos donde la propuesta es aplicada directamente a contornos representados mediante cadenas estas contienen diferentes codificaciones, ya sea una secuencia de coordenadas cartesianas de los puntos del contorno, pares dirección/longitud o códigos de cadena, aunque algunos de los enfoques tratados son potencialmente aplicables a otras representaciones.

Se constató además, un grupo importante de algoritmos que buscan una aproximación a la cadena media a través de sucesivos refinamientos de una solución parcial. Varios de estos no emplean ninguna información heurística que pueda agilizar el proceso de búsqueda. Otro aspecto que parece insuficientemente estudiado es el efecto de realizar varias modificaciones simultáneamente o una a la vez. Estas cuestiones serán abordadas más adelante en la propuesta de un nuevo algoritmo para el cálculo de la cadena media.

Finalmente - constituyendo este un punto fundamental de la propuesta presentada - puede notarse que la generalidad de los esquemas descritos tratan a los contornos como un todo aunque se describan con un modelo estructurado como las cadenas, es decir, no se hace distinción entre diferentes partes de estas, o lo que es lo mismo, sobre diferentes regiones del contorno con determinadas características como pudiera ser un grado de semejanza específico. Sin embargo prestar atención a este aspecto podría además de brindar resultados igualmente satisfactorios a los reportados, ofrecer un punto de vista interpretable que permita identificar aquellos elementos de los contornos que caracterizan una determinada clase o grupo de instancias u otra información de interés.

Sobre estos puntos, el resto de la investigación se centra en la exposición de un enfoque para obtener un grupo de prototipos que representen a un conjunto de contornos. Este se basa en un algoritmo, también propuesto, que permite construir un prototipo para dos contornos a partir del análisis de las partes con mayor similitud entre estos. Estas cuestiones serán minuciosamente abordadas en las siguientes secciones.

Capítulo 3

Algoritmos de selección de prototipos y eliminación de instancias atípicas.

Existen diversos argumentos que sustentan la necesidad de eliminar ciertas instancias del conjunto de entrenamiento de un clasificador, particularmente los basados en la regla de los K-vecinos más cercanos o alguna de sus variantes. Como se ha expuesto, este tipo de algoritmo exhibe muy buen desempeño dado que el número de ejemplos sea lo suficientemente grande. Sin embargo por esta misma razón el tiempo de búsqueda de los vecinos puede tornarse excesivamente largo, por lo que se hace muy deseable la posibilidad de reducir la talla del conjunto de datos de modo que no se afecte la capacidad de generalización.

Por otra parte la presencia de instancias fuera de la región que ocupa su clase (*outliers*) puede ser un problema grave para algunos algoritmos de aprendizaje. Esto es particularmente grave si se debe a ruido en los valores de los atributos (Theodoridis & Koutroumbas, 2006). En el trabajo de Wilson (1972) se demuestra como el desempeño de un clasificador K-NN puede mejorarse al emplear un conjunto del que previamente se eliminaron las instancias mal etiquetadas.

Estas razones han dado lugar a una prolija literatura sobre diversos enfoques para solucionar uno u otro inconveniente. En general los métodos descritos pueden agruparse en *algoritmos de condensado* y *algoritmos de edición*. Los primeros apuntan a la obtención de un subconjunto de los datos de menor talla pero cuidando que el clasificador

pueda generalizar igual de bien. Dentro de este tipo de esquema pueden a su vez pueden distinguirse dos perspectivas distintas, los métodos de selección de instancias y los orientados a sustituir un grupo de objetos por un prototipo, posiblemente artificial, que los represente adecuadamente.

Por su parte, los métodos de *edición* también llamados *filtros de ruido* (Jankowski & Grochowski, 2004a) persiguen eliminar los objetos ubicados en la región de una clase diferente para favorecer la capacidad de generalización.

Si bien es común que los métodos *condensado* tengan una marcada naturaleza heurística mientras que los de *edición* se caracterizan por una fuerte base estadística en muchos casos no existe un límite riguroso entre uno y otro tipo. La literatura recoge diferentes taxonomías para agrupar estos métodos atendiendo a diferentes factores. Wilson & Martínez (2000) distinguen los enfoques *incrementales* que comienzan con un conjunto vacío al que añaden las instancias que cumplen las condiciones requeridas. Esta estrategia es opuesta a la que implementan los enfoques *decrementales* donde partiendo del conjunto original se van excluyendo las instancias. En estos esquemas se permite que se modifique el conjunto editado antes de verificar que todas las instancias cumplan o no la condición para su permanencia por lo que son fuertemente dependientes del orden en que se analizan las instancias. Esta característica los distingue de los algoritmos *por lotes* (*batch*) donde se verifica si la instancia cumple o no la condición de borrado antes de realizar cualquier modificación al conjunto.

Otro criterio para agrupar los algoritmos es definido por Brighton & Mellish (2002) quienes distinguen tres tipos de esquemas: los que *incrementan la competencia* y los que *preservan la competencia* análogos a los que hemos descrito como algoritmos de *edición* y *condensado* respectivamente. El tercer grupo son los enfoques *híbridos*.

En esta sección se han agrupado de forma similar a la taxonomía dada por Brighton & Mellish (2002) tratado de agrupar por una parte los algoritmos presentados en la literatura como propiamente de *edición*

u otros que de una forma más o menos explícita tratan este problema y un grupo de técnicas específicamente consideradas como métodos de *condensado*.

3.1. Algoritmos de edición.

El trabajo de Wilson (1972) es reconocido como pionero dentro de los algoritmos de edición, sirviendo de base a otros enfoques propuestos con posterioridad. Basándose en la idea de que, si una instancia resulta incorrectamente clasificada mediante la regla K-NN entonces debe ser excluida del conjunto de entrenamiento el autor propone un método conocido como *edited nearest neighbour rule*. El algoritmo es simple de implementar, como puede observarse en la especificación dada en la función AlgWilson. La realización del proceso en dos etapas, primero marcando las instancias para su posterior borrado, garantiza la independencia respecto al orden en que se examinan los objetos. Al utilizar todo el conjunto de entrenamiento, salvo la instancia que se está analizando, para determinar los K vecinos más cercanos se tiene que la estimación del error realizada se corresponde con el método *leave one out*. Por su diseño, de la aplicación de este algoritmo resultará un conjunto de instancias relativamente compacto organizadas en grupos homogéneos (Vázquez *et al.*, 2005). Este procedimiento puede ser repetido varias veces hasta que no se verifiquen cambios en el conjunto de datos.

Un método directamente derivado del trabajo de Wilson (1972) es presentado por Tomek (1976a). El autor describe una variante conocida como *All-*KNN. El esquema consiste en aplicar repetidas veces el algoritmo de Wilson pero variando el valor de K, si una instancia resulta incorrectamente clasificada para alguno de estos entonces se elimina. En los experimentos reportados esta variación conduce a resultados superiores respecto a Wilson. Como otros algoritmos, *All-*KNN también puede conservar intactos los puntos interiores. Una versión que particulariza el criterio de borrado es presentada en (Tomek, 1976b). En este caso, se crea una categoría adicional θ_0, correspondiente a los

Función `AlgWilson(`S`,`K`)` :S

```
/* S: conjunto de instancias a editar                    */
/* K: número de vecinos más cercanos                     */
```

para cada *instancia* $S_i \in S$ **hacer**
  clasificar S_i según sus K-vecinos más cercanos en $S \sim S_i$;
  si S_i *es incorrectamente clasificada* **entonces**
    marcar S_i para borrado;
  fin si
fin para cada
borrar de S todas las instancias marcadas;
devolver S;

prototipos rechazados. Si la clase asignada θ_i por la regla K*-NN con rechazo* (Hellman, 1970) no coincide con la etiqueta de la instancia entonces, si $\theta_0 = \theta_i$ se re-etiqueta el objeto como miembro de θ_0, en otro caso se descarta.

Devijver & Kittler (1980) describen un algoritmo también basado en (Wilson, 1972). En este caso se realiza una partición del conjunto de entrenamiento en n subconjuntos disjuntos $C_1, C_2, ..., C_n$ de forma aleatoria. Posteriormente se aplica Wilson a cada partición C_j pero con la particularidad de que los K vecinos de la instancia que se está analizando no se buscan en su propia partición sino en el subconjunto $((j + 1) \bmod n)$. Después de este paso, los objetos que se han conservado se agrupan nuevamente. Debido al carácter aleatorio de la división es posible que no todos los subconjuntos C_j representen adecuadamente a la totalidad de los datos, por lo que pudiera ser conveniente que se atienda a la distribución de las instancias al realizar la partición. Los propios autores describen la posibilidad de aplicar repetidas veces este esquema, lo que se conoce como algoritmo *Multiedit*, hasta que transcurra un número predefinido de iteraciones sin que se excluyan instancias. En (Devijver & Kittler, 1982) se demuestra que para un conjunto infinito se obtienen resultados óptimos. Sin embargo en la práctica, cuando hay pocos objetos, pueden eliminarse todas las instancias de una o más clases (García-Borroto *et al.*, 2011), en estos casos el algoritmo de Wilson puede funcionar considerablemente mejor. También siguiendo la idea de contar con una nueva clase θ_0

que representa a ciertos prototipos Koplowitz & Brown (1981) describen un criterio llamado *edición con re-etiquetado*. Si la clasificación otorgada θ_i por la regla K*-NN con rechazo* (Hellman, 1970) no coincide con la clase de la instancia entonces esta se re-etiqueta como θ_i a menos que $\theta_i = \theta_0$ en cuyo caso se descarta la instancia.

Partiendo del llamado *IB1*, similar a Wilson con $K = 1$ pero buscando los K-vecinos en el conjunto editado en vez de hacerlo en el conjunto original Aha *et al.* (1991) define varios algoritmos. El primero de estos, *IB2*, en lugar de enfocarse en descartar instancias mal etiquetadas tiende a conservarlas ya que un criterio para incluir un objeto en el conjunto editado es precisamente que sea incorrectamente clasificada. Para resolver este posible inconveniente el algoritmo *IB3* mantiene información sobre la cantidad de veces que una instancia interviene en clasificaciones correctas eliminando posteriormente aquellas que no lo hacen con una determinada frecuencia, comportamiento típico de las instancias ruidosas. Otros esquemas incluidos en este trabajo son los algoritmos *IB4* e *IB5*, especializaciones del *IB3* realizando modificaciones a los pesos de los atributos que se emplean en el cálculo de la similitud entre los objetos.

Otro método que también realiza particiones en el conjunto de instancias es descrito por Ferri & Vidal (1992). En este caso con el objetivo de resolver uno de los principales inconvenientes del algoritmo de Wilson, la falta de independencia estadística debido al esquema *leave one out* empleado. Además esta propuesta apunta a resolver problemas que pueden presentar los métodos basados en particiones cuando la cantidad de muestras disponibles es pequeña. Los autores proponen dividir los datos en n bloques aplicando Wilson en cada uno de estos. Al igual que en otros casos, también es posible aplicar el esquema de edición repetidas veces hasta que finalice una iteración sin producirse la eliminación de alguna instancia.

Un algoritmo que además de reducir la talla del conjunto de entrenamiento intenta eliminar objetos atípicos es el descrito por Lowe (1995). El autor describe un algoritmo de aprendizaje llamado *variable-kernel similarity metric* (VSM). Para reducir el tamaño del

conjunto de entrenamiento cualquier instancia S_i es descartada si todos sus K-vecinos concuerdan en la clasificación usando el método (VSM) y lo hacen con una probabilidad superior a 0.6. No hay restricción respecto a que el objeto borrado emita la misma clasificación que sus vecinos en cuyo caso se considera ruidosa.

Aunque no diseñado específicamente para descartar instancias mal etiquetadas Kubat & Matwin (1997) presentan una extensión al algoritmo descrito por Tomek (1976a) capaz de borrar tanto instancias en las fronteras de las clases como puntos interiores. Para esto primero aplican una variante del algoritmo de Hart (Hart, 1968) para luego eliminar pares de instancias de diferente clase que se tienen mutuamente como vecinos más cercanos.

Un algoritmo enfocado principalmente en reducir la talla del conjunto de entrenamiento pero que además puede eliminar objetos atípicos es descrito por Zhang (1992). El procedimiento se basa en la llamada *tipicidad* de una instancia que se define como el radio de su *similitud* promedio respecto a instancias de una misma clase en relación a la similitud promedio con instancias de categoría diferente. En la vecindad de una instancia típica se encontrarán mayormente objetos de la misma clase. A su vez la semejanza entre dos instancias S_i e S_j se mide como $1 - distancia(S_i, S_j)$. En el caso del artículo, se emplea una distancia para vectores. Además cada instancia tiene asociado un peso W. Para obtener un conjunto editado $S\prime$ a partir del conjunto S se localizan en este la instancia más típica S_i que no es correctamente clasificada por las instancias en $S\prime$ y aquella S_j que sea la más típica entre las que causan una clasificación correcta de S_i. Luego S_j se añade a $S\prime$ y se modifica su peso asociado como $1/tipicidad(S_j)$. El proceso se repite hasta que todas las instancias en S sean correctamente clasificadas. Cuando las instancias de una clase pueden agruparse en regiones disjuntas puede ser necesario realizar modificaciones para lidiar con este caso, dado que en el cálculo de la tipicidad se tienen en cuenta todas las instancias de la clase este valor podría resultar sesgado.

Otro método que puede ser empleado para excluir instancias ruidosas es el propuesto por Brodley (1993). Para cada instancia se cuenta la cantidad de veces que se encuentra entre los K-vecinos más cercanos de alguna instancia de su propia clase y cuantas de instancias de categoría diferente. Si este último valor es mayor entonces se elimina.

Wilson & Martínez (2000) estudian varios algoritmos que emplean el concepto de *instancia asociada*. Una instancia S_j se dice asociada a S_i si esta última se encuentra entre los K-vecinos de S_j. El procedimiento *DROP1* excluye a S_i si esto mejora o mantiene igual la clasificación de sus asociadas que estén incluidas en el conjunto editado, estas deberán incluir un nuevo elemento entre sus K-vecinos. De este modo las instancias ruidosas se eliminan dado que estas usualmente aparecen asociadas a instancias de otra clase. Condicionado por el orden en que se excluyen las instancias, puede ocurrir que el borrado de un objeto ruidoso provoque que sus asociadas sean clasificadas erróneamente. Este inconveniente es tratado por la variante *DROP2* donde se verifica cómo afecta la exclusión de S_i en la clasificación de sus asociadas dentro del conjunto original, en lugar de solo revisar las asociadas en el conjunto editado como plantea *DROP1*. Por otra parte, *DROP3* y *DROP4* están centrados en realizar un filtrado más cuidadoso de las instancias ruidosas. *DROP3* es semejante a su predecesor pero antes aplica un algoritmo análogo al descrito por Wilson (1972) para descartar instancias mal etiquetadas y suavizar las fronteras entre clases. *DROP4* incluye una restricción adicional para desechar una instancia en el paso del filtrado de ruido, además de ser incorrectamente clasificada su eliminación no deberá incidir negativamente en la clasificación de otras instancias. Finalmente *DROP5* modifica el orden en que son examinadas las instancias en *DROP2* de modo que además de suprimir el ruido se obtengan fronteras mejor identificadas. La efectividad de los enfoques propuestos es validada en una comparación con varios de los algoritmos descritos en la literatura utilizando datos del repositorio UCI *Machine Learning Database*

Un método que en evaluaciones experimentales logra un desempeño similar a la propuesta de Wilson (1972) es descrito por Eick *et al.* (2004). La idea principal consiste en sustituir un grupo de instancias o

clúster por un prototipo representativo. En diferentes pruebas con algunas bases de datos del repositorio UCI se comprueba que esta estrategia logra mayores índices de reducción que el algoritmo de Wilson. Aunque no directamente, este esquema descarta las instancias ruidosas. Estas tenderán a ser incluidas en un clúster donde predomina una clase diferente y por tanto, el prototipo que representa a este grupo estará etiquetado como la clase mayoritaria.

En contraste con el algoritmo de Wilson y otros algoritmos derivados, Vázquez *et al.* (2005) sustituye la decisión mediante el voto mayoritario empleada en la regla K-NN por una estimación de la probabilidad de que la instancia S_i analizada pertenezca a una determinada clase. De este modo se conjugan información tanto sobre la vecindad de la instancia - información local - como acerca de la distribución probabilística subyacente en la K-vecindad. La probabilidad P_i de que un ejemplo S_i pertenezca a la clase θ_i se calcula según la expresión (e3.1) donde p_i^j denota la probabilidad de que el K-vecino S_j sea de la clase θ_i. Este valor es inicialmente 1 para la categoría con que está etiquetada la instancia S_j y 0 para las restantes clases.

$$P_\theta(S_i) = \sum_{j=1}^{k} p_i^j \frac{1}{(1 + d(S_i, S_j))} \qquad \text{(e3.1)}$$

Atendiendo a este valor los autores presentan dos enfoques llamados *WilsonProb* y *WilsonTh*. En el primero se clasifica a S_i siguiendo la nueva regla de decisión, es decir, se asigna la clase θ con mayor valor de $P_\theta(S_i)$. Si la clasificación no es correcta entonces S_i es descartada. Por otra parte, la variante *WilsonTh* define un umbral, $0 < \mu < 1$, de modo que S_i es eliminada no solo si resulta mal clasificada sino también en caso de que la probabilidad de pertenecer a la clase que tiene predefinida no sobrepase el valor de μ. Los autores reportan una comparación con las propuestas de Wilson (1972) y Devijver & Kittler (1980) utilizando bases datos del repositorio UCI en una tarea de clasificación estimando el error mediante una validación cruzada de 5 particiones. En todos los casos las instancias son representadas mediante vectores. Respecto a los otros métodos estudiados en general

WilsonProb y *WilsonTh* presentan resultados parecidos. No obstante, es notable que independientemente del algoritmo utilizado, en 6 de los 14 experimentos realizados el proceso de edición condujo a un peor desempeño respecto a la clasificación con el conjunto no editado.

En general los enfoques propuestos intentan determinar si una instancia debe excluirse analizando todas las muestras disponibles, tanto las que pertenecen a la supuesta clase de la instancia analizada como aquellas que no. Una estrategia diferente es descrita por Guan *et al.* (2009), estos autores también incluyen información que proveen ejemplos no etiquetados, es decir de clase desconocida, en la decisión sobre la permanencia o no de una instancia en el conjunto editado. Los autores realizan comparaciones con las propuestas presentadas por Wilson (1972) y Tomek (1976a) reportando obtener mejores resultados.

3.2. Algoritmos de condensado.

El algoritmo conocido por *CNN* o algoritmo de Hart (Hart, 1968) del inglés *condensed nearest neighbor* marca el inicio del desarrollo de este tipo de procedimiento. Está basado en el concepto de *subconjunto consistente*, que es un subconjunto de los datos originales S suficiente para clasificar correctamente cada instancia en S empleando la regla 1-NN. El conjunto condensado $S\prime$ se construye paso a paso incluyendo instancias de S mal clasificadas por los elementos en $S\prime$. Respecto a los objetos atípicos estos tienden a ser incluidos en el conjunto editado ya que son mal clasificados erróneamente por sus vecinos. El procedimiento empleado no garantiza encontrar el *subconjunto consistente* de tamaño mínimo, problema de la clase *NP-Completo* según demuestra Wilfong (1991)

Gates (1972) propone un esquema estructurado en dos partes. Primero se aplica el algoritmo de Hart. Posteriormente, para cada instancia del conjunto editado, se verifica si al eliminarla no se pierde la condición de consistencia en cuyo caso queda definitivamente descartada. Aun

cuando usualmente se alcanzan niveles de compactación superiores no es posible garantizar que el conjunto editado tenga la menor talla posible, por lo que en numerosas ocasiones no se justifica el tiempo de cómputo adicional que se requiere.

A diferencia del enfoque de Hart, Swonger (1972) propone el método llamado *Iterative Condensation Algorithm*. Este enfoque permite excluir objetos del conjunto condensado. Además es tolerante a la presencia de instancias ruidosas.

Ullman (1974) presenta dos nuevas variantes. En la primera se establece un umbral μ llamado de *zona muerta*. Un objeto es descartado solo cuando la diferencia entre las distancias al objeto más cercano de igual clase y al de clase diferente sobrepase el valor de μ. La otra propuesta establece un orden para analizar las instancias basado en su distancia a las restantes clases.

Un ejemplo de algoritmo donde el conjunto editado incluye tanto instancias originales como otras artificiales es el descrito por Chang (1974). La idea consiste en comenzar con todos los ejemplos y en sucesivas iteraciones ir sustituyendo los pares de instancias más cercanos pertenecientes a la misma clase por un nuevo prototipo que las represente a las dos. Este proceso se repite mientras la capacidad de generalización no se deteriore. Distintos experimentos empleando bases de datos sintéticas y otras extensamente estudiadas como la *iris plant database* (Iris) muestran que se pueden alcanzar altos índices de reducción.

En (Ritter *et al.*, 1975) los autores añaden restricciones adicionales en la construcción del conjunto editado respecto a las implementadas por el algoritmo de Hart. Denominando *enemigo más próximo* de la instancia S_i al objeto S_j más cercano de clase distinta una nueva condición establece que en el conjunto editado debe incluirse al menos una instancia S'_i de igual clase que S_i de modo que S'_i esté más próxima a S_i que S_j. Los objetos S'_i se conocen como *asociados* de S_i. El conjunto que cumple tanto esta condición como la de *consistencia* se denomina *conjunto selectivo*. El algoritmo propuesto por los

autores se enfoca entonces en la construcción de un conjunto con esta propiedad que sea de tamaño mínimo.

Dasarathy (1994) también estudia la obtención de un conjunto consistente de tamaño mínimo. Retomando el concepto de objeto asociado utilizado por Ritter *et al.* (1975) se dice que S_j soporta a S_i si es su asociado. De acuerdo a este criterio se ordenan los objetos descendentemente según el número de instancias que soportan. El primero de esta lista se retiene, borrándose los objetos soportados por este. Este paso se repite mientras puedan eliminarse objetos, en este caso se vuelve a aplicar el algoritmo sobre las instancias que permanecen. Aunque el algoritmo descrito no cumple este propósito en la totalidad de los casos es un enfoque ampliamente utilizado en experimentos comparativos pues ofrece la posibilidad de encontrar conjuntos de talla bastante reducida de forma determinística.

Un ejemplo de la diversidad de enfoques propuestos para reducir el tamaño del conjunto de entrenamiento se encuentra en el trabajo de Skalak (1994). El autor propone dos métodos basados en selección aleatoria de instancias. Una variante consiste en generar soluciones, muestras de n instancias, mediante un procedimiento de tipo *Monte Carlo* verificando si el nuevo conjunto mejora o no la clasificación para luego devolver la mejor solución encontrada. El segundo enfoque propuesto realiza la selección mediante un *ascenso de colina por mutaciones aleatorias*. Se comienza con n instancias elegidas al azar en el conjunto original S y en cada iteración - *mutación aleatoria* - se sustituye una instancia del conjunto editado $S\prime$ por una en $S \sim S\prime$ ambas seleccionadas de forma aleatoria. Este proceso se repite un número dado de veces, 100 en el trabajo.

Cameron-Jones (1995) utiliza una función (J) de coste que depende del tamaño de los conjuntos original y condensado $S\prime$ así como de la cantidad de instancias mal clasificadas por los ejemplos en S. Atendiendo a esta función se proponen tres enfoques: *ELH*, *ELGrow* y *Explore* que varían el modo en que son seleccionadas las instancias. *ELH* comienza con un conjunto vacío y agrega instancias solo si estas minimizan J. *ELGrow* intenta además eliminar instancias previamen-

te incluidas en el conjunto condensado si esto disminuye el valor de J. La última propuesta es una extensión al procedimiento *ELGrow* que realiza borrados o inserciones de instancias de forma estocástica si esto minimiza J.

Chen & Józwik (1996) describen un método que posibilita controlar el tamaño del conjunto condensado al establecer la número de prototipos n que se desean. En este caso los datos se dividen en n grupos para seguidamente obtener su respectivo centro de gravedad. Cada grupo será reemplazado por su correspondiente centro, un prototipo etiquetado según la clase mayoritaria. Un efecto adicional de este procedimiento es que las instancias ruidosas quedarán descartadas.

Por otra parte, Kuncheva & Bezdek (1998) estudian varias estrategias de obtención de prototipos para un clasificador K-NN como la creación de instancias artificiales o la selección de un subconjunto de los datos originales mediante Algoritmos Genéticos o *Random Search*. Experimentos realizados con el repositorio Iris indican que la selección, particularmente mediante algoritmos genéticos, conduce a mejores resultados en términos del error de clasificación.

Un esquema similar a la regla *CNN* es el que describe Mitra *et al.* (2000) pero realizando la clasificación mediante Máquinas de Vectores de Soporte y seleccionando los vectores de soporte como prototipos. Resultados experimentales empleando las bases de datos *Cancer*, *Monks*-3 y *Heart* del repositorio UCI muestran que en los dos primeros casos la propuesta alcanza mayor condensado que la regla *CNN* lográndose incluso reducir el error en la clasificación. Este mismo autor propone otro algoritmo de condensado en Mitra *et al.* (2002) seleccionando puntos con una alta densidad local, es decir, aquellos con una distancia promedio menor a sus K-vecinos. Los resultados son validados empíricamente empleando algunas bases de datos del repositorio UCI.

En Jankowski & Grochowski (2004a) y Jankowski & Grochowski (2004b) puede encontrarse un estudio comparativo de varios algoritmos de edición.

3.3. Conclusiones parciales.

En el capítulo se examinó una extensa lista de algoritmos para reducir la talla del conjunto de entrenamiento de un clasificador y eliminar instancias ruidosas. En general, las validaciones experimentales se han realizado empleado conocidas bases de datos donde las instancias son representadas mediante vectores de atributos. Como se ha comprobado existe una gran variedad de enfoques, sin embargo seleccionar el óptimo en cada aplicación puede no ser simple debido a que en las diferentes bases de datos, distintos algoritmos ofrecen el mejor resultado.

Por otra parte, se ha señalado que en Reconocimiento de Formas Sintáctico Estructural se emplean otras representaciones como los árboles o las cadenas, motivando la necesidad de crear o adaptar esquemas de clasificación adecuados para este tipo de codificación. En el capítulo anterior se han citado varios trabajos donde se resuelven las tareas de recuperación, agrupamiento y construcción de prototipos para contornos representados por diferentes tipos de cadenas. En estos casos, particularmente en la clasificación mediante la regla de los K-vecinos más cercanos, también puede ser necesario descartar instancias no importantes o eliminar aquellas etiquetadas incorrectamente. Sin embargo, no abundan las referencias sobre la aplicación de las técnicas analizadas en el capítulo concretamente en esta área. Este hecho ha servido como incentivo a nuestro interés de aplicar las aproximaciones a la media entre dos contornos y la identificación de similitudes en algoritmos para reducir la talla y filtrar instancias ruidosas del conjunto de entrenamiento de un clasificador K-NN.

Parte II

Estudios basados en la Distancia de Levenshtein.

Capítulo 4

Detección de similitudes entre contornos.

Como se acotó en la parte introductoria, el problema de la detección de similitudes entre dos contornos se abordará como la identificación de similitudes entre los códigos de cadenas que los representan. En el capítulo se describirá un algoritmo heurístico que dados dos contornos S_1 y S_2 permite identificar pares de segmentos, uno en cada contorno, con los que se construirá un prototipo R que codifica los elementos comunes de ambos. Los segmentos en R son obtenidos mediante la operación de *fusión* definida en la siguiente sección. Este prototipo será empleado en un algoritmo para reducir la talla del conjunto de entrenamiento de un clasificador K-NN. Finalmente se detallarán los resultados de varios experimentos diseñados para validar empíricamente los algoritmos tratados.

4.1. Operación de fusión entre cadenas.

Sean S_1 y S_2 cadenas de Freeman la *fusión* entre cadenas se define como $R = Fusión(S_1, S_2)$ donde R satisface las restricciones (e4.1) y (e4.2). Es importante señalar que se sobreentiende que símbolo ε no está incluido en ninguna de las cadenas, es decir, sea por ejemplo $S_1 = \{1, \varepsilon, 2\}$ entonces se interpretará como $S_1 = \{1, 2\}$.

$$D(S_1, S_2) \geqslant D(S_1, R). \tag{e4.1}$$

$$D(S_1, S_2) \geqslant D(S_2, R). \tag{e4.2}$$

Si $D(S_k, S_l)$ mide que tan bien la cadena S_k es representada por S_l, entonces $D(S_k, S_j) < D(S_k, S_l)$ significa que S_k queda mejor representada por S_j que por S_l. Según este criterio, tanto S_1, S_2 como su media - o una aproximación - pueden satisfacer los requerimientos anteriores. En tal caso, se preferirá la cadena R que satisfaga (e4.3). Lógicamente esta tenderá a situarse a igual distancia de ambas cadenas por lo que será un mejor representante que otra cadena con mayor valor, la que estará sesgada en favor de S_1 o S_2.

$$argmin_R\{|D(R, S_1) - D(R, S_2)|\} \tag{e4.3}$$

Para definir la *fusión* entre S_1 y S_2 sean L_{S1} y L_{S2} sus longitudes respectivas, entonces la longitud de R será:

$$L_R = \left\lfloor \frac{L_{S1} + L_{S2}}{2} \right\rfloor \tag{e4.4}$$

Si $L_{S1} = L_{S2}$, una forma de construir R es tomar como su primer carácter $R[1]$ el símbolo m que minimiza $|w(m, S_1[1]) - w(m, S_2[1])|$. Los otros símbolos de R se obtienen de manera similar.

Como quiera, debe considerarse un esquema más general para obtener R. En la mayor parte de los casos $L_{S1} \neq L_{S2}$ por lo que el sencillo enfoque anterior no es adecuado. Sin perder generalidad, sea $L_{S1} > L_{S2}$, entonces al calcular la longitud de R mediante la fórmula (e4.4) se tiene que $L_{S2} \leqslant L_R < L_{S1}$. Esto significa que los símbolos de R no pueden obtenerse como la media de solamente un símbolo en S_1 y su correspondiente en S_2. En este caso cada $R[i]$ es calculado teniendo en cuenta los votos de todos los símbolos en S_1 y S_2, donde el peso W del j-ésimo símbolo de una cadena estará dado por la expresión (e4.5).

$$W = \begin{cases} 1 & \text{si} \quad j \geqslant i\frac{L_S}{L_R} \wedge (j+1) \leqslant (i+1)\frac{L_S}{L_R} \\ (i+1)\frac{L_S}{L_R} - j & \text{si} \quad i\frac{L_{S1}}{L_R} \leqslant j \leqslant (i+1)\frac{L_S}{L_R} \wedge (j+1) > (i+1)\frac{L_S}{L_R} \\ (j+1) - i\frac{L_S}{L_R} & \text{si} \quad j < i\frac{L_S}{L_R} \wedge i\frac{L_S}{L_R} < (j+1) < (i+1)\frac{L_S}{L_R} \\ \frac{L_S}{L_R} & \text{si} \quad j < i\frac{L_S}{L_R} \wedge (j+1) \geqslant (i+1)\frac{L_S}{L_R} \end{cases} \quad \text{(e4.5)}$$

Dado que en este caso cada símbolo representa una dirección d de Freeman, no tiene sentido multiplicarlo por su peso W y luego sumarlo para obtener el resultado de la votación. En lugar de esto, los pares $< d, W >$ se interpretan como vectores en un sistema de coordenadas polares con dirección d y magnitud W. Los símbolos de R se definen como la dirección del vector resultante al sumar cada $< d, W >$. La figura (4.1) muestra la cadena que resulta cuando se fusionan $S_1 = \{0, 0, 0, 0\}$ y $S_2 = \{2, 2\}$.

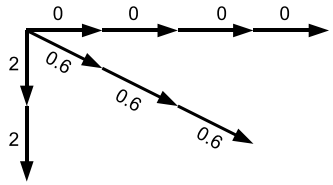


Figura 4.1: Resultado de la operación de fusión entre las cadenas $S_1 = \{0, 0, 0, 0\}$ y $S_2 = \{2, 2\}$.

Debe notarse que la definición dada para $Fusión(S_1, S_2)$ depende de los valores fijados para $w(a, \varepsilon)$ y $w(\varepsilon, b)$ para satisfacer las restricciones impuestas por (e4.1) y (e4.2). El siguiente ejemplo ilustra lo anterior.

Sean $S_1 = \{3, 1, 1, 4\}$ y $S_2 = \{1, 3, 3, 6\}$ de donde $R = Fusión(S_1, S_2) = \{2, 2, 2, 5\}$. Cuando $w(a, \varepsilon) = w(\varepsilon, b) = 0.5$ se tiene que $D(S_1, S_2) = 3$ mientras que $D(S_1, R) = D(S_2, R) = 4$. En cambio, haciendo que las inserciones y borrados tengan coste 1 se obtiene $D(S_1, S_2) = 5$ y $D(S_1, R) = D(S_2, R) = 4$ satisfaciéndose (e4.1)

y (e4.2). El ejemplo anterior muestra que los valores para $w(a, \varepsilon)$ y $w(\varepsilon, b)$ deben ser elegidos cuidadosamente.

4.2. Algoritmo para la detección de similitudes entre contornos.

La secuencia de operaciones Q con coste mínimo para transformar S_1 en S_2 establece un alineamiento entre los símbolos de ambas cadenas. Las sustituciones hacen corresponder a un símbolo en S_1 uno en S_2, por otra parte, los borrados e inserciones denotan respectivamente un símbolo de S_1 o S_2 al que le corresponde el símbolo ε. A modo de ejemplo, sean $S_1 = \{5, 5, 5, 6, 6, 6, 6, 5, 7, 7, 1, 1, 0, 1, 1, 3, 3, 1, 2, 3, 3\}$ y $S_2 = \{5, 5, 6, 6, 6, 5, 7, 7, 7, 7, 1, 1, 2, 3, 3, 3, 1, 1, 3, 3\}$ las representaciones mediante cadenas de Freeman de dos contornos. La tabla (4.1) muestra el coste de cada operación de edición.

Otro modo de ver el alineamiento es que Q transforma las cadenas S_1 y S_2, no necesariamente de la misma longitud, en nuevas cadenas $S_{1\prime}$ y $S_{2\prime}$ de igual longitud insertando el símbolo ε en estas. Q establece un correspondencia entre los caracteres de $S_{1\prime}$ y $S_{2\prime}$ situados en la misma posición, lo que se ilustra en la figura (4.2). Es decir, una subsecuencia $Q^{k,l}$ de Q establece una correspondencia entre $S_{1\prime}^{k,l}$ y $S_{2\prime}^{k,l}$. Luego, el coste $E_{Q^{k,l}}$ puede ser calculado como la suma de los costes individuales de las operaciones en $Q^{k,l}$. Por ejemplo, para $Q^{11,16} = \{w(1,1), w(1,1), w(0,\varepsilon), w(1,2), w(1,3), w(3,3)\}$ se tiene como resultado $E_{Q^{11,16}} = 0.0 + 0.0 + 2.0 + 1.0 + 2.0 + 0.0 = 5.0$ según los costes mostrados en la tabla (4.1).

5	5	5	6	6	6	6	5	7	7	1	1	0	1	1	3	3	1	2	3	3
5	5	6	6	6	5	7	7	7	7	1	1	ε	2	3	3	3	1	1	3	3

Figura 4.2: Correspondencia entre los símbolos de $S_{1\prime}$ y $S_{2\prime}$ dada por Q.

Tabla 4.1: Secuencia de edición de coste mínimo.

Q	Coste
$w(5,5)$	0.0
$w(5,5)$	0.0
$w(5,6)$	1.0
$w(6,6)$	0.0
$w(6,6)$	0.0
$w(6,5)$	1.0
$w(6,7)$	1.0
$w(5,7)$	2.0
$w(7,7)$	0.0
$w(7,7)$	0.0
$w(1,1)$	0.0
$w(1,1)$	0.0
$w(0,\varepsilon)$	2.0
$w(1,2)$	1.0
$w(1,3)$	2.0
$w(3,3)$	0.0
$w(3,3)$	0.0
$w(1,1)$	0.0
$w(2,1)$	1.0
$w(3,3)$	0.0
$w(3,3)$	0.0

El algoritmo se propone construir un prototipo R identificando pares $\langle S_{1\prime}^{k,l}, S_{2\prime}^{k,l} \rangle$ de segmentos correspondientes en los contornos. Algunos de estos pares se corresponderán con similitudes entre los contornos mientras que otros representarán secciones que no satisfacen el criterio establecido de semejanza. Es decir, se encontrarán dos conjuntos C_S y C_T de pares donde los elementos en C_S codifican segmentos de los contornos que cumplen el criterio de similitud mientras que C_T contiene información sobre los segmentos que no lo hacen. Cada elemento en estos conjuntos determina un segmento del contorno prototipo R.

Para determinar cuáles segmentos serán considerados similares y cuáles no, se introduce el parámetro $0 \leqslant T \leqslant 1$ llamado *factor de tolerancia*. Los segmentos $S_{1\prime}^{k,l}$ y $S_{2\prime}^{k,l}$ serán similares si $E_{Q^{k,l}} \leqslant D(S_1, S_2)T$, siendo el valor de T quien controla el criterio de similitud. Esto es, si se fija cercano a 0 el coste conjunto de las operaciones en $Q^{k,l}$ también deberá ser próximo a 0 para que los segmentos $S_{1\prime}^{k,l}$ y $S_{2\prime}^{k,l}$ sean considerados similares. En cambio, si T se toma cercano a 1 entonces se clasificarán como similares segmentos de contornos deter-

minados por un $Q^{k,l}$ con un coste similar a $D(S_1, S_2)$. De este modo, el parámetro T determina un conjunto C_S de subsecuencias disjuntas de Q que satisfacen la restricción impuesta por la expresión (e4.6).

$$\sum E_{Q^{k,l}} \leqslant D(S_1, S_2)T \mid Q^{k,l} \in C_S \ . \tag{e4.6}$$

Es decir los distintos segmentos de Q en C_S establecen mapeos entre segmentos de S_1 y S_2 cuyo coste conjunto de transformación, coste de los diferentes $Q^{k,l}$, no excede una fracción de la distancia total $D(S_1, S_2)$.

Dado que la restricción (e4.6) puede ser satisfecha por varios conjuntos, se preferirá aquel que abarque la mayor longitud posible de los contornos lo que se logra teniendo también en cuenta la condición (e4.7)

$$argmax_{C_S}\{\sum L_{Q^{k,l}} \mid Q^{k,l} \in C_S\} \ . \tag{e4.7}$$

Finalmente, las soluciones sujetas a (e4.8) serán favorecidas. Esto es, C_S contendrá unas pocas subsecuencias de Q pero de larga longitud en lugar de un gran número de subcadenas de corta longitud.

$$argmin_{C_S}\{|C_S|\} \tag{e4.8}$$

El conjunto que satisface las restricciones (e4.6), (e4.7) y (e4.8) es construido por un procedimiento heurístico que permite obtener una solución lo suficientemente buena. Para esto, primeramente es necesario definir una medida, ecuación (e4.9), que describe que tan buena es una subcadena de Q para ser incluida en C_S.

$$P_{Q^{k,h}} = \begin{cases} 2 & \text{si } Q^{n,k-1}, Q^{h+1,m} \in C_S. \\ 1 & \text{si } Q^{n,k-1} \in C_S \\ & \quad \wedge\ Q^{h+1,m} \ni C_S \text{ o viceversa} \ . \\ 0 & \text{si } Q^{n,k-1}, Q^{h+1,m} \ni C_S. \end{cases} \tag{e4.9}$$

Asignando los valores de $P_{Q^{k,h}}$ de la manera antes descrita, el procedimiento *BuscarSegmentosSimilares* permite construir C_S.

4.3. Construcción del prototipo.

Una vez identificados los segmentos similares de los contornos S_i y S_j se puede construir un prototipo R representándolos. En la figura (4.3) se describe gráficamente los distintos pasos para construir R. Como se comentó anteriormente, el conjunto C_S contiene información sobre los segmentos clasificados como similares en ambos contornos, mientras que en C_T quedan los segmentos no similares según el criterio establecido. Cada subcadena $Q^{k,l} \in C_S$, determinará los segmentos $S_{i\prime}^{k,l}$ y $S_{j\prime}^{k,l}$ que se utilizarán para calcular el correspondiente fragmento de R, dado por $R^{k,l} = Fusión(S_{i\prime}^{k,l}, S_{j\prime}^{k,l})$. Por otra parte, la longitud de los segmentos de $R^{g,h}$ originados por los elementos en $Q^{g,h} \in C_T$ también estará determinada por la expresión (e4.4). En este caso, cada $R^{g,h}[i]$ será el símbolo "?", que como se acotó anteriormente no se refiere a una dirección en particular. Esto provoca que se pierda cierta cantidad de información - direcciones - pérdida que cuantifica la ecuación en (e4.10) donde $A_?(S_x)$ es el número de símbolos "?" en la cadena S_x. Esta definición hace a I_R proporcional a $A_?(R)$, valor indirectamente determinado por T.

$$I_R = \frac{L_R - A_?(R)}{L_R} \tag{e4.10}$$

Finalmente, R se construye concatenando cada uno de los $R^{g,h}$ previamente ordenados por g. El siguiente ejemplo ilustra el funcionamiento del algoritmo descrito.

Función `BuscarSegmentosSimilares(`Q`,`T`)` :C_S

```
/* Q: secuencia de edición óptima para transformar S1 en S2   */
/* C_T: conjunto de todas las subsecuencias Q^{k,k}|k = 0..L_Q */
/* C_S = ∅: conjunto con los segmentos similares de S1 y S2   */
/* T: factor de tolerancia                                    */
```

mientras *($C_T \neq \varnothing$)* **y** *($\sum E_{Q^{k,l}} \leqslant D(S_1, S_2)T \mid Q^{k,l} \in C_S$)*
hacer

- $Q^{g,g} = argmin\{E_{Q^{k,k}} | Q^{k,k} \in C_T\}$: si hay dos o más seleccionar aquella con el mayor valor de $P_{Q^{g,g}}$;
- $C_S = C_S \cup Q^{g,g}$;
- $C_T = C_T \sim Q^{g,g}$;
- **seleccionar** $P_{Q^{g,g}}$ **:**
  - **caso** 2 :
    - $C_S = C_S \sim Q^{n,g-1}$;
    - $C_S = C_S \sim Q^{g+1,m}$;
    - $C_S = C_S \sim Q^{g,g}$;
    - $C_S = C_S \cup Q^{n,m}$;
  - **caso** 1 :
    - **si** *($Q^{n,g-1} \in C_S$)* **entonces**
      - $C_S = C_S \sim Q^{n,g-1}$;
      - $C_S = C_S \sim Q^{g,g}$;
      - $C_S = C_S \cup Q^{n,g}$;
    - **sino** /* $Q^{g+1,m} \in C_S$ */
      - $C_S = C_S \sim Q^{g+1,m}$;
      - $C_S = C_S \sim Q^{g,g}$;
      - $C_S = C_S \cup Q^{g,m}$;
    - **fin si**
  - **fin seleccionar**

fin mientras
devolver C_S;

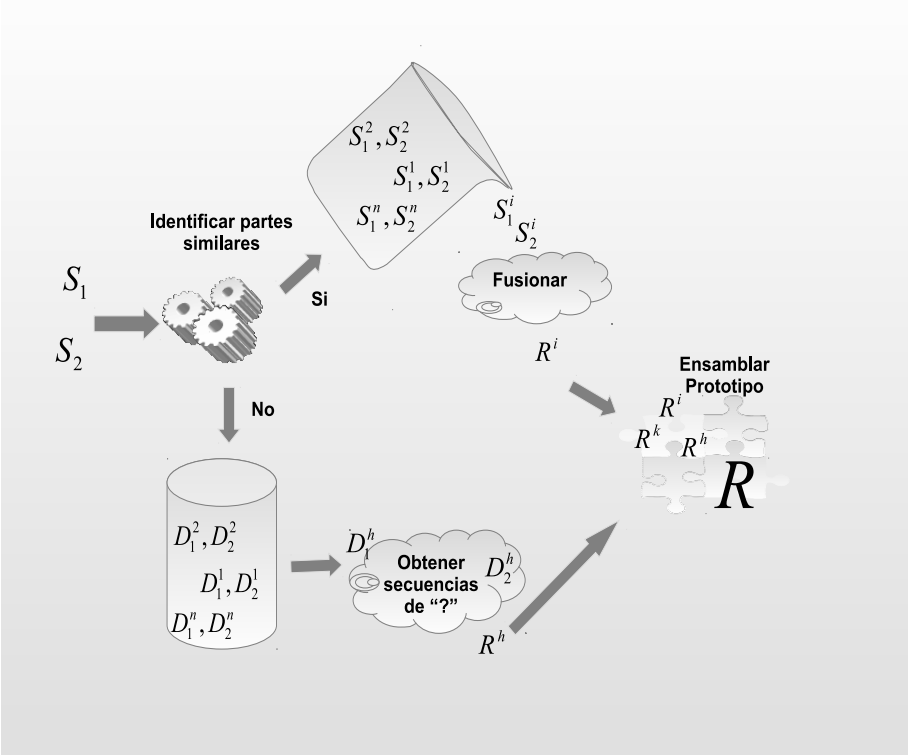


Figura 4.3: Esquema general del algoritmo propuesto.

4.4. Ejemplo del algoritmo para la detección de similitudes entre contornos.

Como antes, sean $S_1 = \{5,5,5,6,6,6,6,5,7,7,1,1,0,1,1,3,3,1,2,3,3\}$ y $S_2 = \{5,5,6,6,6,5,7,7,7,7,1,1,2,3,3,3,1,1,3,3\}$ las cadenas que representan los contornos en la figura (4.4a). Haciendo $T = 0.2$ se tiene que $C_S = \{Q^{1,5}, Q^{9,12}, Q^{16,21}\}$ y $C_T = \{Q^{6,8}, Q^{13,15}\}$. Las operaciones de edición en $Q^{1,5}$ determinan $S_{1\prime}^{1,5} = \{5,5,5,6,6\}$ y $S_{2\prime}^{1,5} = \{5,5,6,6,6\}$ cuya fusión es $R^{1,5} = \{5,5,5.5,6,6\}$. De $Q^{9,12}$ se obtiene $S_{1\prime}^{9,12} = \{7,7,1,1\}$ y $S_{2\prime}^{9,12} = \{7,7,1,1\}$ de donde $R^{9,12} = \{7,7,1,1\}$. Finalmente de $Q^{16,21}$ resultan $S_{1\prime}^{16,21} = \{3,3,1,2,3,3\}$ y $S_{2\prime}^{16,21} = \{3,3,1,1,3,3\}$ $R^{16,21} = \{3,3,1,1.5,3,3\}$. En el caso de C_T se tiene que de $Q^{6,8}$ resultan $S_{1\prime}^{6,8} = \{6,6,5\}$ y $S_{2\prime}^{6,8} = \{5,7,7\}$

de donde $R^{6,8} = \{\text{“?”}, \text{“?”}, \text{“?”}\}$. Por último de $Q^{13,15}$ se obtiene $S_{1\prime}^{13,15} = \{0, 1, 1\}$ y $S_{2\prime}^{13,15} = \{\varepsilon, 2, 3\}$ de donde $R^{13,15} = \{\text{“?”}, \text{“?”}\}$. La figura (4.4b) muestra las diferentes regiones en cada contorno.

Concatenando $R^{1,5}$, $R^{6,8}$, $R^{9,12}$,$R^{13,15}$ y $R^{16,21}$ se construye el nuevo prototipo:

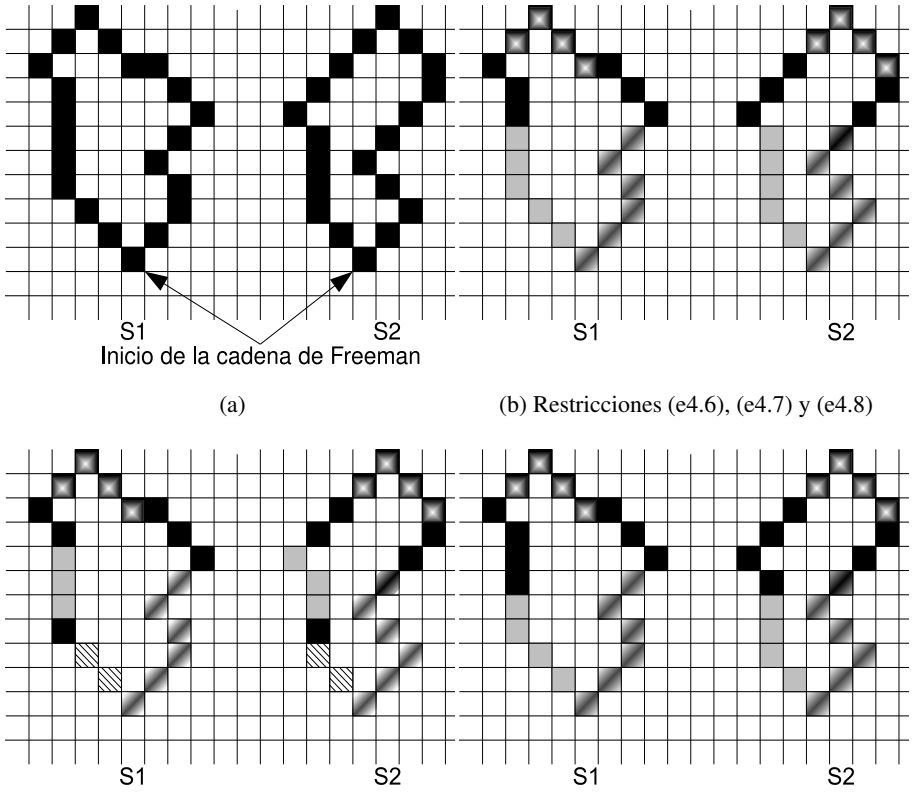
$$R = \{5, 5, 5.5, 6, 6, \text{“?”}, \text{“?”}, \text{“?”}, 7, 7, 1, 1, \text{“?”}, \text{“?”}, 3, 3, 1, 1.5, 3, 3\}.$$

Este puede “interpretarse” como que las cadenas, es decir los contornos, que representa pueden ser caracterizados por 5 segmentos. Primero un fragmento que luce similar a $R^{1,5}$ seguido de un segmento de longitud promedio 3 que no satisface el criterio de similitud. A este siguen otro fragmento similar a $R^{6,8}$, un segmento de longitud promedio 2 que tampoco es similar y por último un segmento de contorno similar a $R^{16,21}$.

Por último la figura (4.4) ilustra gráficamente el efecto de cada una de las restricciones (e4.6), (e4.7) y (e4.8) al identificar las similitudes entre los contornos en (4.4a). En (4.4b) se muestran los segmentos similares al tener en cuenta todas las restricciones anteriores. En (4.4c) se observan las regiones similares que se obtienen al mantener (e4.6) y (e4.7) pero sin atender a (e4.8), es decir, cubriendo la misma longitud de contorno pero permitiendo más segmentos de menor longitud. Por otra parte, en la figura (4.4d) pueden verse las regiones similares identificadas sin tener en cuenta (e4.7) por lo que, aun cuando $\sum E_{Q^{k,l}} \, |Q^{k,l} \in C_S$ y $|C_S|$ no varían respecto al ejemplo analizado, se tiene que es cubierta una menor longitud de los contornos.

4.5. Construcción de prototipos para un conjunto de instancias.

Asumiendo que a partir de dos contornos el algoritmo propuesto permite construir un prototipo que los represente adecuadamente, entonces puede aplicarse para obtener un grupo de prototipos que describa



(a)

(b) Restricciones (e4.6), (e4.7) y (e4.8)

(c) Restricciones (e4.6) y (e4.7)

(d) Restricciones (e4.6) y (e4.8)

Figura 4.4: Efecto de cada una de las restricciones (e4.6), (e4.7) y (e4.8) al identificar las similitudes entre los contornos que representa (4.4a). En (4.4b), (4.4c) y (4.4d) señalados con una misma textura se muestran los segmentos similares, mientras los píxeles en negro indican los que no lo son. En cada caso se han indicado las restricciones que se tuvieron en cuenta.

un determinado conjunto de contornos. Debido a que el algoritmo solo permite calcular el prototipo representativo para dos instancias a la vez, es necesario definir cómo utilizarlo para construir prototipos para un conjunto S donde $|S| > 2$. En este caso, se ha seguido el esquema descrito por Sánchez *et al.* (2002). La idea consiste en tomar pares de contornos, construir el prototipo que los represente y que los sustituirá en S. Este procedimiento se repite en el conjunto de todas las instancias de una misma clase y mientras se cumplan determinadas restricciones que constituirán la condición de parada; concretamente que al construir el prototipo la pérdida de información caracterizada por (e4.10) no sobrepase un valor previamente fijado. Es decir, mientras $I_R \leqslant P|0 \leqslant P \leqslant 1$ donde P, llamado *factor de persistencia* es un parámetro fijado por el usuario para detener el proceso cuando la pérdida de información sobrepasa el valor deseado.

Respecto al orden para elegir los pares de contornos en S, se seleccionan aquellos que se encuentren a menor distancia por dos razones fundamentales. Primero, al tener en cuenta el principio de que la calidad del prototipo depende en gran medida del grado de similitud entre las instancias que le dieron origen. En segundo lugar, tomar los pares de instancias de esta forma permite asumir que si al concluir el proceso aún quedan en el conjunto instancias originales, estas son miembros de la clase bastante diferentes del resto de los elementos. La función *ConstruirPrototipos* detalla los pasos descritos para construir prototipos caracterizando a un conjunto de instancias.

4.6. Análisis del coste computacional.

Como se ha explicado, para identificar los segmentos similares en los contornos S_1 y S_2 se requiere la realización de varios pasos. Primeramente es necesario calcular la distancia de Levenshtein, que, según la implementación propuesta por Wagner & Fischer (1974) requiere $\mathcal{O}(L^2)$, donde $L = max\{L_{S1}, L_{S2}\}$. Mediante este procedimiento también se puede encontrar la secuencia de operaciones de edición con coste mínimo Q en tiempo proporcional a $\mathcal{O}(L)$.

Función `ConstruirPrototipos(`S`,`μ`)`:S

```
/* S: conjunto de instancias para construir prototipos        */
/* μ: umbral para el factor de persistencia                   */
```

mientras $|S| > 1$ **hacer**
  encontrar $S_1, S_2 \in S$ tal que $D(S_1, S_2)$ sea mínima;
  $R = Fusión(S_1, S_2)$;
  calcular P;
  si $P > \mu$ **entonces**
    $S \leftarrow S \cup R$;
    $S \leftarrow S \sim S_1$;
    $S \leftarrow S \sim S_2$;
  fin si
fin mientras
devolver S;

Un segundo paso implica la selección de los segmentos de los contornos que satisfacen las restricciones (e4.6), (e4.7) y (e4.8). Esto requiere examinar cada operación en Q para decidir si será incluida o no en C_S. En el peor caso L pares $Q^{k,k}$ deben ordenarse por su valor $P_{Q^{k,k}}$ lo que puede hacerse en tiempo $\mathcal{O}(L \log L)$. De lo anterior resulta que el coste computacional del algoritmo es proporcional a $\mathcal{O}(L) + \mathcal{O}(L \log L) + \mathcal{O}(L^2)$ que de acuerdo a la regla de la suma (Aho *et al.*, 1983) queda $\mathcal{O}(L^2)$.

4.7. Resultados experimentales.

Luego de implementada nuestra propuesta un paso fundamental consiste en mostrar mediante diversos experimentos la efectividad del enfoque descrito. Es decir, partiendo de las cadenas de Freeman que codifican dos contornos y un determinado criterio de similitud, se debe obtener una tercera cadena que represente un contorno prototipo de los anteriores. En este las zonas con un grado de semejanza específico quedarán codificadas mediante direcciones reales y aquellas que no, se describen empleando secuencias del símbolo "?". En la sección se

presentan y analizan los resultados de varios experimentos diseñados para este propósito. En primer lugar, se realiza una valoración empírica del procedimiento de fusión entre contornos descrito en el epígrafe (4.1). Un segundo experimento está dirigido a evaluar el desempeño de un clasificador K-NN utilizando un conjunto preprocesado mediante el procedimiento abordado en el apartado (4.5).

4.7.1. Validación del procedimiento de fusión entre cadenas.

Según se acotó en la sección (4.1), el algoritmo de fusión no garantiza que la cadena resultante R cumpla los requerimientos dados por las expresiones (e4.1) y (e4.2), mostrándose que en el resultado influyen los valores seleccionados como coste para las operaciones de inserción y borrado.

Para evaluar la definición dada, se desarrollaron dos experimentos involucrando las siguientes variables:

Variable independiente: Algoritmo de fusión entre cadenas de Freeman.

Definición conceptual: Se utilizará el algoritmo descrito en el epígrafe (4.1) para, dadas dos cadenas de Freeman representando contornos construir una nueva cadena que satisfaga las restricciones impuestas en (e4.1) y (e4.2).

Variable dependiente: Cumplimiento o no de los requerimientos dados por (e4.1) y (e4.2)

Definición conceptual: Las restricciones indican que la cadena resultado de la fusión representa mejor a las cadenas fusionadas de lo que estas lo hacen entre ellas.

Definición operacional: Para medir esta variable es preciso implementar un algoritmo que permita el cálculo de la distancia de

edición de Levenshtein. Los costes asociados a cada operación de edición se toman de acuerdo a como se definieron en la sección (1.5.2). Se deben tomar dos cadenas de Freeman, fusionarlas y comprobar que la cadena resultante cumpla las especificaciones establecidas.

En un análisis preliminar, se analizaron todas las posibles fusiones dos a dos sobre un conjunto de 100 cadenas de Freeman de longitud 100 generadas aleatoriamente para comprobar si el algoritmo de fusión descrito es capaz de obtener una cadena que cumpla (e4.1) y (e4.2). En esta prueba, la cadena resultante R satisfizo las restricciones impuestas en el $100\,\%$ de los casos.

En un segundo experimento se extrajo una muestra de 2 instancias por clase procedentes de las bases de datos que se describen en la sección (1.5.5), las características de estos subconjuntos se describen en la tabla (4.2). Las fusiones dos a dos se realizaron sobre cada posible par de instancias en este conjunto, es decir, fueron fusionadas no solo instancias pertenecientes a la misma clase, sino pares heterogéneos para asegurar una mayor diversidad en las cadenas en cuanto a clases representadas y longitudes de las cadenas de Freeman. En este caso, el $99.2\,\%$ de las cadenas obtenidas en la fusión cumplieron los requerimientos impuestos por (e4.1) y (e4.2).

Tabla 4.2: Características de los datos empleados en el experimento para validar la operación de fusión entre cadenas.

Corpus	Clases	Instancias x Clase	Longitud de las cadenas		
			Promedio	Max.	Min.
NIST-19	26	2	200 ± 70	381	41
USPS	10	2	150 ± 40	235	87
MPEG-7	11	2	1400 ± 900	3431	281

4.7.2. Validación del algoritmo de detección de similitudes entre contornos.

El método desarrollado se propone ser adecuado para, a partir de dos contornos, construir un prototipo de estos empleando aquellas partes con grado de similitud mayor que uno fijado. Luego, asumiendo como cierta esta hipótesis, el algoritmo podría emplearse para obtener un grupo de prototipos que caractericen un conjunto dado de contornos lo que puede conducir a disminuir el tiempo de respuesta un clasificador K-NN, o identificar aquellos elementos comunes a una familia de contornos. En este sentido se han propuesto diferentes trabajos abordados previamente.

Acorde a lo anterior, se han organizado un grupo de experimentos cuya descripción general se ilustra en la figura (4.5). El objetivo es mostrar que aplicando el método propuesto en la sección (4.5) es posible reducir las bases de datos manteniendo la relación compresión/desempeño en la clasificación razonable.

En los experimentos se consideran las siguientes variables:

Variable independiente: Identificación de partes similares entre los contornos, atendiendo a un criterio de similitud dado.

Definición conceptual: Dos segmentos provenientes de diferentes contornos se consideran similares si la distancia de edición entre ellos no sobrepasa cierto umbral.

Definición operacional: Estas partes similares son obtenidas al aplicar el algoritmo propuesto, que permite calcular un conjunto de segmentos similares de acuerdo al criterio definido y que además cubren la mayor extensión posible del contorno.

Variable dependiente: Conjunto de prototipos representativos.

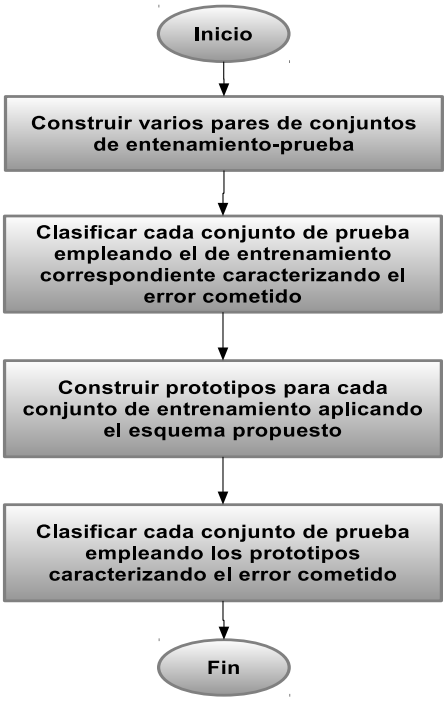


Figura 4.5: Esquema general de los experimentos de validación.

Definición conceptual: Se entiende por prototipos representativos aquellos que cumplen determinado criterio de representatividad respecto al grupo de contornos. En este caso, dicho criterio se define como la variación del error cometido al clasificar ciertas instancias de prueba, empleando un conjunto de entrenamiento y el error al clasificar con los prototipos obtenidos para dicho conjunto.

Definición operacional: Para evaluar el criterio de representatividad es preciso medir el desempeño alcanzado en la clasificación, en términos del error absoluto, por el conjunto de contornos, y comparar este valor con el error obtenido al clasificar empleando los prototipos. Si dicha diferencia es menor que un valor fijado a

conveniencia del usuario, se dice que los prototipos son representativos.

Con el objetivo validar los resultados, y tener un mejor criterio sobre el papel de los diferentes parámetros en el desempeño del algoritmo, se realizó una validación cruzada de 4 particiones a partir de los datos descritos en la sección (1.5.5).

Dado que uno de los factores que influyen en el valor de la distancia de edición entre dos cadenas es la longitud de estas, se prestó especial atención a los valores promedio y desviación típica de la longitud para las cadenas de Freeman que representan cada una de las instancias seleccionadas para los experimentos. En todas las bases de datos estudiadas puede observarse variaciones en las longitudes promedios de cada clase. A pesar de esto, exceptuando las instancias correspondientes al dígito "*5*" y a "*phone*", ninguna otra clase muestra diferencias sustanciales respecto al resto pues al menos otra tiene una longitud promedio similar.

Otras variables que pueden afectar los resultados de la clasificación y la correcta separación de las instancias de una clase respecto al resto son, las distancias *intra-clase* definida como la distancia promedio entre todas las instancias de una clase, y la *inter-clase* que es el promedio de la distancia de todas las instancias de una clase a las instancias de las otras clases. Los datos compilados, ver tablas (7.1), (7.2) y (7.3) de los anexos, muestran una notable diferencia en el comportamiento de estos parámetros en las distintas bases de datos. En el caso de los caracteres literales existen clases donde incluso la distancia *intra-clase* es mayor que la *inter-clase* con alguna otra. Tal es el caso de las instancias correspondientes a los caracteres "*D*","*G*","*H*","*I*","*W*" que se encuentran como promedio más próximas a las clases "*0*","*C*","*A*","*O*","*V*" respectivamente. Si bien esta situación no se produce en la base de dígitos también en esta pueden encontrarse pares de clases muy similares, como es el caso de las correspondientes a los dígitos "*8*" y "*0*" o "*4*" y "*9*". A diferencia de las anteriores, en la base MPEG-7 es notable la diferencia entre la distancia *intra-clase* de cualquier clase y la *inter-clase* hasta la clase más

próxima, es decir, al menos con respecto a estas variables existe una total diferenciación entre las instancias de una clase y las restantes.

Definiendo por Tr_i y Te_i los conjuntos de entrenamiento y prueba de la i-ésima partición respectivamente, la primera parte del experimento consistió en clasificar mediante la regla 1-NN a cada conjunto de prueba empleando el de entrenamiento correspondiente para medir el error E_{Or} cometido como porcentaje de instancias incorrectamente clasificadas. La tabla (4.3) muestra el valor promedio de E_{Or} en cada uno de los conjuntos estudiados y la respectiva desviación típica (σ).

Tabla 4.3: Valores promedios de E_{Or} *(error en la clasificación)* en las diferentes particiones y conjuntos de datos.

Partición	Corpus		
	NIST-19	USPS	MPEG-7
0	13.3	1.5	5.4
1	14.0	3.5	0.0
2	15.5	1.5	1.8
3	12.1	0.5	1.8
Promedio	13.7	1.7	2.3
σ	1.4	1.2	2.3

Luego, el procedimiento descrito en la sección (4.5) fue aplicado de forma independiente sobre las instancias de una misma clase en cada Tr_i para obtener un nuevo conjunto PTr_i de menor tamaño, o a lo sumo igual, que Tr_i, donde $100 * \frac{|PTr_i|}{|Tr_i|}$ es el porcentaje que representa el tamaño de PTr_i respecto al de Tr_i. Para medir la reducción del tamaño del conjunto procesado respecto al original se define en (e4.11) el *índice de reducción de instancias*. En este paso se probaron diferentes combinaciones de los valores de T y P variando cada uno de estos en 0.2 en el intervalo $[0, 1]$ a partir de 0.1, para un total de veinticinco posibles combinaciones. Estos fueron los únicos parámetros estudiados ya que, manteniendo fijos los costes de las operaciones de edición, el comportamiento del algoritmo de fusión queda totalmente determinado por T. En la etapa de construcción del prototipo, una vez fijado T el otro parámetro cuya modificación puede conducir a resultados diferentes es P.

$$\%D = 100 - 100 * \frac{|PTr_i|}{|Tr_i|}. \quad \text{(e4.11)}$$

Finalmente, las instancias en cada Te_i fueron clasificadas nuevamente, pero esta vez tomando como conjunto de entrenamiento el correspondiente PTr_i, denominando por E_{Proc} el porcentaje de instancias mal clasificadas en este caso. Para medir cómo el error en la clasificación se afecta cuando PTr_i es utilizado en lugar de Tr_i se define $\Delta E = E_{Proc} - E_{Or}$ llamado *coeficiente de variación del error*. Tanto $\%D$ como ΔE caracterizan el comportamiento del algoritmo dado que se espera aumentar el *índice de reducción de instancias* sin aumentar sensiblemente el error en la clasificación.

Las tablas (4.5) y (4.4) muestran el promedio y la correspondiente desviación típica para los valores de ΔE y $\%D$ obtenidos en cada partición del corpus NIST-19. Se aprecia que para algunos valores de los parámetros, como por ejemplo $T = 0.1$ y $P = 0.5$ el algoritmo se comporta satisfactoriamente, con un *índice de reducción de instancias* del $45.6\,\%$ mientras el incremento del error fue de $4.9\,\%$. No obstante, en general no se alcanzan altos valores para $\%D$ sin afectar perceptiblemente ΔE.

Tabla 4.4: Valores de ($\%D$) promedios (4 particiones) para los diferentes valores de T y P (NIST-19).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	76.0±2.0	60.6±0.8	46.0±2.0	17.4±0.3	0.0±0.0
0.3	92.8±1.1	89.5±1.0	78.8±0.8	45.8±1.5	0.8±0.2
0.5	96.5±0.2	95.8±0.4	92.5±1.1	80.5±0.5	9.2±1.4
0.7	96.7±0.0	96.7±0.0	96.2±0.3	94.0±0.6	43.3±0.8
0.9	96.7±0.0	96.7±0.0	96.7±0.0	96.6±0.1	91.5±1.8

Como se apuntó anteriormente, en el caso de la base de datos USPS se obtuvieron menores valores para E_{Or}, es decir, al clasificar las instancias en los conjuntos de pruebas utilizando los conjuntos de entrenamiento originales. En las tablas (4.6) y (4.7) pueden observarse respectivamente los valores para $\%D$ y ΔE obtenidos para las diferentes combinaciones de T y P. En esta base de datos también se

Tabla 4.5: Incremento del error (ΔE) promedios (4 particiones) al clasificar empleando las bases de datos reducidas según la tolerancia y persistencia indicadas (NIST-19).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	24.0±2.0	9.0±3.0	4.9±1.3	2.8±0.9	0.0±0.0
0.3	49.0±3.0	41.0±5.0	19.0±3.0	5.5±1.6	0.1±0.2
0.5	54.0±3.0	51.0±3.0	44.0±3.0	20.0±3.0	1.2±1.4
0.7	57.0±4.0	57.0±4.0	56.0±5.0	51.0±3.0	7.1±1.6
0.9	60.1±1.4	60.1±1.4	60.1±1.4	60.0±1.3	51.0±4.0

constata que aunque no para todos los valores de los parámetros se obtiene un buen radio de $\%D$ respecto a ΔE, sí es posible en algunos casos como cuando se toma $T = 0.1$ y $P = 0.5$.

Tabla 4.6: Valores de ($\%D$) promedios (4 particiones) para los diferentes valores de T y P (USPS).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	90.3±0.6	73.0±2.0	53.1±1.1	17.0±2.0	0.0±0.0
0.3	96.7±0.0	96.6±0.1	91.6±1.3	62.0±3.0	0.8±0.3
0.5	96.7±0.0	96.7±0.0	96.7±0.0	92.4±0.9	10.0±4.0
0.7	96.7±0.0	96.7±0.0	96.7±0.0	96.6±0.1	56.0±3.0
0.9	96.7±0.0	96.7±0.0	96.7±0.0	96.7±0.0	94.0±1.9

Tabla 4.7: Incremento del error (ΔE) promedios (4 particiones) al clasificar empleando las bases de datos reducidas según la tolerancia y persistencia indicadas (USPS).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	19.0±3.0	3.0±1.2	0.8±1.2	0.9±0.3	0.0±0.0
0.3	18.0±5.0	18.0±5.0	12.0±4.0	2.6±1.4	0.0±0.0
0.5	21.0±4.0	21.0±4.0	21.0±4.0	11.0±5.0	0.5±0.4
0.7	25.0±6.0	25.0±6.0	25.0±6.0	23.0±5.0	3.8±1.3
0.9	30.0±3.0	30.0±3.0	30.0±3.0	30.0±3.0	28.0±8.0

De forma similar a la base de datos USPS, en el caso de la MPEG-7 los valores promedios para E_{Or} fueron pequeños respecto al corpus NIST-19. Al analizar los datos mostrados en las tablas (4.8) y (4.9) se comprueba como en general en esta base de datos se pueden obtener valores elevados para $\%D$ acompañados de pequeños valores para ΔE, registrándose casos donde se redujo el tamaño del conjunto de

entrenamiento en un $80.8\,\%$ mientras que no se aumentó el error en la clasificación, resultado obtenido al tomar $T = 0.7$ y $P = 0.9$.

Tabla 4.8: Valores de ($\%D$) promedios (4 particiones) para los diferentes valores de T y P (MPEG-7).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	86.2±0.3	84.2±0.9	81.2±0.5	61.0±2.0	12.0±0.8
0.3	86.7±0.0	86.4±0.3	84.8±0.5	82.4±1.5	24.0±3.0
0.5	86.7±0.0	86.7±0.0	86.7±0.0	85.6±0.6	69.0±4.0
0.7	86.7±0.0	86.7±0.0	86.7±0.0	86.2±0.6	80.8±0.9
0.9	86.7±0.0	86.7±0.0	86.7±0.0	86.7±0.0	86.4±0.6

Tabla 4.9: Incremento del error (ΔE) promedios (4 particiones) al clasificar empleando las bases de datos reducidas según la tolerancia y persistencia indicadas (MPEG-7).

T/P	0.1	0.3	0.5	0.7	0.9
0.1	6.0±2.0	1.8±1.5	0.5±0.9	-0.5±0.9	0.0±0.0
0.3	7.0±2.0	5.0±3.0	0.0±0.0	0.0±0.0	0.0±0.0
0.5	8.0±3.0	8.0±3.0	8.0±3.0	0.9±1.0	-0.5±0.9
0.7	7.0±3.0	7.0±3.0	7.0±3.0	4.0±2.0	0.0±0.0
0.9	8.0±4.0	8.0±4.0	8.0±4.0	8.0±4.0	6.0±4.0

La tabla (4.10) muestra un resumen de los resultados obtenidos tomando $T = 0.1$ y $P = 0.5$ en las distintas particiones y datos.

Tabla 4.10: Valores para $\%D$ y ΔE al tomar $T = 0.1$ y $P = 0.5$.

	NIST-19		USPS		MPEG-7	
Partición	$\%D$	ΔE	$\%D$	ΔE	$\%D$	ΔE
0	48.3	3.8	52.5	0.0	81.2	0.0
1	44.9	4.8	53.0	0.5	80.6	0.0
2	43.3	4.2	54.7	0.0	81.8	1.8
3	45.8	6.7	52.3	2.5	81.2	0.0
Promedio	46.0	4.9	53.1	0.75	81.2	0.4
σ	2.1	1.3	1.1	1.2	0.5	0.9

Como se ha explicado, el diseño de los experimentos se encaminó a comparar los resultados obtenidos al fijar diferentes valores para T y P. Es decir, analizar el comportamiento de $\%D$ y ΔE si por ejemplo, se mantiene constante T variando P. Esto permitió determinar

las mejores o peores combinaciones para estos parámetros en los experimentos realizados con cada base datos.

De lo anterior se desprende que un criterio importante para evaluar la relación de los valores de T y P con los resultados es comparar si esta relación es similar en las diferentes conjuntos estudiados. Como se muestra en la figura (4.6a), cuando T es constante y se cambia P se observa una fuerte correlación de la variación del error en cada base de datos. Al dejar fijo P y variando T se llegan a conclusiones similares, como ilustra la figura (4.6b). De las gráficas mostradas en las figuras (4.6c) y (4.6d) se desprenden iguales conclusiones respecto a $\%D$.

Al comparar los resultados obtenidos en los diferentes corpus, se observa que estos son mejores en los correspondientes a MPEG-7 y USPS. Una posible explicación para esto es que la base de datos NIST-19 contiene un número mayor de clases. Otros elementos que muestran relación directa con los resultados son el comportamiento de las distancias *intra* e *inter* clase. Con respecto a estas variables se observan mejores resultados en las bases de datos donde en general la distancia *intra-clase* es significativamente menor que la distancia *inter-clase* a cualquiera de las restantes clases del conjunto, es decir, las bases MPEG-7 y USPS.

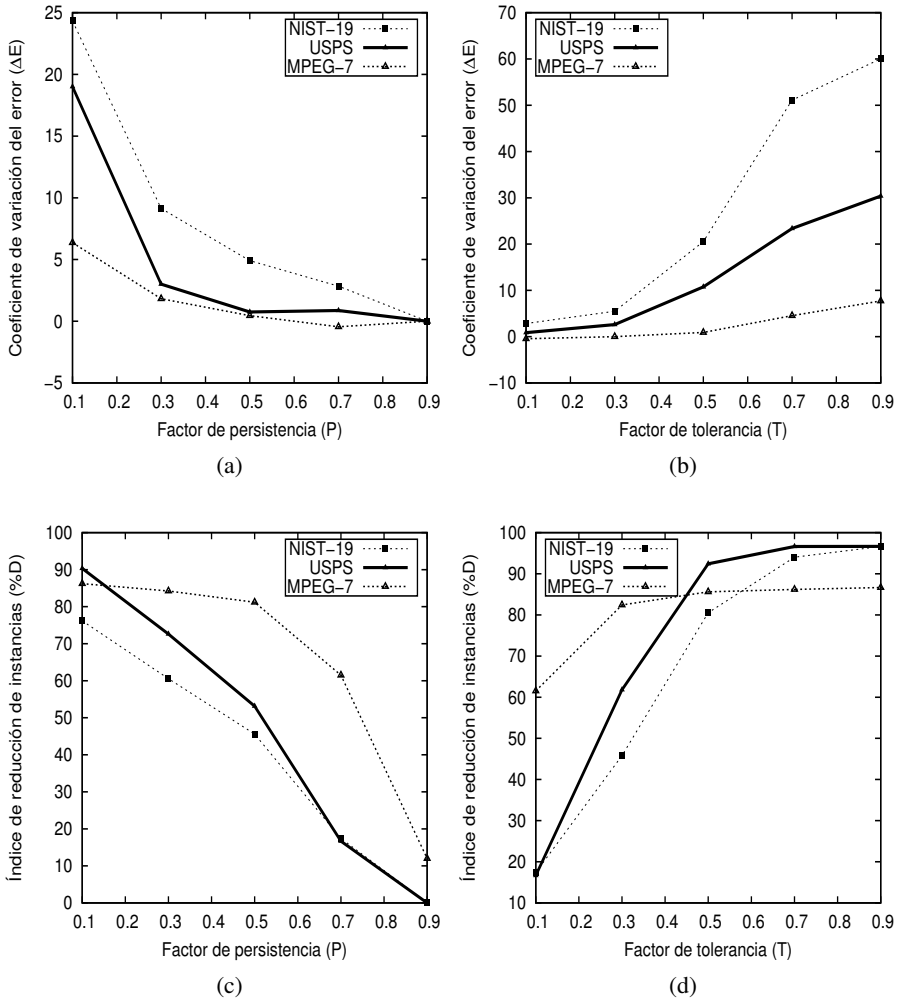


Figura 4.6: Valores para ΔE y $\%D$ variando P y haciendo $T = 0.1$ (4.6a) y (4.6c). Comparación al hacer constante $P = 0.7$ y variar T (4.6b) y (4.6d).

4.8. Conclusiones parciales.

De acuerdo a los objetivos trazados, en el capítulo se ha propuesto un algoritmo para construir un prototipo que representa a dos contornos directamente de la codificación de estos mediante cadenas de Freeman. El esquema se basa en empleo de la distancia de edición de Levenshtein y en la cadena de operaciones con coste mínimo para seleccionar regiones semejantes entre los contornos. Con estas y empleando secuencias del símbolo "?" se conforma el contorno prototipo.

Es preciso notar que el enfoque presentado puede extenderse a otras representaciones, siendo necesario definir una forma para discriminar aquellos pares de partes en los contornos que cumplen los criterios establecidos y una función que dados estos calcule su correspondiente en la cadena resultante.

Se mostró que el algoritmo descrito presenta un coste computacional proporcional a $\mathcal{O}(L^2)$; en este aspecto, comparable a otros recogidos en la literatura previamente consultada.

Resultados experimentales indican que este enfoque puede emplearse para caracterizar una clase mediante prototipos que codifican elementos comunes a los contornos. Así mismo, puede ser adecuado para reducir la talla de un conjunto de datos sin reducir notablemente su poder de representación en el ámbito de una tarea de clasificación mediante la regla de los K-vecinos más cercanos ya que en algunas de las pruebas se logró obtener una reducción cercana al $80\,\%$ con un incremento del error del $0.45\,\%$.

Capítulo 5

Aproximación a la media entre dos contornos.

Dado un conjunto de contornos, la extensión del concepto *promedio* a estos es una necesidad en diferentes aplicaciones como la definición de Mapas Auto-organizados para cadenas, identificación de agrupamientos o los algoritmos de condensado entre otros ejemplos. Los enfoques propuestos para este propósito dependerán del formalismo seleccionado para representar los contornos. En el caso de los códigos de cadenas, esta extensión puede darse por el concepto de *cadena media*, problema abordado desde muy diversas aristas. En el capítulo se explicarán diferentes enfoques para obtener la media entre dos cadenas de Freeman representando contornos así como un nuevo algoritmo de edición basado en la propuesta descrita por Wilson (1972) que requiere el cálculo de la media entre contornos.

5.1. Algoritmo para el cálculo del contorno medio.

El enfoque propuesto permite construir la media R entre dos cadenas S_1 y S_2 mediante la inclusión de ciertos símbolos de estas en R. Para determinar cuándo incluir o no un símbolo cada operación en la secuencia de edición con coste mínimo Q es examinada para ver cómo se afectarán $D(R, S_1)$ y $D(R, S_2)$ dado que se busca una cadena satisfaciendo (e1.2) y (e4.3). Estos requerimientos indican que

se preferirá una cadena R equidistante de S_1 y S_2 y que a su vez se encuentre lo más cerca posible de ambas.

Como se explicó en la sección (1.5.2) las operaciones de *borrado* en Q involucran un símbolo de S_1, las *inserciones* uno de S_2 mientras las *sustituciones* relacionan un símbolo de S_1 con uno en S_2. El algoritmo se basa en la idea de que incluyendo o no el símbolo involucrado en la operación $q_i \in Q$ se afectaran las distancias de R a S_1 y S_2. En el caso de las *inserciones* y *borrados* se verifican los siguientes casos:

La operación q_i es aceptada:

- Si q_i es un *borrado* entonces un símbolo de S_1 será incluido en R haciendo que esta se parezca a S_1.

- q_i es una *inserción*, en este caso se incluirá un símbolo de S_2 en R que se hará similar a S_2.

En caso de que q_i sea rechazada:

- q_i es una *inserción*, en este caso el símbolo de S_2 será excluido de R que permanecerá similar a S_1.

- q_i es un *borrado*, entonces el símbolo excluido de R será de S_1 haciendo que R se parezca a S_2.

Analizando detenidamente cada posible operación se puede explicar cómo incluyendo o no un símbolo se afectarán $D(R, S_1)$ y $D(R, S_2)$.

Para las *inserciones* sea b^i_{S2} el i-ésimo símbolo de S_2. La operación $q_i = w(\varepsilon, b^i_{S2})$ indica la inserción de este símbolo en S_1 para obtener R. Suponiendo que se decida incluir b^i_{S2}, operación aceptada, entonces puede esperarse que la secuencia de edición óptima para transformar R en S_2 no involucre la inserción de este símbolo en R dado que fue incluido previamente. De acuerdo a lo anterior $D(R, S2)$ no

cambiará mientras $D(R, S_1)$ lo hará en $w(b^i_{S2}, \varepsilon)$ dado que el símbolo será borrado en este caso. Si se decide no incluir b^i_{S2}, esto es rechazar la operación, entonces $D(R, S_2)$ se incrementará en $w(\varepsilon, b^i_{S2})$ dado que b^i_{S2} debe insertarse en R para obtener S_2. La distancia de R a S_1 no se afectará.

A su vez, para los *borrados* aceptados $w(b^i_{S1}, \varepsilon)$ el símbolo b^i_{S1} será insertado en R, de este modo $D(R, S_1)$ permanecerá inalterada mientras que $D(R, S_2)$ se verá incrementada en $w(b^i_{S1}, \varepsilon)$. En caso de rechazarse $D(R, S_1)$ aumentará en $w(\varepsilon, b^i_{S1})$ en tanto $D(R, S_2)$ no sufrirá modificaciones.

Cuando se tratan las sustituciones $w(b_{S1}, b_{S2})$ siempre será incluido un símbolo en R, pero cuando sea posible se elegirá un símbolo m que satisfaga (e5.1) y (e5.2) en lugar de b_{S1} o b_{S2}. Este criterio intenta hacer a R similar a ambas S_1 y S_2 de modo que las distancias de R a las cadenas crecerán en $w(b_{S1}, m)$ y $w(m, b_{S2})$ respectivamente.

$$w(b_{S_1}, b_{S_2}) \geqslant w(b_{S_1}, m) + w(m, b_{S_2}). \tag{e5.1}$$

$$argmin_m\{|w(m, b_{S_1}) - w(m, b_{S_2})|\}. \tag{e5.2}$$

Los procedimientos FMSC y *BuscarOp* implementan el algoritmo, explorando las diferentes alternativas de incluir o no un símbolo para obtener una cadena media entre S_1 y S_2 que cumpla con las restricciones establecidas.

5.2. Ejemplo del algoritmo para el cálculo del contorno medio.

Para ilustrar mejor las ideas antes descritas puede analizarse el siguiente ejemplo. Sean $S_1 = \{2, 3, 4\}$, $S_2 = \{6, 0\}$ con costes de sustitución según se definió en el apartado (1.5.2) pero tomando valor 1 para las *inserciones* y *borrados* entonces $Q = \{w(2, \varepsilon), w(3, \varepsilon), w(4, 6),$

Función `BuscarOp(`i,a_{S1},a_{S2}`)` : $\langle d, r \rangle$

```
/* Q: secuencia de edición óptima para transformar S1 en S2   */
/* q_i: índice de la operación q ∈ Q que se analizará         */
/* r: cadena resultante de aplicar a S1 las operaciones en Q  */
/* a_S1 = 0 y a_S2 = 0 : distancias acumuladas D(r,S1) y D(r,S2)
   respectivamente                                            */
/* d: diferencia entre a_S1 y a_S2                            */
/* mejor = ⟨∞,∅⟩:mejor resultado parcial, mejor[0] distancia,
   mejor[1] respectivo r                                      */
```

si $i == 0$ **entonces** $mejor \leftarrow \langle a_{S1} - a_{S2}, \varnothing \rangle$ **sino**

  seleccionar $Q[i]$ **:**

    caso $w(b_{S1}, \varepsilon,)$: `/* Borrado */`

      $\langle d_{no}, r_{no} \rangle \leftarrow BuscarOp(q_i - 1, a_{S1} + w(\varepsilon, b_{S1}), a_{S2})$ `/* Rechazada */`

      $\langle d_{si}, r_{si} \rangle \leftarrow BuscarOp(q_i - 1, a_{S1}, a_{S2} + w(b_{S1}, \varepsilon))$ `/* Aceptar */`

      si $|d_{si}| < |d_{no}|$ **entonces**

      $mejor \leftarrow \langle d_{si}, r_{si} \cup \{b_{S1}\} \rangle$ **sino**

        $mejor \leftarrow \langle d_{no}, r_{no} \rangle$

      fin si

    caso $w(\varepsilon, b_{S2})$: `/* Inserción */`

      $\langle d_{no}, r_{no} \rangle \leftarrow BuscarOp(op_i - 1, a_{S1}, a_{S2} + w(\varepsilon, b_{S2}))$ `/* Rechazada */`

      $\langle d_{si}, r_{si} \rangle \leftarrow BuscarOp(op_i - 1, a_{S1} + w(b_{S2}, \varepsilon), a_{S2})$ `/* Aceptar */`

      si $|d_{si}| < |d_{no}|$ **entonces**

      $mejor \leftarrow \langle d_{si}, r_{si} \cup \{b_{S2}\} \rangle$ **sino**

        $mejor \leftarrow \langle d_{no}, r_{no} \rangle$;

      fin si

    caso $w(b_{S1}, b_{S2})$: `/* Sustitución */`

      para cada *símbolo* m *satisfaciendo (e5.1) y (e5.2)* **hacer**

        $\langle d, r \rangle \leftarrow BuscarOp(op_i - 1, a_{S1} + w(m, b_{S1}), a_{S2} + w(m, b_{S2}))$;

        si $|d| < |mejor[0]|$ **entonces**

        $mejor \leftarrow \langle d, r \cup \{m\} \rangle$;

      fin para cada

  fin seleccionar

fin si

devolver $mejor$

Función `FMSC` (S_1,S_2) :R

```
/* S1 y S2: cadenas para calcular su media aproximada R      */
```

calcular $D(S_1, D_2)$ para obtener Q;
$\langle d, R\rangle \leftarrow$ BuscarOp($L_Q, 0, 0$);
devolver R;

$w(\varepsilon, 0)\}$. Las diferentes combinaciones sobre la inserción en R de los símbolos que relacionan las operaciones en Q conducen al árbol que se muestra en la figura (5.1) donde cada nodo hoja representa una cadena media R candidata que incluye los símbolos de las operaciones aceptadas a lo largo de la rama. Entre corchetes las distancias $D(R, S_1)$ y $D(R, S_2)$ acumuladas que resultan de las operaciones en el camino hasta el nodo. Por ejemplo, la cadena $R = \{2, 3, 5, 0\}$ en la rama del extremo izquierdo resulta de *aceptar* todas las operaciones en Q.

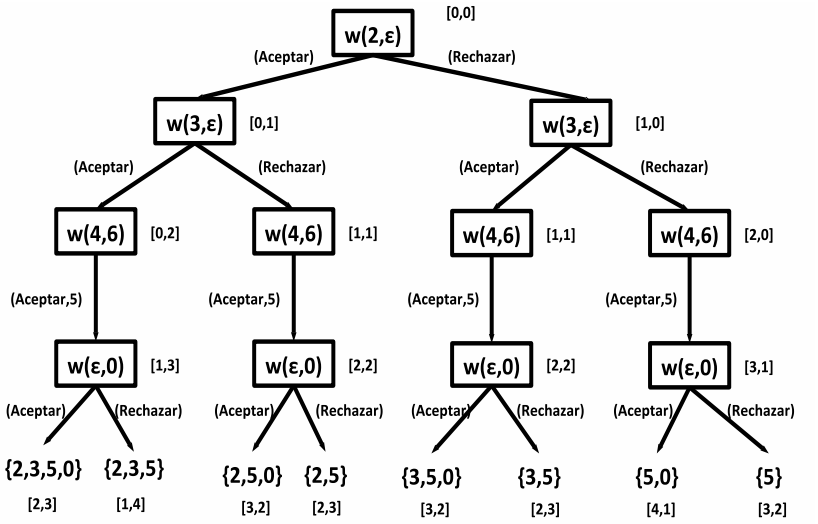


Figura 5.1: Ejemplo del cálculo de la media utilizando el algoritmo FMSC. Cada rama representa una posible decisión sobre las operaciones de edición a aplicar en S_1 para obtener R.

5.3. Análisis del coste computacional.

Sea L_Q la longitud de la secuencia de edición de coste mínimo para transformar S_1 en S_2, en el peor caso se tendrá que $L_Q = L_{S1} + L_{S2} \leqslant 2L$, esto es cuando no contiene sustituciones, donde $L = max\{L_{S1}, L_{S2}\}$. El cómputo de la aproximación R a la media de dos cadenas S_1 y S_2 implica el cálculo de la distancia de edición $D(S_1, S_2)$ lo cual puede hacerse en tiempo proporcional a $\mathcal{O}(L^2)$ como se señaló en la sección (1.5.2).

La búsqueda en el árbol mediante la función *BuscarOp* puede verse como la evaluación de todas las posibilidades de asignar $\{aceptada/rechazada\}$ a cada q_i en Q, lo que totaliza 2^{2L} posibilidades, esto es, el número máximo de ramas en el árbol. De acuerdo a lo anterior, *BuscarOp* requiere un primer paso con coste $\mathcal{O}(L^2)$ seguido de un bloque con coste $\mathcal{O}(2^L)$. Aplicando la *regla de la suma* (Aho *et al.*, 1983) se obtiene que el coste computacional del algoritmo FMSC es $\mathcal{O}(2^L)$. Naturalmente en la práctica este valor es menor dado que el factor de ramificación de los nodos correspondientes a las *sustituciones* es 1.

Por otra parte, examinando el árbol de la figura (5.1) puede verse que varias veces debe resolverse el mismo problema. Es decir, por diferentes ramas se llegan a nodos con idénticos valores para las distancias acumuladas y que al estar situados en el mismo nivel indican que en cada rama restan por examinar el mismo subconjunto de operaciones. En este punto, la decisión sobre las operaciones restantes no dependerá de cómo se ha llegado al nodo, dado que solo interesa el valor de las distancias parciales. Esto es, cada subproblema queda caracterizado por el conjunto de operaciones examinadas hasta el momento y los valores de las distancias acumuladas cuya suma estará acotada por $D(S_1, S_2)$, que se denotará por D para simplificar la notación. De acuerdo a lo anterior aplicando *programación dinámica* en el peor caso existirán $L \times D^2$ subproblemas. Recordando que encontrar la secuencia de operaciones de edición tiene un coste $\mathcal{O}(L^2)$ resulta que en este caso el coste computacional puede rebajarse a $\mathcal{O}(max\{L^2, L \times D^2\})$.

5.4. Variante heurística del algoritmo para el cálculo del contorno medio.

El proceso de búsqueda desarrollado por el algoritmo FMSC necesita explorar un número de alternativas que puede tornarse bastante grande. En esta sección se presenta una variante heurística que reduce el número de ramas que se exploran al descartar algunas de ellas. Durante la ejecución del algoritmo solo dos ramas B_1 y B_2 se mantienen abiertas, explorando el árbol en un recorrido *Primero a lo Ancho*.

Como antes, los nodos relativos a las inserciones y los borrados crean dos nuevas ramas correspondientes a las opciones de aceptación o rechazo. Asumiendo que solo existen dos ramas en el árbol, esto conduce a cuatro nuevas ramas dos de las cuales deben podarse. Una de las ramas sobrevivientes es aquella que conduce a las mejores distancias parciales $D(R, S_1)$ y $D(R, S_2)$ de acuerdo a los requerimientos establecidos por (e4.1), (e4.2) y (e4.3). La otra rama seleccionada es la correspondiente a la decisión contraria en la otra rama explorada.

Por ejemplo, sea la mejor alternativa aceptar el símbolo de la operación en la rama B_1, entonces la otra rama que se mantendrá será la resultante de rechazar la operación en la rama B_2. En el caso de las sustituciones estas se siguen tratando igual que antes. El árbol en la figura (5.2) ilustra estas ideas de forma gráfica.

5.5. Análisis del coste computacional.

Sean L y L_Q tal como se definieron en la sección (5.3). El algoritmo heurístico también requiere calcular $D(S_1, S_2)$ que como se explicó puede hacerse en tiempo proporcional a $\mathcal{O}(L^2)$.

Respecto al bucle **para cada** en el procedimiento FMSC-Voraz, este requiere examinar las L_Q operaciones en Q pero solamente una vez, esto es, en tiempo $\mathcal{O}(L)$.

Función FMSC-Voraz(S_1,S_2):R

/* S_1 y S_2: cadenas para calcular su media aproximada R */
/* Q: secuencia de edición óptima para transformar S_1 en S_2 */
/* q_i: i-ésima operación de edición $q \in Q$ */
/* a_{S1}^{B1}, a_{S2}^{B1}: distancias acumuladas $D(R,S_1)$ y $D(R,S_2)$, rama 1 */
/* a_{S1}^{B2}, a_{S2}^{B2}: distancias acumuladas $D(R,S_1)$ y $D(R,S_2)$, rama 2 */

para cada $q_i \in Q$ **hacer**

seleccionar q_i **:**

caso $w(b_{S1}, \varepsilon)$*:* /* Borrado */

$\langle d_{S1}^{B1}, d_{S2}^{B1} \rangle \leftarrow \langle a_{S1}^{B1} + w(\varepsilon, b_{S1}), a_{S2}^{B1} \rangle$; /* Rechazar, rama 1 */

$\langle c_{S1}^{B1}, c_{S2}^{B1} \rangle \leftarrow \langle a_{S1}^{B1}, a_{S2}^{B1} + w(b_{S1}, \varepsilon) \rangle$; /* Aceptar, rama 1 */

$\langle d_{S1}^{B2}, d_{S2}^{B2} \rangle \leftarrow \langle a_{S1}^{B2} + w(\varepsilon, b_{S1}), a_{S2}^{B2} \rangle$; /* Rechazar, rama 1 */

$\langle c_{S1}^{B2}, c_{S2}^{B2} \rangle \leftarrow \langle a_{S1}^{B2}, a_{S2}^{B2} + w(b_{S1}, \varepsilon) \rangle$; /* Aceptar, rama 1 */

ActDist($\langle d_{S1}^{B1}, d_{S2}^{B1} \rangle, \langle c_{S1}^{B2}, c_{S2}^{B2} \rangle, \langle d_{S1}^{B1}, d_{S2}^{B1} \rangle, \langle c_{S1}^{B1}, c_{S2}^{B1} \rangle$);

caso $w(\varepsilon, b_{S2})$*:* /* Inserción */

$\langle d_{S1}^{B1}, d_{S2}^{B1} \rangle \leftarrow \langle a_{S1}^{B1}, a_{S2}^{B1} + w(\varepsilon, b_{S2}) \rangle$; /* Rechazar, rama 1 */

$\langle c_{S1}^{B1}, c_{S1}^{B1} \rangle \leftarrow \langle a_{S1}^{B1} + w(b_{S2}, \varepsilon), a_{S2}^{B1} \rangle$; /* Aceptar, rama 1 */

$\langle d_{S1}^{B2}, d_{S2}^{B2} \rangle \leftarrow \langle a_{S1}^{B2}, a_{S2}^{B2} + w(\varepsilon, b_{S2}) \rangle$; /* Rechazar, rama 2 */

$\langle c_{S1}^{B2}, c_{S2}^{B2} \rangle \leftarrow \langle a_{S1}^{B2} + w(b_{S2}, \varepsilon), a_{S2}^{B2} \rangle$; /* Aceptar, rama 2 */

ActDist($\langle d_{S1}^{B1}, d_{S2}^{B1} \rangle, \langle c_{S1}^{B2}, c_{S2}^{B2} \rangle, \langle d_{S1}^{B1}, d_{S2}^{B1} \rangle, \langle c_{S1}^{B1}, c_{S2}^{B1} \rangle$);

caso $w(b_{S1}, b_{S2})$*:* /* Sustitución */

seleccionar el mejor símbolo m respecto a (e5.1) y (e5.1);

$\langle a_{S1}^{B1}, a_{S2}^{B1} \rangle \leftarrow \langle a_{S1}^{B1} + w(m, b_{S1}), a_{S2}^{B1} + w(m, b_{S2}) \rangle$;

$\langle a_{S1}^{B2}, a_{S2}^{B2} \rangle \leftarrow \langle a_{S1}^{B2} + w(m, b_{S1}), a_{S2}^{B2} + w(m, b_{S2}) \rangle$;

fin seleccionar

fin para cada

seleccionar la mejor opción entre $\langle a_{S1}^{B1}, a_{S2}^{B1} \rangle$ y $\langle a_{S1}^{B2}, a_{S2}^{B2} \rangle$;

devolver R, incluyendo los símbolos involucrados en las operaciones *aceptadas* de la mejor rama.

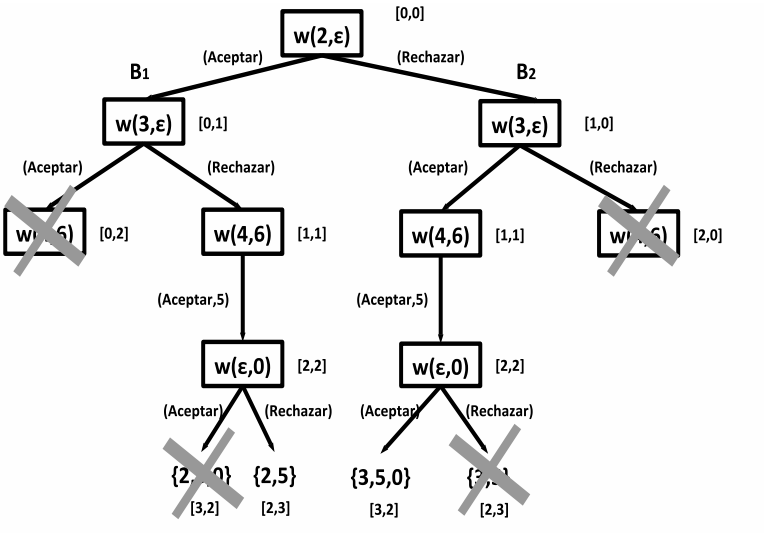


Figura 5.2: Ejemplo de la variante heurística para cálculo de la media. Los nodos marcados representan los ramas podadas en cada nivel.

De lo anterior, aplicando la *regla de la suma* (Aho *et al.*, 1983) resulta que el coste computacional de FMSC-Voraz es $\mathcal{O}(L^2)$.

5.6. Aplicación del algoritmo para el cálculo del contorno medio. Método de edición basado en Wilson

Como ejemplo de aplicación del algoritmo propuesto para el cálculo de la cadena media entre dos instancias se describe su empleo en un nuevo procedimiento de edición basado en el algoritmo descrito en Wilson (1972).

Sea S el conjunto de instancias que se desea editar; los algoritmos de edición basados en la propuesta realizada por Wilson (1972), como los trabajos de Tomek (1976a) o Guan *et al.* (2009) excluyen las instancias S_i incorrectamente clasificadas por sus K-vecinos más cercanos.

Procedimiento `ActDist` ($\langle d_{S1}^{B1}, d_{S2}^{B1}\rangle, \langle c_{S1}^{B2}, c_{S2}^{B2}\rangle, \langle d_{S1}^{B1}, d_{S2}^{B1}\rangle, \langle c_{S1}^{B1}, c_{S2}^{B1}\rangle$)

$\langle t_{S1}, t_{S2}\rangle \leftarrow$ mejor entre $\{\langle d_{S1}^{B1}, d_{S2}^{B1}\rangle, \langle c_{S1}^{B2}, c_{S2}^{B2}\rangle, \langle d_{S1}^{B1}, d_{S2}^{B1}\rangle, \langle c_{S1}^{B1}, c_{S2}^{B1}\rangle\}$;
seleccionar $\langle t_{S1}, t_{S2}\rangle$ **:**

caso $\langle d_{S1}^{B1}, d_{S2}^{B1}\rangle$*:*
$\langle a_{S1}^{B1}, a_{S2}^{B1}\rangle \leftarrow \langle d_{S1}^{B1}, d_{S2}^{B1}\rangle$;
$\langle a_{S1}^{B2}, a_{S2}^{B2}\rangle \leftarrow \langle c_{S1}^{B2}, c_{S2}^{B2}\rangle$;

caso $\langle c_{S1}^{B1}, c_{S2}^{B1}\rangle$*:*
$\langle a_{S1}^{B1}, a_{S2}^{B1}\rangle \leftarrow \langle c_{S1}^{B1}, c_{S2}^{B1}\rangle$;
$\langle a_{S1}^{B2}, a_{S2}^{B2}\rangle \leftarrow \langle d_{S1}^{B2}, d_{S2}^{B2}\rangle$;

caso $\langle d_{S1}^{B2}, d_{S2}^{B2}\rangle$*:*
$\langle a_{S1}^{B1}, a_{S2}^{B1}\rangle \leftarrow \langle c_{S1}^{B1}, c_{S2}^{B1}\rangle$;
$\langle a_{S1}^{B2}, a_{S2}^{B2}\rangle \leftarrow \langle d_{S1}^{B2}, d_{S2}^{B2}\rangle$;

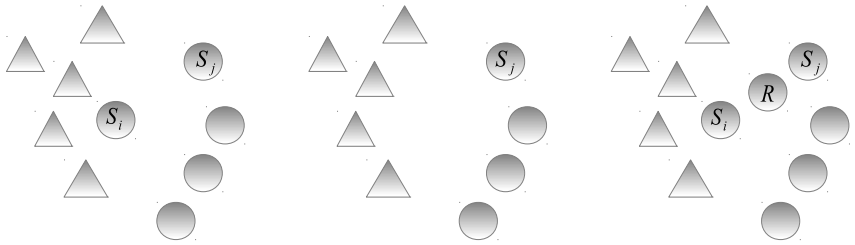
caso $\langle c_{S1}^{B2}, c_{S2}^{B2}\rangle$*:*
$\langle a_{S1}^{B1}, a_{S2}^{B1}\rangle \leftarrow \langle d_{S1}^{B1}, d_{S2}^{B1}\rangle$;
$\langle a_{S1}^{B2}, a_{S2}^{B2}\rangle \leftarrow \langle c_{S1}^{B2}, c_{S2}^{B2}\rangle$;

fin seleccionar

Este tipo de edición elimina regiones donde se solapan diferentes clases al mismo tiempo que se definen más claramente las fronteras entre estas. Un clasificador K-NN que tome un conjunto así editado como datos de entrenamiento debe mejorar sus resultados en la clasificación comparados al hacerlo con el conjunto original.

En cambio, como plantean Wilson & Martínez (2000), en algunos casos el proceso de edición debe hacerse cuidadosamente ya que pueden eliminarse un gran número de instancias a la vez que se deteriora la capacidad de generalización del clasificador. Cuando $K \geqslant 1$, una clasificación incorrecta de la instancia S_i no quiere decir no exista entre sus K-vecinos alguno de su propia clase. Por tal razón, no necesariamente S_i deber ser un objeto atípico, en cambio pudiera ser una instancia situada en las fronteras de las clases útil en la clasificación. En otros casos, debido a distintos factores como la alta dimensionalidad de los datos (Ferri *et al.*, 1999), muchas instancias pueden estar cerca de otros ejemplos de clase diferente. En tal caso, una reducción del conjunto de entrenamiento puede incrementar el error promedio en la clasificación.

La variante de algoritmo de edición propuesta, JWilson en los experimentos, se propone tratar este problema haciendo menos riguroso criterio para borrar una instancia utilizado por Wilson (1972). En este caso, al conjunto S se añaden una o varias instancias artificiales R construidas a partir de S_i y su K-vecino más próximo de igual clase S_j en caso de que este exista. La instancia R etiquetada con la misma clase que S_i debe satisfacer $D(R, S_i) \leqslant D(S_i, S_j)$, luego, incluyéndola se aumenta la oportunidad de que S_i sea correctamente clasificada por sus K-vecinos ya que al cumplir la condición anterior, R se situará en la K-vecindad de S_i. Al mismo tiempo, algunas regiones pobremente cubiertas pueden quedar mejor representadas. Haciendo esta modificación puede esperarse que clasificar con un conjunto así editado conduzca a menores errores en la clasificación. En la figura (5.3) se muestra mediante un ejemplo la diferencia del algoritmo propuesto respecto al descrito por Wilson (1972).



(a) (b) (c)

Figura 5.3: Resultado de aplicar Wilson (5.3b) y JWilson (5.3c) al conjunto en (5.3a).

Una vez definidas las ideas generales del algoritmo de edición, una cuestión importante por resolver es cómo construir la instancia R. En este caso, se ha empleado la "media", es decir, R será la media entre S_i y su K-vecino de más cercano de igual clase S_j. Concretamente cuando las instancias corresponden a cadenas de Freeman representando contornos se propone utilizar el algoritmo descrito en la sección (5.1).

La función JWilson presenta una descripción algorítmica del algoritmo de edición propuesto.

Función `JWilson(`S`,`K`)` :S

```
/* S: conjunto de instancias a editar                    */
/* K: número de vecinos más cercanos                     */
```

para cada *instancia* $S_i \in S$ **hacer**
  clasificar S_i según sus K-vecinos más cercanos en $S - S_i$;
  si S_i *es incorrectamente clasificada* **entonces**
    encontrar S_j, el K-vecino más cercano de igual clase que S_i;
    si *existe* S_j **entonces**
      hacer $R = FMSC(S_i, S_j)$ o
      $R = FMSC - Voraz(S_i, S_j)$;
      hacer $S = S \cup R$;
    sino
      marcar S_i para borrado;
    fin si
  fin si
fin para cada
borrar de S todas las instancias marcadas;
devolver S;

5.7. Resultados experimentales

Anteriormente se presentaron dos algoritmos para el cálculo de una aproximación a la media de dos contornos representados mediante cadenas de Freeman. Como aplicación de estos algoritmos se describió un nuevo método de edición aplicado al preprocesamiento del conjunto de entrenamiento de un clasificador K-NN. Dado que la validación de cada uno de los enfoques propuestos se realizará de forma empírica, deben desarrollarse un grupo de experimentos dirigidos a evaluar los siguientes aspectos:

- Desempeño de los algoritmos FMSC y FMSC-Voraz en el cálculo del contorno promedio de acuerdo a las restricciones especificadas en (e4.1), (e4.2) y (e4.3).

- Desempeño del algoritmo de edición respecto al *baseline* establecido por el algoritmo de Wilson.

Las siguientes secciones quedan dedicadas a cubrir cada uno de los experimentos diseñados para este propósito.

5.7.1. Validación del cálculo del contorno medio.

Un primer experimento fue realizado para comparar la efectividad de FMSC y FMSC-Voraz en el cálculo la media entre dos contornos. Además ambos métodos fueron comparados con las variantes propuestas por Fischer & Zell (2000), Cárdenas (2004) y Martínez-Hinarejos *et al.* (2003). Este último fue inicializado con la mediana y con la aproximación voraz descrita por Casacuberta & Antoni (1997). En las comparativas se denotarán como Hinarejos-R^M e Hinarejos-R^G respectivamente. En todos los casos se adaptó el criterio de búsqueda para encontrar una aproximación a la media sujeta a (e1.2) y (e4.3). Como se apuntó en el capítulo (2) estos métodos calculan la media efectuando sucesivas perturbaciones a una solución parcial, los dos primeros haciendo varias modificaciones simultáneas mientras que Martínez-Hinarejos *et al.* (2003) las realiza de una en una.

Para verificar que los resultados obtenidos sean independientes de los datos utilizados se seleccionaron contornos de diferentes clases y longitudes pertenecientes a las bases de datos descritas en el apartado (1.5.5), la muestra empleada en este caso se detalla en la tabla (5.1). Con estos datos se construyó el contorno medio para cada posible par de objetos dentro del conjunto seleccionado, sin importar que pertenecieran a diferentes clases. La figura (5.4) muestra algunos ejemplos de

Tabla 5.1: Características de los datos empleados en el experimento para comparar FMSC y FMSC-Voraz.

Corpus	Clases	Instancias x Clase	Longitud de las cadenas		
			Promedio	Max.	Min.
NIST-19	26	2	200±70	381	41
USPS	10	2	150 ± 40	235	87
MPEG-7	11	2	1400 ± 90	3431	281

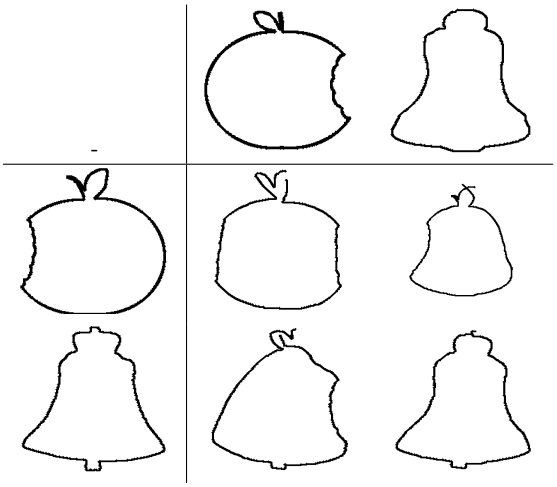


Figura 5.4: Media entre contornos seleccionados de la base de datos MPEG-7. Cada columna muestra la media calculada a partir del primer contorno en las respectivas fila y columna.

los contornos originales y los promedios obtenidos mediante FMSC-Voraz.

En este experimento los valores de (e1.2) y (e4.3) fueron calculados para evaluar cuán bien se comportan los algoritmos propuestos. Más formalmente, se definen:

Variable independiente: Algoritmo utilizado para obtener la media entre dos contornos.

Definición conceptual: Un algoritmo para calcular la media entre dos contornos se propone obtener un nuevo prototipo que se

encuentre “equidistante” de los contornos originales en términos de una cierta métrica que permita calcular la distancia entre estos. En el experimento, se refiere a un algoritmo capaz de obtener una cadena que minimiza (e1.2) y (e4.3) utilizando la métrica descrita en la sección (1.5.2).

Variable dependiente: Suma de las distancias del contorno medio a los contornos en el conjunto.

Definición conceptual: Se entiende por la distancia de acuerdo a la métrica elegida del contorno medio hasta los dos contornos que fueron “promediados” para obtenerlo. Está dada por la fórmula (e1.2).

Definición operacional: Para medir esta variable es preciso implementar un algoritmo que permita el cálculo de la distancia de edición de Levenshtein. Los costes asociados a cada operación de edición se toman de acuerdo a como se definieron en la sección (1.5.2).

Variable dependiente: Diferencia entre las distancias del contorno promedio a los contornos en el conjunto.

Definición conceptual: Está definida por la expresión en (e4.3). Mide que tan “equidistante” se encuentra el contorno medio respecto al resto de los contornos dado que no necesariamente todos los contornos que minimicen (e1.2) serán óptimos en referencia a (e4.3).

Definición operacional: Para medir esta variable es preciso implementar un algoritmo que permita el cálculo de la distancia de edición de Levenshtein. Los costes asociados a cada operación de edición se toman de acuerdo a como se definieron en la sección (1.5.2).

Respecto a (e1.2) no se encontraron diferencias entre ninguno de los métodos evaluados salvo Hinarejos-R^G que condujo a los peores re-

sultados. En el caso de (e4.3) la tabla (5.2) muestra que FMSC y FMSC-Voraz se comportan muy bien, esto es, la instancia promedio se encuentra situada en el "medio" de las dos instancias originales. Hinarejos-R^M también tiene un buen desempeño, pero el valor elevado de la desviación estándar indica que en algunos casos la media aproximada está sesgada hacia uno de los contornos originales. Para esta magnitud nuevamente Hinarejos-R^G alcanzó los peores resultados entre los algoritmos que realizan las perturbaciones una a la vez, aunque este grupo muestra un desempeño significativamente superior al resto de los enfoques incluidos en la prueba.

Tabla 5.2: Comparación de diferentes algoritmos en el cálculo del contorno promedio en términos del promedio y la desviación típica de $|D(R, S_1) - D(R, S_2)|$.

Algoritmo	Promedio$\pm\sigma$
FMSC	0.5 ± 0.5
FMSC-Voraz	0.9 ± 0.7
Hinarejos-R^G	0.0 ± 44.0
Hinarejos-R^M	0.7 ± 5.0
Fischer	42.0 ± 21.0
Cárdenas	21.0 ± 10.0

Debe señalarse que FMSC-Voraz es tan efectivo como FMSC e Hinarejos-R^M pero requiriendo significativamente menos tiempo de cómputo. Por ejemplo, cada vez que una cadena candidata es modificada para verificar si mejora la aproximación a la media deben calcularse dos distancias. En el caso de Hinarejos-R^M se calcularon como promedio 43218.2 distancias mientras que Hinarejos-R^G requirió 73143.3. En contraste, FMSC-Voraz solo precisa calcular una sola distancia. Los algoritmos descritos por Fischer & Zell (2000) y Cárdenas (2004) convergen rápidamente pero como se aprecia no alcanzan tan buenos resultados. Aunque teóricamente FMSC-Voraz presenta un coste computacional menor respecto a FMSC, se realizó una comparación. Como promedio la aproximación voraz fue cerca de 16 veces más rápida.

5.7.2. Experimento de validación. Algoritmo de edición.

Como se apuntó anteriormente, otro de los objetivos de los experimentos propuestos es evaluar el desempeño del algoritmo de edición descrito; en este caso a través de una comparación con el algoritmo de Wilson. Como criterio comparativo se tomará el error en la clasificación obtenido por un clasificador K-NN al utilizar de forma independiente los siguientes conjuntos:

- Conjunto de datos original.
- Conjunto preprocesado mediante el algoritmo de Wilson.
- Conjunto preprocesado mediante el algoritmo de edición propuesto, identificado por JWilson.

Más formalmente, quedan definidas las siguientes variables:

Variable independiente: Algoritmo de edición.

Definición conceptual: Es un procedimiento que tiene como objetivo eliminar instancias atípicas del conjunto de entrenamiento de un clasificador K-NN de modo que la efectividad de este se incremente. En este caso la entrada consiste en un grupo de contornos codificados mediante cadenas de Freeman. El clasificador K-NN utiliza la métrica descrita en el apartado (1.5.2).

Variable independiente: Cantidad K de vecinos más cercanos considerados por el algoritmo de edición.

Definición conceptual: Diferentes algoritmos de edición descritos y concretamente los que se comparan en el trabajo deciden si una instancia fue etiquetada erróneamente examinando una vecindad de esta. El valor de K define pues, la cantidad de vecinos más próximos que se tendrán en cuenta.

Variable independiente: Cantidad K de vecinos más cercanos considerados en la clasificación.

Definición conceptual: El valor de K se define como la cantidad de vecinos más próximos que se tendrán en cuenta al realizar una clasificación utilizando la regla K-NN descrita en la sección (1.5.4).

Variable dependiente: Error en la clasificación de un clasificador K-NN.

Definición conceptual: Se define como el por ciento de instancias incorrectamente clasificadas respecto al total.

Variable dependiente: Cantidad de ejemplos en el conjunto de entrenamiento editado.

También en este experimento se utilizaron los datos descritos en la sección (1.5.5). Las 80 instancias seleccionadas de forma aleatoria fueron divididas en 4 porciones para realizar una *validación cruzada* de modo que el conjunto de entrenamiento queda formado por 60 instancias y el correspondiente conjunto de prueba por las 20 restantes. Debe apuntarse que el conjunto correspondiente al corpus MPEG-7 fue descartado debido a que solamente hay disponibles 20 instancias por clase. De acuerdo a la cantidad de particiones tomadas en la validación cruzada, el conjunto de entrenamiento tendría 15 instancias por clase lo que es contraproducente ya que se probaron valores de K hasta 17.

Inicialmente cada conjunto de entrenamiento fue editado utilizando Wilson para posteriormente clasificar el correspondiente conjunto de prueba. Este procedimiento de edición-clasificación fue repetido dos veces, pero utilizando JWilson; una vez tomando FMSC para construir las instancias artificiales y FMSC-Voraz en el otro caso. Además de comparar entre sí los resultados obtenidos por diferentes esquemas de edición, se computó el error en la clasificación tomando los conjuntos de entrenamiento sin editar. Los correspondientes conjuntos de

entrenamiento fueron editados tomando valores de K impares en el rango de 3 a 17 mientras para la clasificación varió de 1 a 17. Como medida de similitud se tomó la distancia de Levenshtein tal como quedó definida en la sección (1.5.2).

Las tablas (5.3) y (5.4) muestran el error de clasificación promedio y la correspondiente desviación típica obtenidas en las 4 particiones para los distintos valores de K y algoritmos de edición. Los datos en la fila etiquetada como *No editado* muestran los resultados de clasificar tomando el conjunto sin editar. Como puede apreciarse, en ninguno de estos experimentos Wilson logra reducir el error en la clasificación base, esto es al clasificar con los conjuntos de entrenamiento sin editar. En cambio este resultado es mejorado al utilizar la propuesta JWilson y FMSC para construir los prototipos artificiales. Mediante este enfoque se logra reducir el error en un $87.5\,\%$ de los experimentos edición-clasificación en la base de datos NIST-19 y en el $79.2\,\%$ para el corpus USPS. Estos resultados aparecen resaltados en **negritas** en las tablas. Un desempeño similar se alcanzó al emplear FMSC-Voraz.

Tabla 5.3: Error promedio (4-particiones) como por ciento en la clasificación usando diferentes conjuntos editados en la base de datos (NIST-19).

			K en la clasificación								
			1	3	5	7	9	11	13	15	17
K en la edición	No editado		13.7±1.4	14.7±1.9	15.4±1.5	16.0±0.8	17.1±1.1	17.7±1.7	18.0±2.0	18.6±1.8	19.6±1.8
	3	Wilson	16.0±2.0	17.6±1.8	17.6±1.0	19.4±1.4	19.5±1.8	20.0±2.0	21.0±1.8	22.0±1.5	22.1±1.6
		FMSC	14.5±1.9	15.5±1.7	**15.3±1.4**	16.6±1.1	**16.4±1.9**	**18.0±2.0**	18.0±2.0	19.0±2.0	**19.0±1.9**
		Voraz	14.4±1.8	15.0±2.0	**15.3±1.9**	16.2±1.5	**16.0±2.0**	**17.0±2.0**	18.0±2.0	19.0±2.0	**19.0±2.0**
	5	Wilson	15.8±1.7	17.6±1.7	17.9±1.3	19.6±1.4	20.1±1.7	21.0±2.0	21.0±2.0	21.6±1.5	22.7±1.7
		FMSC	14.0±1.6	**14.5±1.8**	**14.5±1.7**	**15.6±1.1**	**16.0±1.8**	**16.3±1.7**	**17.5±1.6**	**17.7±1.7**	**18.4±1.9**
		Voraz	13.8±1.9	**14.0±2.0**	**14.7±1.9**	**15.0±1.2**	**15.5±1.8**	**16.1±1.5**	**17.1±1.4**	**18.1±1.8**	**18.0±2.0**
	7	Wilson	16.3±1.9	17.5±1.2	17.9±1.3	19.9±1.6	20.0±2.0	21.0±2.0	22.0±2.0	22.3±1.4	23.0±1.4
		FMSC	13.9±1.6	**14.4±1.7**	**14.2±1.6**	**15.3±1.0**	**15.6±1.3**	**16.3±1.0**	**16.6±1.4**	**17.3±1.4**	**17.8±1.7**
		Voraz	13.9±1.7	**14.0±2.0**	**14.1±1.3**	**15.1±0.5**	**15.0±1.1**	**15.9±1.4**	**16.4±1.2**	**17.1±1.8**	**18.0±2.0**
	9	Wilson	17.0±2.0	17.8±1.7	18.5±1.8	19.8±1.9	20.0±2.0	21.0±3.0	22.0±2.0	22.6±1.6	23.8±1.6
		FMSC	13.8±1.4	**14.2±1.3**	**14.0±1.5**	**15.0±0.8**	**15.4±0.9**	**16.2±0.9**	**16.8±0.9**	**17.1±0.8**	**17.6±1.2**
		Voraz	13.7±1.4	**14.0±1.7**	**14.0±0.9**	**14.7±0.2**	**14.3±1.0**	**15.8±1.1**	**16.4±0.9**	**16.4±1.5**	**17.2±1.8**
	11	Wilson	17.1±1.9	18.8±1.7	18.9±1.6	20.0±2.0	21.0±2.0	22.0±2.0	22.0±2.0	23.3±1.4	24.0±1.4
		FMSC	**13.6±1.4**	**14.0±1.6**	**14.1±1.6**	**14.9±0.7**	**15.2±1.0**	**15.6±1.0**	**16.3±1.1**	**16.5±0.8**	**17.3±1.2**
		Voraz	**13.5±1.4**	**14.0±1.5**	**13.8±1.2**	**14.5±0.5**	**14.3±1.2**	**15.4±1.1**	**15.8±1.0**	**15.9±1.5**	**16.9±1.6**
	13	Wilson	17.1±1.7	18.9±1.3	19.5±1.4	21.0±2.0	22.0±2.0	22.0±2.0	22.8±1.6	23.5±1.5	24.0±1.5
		FMSC	**13.6±1.3**	**14.2±1.5**	**14.1±1.5**	**14.6±1.0**	**15.1±1.0**	**15.6±1.1**	**16.3±1.1**	**16.6±1.2**	**17.2±1.2**
		Voraz	**13.5±1.3**	**14.3±1.7**	**13.9±1.3**	**14.3±0.7**	**14.6±1.2**	**15.3±1.5**	**15.5±1.0**	**16.1±1.3**	**17.0±2.0**
	15	Wilson	17.5±1.8	19.6±1.6	19.8±1.4	20.8±1.8	22.0±2.0	22.0±2.0	23.3±1.6	23.7±1.6	24.6±1.4
		FMSC	**13.4±1.3**	**13.9±1.3**	**13.8±1.4**	**14.7±1.0**	**15.3±1.2**	**15.6±1.2**	**16.2±0.9**	**16.5±1.4**	**17.0±1.4**
		Voraz	**13.3±1.3**	**13.9±1.4**	**13.7±1.3**	**14.1±0.5**	**14.1±0.9**	**15.2±1.3**	**15.3±0.9**	**16.1±1.7**	**16.6±1.7**
	17	Wilson	17.8±1.5	19.9±1.2	20.2±1.2	21.3±1.5	22.2±1.6	23.1±1.7	23.7±1.9	24.2±1.3	24.8±1.3
		FMSC	**13.5±1.2**	**14.0±1.1**	**13.9±1.4**	**14.8±1.0**	**15.3±0.9**	**15.7±0.9**	**16.0±1.0**	**16.3±0.9**	**16.6±1.5**
		Voraz	**13.3±1.1**	**13.9±1.0**	**13.8±1.1**	**14.5±0.4**	**14.2±0.7**	**15.0±0.8**	**15.3±1.0**	**15.9±1.4**	**16.3±1.5**

Tabla 5.4: Error promedio (4-particiones) como por ciento en la clasificación usando diferentes conjuntos editados en la base de datos (USPS).

K en la edición			K en la clasificación								
			1	3	5	7	9	11	13	15	17
	No editado		1.8±1.3	2.0±0.7	3.0±0.4	3.5±0.4	3.6±0.3	4.1±0.5	4.4±0.9	4.8±1.3	4.9±0.9
	3	Wilson	2.8±0.9	3.1±0.8	3.6±0.3	4.3±0.9	4.3±0.9	4.5±0.8	4.8±0.6	5.1±1.1	5.1±0.9
		FMSC	2.0±1.1	2.3±0.5	**2.9±0.3**	**3.4±0.5**	**3.5±0.7**	**3.9±0.5**	**4.1±0.8**	**4.5±1.3**	**4.6±0.5**
		Voraz	1.9±1.0	2.3±0.5	**2.9±0.3**	**3.3±0.3**	**3.5±0.7**	**3.9±0.5**	**4.3±0.6**	**4.6±1.4**	**4.8±0.6**
	5	Wilson	2.6±0.9	2.9±0.6	3.6±0.3	4.3±0.6	4.1±0.6	4.5±0.8	4.8±0.6	5.1±1.1	5.1±0.9
		FMSC	1.9±1.1	2.4±0.6	**2.8±0.3**	**3.0±0.4**	**3.1±0.3**	**3.8±0.5**	**3.9±0.5**	**4.3±1.4**	**4.4±0.6**
		Voraz	1.8±1.0	2.3±0.5	**2.8±0.3**	**3.0±0.4**	**3.1±0.3**	**3.5±0.4**	**3.8±0.3**	**4.3±1.4**	**4.4±1.0**
	7	Wilson	2.5±1.1	3.0±0.7	3.8±0.5	4.3±0.9	4.3±0.9	4.6±0.9	5.0±0.9	5.4±1.3	5.4±0.6
		FMSC	1.8±1.0	2.1±0.6	**2.5±0.6**	**2.9±0.8**	**3.1±0.5**	**3.6±0.6**	**3.9±0.5**	**4.1±1.3**	**4.3±0.6**
		Voraz	**1.6±1.0**	2.0±0.4	**2.5±0.6**	**2.9±0.8**	**3.1±0.5**	**3.4±0.6**	**3.8±0.3**	**4.0±1.2**	**4.3±1.0**
	9	Wilson	2.9±1.1	3.3±0.9	4.3±0.3	4.5±0.7	4.6±0.6	4.6±0.8	5.0±0.9	5.5±1.1	5.5±0.6
		FMSC	1.8±1.0	2.0±0.4	**2.5±0.6**	**2.6±0.5**	**3.0±0.4**	**3.4±0.3**	**3.5±0.7**	**4.0±1.1**	**4.1±0.9**
		Voraz	**1.6±1.0**	**1.9±0.3**	**2.5±0.6**	**2.6±0.5**	**2.9±0.5**	**3.3±0.5**	**3.5±0.7**	**3.8±0.9**	**4.1±1.3**
	11	Wilson	2.8±1.0	3.4±0.9	4.3±0.3	4.6±0.6	4.9±0.8	4.6±0.8	5.3±1.0	5.5±1.1	5.5±0.9
		FMSC	1.8±1.0	2.1±0.6	**2.5±0.6**	**2.6±0.5**	**2.9±0.6**	**3.0±0.7**	**3.3±0.5**	**3.9±0.9**	**4.0±0.9**
		Voraz	**1.6±1.0**	2.0±0.4	**2.5±0.6**	**2.6±0.5**	**2.8±0.6**	**2.9±0.9**	**3.3±0.5**	**3.6±0.8**	**4.0±1.3**
	13	Wilson	2.8±0.9	3.4±0.6	4.1±0.5	4.8±0.3	4.9±0.8	4.8±0.9	5.6±1.4	6.1±1.5	6.1±1.4
		FMSC	1.8±1.0	2.1±0.6	**2.4±0.5**	**2.6±0.5**	**2.9±0.6**	**3.0±0.7**	**3.3±0.5**	**3.9±0.9**	**4.0±0.9**
		Voraz	**1.6±1.0**	2.0±0.4	**2.4±0.5**	**2.6±0.5**	**2.8±0.6**	**2.9±0.9**	**3.3±0.5**	**3.8±0.9**	**4.0±1.3**
	15	Wilson	2.8±0.9	3.8±0.9	4.3±0.6	5.0±0.7	5.1±0.9	4.9±0.8	5.9±1.3	6.3±1.3	6.0±1.6
		FMSC	1.8±1.0	2.1±0.6	**2.4±0.5**	**2.6±0.5**	**2.9±0.6**	**3.1±0.5**	**3.3±0.5**	**3.6±0.9**	**3.9±1.1**
		Voraz	**1.6±1.0**	2.0±0.4	**2.4±0.5**	**2.6±0.5**	**2.9±0.9**	**2.9±0.9**	**3.3±0.5**	**3.5±0.7**	**3.9±1.1**
	17	Wilson	2.8±0.9	3.9±1.0	4.4±0.8	5.1±0.6	5.4±0.8	5.3±0.5	5.9±1.0	6.3±1.3	6.1±1.4
		FMSC	**1.6±1.0**	2.0±0.7	**2.4±0.9**	**2.6±0.5**	**3.0±0.8**	**3.3±0.6**	**3.3±0.5**	**3.8±1.0**	**3.9±1.1**
		Voraz	**1.6±1.0**	2.0±0.4	**2.4±0.5**	**2.6±0.5**	**2.9±0.9**	**3.0±1.1**	**3.3±0.5**	**3.6±0.9**	**3.9±1.1**

Los resultados previamente comentados muestran que con respecto al error en la clasificación, como regla el enfoque descrito ofrece una mejor alternativa al algoritmo de Wilson. Se comprueba además que los mejores resultados se obtienen cuando la edición se realiza fijando altos valores para K mientras se toman bajos en la clasificación. Este resultado se repite en cada una de las bases de datos estudiadas, como se puede observar en los mapas de calor en la figura (5.5) que muestran de forma gráfica los datos en las tablas (5.3) y (5.4). Lo anterior sugiere que un incremento en el valor de K durante el proceso de filtrado puede aumentar la posibilidad de que una instancia de la misma clase se encuentre dentro de la K-vecindad de la instancia mal clasificada. Lo anterior induce al algoritmo a mantener más instancias y a poblar algunas regiones con instancias artificiales. En cambio, en la etapa de clasificación un problema con la regla K-NN es que pueden haber pocos objetos de su misma clase alrededor de la instancia que se desea clasificar por lo que el voto puede decantarse a favor de otra clase. En varias situaciones este problema puede evitarse tomando un número reducido de vecinos lo que es consistente con los resultados obtenidos.

Por otra parte, aun cuando teóricamente la eliminación de instancias ruidosas del conjunto de entrenamiento de un clasificador K-NN conduce a una mejora su desempeño, se pudo verificar que en el problema tratado este supuesto no fue válido en la amplia mayoría de los casos. Este resultado es consistente con Ferri *et al.* (1999) y Rico-Juan & Iñesta (2012) siendo una posible causa la alta dimensionalidad de los datos. Una conclusión importante es que el procedimiento descrito permite mejorar el resultado en la clasificación tanto respecto a la regla NN editada como a la clasificación utilizando el conjunto de entrenamiento sin preprocesamiento alguno. Es decir, la adición de los prototipos artificiales calculados como la media entre una supuesta instancia mal etiquetada y su K-vecino de igual clase más próximo favorece el efectividad del clasificador.

Otro elemento interesante a analizar es la cantidad de instancias en el conjunto editado. Wilson descarta todas las muestras incorrectamente clasificadas por lo que el tamaño del conjunto editado nunca se incre-

mentará. Por otra parte, el enfoque propuesto puede no borrar cada instancia marcada para tal por Wilson, llegando incluso a añadir nuevos prototipos a los datos. Como muestran los experimentos, en todos los casos el tamaño del conjunto editado S' parece crecer en un factor lineal acotado por K. Esto es, $|S'| \leqslant |S| + \frac{K*|S|}{100}$.

Finalmente y tal como se esperaba, se evidencia que la utilización de FMSC-Voraz mantiene los mismos resultados. Un ejemplo puede verse en la figura (5.6) que muestra el error promedio al clasificar empleando diferentes conjuntos editados. Para ilustrar se han tomado los extremos del rango de valores evaluados para K en la clasificación, es decir $K = 1$ y $K = 17$. Como se observa, las gráficas correspondientes a FMSC y FMSC-Voraz para un mismo valor de K al clasificar son muy similares lo que indica resultados equivalentes.

5.8. Conclusiones parciales.

En el capítulo se abordó el problema de la obtención del promedio de dos contornos representados mediante códigos de Freeman. Con este propósito se describió un esquema que permite encontrar una cadena que cumple los requerimientos dados para la cadena que representará el contorno promedio. También se cubrió una implementación heurística de dicho procedimiento que permite acelerar el cálculo del contorno medio, rebajando el coste computacional de $\mathcal{O}(max\{L^2, L \times D^2\})$ a $\mathcal{O}(L^2)$. Como ejemplo de problema donde se requiere la construcción de un contorno medio se describió un nuevo algoritmo de edición que atiende algunas de las deficiencias que puede presentar el algoritmo de Wilson y otros derivados de este.

Se realizaron diferentes experimentos utilizando tres conocidas bases de datos para validar la efectividad de cada uno de los algoritmos descritos. En un primer caso, las dos propuestas que se presentan en la tesis para el cálculo del contorno medio fueron comparadas con otros enfoques descritos en la literatura uno de los cuales brinda muy buenos resultados. Se pudo comprobar que independientemente de la

longitud de las cadenas y el contorno que representan, es posible obtener mediante el enfoque propuesto una aproximación al contorno promedio adecuada de acuerdo a los requerimientos dados. Un elemento importante es que para el problema particular tratado, los procedimientos descritos permiten reducir el coste computacional, esto sin degradar la calidad de la respuesta obtenida, superando o igualando en el peor de los casos a los enfoques tomados como referencia. En particular, los bajos tiempos de ejecución de la variante heurística la hacen muy adecuada cuando se trabaja con cadenas de gran longitud.

Respecto al algoritmo de edición presentado, se logró reducir el error en la clasificación en cerca del $83\,\%$ de las pruebas notándose resultados equivalentes al emplear una u otra variante para construir la instancia artificial. Esto confirma la idea de que la adición de instancias artificiales puede ser beneficiosa. No obstante, aun cuando en este sentido los resultados son muy alentadores, se pudo verificar una correlación positiva entre la disminución del error y el aumento de la talla del conjunto editado. Este fenómeno puede hacer inadecuado al enfoque que se propone en tareas donde no solo se requiera disminuir el error en la clasificación sino también reducir el tamaño de los datos de entrenamiento.

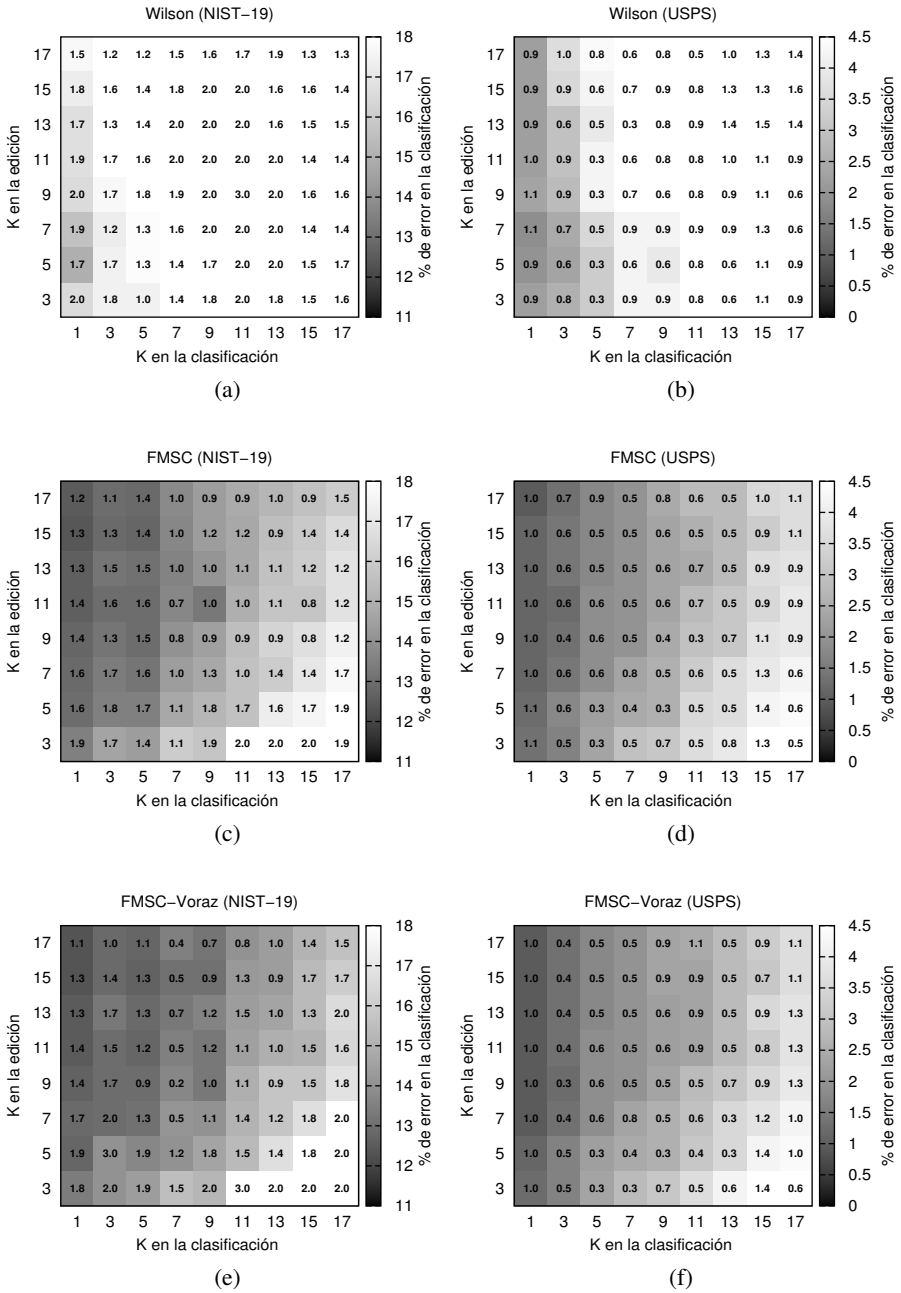


Figura 5.5: Mapas de calor para el error promedio de los diferentes algoritmos y bases de datos. Los mayores niveles de gris se corresponden con los mejores resultados y el número en cada casilla indica la desviación típica.

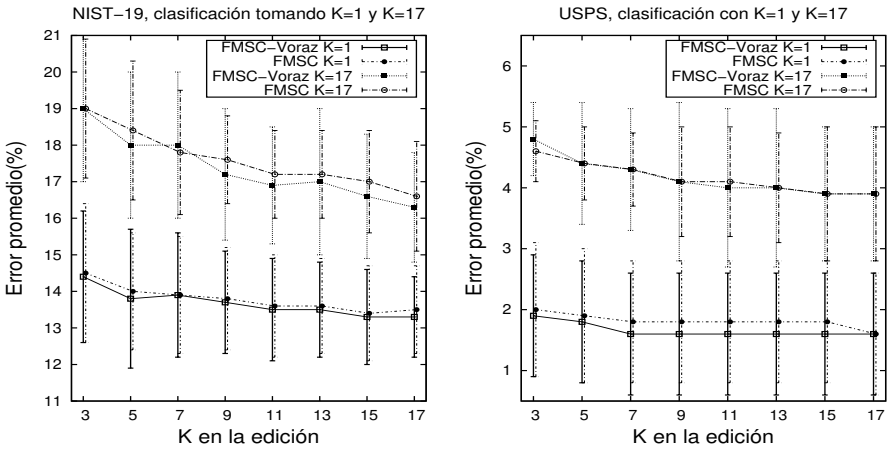


Figura 5.6: Error promedio en la clasificación tomando K=1 y K=17 utilizando los conjuntos editados con diferentes valores de K en las bases de datos NIST-19 (5.6a) y USPS (5.6b).

Capítulo 6

Cálculo de la cadena media.

Según se ha apuntado, dado un alfabeto $\sum$, un conjunto de cadenas S sobre este y una métrica que permita caracterizar la distancia entre las cadenas se define como cadena media aquella que minimiza $SOD(R^*, S) = \sum D(R^*, S_j)|S_j \in S$. Dado que en la mayoría de los casos encontrarla se vuelve un problema intratable computacionalmente se han descrito variados enfoques que permiten sino encontrar la media verdadera al menos obtener una buena aproximación. Se ha visto que un esquema general para buscar una aproximación se puede estructurar en los siguientes pasos:

- seleccionar una aproximación burda a la media, como por ejemplo la mediana de los elementos de un conjunto.
- generar nuevas soluciones haciendo una o más modificaciones a la solución actual.
- repetir mientras alguna modificación conduzca a una mejora o se verifique otra condición de parada.

De un análisis de este enfoque se desprende que es importante definir cómo se seleccionan las modificaciones adecuadas y cuántas realizar simultáneamente. Como se ha visto, estos aspectos pueden incidir directamente en la velocidad de convergencia y en la calidad de la solución obtenida.

En el capítulo se describirá un algoritmo para obtener una aproximación a la cadena media basado en dos ideas fundamentales. En primer lugar, seleccionar la modificación adecuada atendiendo a información estadística resultante del cálculo de la distancia de edición de la solución parcial a cada cadena en el conjunto. Una de las hipótesis es que de este modo se puede evitar probar un gran número de soluciones malas, lo que se traduce en una reducción del número de distancias calculadas. La otra característica importante es efectuar las modificaciones de una en una.

6.1. Algoritmo para obtener una aproximación a la cadena media.

Al calcular la distancia de la solución parcial R^t a cada elemento del conjunto $S = \{S_1, S_2, ..., S_N\}$ puede obtenerse también una lista de operaciones relativas a cada posición de esta cadena. Por ejemplo, al calcular $D(R^t, S_1)$ podría resultar como operación en la secuencia de coste mínimo un borrado del primer símbolo de R^t, de $D(R^t, S_2)$ que este símbolo debe ser sustituido por otro, de $D(R^t, S_3)$ que debe ser eliminado y así para el resto de las cadenas. Al final se tendrá qué operación se realizó en cada caso. Con esta información la operación de edición más adecuada en el paso t será seleccionada empleando dos enfoques diferentes. El primero simplemente ordena las operaciones de forma decreciente de acuerdo a su *frecuencia* al calcular la distancia de la solución parcial a todas las cadenas del conjunto, de forma análoga a como proponen Fischer & Zell (2000). Puede esperarse que al aplicar esta operación se elimine una diferencia de R^t que es frecuente respecto a los elementos en S. Es decir, dada una operación q_i las cadenas en S pueden dividirse en dos grupos C_{SI}, aquellas que votan por q_i, y aquellas que no lo hacen C_{NO}. Al aplicar q_i al menos se garantiza que la distancia de la nueva solución R^t disminuya respecto a cualquiera de las cadenas en C_{SI}, intuitivamente puede esperarse que mientras más sean, más prometedora será la operación, esto es, que se logrará una mayor reducción del valor de (e1.2). Claramente,

para una mejor evaluación de qué tan bueno será realizar o no q_i debe considerarse también el valor de las distancias de R^{t+1} a las cadenas en C_{NO}.

Otro elemento también importante es que, al emplear un esquema más general para asignar costes a las operaciones de edición la frecuencia puede no ser la mejor medida para evaluar qué tan prometedora resulta una transformación. Por ejemplo, sin perder generalidad supongamos que la operación q_1 con mejor ranking es una sustitución con frecuencia 2 y coste 1 aplicada a alguna posición de R^t cuando $D(R^t, S_1)$ y $D(R^t, S_2)$ se calcularon. Supongamos además que existe otra sustitución q_2 con frecuencia 1 pero coste 3. De acuerdo a los resultados presentados por Bunke *et al.* (2002) se tiene que una cadena R^{t+1} obtenida al aplicar q_1 deberá satisfacer que $D(R^{t+1}, S_1) = D(R^t, S_1) - 1$ y $D(R^{t+1}, S_2) = D(R^t, S_2) - 1$. Sin tener en cuenta cuál sea el valor de $D(R^{t+1}, S_3)$ puede esperarse que (e1.2) disminuirá en 2. Un análisis similar indica que aplicando q_2 se podría lograr una reducción de 3. El problema radica en que al utilizar solo la *frecuencia* no se tiene en cuenta el coste de la operación, que es otra variable que influye en el valor de la reducción de la distancia. En este caso, se propone emplear *frecuencia* $\times$ *coste* como índice de calidad que puede brindar información más completa sobre qué tan promisoria es la operación.

El procedimiento *AppMedia* esboza como obtener la aproximación a la cadena media.

El siguiente ejemplo ilustra el funcionamiento del algoritmo. Sea $R^t = \{5, 5, 0\}$, $S_1 = \{3, 1, 1, 2\}$ y $S_2 = \{0, 6, 1, 6\}$. En la tabla (6.1) se muestra el cálculo de la distancia de edición de R^t a S_1 y S_2. En el primer caso puede extraerse como secuencia óptima de operaciones de edición $\{w(5, 3), w(5, 1), w(0, 1), w(\varepsilon, 2)\}$ mientras que de $D(R^t, S_2)$ resulta $\{w(5, 0), w(5, 6), w(0, 1), w(\varepsilon, 6)\}$. La tabla (6.2) muestra las operaciones ordenadas de forma descendente de acuerdo a su frecuencia. Debe notarse como un índice de calidad diferente conduce a un ranking distinto. Aplicando la mejor operación $w(0, 1)$ en la posición 3 se obtiene $R^{t+1} = \{5, 5, 1\}$ que conduce a una mejora dado que $D(R^{t+1}, S_1) = 8$ y $D(R^{t+1}, S_2) = 6$. En caso de que la mejor opera-

Función `AppMedia(`S`,`R`)` :R'

```
/* S: conjunto de cadenas para obtener la media                    */
/* R: aproximación inicial                                         */
```

$R' = R$;
hacer
  $M = R'$;
  para cada *instancia* $S_i \in S$ **hacer**
    calcular $D(R', S_i)$;
    obtener Q, la secuencia de edición óptima para transformar R' en S_i;
    actualizar la información estadística para cada posición de R';
  fin para cada
  sea P_p una cola de operaciones ordenadas de acuerdo a su índice de calidad;
  mientras $P_p \neq \varnothing$ **y** $\sum_{S_i \in S} D(M, S_i) \leqslant \sum_{S_i \in S} D(R', S_i)$ **hacer**
    $q_i = P_p.frente$;
    aplicar q_i a R para obtener R';
  fin mientras
mientras *alguna modificación mejore* ;
devolver el último R' encontrado con valor mínimo de $\sum_{s_i \in T} D(R', S_i)$;

ción no mejore los resultados se prueba la segunda mejor y así sucesivamente mientras resten operaciones por evaluar. El proceso se repite comenzado por la nueva solución si alguna de las operaciones condujo a una mejor aproximación a la media.

El ejemplo anterior también ilustra cómo ordenando por *frecuencia* $\times$ *coste* es posible obtener mejores resultados. Como se explicó, aplicando $w(0,1)$ se llega a $SOD(R^{t+1},S)=14$. Sin embargo si $w(5,1)$ en la posición 2 fuese seleccionada entonces $R^{t+1}=\{5,1,0\}$ de donde $D(R^{t+1},S_1)=5$ y $D(R^{t+1},S_2)=5$, es decir $SOD(R^{t+1},S)=10$.

Tabla 6.1: Cálculo de la distancia de edición de R^t a S_1 y S_2. La secuencia óptima aparece sombreada para su mejor identificación.

(a)

		3	1	1	2
	0	2	4	6	8
5	2	2	4	6	8
5	4	4	6	8	9
0	6	6	5	7	9

(b)

		0	6	1	6
	0	2	4	6	8
5	2	3	3	5	7
5	4	5	4	6	6
0	6	4	6	5	7

Tabla 6.2: Operaciones ordenadas de acuerdo a la frecuencia.

Operación	Posición	Frecuencia	Coste	Frecuencia $\times$ Coste
$w(0,1)$	3	2	1	2
$w(5,0)$	1	1	3	3
$w(5,1)$	2	1	4	4
$w(5,6)$	2	1	1	1
$w(5,3)$	1	1	2	2
$w(\varepsilon,2)$	3	1	2	2
$w(\varepsilon,6)$	3	1	2	2

6.2. Análisis del coste computacional.

El procedimiento empleado para calcular la aproximación a la cadena media requiere calcular la distancia desde la solución parcial a cada elemento en el conjunto. Empleando la distancia de Levenshtein esto

puede hacerse en tiempo proporcional a $\mathcal{O}(L^2)$ de acuerdo al algoritmo de programación dinámica descrito por Wagner & Fischer (1974) donde L es la longitud de la cadena más larga. La instrucción **para cada** se repite tantas veces como cadenas tenga el conjunto, es decir N, por lo que la primera etapa del algoritmo consume un tiempo proporcional a $\mathcal{O}(N \times L^2)$.

Asumiendo que ninguna perturbación refina la solución, la instrucción **mientras** interior examina completamente la cola P_p. Supongamos que la solución candidata R^t tiene a lo sumo longitud L y que $|\sum| > N$. Entonces se tienen $\mathcal{O}(L \times |\sum|)$ posibles sustituciones, $|\sum|$ por cada uno de los L símbolos en R^t. El mismo resultado se obtiene para las inserciones. En el caso de los borrados, solo son posibles L. De acuerdo a lo anterior, una cota superior pesimista para $|P_p|$ puede fijarse como $\mathcal{O}((L \times |\sum|) + L)$. En el peor caso, cada operación P_p implica calcular la distancia de R^t a todas las cadenas lo que requiere $\mathcal{O}(N \times L^2)$. Según estas suposiciones el ciclo **mientras** consume un tiempo proporcional a $\mathcal{O}(N \times L^3 \times |\sum|)$.

Sea además K el número de veces que se repite la instrucción **mientras** más exterior, entonces el algoritmo tiene coste $\mathcal{O}(K \times N \times L^3 \times |\sum|)$ que es similar al algoritmo propuesto por Martínez-Hinarejos *et al.* (2003). Como quiera, en la práctica el enfoque descrito se comporta mucho mejor.

6.3. Resultados experimentales.

Para evaluar el desempeño del algoritmo propuesto se desarrollaron varios experimentos. En este caso $\sum = \{0, 1, 2, 3, 4, 5, 6, 7\}$, símbolos que se corresponden a las direcciones de los códigos de Freeman. Las cadenas en cada conjunto no fueron generadas aleatoriamente sino que se corresponden a la representación mediante códigos de Freeman de una muestra de contornos seleccionada de las bases de datos descritas en el epígrafe (1.5.5). Se tomaron 20 instancias por cada una

de las categorías, en la tabla (6.3) pueden verse más detalles de la muestra.

Tabla 6.3: Características de los datos empleados en el experimento para validar el cálculo de la cadena media.

Corpus	Clases	Instancias x Clase	Longitud de las cadenas		
			Promedio	Max.	Min.
NIST-19	26	20	200±60	450	14
USPS	10	20	160± 40	268	78

Con estos datos se calculó una aproximación a la cadena media para cada conjunto mediante el algoritmo propuesto. Éste, identificado por *JR-S* fue comparado con los métodos descritos por Fischer & Zell (2000) y Cárdenas (2004) en los que se realizan modificaciones simultáneas a la solución candidata y con la propuesta de Martínez-Hinarejos *et al.* (2003) donde se aplica una perturbación cada vez. De manera más precisa se estudian las siguientes variables:

Variable independiente: Algoritmo utilizado para obtener una aproximación a la cadena media.

Definición conceptual: Dado un conjunto de cadenas, estos algoritmos se proponen encontrar aquella que minimice (e1.2) o en su defecto una buena aproximación de acuerdo a una métrica dada, en este caso la descrita en (1.5.2).

Variable dependiente: Suma de las distancias de la aproximación a la cadena media a cada elemento en el conjunto.

Definición conceptual: Se entiende por la distancia de acuerdo a la métrica elegida de la solución propuesta hasta las cadenas en el conjunto. Está dada por la fórmula (e1.2).

Definición operacional: Para medir esta variable es preciso implementar un algoritmo que permita el cálculo de la distancia de edición de Levenshtein. Los costes asociados a cada operación de

edición se toman de acuerdo a como se definieron en la sección (1.5.2).

Las tablas (6.4) y (6.5) muestran los resultados para cada conjunto en la correspondiente base de datos. En cada caso ε, R^M o R^G se refieren a la solución inicial, que son la cadena vacía, la mediana y la inicialización voraz propuesta por Casacuberta & Antoni (1997). En las tablas (6.6) y (6.7) se muestra el coeficiente $\frac{SOD(R,S)}{SOD(R^M,S)}$ donde R^M es la cadena mediana, R la aproximación a la media obtenida por alguno de los métodos estudiados y S el conjunto de cadenas. De acuerdo a esta definición, cuanto mejor sea la aproximación obtenida respecto a la mediana menor será este valor. Analizando el promedio y la desviación típica se comprueba que el algoritmo propuesto y los métodos descritos por Martínez-Hinarejos *et al.* (2003) conducen a las mejores aproximaciones siendo muy similares los resultados de uno y otro enfoque.

Dado que todos los algoritmos evaluados trabajan de manera iterativa, se estudió el número total de distancias calculadas. En las tablas (6.8) and (6.9) puede verse una comparación.

Los resultados sugieren que aplicar las perturbaciones de una en una conduce a mejores aproximaciones a la media. En cada conjunto los mejores resultados se alcanzan al aplicar el algoritmo propuesto o el descrito por Martínez-Hinarejos *et al.* (2003). En general *JR-S* brinda soluciones equivalentes o incluso mejores que las obtenidas por Martínez-Hinarejos *et al.* (2003) pero como muestran las tablas (6.8) y (6.9) el algoritmo descrito solamente calcula una fracción mucho menor de las distancias requeridas por Hinarejos-R^M o Hinarejos-R^G. Como quiera, aunque establecer el ranking de operaciones de acuerdo a *frecuencia* $\times$ *coste* conduce en algunos casos a soluciones ligeramente mejores, en general también implica calcular una cantidad adicional de distancias respecto a los resultados obtenidos al ordenar las operaciones solo por la *frecuencia*.

Por otra parte, aunque los resultados no son tan buenos los métodos descritos por Fischer & Zell (2000) y Cárdenas (2004) necesitan cal-

cular solo unas pocas distancias para mejorar considerablemente la aproximación dada por la mediana del conjunto. En ambos casos los resultados sugieren que el algoritmo se atasca en un mínimo local luego de un pequeño número de iteraciones.

Tabla 6.4: Distancia promedio de la media aproximada a las cadenas en el conjunto (NIST-19).

Clase	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
A	105.1	101.6	100.1	102.9	98.6	93.3	93.2	93.3	93.5	93.6	93.1
B	123.9	120.1	120.8	121.3	119.3	110.3	110.1	110.4	110.0	110.0	110.0
C	132.1	121.6	122.1	122.0	122.3	115.5	115.8	115.4	115.4	115.4	115.7
D	173.0	179.4	172.9	179.1	172.9	163.0	163.2	163.0	162.7	162.8	162.8
E	184.4	177.5	175.9	175.7	174.4	166.5	166.3	166.0	165.9	166.3	166.2
F	156.0	150.9	149.8	156.1	149.3	141.4	141.4	141.4	141.3	141.3	141.3
G	184.6	180.5	178.1	177.3	176.5	167.7	167.6	167.8	167.8	167.8	167.5
H	160.5	158.4	160.4	156.5	157.3	146.7	146.3	146.9	146.5	146.2	146.8
I	125.7	136.6	125.7	130.4	125.7	120.1	120.6	120.5	120.7	120.9	120.3
J	190.4	185.3	185.3	183.6	182.8	169.9	170.0	170.1	170.0	170.0	169.9
K	172.1	165.6	165.4	162.7	163.6	154.2	154.7	154.2	155.1	154.1	154.2
L	85.0	82.4	79.3	84.1	79.3	77.2	77.4	77.2	77.5	77.2	77.3
M	215.5	217.1	215.4	211.3	215.4	197.3	197.2	196.8	197.2	197.2	197.1
N	174.3	173.9	173.1	172.6	170.9	164.0	164.1	164.0	163.6	164.5	163.5
O	76.8	74.6	75.4	76.6	75.8	71.4	71.2	71.7	71.1	71.0	71.0
P	87.8	83.2	86.8	81.7	84.9	78.2	78.6	78.5	78.3	78.4	78.3
Q	104.1	105.8	104.1	101.7	104.1	93.5	93.7	93.5	93.9	93.8	93.7
R	157.1	160.1	157.1	156.3	155.2	143.3	143.5	143.2	143.6	142.7	143.0
S	147.5	143.4	141.8	142.8	138.7	133.1	133.2	133.5	133.1	133.2	133.3
T	133.9	141.8	133.9	133.8	133.9	127.1	127.2	127.2	127.4	126.9	127.1
U	166.1	175.4	166.1	174.0	166.1	158.7	159.1	159.1	159.9	158.8	159.6
V	130.8	129.0	130.8	127.3	128.7	123.2	123.5	123.6	123.5	123.5	123.1
W	210.9	207.2	207.1	204.0	201.9	192.5	192.5	192.3	192.6	192.4	192.6
X	144.9	143.9	142.3	139.5	141.2	132.5	132.4	132.5	133.3	132.4	133.0
Y	149.7	152.9	149.6	150.1	149.6	141.0	141.8	141.0	141.6	141.6	141.1
Z	162.1	160.6	160.7	162.4	157.6	150.6	150.5	150.4	150.6	150.5	150.7

6.4. Conclusiones parciales.

Se ha presentado un nuevo enfoque para obtener una aproximación a la media de un conjunto de cadenas. Para seleccionar qué opera-

Tabla 6.5: Distancia promedio de la media aproximada a las cadenas en el conjunto (USPS).

Clase	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
0	53.0	49.9	50.5	54.3	50.5	47.5	47.4	47.8	47.5	47.6	47.5
1	46.0	43.2	44.0	46.5	44.6	41.3	41.6	41.7	41.6	41.9	41.2
2	117.4	111.1	111.6	113.8	109.7	105.3	104.9	104.7	105.3	105.1	105.0
3	120.1	110.0	109.3	113.7	108.9	103.9	104.9	104.3	104.0	104.5	104.5
4	149.7	146.4	145.0	146.6	144.5	138.3	138.0	138.9	138.4	138.1	138.3
5	147.8	135.7	137.4	140.2	135.2	128.8	128.5	128.6	128.3	128.3	128.2
6	82.7	80.3	80.1	85.6	79.7	75.4	75.7	75.6	75.6	75.5	75.7
7	105.2	98.1	95.3	99.3	96.7	91.7	91.5	92.7	91.7	91.5	91.4
8	91.2	88.8	84.9	90.3	87.0	80.9	80.5	80.4	80.3	80.6	80.8
9	87.8	83.9	82.7	85.8	83.3	79.3	79.0	79.3	79.1	79.7	79.6

ción realizar se tiene en cuenta su *coste* y su *frecuencia* al calcular la distancia de la solución intermedia a las cadenas del conjunto. El algoritmo realiza sucesivas mejoras a una solución parcial, aplicando las modificaciones una a la vez de forma similar a como se sugiere en Martínez-Hinarejos *et al.* (2003).

Resultados empíricos muestran que este enfoque conduce a mejores aproximaciones a la cadena media que los métodos que aplican múltiples modificaciones de forma simultánea, aunque estos tienen tiempos de ejecución mucho menores. Comparaciones con la propuesta de Martínez-Hinarejos *et al.* (2003) indican que el algoritmo propuesto es capaz de calcular aproximaciones muy buenas a la cadena media lo que unido a su mayor velocidad de convergencia lo hace muy adecuado en aplicaciones donde se requiera una aproximación precisa.

Tabla 6.6: Coeficiente entre la distancia acumulada de la aproximación obtenida por cada algoritmo y la cadena mediana. Los valores menores indican un mejor resultado respecto a la mediana en términos de (e1.2) (NIST-19).

Clase	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
A	1.00	0.97	0.95	0.98	0.94	0.89	0.89	0.89	0.89	0.89	0.89
B	1.00	0.97	0.97	0.98	0.96	0.89	0.89	0.89	0.89	0.89	0.89
C	1.00	0.92	0.92	0.92	0.93	0.87	0.88	0.87	0.87	0.87	0.88
D	1.00	1.04	1.00	1.04	1.00	0.94	0.94	0.94	0.94	0.94	0.94
E	1.00	0.96	0.95	0.95	0.95	0.90	0.90	0.90	0.90	0.90	0.90
F	1.00	0.97	0.96	1.00	0.96	0.91	0.91	0.91	0.91	0.91	0.91
G	1.00	0.98	0.96	0.96	0.96	0.91	0.91	0.91	0.91	0.91	0.91
H	1.00	0.99	1.00	0.98	0.98	0.91	0.91	0.92	0.91	0.91	0.91
I	1.00	1.09	1.00	1.04	1.00	0.96	0.96	0.96	0.96	0.96	0.96
J	1.00	0.97	0.97	0.96	0.96	0.89	0.89	0.89	0.89	0.89	0.89
K	1.00	0.96	0.96	0.95	0.95	0.90	0.90	0.90	0.90	0.90	0.90
L	1.00	0.97	0.93	0.99	0.93	0.91	0.91	0.91	0.91	0.91	0.91
M	1.00	1.01	1.00	0.98	1.00	0.92	0.92	0.91	0.92	0.92	0.91
N	1.00	1.00	0.99	0.99	0.98	0.94	0.94	0.94	0.94	0.94	0.94
O	1.00	0.97	0.98	1.00	0.99	0.93	0.93	0.93	0.93	0.92	0.93
P	1.00	0.95	0.99	0.93	0.97	0.89	0.90	0.89	0.89	0.89	0.89
Q	1.00	1.02	1.00	0.98	1.00	0.90	0.90	0.90	0.90	0.90	0.90
R	1.00	1.02	1.00	0.99	0.99	0.91	0.91	0.91	0.91	0.91	0.91
S	1.00	0.97	0.96	0.97	0.94	0.90	0.90	0.91	0.90	0.90	0.90
T	1.00	1.06	1.00	1.00	1.00	0.95	0.95	0.95	0.95	0.95	0.95
U	1.00	1.06	1.00	1.05	1.00	0.96	0.96	0.96	0.96	0.96	0.96
V	1.00	0.99	1.00	0.97	0.98	0.94	0.94	0.94	0.94	0.94	0.94
W	1.00	0.98	0.98	0.97	0.96	0.91	0.91	0.91	0.91	0.91	0.91
X	1.00	0.99	0.98	0.96	0.97	0.91	0.91	0.91	0.92	0.91	0.92
Y	1.00	1.02	1.00	1.00	1.00	0.94	0.95	0.94	0.95	0.95	0.94
Z	1.00	0.99	0.99	1.00	0.97	0.93	0.93	0.93	0.93	0.93	0.93
Promedio	1.00	0.99	0.98	0.98	0.97	0.92	0.92	0.92	0.92	0.92	0.92
σ	0.00	0.04	0.02	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02

Tabla 6.7: Coeficiente entre la distancia acumulada de la aproximación obtenida por cada algoritmo y la cadena mediana. Los valores menores indican un mejor resultado respecto a la mediana en términos de (e1.2) (USPS).

Clase	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
0	1.00	0.94	0.95	1.02	0.95	0.90	0.89	0.90	0.90	0.90	0.90
1	1.00	0.94	0.96	1.01	0.97	0.90	0.90	0.91	0.90	0.91	0.89
2	1.00	0.95	0.95	0.97	0.93	0.90	0.89	0.89	0.90	0.89	0.89
3	1.00	0.92	0.91	0.95	0.91	0.87	0.87	0.87	0.87	0.87	0.87
4	1.00	0.98	0.97	0.98	0.97	0.92	0.92	0.93	0.92	0.92	0.92
5	1.00	0.92	0.93	0.95	0.91	0.87	0.87	0.87	0.87	0.87	0.87
6	1.00	0.97	0.97	1.04	0.96	0.91	0.91	0.91	0.91	0.91	0.92
7	1.00	0.93	0.91	0.94	0.92	0.87	0.87	0.88	0.87	0.87	0.87
8	1.00	0.97	0.93	0.99	0.95	0.89	0.88	0.88	0.88	0.88	0.89
9	1.00	0.96	0.94	0.98	0.95	0.90	0.90	0.90	0.90	0.91	0.91
Promedio	1.00	0.95	0.94	0.98	0.94	0.89	0.89	0.89	0.89	0.89	0.89
σ	0.00	0.02	0.02	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02

Tabla 6.8: Número de distancias calculadas (en miles)(NIST-19).

Clase	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
A	0.2	0.3	0.2	0.3	0.3	47.9	47.9	31.0	23.5	550.2	367.0
B	0.2	0.3	0.2	0.3	0.2	60.4	47.3	37.3	44.5	328.8	425.9
C	0.2	0.3	0.3	0.3	0.3	49.7	47.6	32.9	26.8	1116.0	529.0
D	0.2	0.3	0.2	0.4	0.2	117.6	76.5	95.3	98.5	310.1	428.1
E	0.2	0.3	0.2	0.5	0.2	78.1	85.5	76.6	55.6	1075.7	565.6
F	0.2	0.3	0.3	0.4	0.2	70.9	72.6	62.0	51.8	644.4	458.9
G	0.2	0.3	0.2	0.4	0.3	122.1	84.0	71.3	100.6	1177.7	693.9
H	0.2	0.3	0.2	0.3	0.3	79.1	56.9	56.2	54.4	661.3	691.0
I	0.2	0.2	0.2	0.2	0.2	62.6	34.8	40.1	31.0	366.7	166.4
J	0.2	0.3	0.3	0.3	0.3	91.2	86.3	98.8	91.6	553.8	467.1
K	0.2	0.4	0.3	0.4	0.3	115.1	77.1	91.2	64.4	768.0	534.8
L	0.2	0.5	0.2	0.4	0.2	30.9	20.7	18.6	14.7	232.5	273.9
M	0.2	0.3	0.2	0.4	0.2	173.2	127.7	166.5	88.0	661.9	695.8
N	0.2	0.3	0.3	0.4	0.3	135.5	126.0	101.6	110.7	772.7	536.4
O	0.2	0.4	0.2	0.3	0.2	30.5	22.4	17.5	18.2	226.5	210.2
P	0.2	0.4	0.3	0.4	0.3	33.9	26.1	30.9	40.3	290.2	337.5
Q	0.2	0.2	0.2	0.2	0.2	46.1	36.9	35.3	24.4	289.1	354.2
R	0.2	0.3	0.2	0.3	0.2	98.3	61.5	83.0	71.8	953.6	426.9
S	0.2	0.3	0.2	0.2	0.2	120.1	73.4	78.6	66.1	621.3	260.2
T	0.2	0.2	0.2	0.3	0.2	48.5	42.0	35.2	33.7	729.3	347.3
U	0.2	0.4	0.2	0.4	0.2	106.9	104.6	71.2	57.9	676.5	437.0
V	0.2	0.3	0.2	0.3	0.3	73.4	48.9	34.0	30.0	713.9	450.2
W	0.2	0.3	0.3	0.4	0.3	142.8	107.6	115.9	76.1	922.2	681.6
X	0.2	0.3	0.2	0.4	0.3	106.5	78.7	67.0	44.8	675.5	436.3
Y	0.2	0.3	0.2	0.4	0.2	134.3	70.9	89.6	44.5	539.3	482.3
Z	0.2	0.3	0.3	0.5	0.3	91.9	46.8	71.5	45.1	343.9	402.2
Promedio	0.2	0.3	0.2	0.4	0.2	87.2	65.8	65.7	54.2	623.1	448.4
σ	0.0	0.1	0.0	0.1	0.0	38.6	29.7	34.8	27.2	276.0	143.6

Tabla 6.9: Número de distancias calculadas (en miles)(USPS).

Classe	Mediana	Cárdenas-ε	Cárdenas-R^M	Fischer-ε	Fischer-R^M	JR-S ε Freq x Cost	JR-S R^M Freq x Cost	JR-S ε Freq	JR-S R^M Freq	Hinarejos-R^G	Hinarejos-R^M
0	0.2	0.3	0.2	0.3	0.2	13.5	12.1	22.7	10.9	217.6	233.0
1	0.2	0.2	0.2	0.3	0.2	10.2	9.3	11.1	13.1	160.7	224.2
2	0.2	0.3	0.3	0.3	0.3	55.9	50.2	46.5	38.1	492.7	247.6
3	0.2	0.3	0.2	0.3	0.2	81.2	35.4	48.2	53.5	751.3	484.4
4	0.2	0.2	0.3	0.3	0.3	90.6	78.0	50.5	63.0	857.6	499.7
5	0.2	0.3	0.3	0.4	0.3	69.0	64.7	67.6	39.2	827.5	514.8
6	0.2	0.3	0.2	0.3	0.3	30.2	32.9	24.6	18.4	645.9	280.4
7	0.2	0.3	0.3	0.3	0.3	47.3	52.8	31.0	41.5	623.6	554.2
8	0.2	0.2	0.3	0.2	0.3	29.2	26.1	53.5	29.4	296.6	277.6
9	0.2	0.2	0.3	0.3	0.3	26.4	25.6	39.5	29.8	589.6	295.6
Promedio	0.2	0.3	0.2	0.3	0.2	45.3	38.7	39.5	33.7	546.3	361.1
σ	0.0	0.0	0.0	0.0	0.0	28.1	22.4	17.0	16.9	248.9	133.8

Parte III

Conclusiones finales y trabajos futuros

Capítulo 7

Conclusiones

En el trabajo se ha abordado el empleo de las cadenas de Freeman y la distancia de Levenshtein para la identificación de similitudes entre contornos. Se estudió la posibilidad de construir prototipos que representen un conjunto de contornos a partir de los elementos similares entre estos y su aplicación en la reducción de la talla del conjunto de entrenamiento de un clasificador K-NN. Además se han propuesto dos enfoques para obtener una aproximación al promedio de dos contornos. Este prototipo artificial se utilizó en un nuevo algoritmo de edición que permite elevar el desempeño de un clasificador K-NN. Se abordó además el problema de obtener una aproximación a la cadena media describiéndose un algoritmo de propósito general aplicándolo en este caso a cadenas de Freeman.

Seguidamente se relacionan las principales aportaciones de la tesis, se sugieren posibles direcciones de trabajo futuro y se lista la producción científica obtenida en el desarrollo de la investigación.

7.1. Principales aportaciones.

Las principales aportaciones de la tesis han sido:

✓Un sumario y análisis crítico de las principales propuestas como solución a las problemáticas abordadas en la tesis, específicamente:

- Identificación de similitudes entre contornos.
- Construcción de prototipos para representar un conjunto de contornos codificados mediante algún tipo de código de cadena.
- Algoritmos para el cálculo de la cadena media.
- Algoritmos de edición.

✓Un método para dados dos segmentos señalados como similares en sus respectivos contornos obtener un nuevo segmento que los represente.

✓Un algoritmo que permite identificar partes similares entre contornos a partir del cálculo de la distancia de edición entre las cadenas de Freeman que los representan.

✓Dos procedimientos para construir una aproximación al promedio entre dos contornos codificados mediante cadenas de Freeman.

✓Un nuevo enfoque para obtener una aproximación a la cadena media.

✓Un nuevo algoritmo de edición que relaja las condiciones para borrar una instancia establecidas por el método de Wilson conduciendo a resultados superiores respecto a la precisión en la clasificación mediante la regla K-NN.

✓Extenso grupo de experimentos para validar cada uno de los enfoques propuestos y que sirven de referencia en futuras investigaciones.

7.2. Trabajo futuros.

En la tesis se han relacionado diferentes cuestiones que constituyen por si solas interesantes temas de investigación. Como quiera consideramos que las siguientes son algunas de las direcciones a las que pueden enfocarse ulteriores trabajos:

- Estudiar posibles mejoras a los algoritmos para la identificación de similitudes entre las cadenas.

Los enfoques propuestos para identificar similitudes entre los contornos tienen una naturaleza eminentemente heurística, por lo que los resultados obtenidos no necesariamente tienen que ser óptimos para los requerimientos dados.

- Extensión de los algoritmos para la detección de similitudes y la construcción del contorno promedio al caso de más de dos contornos.

  Sin dudas una limitación importante de estos algoritmos es la cantidad de patrones que pueden tratar simultáneamente. Como quiera, la extensión a un caso más general presenta dificultades adicionales. En primer lugar, es preciso calcular la distancia de edición, esto es establecer un alineamiento entre los símbolos de las cadenas, lo que en el caso abordado puede lograrse en tiempo polinómico. Sin embargo el alineamiento de múltiples cadenas es un problema NP-Duro como sugieren Wang & Jiang (1994). Por esta razón deben estudiarse diferentes aproximaciones polinómicas para seleccionar aquellas con mejores cotas para el error cometido. Posteriormente puede ser necesario redefinir el concepto de operación dado que más de dos símbolos pueden alinearse al mismo tiempo, incluido el carácter ε.

- Estudiar el modo de utilizar la información que provee el cálculo de la distancia de edición en la identificación de similitudes, construcción de contornos medios y prototipos codificados mediante otros tipos de formalismos como los *conjuntos de vectores ordenados* o los *códigos direccionales relativos*. Pueden considerarse otras distancias de edición como *Dynamic Time Warping*.

  Las cadenas de Freeman utilizadas para representar los contornos tienen varios inconvenientes que han motivado otras derivaciones orientadas a resolverlas. Se han mencionado varias de las propuestas descritas en la literatura para este propósito. En general para estos formalismos pueden utilizarse distintas formulaciones de la distancia de edición o definir algunas a la medida.

- Profundizar en el estudio de las relaciones *contorno medio/cadena media* y *subsecuencias similares en las cadenas/similitudes entre contornos* desde el punto de vista semántico.

La codificación de un contorno mediante un determinado formalismo puede verse como una proyección a un nuevo dominio. En el caso de las cadenas de Freeman este espacio sería el conjunto de todas las cadenas sobre el alfabeto $\sum = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Tanto en esta investigación como en varios de los trabajos consultados, una vez que se han codificado los patrones mediante una representación estructural se aplican los algoritmos sobre éstos para obtener una instancia con las características deseadas, como por ejemplo sería el contorno medio representado por la cadena media. Los resultados demuestran que la instancia así obtenida podría ser satisfactoria para los requerimientos establecidos como tener menor distancia al resto de los elementos del conjunto. Sin embargo esta misma instancia podría no tener una semántica válida en el dominio inicial donde se trabaja.

Por ejemplo, dados los contornos en las figuras (7.1a) y (7.1b) representados por las cadenas $S_1 = \{4, 4, 4, 4, 4, 4, 4, 4, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3\}$ y $S_2 = \{4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 6, 7, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3\}$ donde $D(S_1, S_2) = 24$ supongamos que se desea encontrar una aproximación R a la media sujeta a (e1.2) y (e4.3) es decir que minimice la distancia a S_1 y S_2 al mismo tiempo que se encuentre equidistante de ambas. En la figura (7.1c) se muestra el contorno correspondiente a la cadena $R^B = \{4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3\}$ obtenida utilizando el método descrito por Bunke *et al.* (2002). En este caso tenemos que $D(R^B, S_1) = 12$ y $D(R^B, S_2) = 12$. En la figura (7.1d) se observa el contorno codificado por la cadena $R^{FMSC} = \{4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 6, 7, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3\}$ computada utilizando el algoritmo FMSC. Para esta cadena nuevamente se cumple que $D(R^{FMSC}, S_1) = 12$ y $D(R^{FMSC}, S_2) = 12$. Por último la figura en (7.1e) representa el contorno que resulta de la cadena $R^V = \{4, 4, 4, 4, 4, 4, 4, 4, 5, 6, 7, 6, 7, 6, 7, 7, 7, 7, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3\}$ construida de forma manual pero que podría parecer más adecuada como contorno medio, es decir, es semánticamente más significativa. Sin embargo en este caso se tiene que $D(R^V, S_1) = 9$ y $D(R^V, S_2) = 15$, esto es que no

cumple la restricción de estar a igual distancia de las cadenas S_1 y S_2.

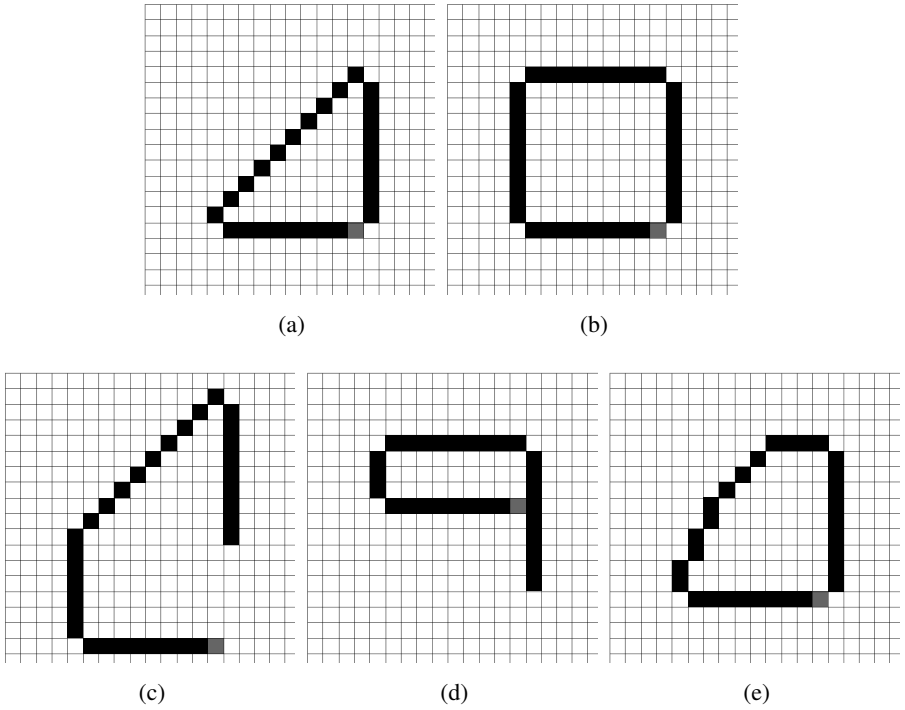


Figura 7.1: Diferencias visuales entre distintas aproximaciones a la media de los contornos en (7.1a) y (7.1b). En (7.1c) y (7.1d) los resultados obtenidos utilizando el algoritmo descrito por Bunke *et al.* (2002) y la propuesta realizada en la tesis respectivamente. Por último en (7.1e) un contorno que no cumple los requerimientos establecidos como contorno medio pero que visualmente es más adecuado como tal. Se ha señalado el pixel correspondiente al comienzo de la cadena en cada caso.

7.3. Producción científica.

El desarrollo de esta investigación ha conducido a las publicaciones que a continuación se mencionan:

En relación con el capítulo (4):

- **Abreu, J.**, Rico-Juan J., *Characterization of contour regularities based on the Levenshtein edit distance.* Pattern Recognition Letters, 2011. Vol. 32, pp. 1421-1427. Factor de Impacto: 1,034
- **Abreu, J.**, Rico-Juan J., *Contour regularity extraction based on string edit distance.* Pattern Recognition and Image Analysis, 2009. LNCS Vol. 5524, pp. 160-167.
  - Este trabajo fue presentado en: 4th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA2009). *Conference Ranking:* C.

Relacionadas con el capítulo (5):

- **Abreu, J.**, Rico-Juan J., *A new editing scheme based on fast two-string median computation applied to OCR.* Structural, Syntactic, and Statistical Pattern Recognition, 2010. LNCS Vol. 6218, pp. 748-756.
  - Este trabajo fue presentado en: 13th International Workshop on Structural and Syntactic Pattern Recognition y 8th International Workshop on Statistical Pattern Recognition (S+SSPR 2010). *Conference Ranking:* A.
- En proceso de revisión:
  - **Abreu, J.** , Rico-Juan, J., *An improved fast edit approach for two-string approximated mean computation applied to OCR.* Enviado a *Pattern Recognition Letters*, 2012.

En referencia al capítulo (6):

- En proceso de revisión:

- **Abreu, J.** , Rico-Juan, J., *A new iterative algorithm for computing an approximated mean of strings based on edit operations.* Enviado a *Pattern Recognition Letters*, 2012.

Parte IV

Bibliografía consultada y apéndices.

Bibliografía

Aha, D. W., Kibler, D., & Albert, M.K. 1991. Instance-based learning algorithms. *Machine Learning*, **6**, 37–66.

Aho, A., Hopcroft, J., & Ullman, J. 1983. *Data Structures and Algorithms*. Addison-Wesley.

Angiulli, Fabrizio. 2005. Fast condensed nearest neighbor rule. *Pages 25–32 of: Proceedings of the 22nd international conference on Machine learning*. ICML '05. New York, NY, USA: ACM.

Angiulli, Fabrizio. 2007. Fast Nearest Neighbor Condensation for Large Data Sets Classification. *IEEE Transactions on Knowledge and Data Engineering*, **19**, 1450–1464.

Arslan, A.N. 2006. An algorithm for string edit distance allowing substring reversals. *Pages 220–226 of: 6th IEEE Symp. on BioInformatics and BioEngineering (BIBE)*.

Bai, Xiang, Yang, Xingwei, & Latecki, Longing Jan. 2008. Detection and recognition of contour parts based on shape similarity. *Pattern Recognition*, **41**, 2189–2199.

Bontempi, B., & Marcelli, A. 1995. Towards a genetic based prototyper for character shapes. *Pages 694–697 of: 3rd International Conference on Document Analysis and Recognition.*, vol. 2.

Brighton, H., & Mellish, C. 2002. Advances in instance -based learning algorithms. *Data Mining and Knowledge Discovery*, **6**, 153–172.

Brodley, C. E. 1993. Addressing the selective superiority problem: automatic algorithm/model class selection. *10th Int. Machine Learning Conference*, 17–24.

Bunke, Horst, Jiang, Xiaoyi, Abegglen, Karin, & Kandel, Abraham. 2002. On the Weighted Mean of a Pair of Strings. *Pattern Analysis & Applications*, **5**, 23–30. 10.1007/s100440200003.

Cameron-Jones, R. M. 1995. Instance selection by enconding lenght heuristicwith random mutation hill climbing. *8th Australian Joint Conf. on Artificial Intelligence*, 99–106.

Cárdenas, R. 2004. A learning model for multiple-prototype classification of strings. *Pages 420–423 of: 17th Int. Conf. on Pattern Recognition (ICPR)*, vol. 4.

Casacuberta, F. 2000. Topology of strings: median string is NPComplete. *Theoretical Computer Science.*, **230**, 39–48.

Casacuberta, Francisco, & Antoni, M.D. 1997. A greedy algorithm for computing approximate median strings. *7th Spanish Symposium on Pattern Recognition and Image Analysis.*, 193–198.

Chang, C. L. 1974. Finding prototypes for nearest neighbor classifier. *IEEE Trans. on Computers*, **23**, 1179–1184.

Chen, Ch., & Józwik, A. 1996. A sample set condensation algorithm for the class sensitive artificial neural network. *Pattern Recognition Letters*, **17**, 819–823.

Cover, T., & Hart, P. 1967. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory.*, **13**, 21–27.

Dasarathy, B. D. 1994. Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Trans. on Systems, Man and Cybernetics*, **24**, 511–517.

Devijver, P. A., & Kittler, J. 1980. On the edited nearest neighbour rule. *Pages 72–80 of: 5th Int. Conf. on Pattern Recognition*.

Devijver, P. A., & Kittler, J. 1982. *Pattern Recogniton, A Statistical Approach*. Prentice-Hall, Englewood Cliffs.

Duda, R., & Hart, P. 2001. *Pattern Classification*. Willey-Interscience.

Duta, N., Sonka, M., & Jain, A. 1999. Learning shape models from examples using automatic shape clustering and procrustes analysis. *LNCS, Information Processing in Medical Imaging*, **1613**, 370–375.

Duta, N., Jain, A., & Dubuisson-Jolly, M. 2001. Automatic construction of 2D shape models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**, 433–446.

Eick, C.F., Zeidat, N., & Vilalta, R. 2004. Using representative-based clustering for nearest neighbor dataset editing. *Pages 375–378 of: 4th IEEE Int. Conf. on Data Mining (ICDM)*.

Fernández, A., & Díaz, J. 2006. Extensión del algoritmo del cálculo de la distancia. *Comunicación personal.*

Ferri, F. J., & Vidal, E. 1992. Comparison of several editing and condensing techniques for colour image segmentation and object location.

Ferri, F.J., Albert, J.V., & Vidal, E. 1999. Considerations about sample-size sensitivity of a family of edited nearest-neighbor rules. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, **29**(5), 667 –672.

Fischer, Igor, & Zell, Andreas. 2000. String Averages and Self-Organizing Map for Strings. *Pages 208–215 of: Proceedings of the Neural Computation 2000, Canada / Switzerland, ICSC*. Academic Press.

Freeman, H. 1974. Computer processing of line-drawing data. *Computer Surveys*, **6**, 57–96.

García-Borroto, Milton, Trinidad, José Francisco Martínez, Ochoa, Jesús Ariel Carrasco, & Ruiz-Shulcloper, José. 2011. *Estado del arte de las sinergias de métodos de selecció de objetos.* Tech. rept. Centro de Aplicaciones de Tecnologías de Avanzada, Cuba.

García-Díez, Silvia, Fouss, François, Shimbo, Masashi, & Saerens, Marco. 2011. A sum-over-paths extension of edit distances accounting for all sequence alignments. *Pattern Recognition*, **44**(6), 1172 – 1182.

Gates, G. 1972. The reduce nearest neighbor rule. *IEEE Trans. on Information Theory*, **18**, 431–433.

Ghosh, Pijush K., & Deguchi, Koichiro. 2008. *Matehamitcs of Shape Description. A Morphological Approach to Image Processing and Computer Graphics.* John Wiley & Sons.

González-Rubio, J., & Casacuberta, F. 2010 (aug.). On the Use of Median String for Multi-source Translation. *Pages 4328 –4331 of: Pattern Recognition (ICPR), 2010 20th International Conference on*.

Guan, Donghai, Yuan, Weiwei, Lee, Young-Koo, & Lee, Sungyoung. 2009. Nearest neighbor editing aided by unlabeled data. *Informa-*

tion Sciences, **179**(13), 2273 – 2282. Special Section on High Order Fuzzy Sets.

Han-fa, Xing, Bing-Liang, Cui, & Xiao-guang, Zhou. 2010. Incremental contour fusion based on line/line topological relation. *Pages 1–5 of: Int. Conf. on Multimedia Technology (ICMT)*.

Hart, P. E. 1968. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory*, **14**, 515–516.

Hellman, M. 1970. The nearest neighbour classification rules with a reject option. *IEEE Trans. on Systems, Man and Cybernetics*, **6**, 179–185.

Jain, A.K., & Zongker, D. 1997. Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**, 1386 –1390.

Jankowski, Norbert, & Grochowski, Marek. 2004a. Comparison of instances selection algorithms I. Algorithms survey. *In:* Rutkowski, Leszek, Siekmann, Jörg, Tadeusiewicz, Ryszard, & Zadeh, Lotfi (eds), *Artificial Intelligence and Soft Computing - ICAISC 2004*. Lecture Notes in Computer Science, vol. 3070.

Jankowski, Norbert, & Grochowski, Marek. 2004b. Comparison of instances selection algorithms II. Results and Comments. *Pages 580–585 of:* Rutkowski, Leszek, Siekmann, Jörg, Tadeusiewicz, Ryszard, & Zadeh, Lotfi (eds), *Artificial Intelligence and Soft Computing - ICAISC 2004*. Lecture Notes in Computer Science, vol. 3070. Springer Berlin / Heidelberg. 10.1007/978-3-540-24844-6-87.

Jiang, X., Schiffmann, L., & Bunke, H. 2000. Computation of median shapes. *In: 4th Asian Conf. on Computer Vision*.

Jiang, X., Abegglen, K., Bunke, H., & Csirik, Janos. 2003. Dynamic computation of generalised median strings. *Pattern Analysis & Applications*, **6**, 185–193. 10.1007/s10044-002-0184-4.

Jiang, Xiaoyi, Wentker, Jöran, & Ferrer, Miquel. 2011. Generalized median string computation by means of string embedding in vector spaces. *Pattern Recognition Letters*, –.

Jusoh, Nor Amizam, & Zain, Jasni Mohamad. 2011. Application of Freeman Chain Codes: An Alternative Recognition Technique for Malaysian Car Plates. *CoRR*, **abs/1101.1602**.

Kohonen, T. 1985. Median Strings. *Pattern Recognition Letters*, **3**, 309–313.

Kohonen, T. 1998. Self-organizing maps of symbols strings. *Neurocomputing*, **21**, 19–30.

Kontschieder, Peter, Donoser, Michael, & Bischof, Horst. 2010. Beyond Pairwise Shape Similarity Analysis. *Pages 655–666 of:* Zha, Hongbin, Taniguchi, Rin-ichiro, & Maybank, Stephen (eds), *Computer Vision-ACCV 2009*. Lecture Notes in Computer Science, vol. 5996. Springer Berlin / Heidelberg. 10.1007/978-3-642-12297-2-63.

Koplowitz, J., & Brown, T. 1981. On the relation of performance to editing in nearest neighbour rules. *Pattern Recognition*, **13**, 251–255.

Kruskal, J. 1983. An overview of sequence comparison. Time warps, string edits and macromolecules. *SIAM Reviews*, **2**, 201–2037.

Kruzslicz, F. 1999. Improved greedy algorithm for computing approximate median strings. *Acta Cybernetica*, **14**, 331–339.

Kubat, M., & Matwin, S. 1997. Addressing the curse of imbalanced training sets: one-side selection. *14th Int. Conf. on Machine Learning*, 179–186.

Kuncheva, Liudmila I., & Bezdek, James C. 1998. Nearest prototype classification: clustering, genetic algorithms, or random search. *IEEE Trans. on Systems, Man and Cybernetics*, **28**, 160–164.

Levenshtein, Vladimir I. 1966. *Binary codes capable of correcting deletions, insertions, and reversals*. Tech. rept. 8.

Li, M., Ma, B., & Wang, L. 1999. Finding similar regions in many strings. *31th Annual ACM Symposium on Theory of Computing*, 473–482.

Liu, Yong Kui, & Žalik, Borut. 2005. An efficient chain code with Huffman coding. *Pattern Recognition*, **38**(4), 553 – 557.

Lourenço, A., & Fred, A. 2005. Ensemble methods in the clustering of string patterns. *Pages 143 – 148 of: 7th IEEE Workshops on Application of Computer Vision (WACV/MOTIONS)*, vol. 1.

Lowe, D. G. 1995. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, **7**, 72–85.

Martínez-Hinarejos, Carlos David. 2003. *La cadena media y su aplicación en reconocimiento de formas.* Ph.D. thesis.

Martínez-Hinarejos, C.D., Juan, A., & Casacuberta, F. 2003. Median strings for k-nearest neighbour classification. *Pattern Recognition Letters*, **24**(1-3), 173–181.

Marzal, A., Palazón, V., & Peris, G. 2006. Contour-based shape retrieval using dynamic time warping. *LNCS Current Topics in Artificial Intelligence*, **4177**, 190–199.

Michie, D., Spiegelhalter, D., & Taylor, C. 1994. *Machine Learning, Neural and Statistical Classification*.

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill.

Mitra, Pabitra, Murthy, C. A., & Pal, Sankar K. 2000. Data condensation in large databases by incremental learning with suppor vector machines. *IEEE*, **-**, –.

Mitra, Pabitra, Murthy, C. A., & Pal, Sankar K. 2002. Density-Based Multiscale Data Condensation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 734–747.

Moreno-Seco, Francisco, Micó, Luisa, & Oncina, José. 2004. A new classification rule based on nearest neighbour search. *Pages 408–411 of: 17th Int. Conf. on Pattern Recognition (ICPR)*, vol. 4.

Nallaperumal, K., Varghese, J., Saudia, S., Ranjani, J.J., Velu, K., & Kannan, S.S. 2006. An efficient extension to chain codes for external image representation. *In: IFIP Int. Conf. on Wireless and Optical Communications*.

Nicolas, François, & Rivals, Eric. 2005. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *Journal of Discrete Algorithms*, **3**(2-4), 390–415. Combinatorial Pattern Matching (CPM) Special Issue. The 14th annual Symposium on combinatorial Pattern Matching.

Ong, Ju Lynn, & Seghouane, Abd-Krim. 2008. Mean shape models for polyp detection in CT Colonography. *Digital Image Computing: Techniques and Applications*, 287–293.

Ozkan, K., Topal, C., & Akinlar, C. 2011. Corner detection via trilateral filtering of chain codes. *Pages 168 – 172 of: Int. Symposium on Innovations in Intelligent Systems and Applications (INISTA)*.

Platonov, Juri, & Langer, Marion. 2007. Automatic contour model creation out of polygonal CAD models for markless Augment Reality. -, **-**, –.

Rico-Juan, J., & Micó, L. 2003. Comparison of AESA and LAESA search algorithms using string and tree-edit-distances. *Pattern Recognition Letters*, **24**(9-10), 1417–1426.

Rico-Juan, Juan Ramón, & Iñesta, José Manuel. 2012. New rank methods for reducing the size of the training set using the nearest neighbor rule. *Pattern Recognition Letters*, **33**(5), 654 – 660.

Ristad, E.S., & Yianilos, P.N. 1998. Learning string-edit distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **20**, 522 – 532.

Ritter, G. L., Woodruff, H. B., Lowry, S. R., & Isenhour, T. L. 1975. An algorithmfor a selective nearest neighbor decision rule. *IEEE Trans. on Information Theory*, **21(6)**, 665–669.

Rodríguez, Cristian Olivares, & Carratalá, José Oncina. 2008. A stochastic approach to median string computation. *Lecture Notes on Computer Science, SSPR&SPR 2008*, **5342**, 431–440.

Salleh, S.S., Aziz, N.A.A., Mohamad, D., & Omar, M. 2011. Combining Mahalanobis and Jaccard to improve shape similarity measurement in sketch recognition. *Pages 319 – 324 of: 13th Int. Conf. on Computer Modelling and Simulation (UKSim)*.

Sánchez, G., Lladós, J., & Tombre, K. 2002. A mean string algorithm to compute the average among a set of 2D shapes. *Pattern Recognition Letters*, **23**, 203–213.

Skalak, D. B. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Int. Conf. on Machine Learning*, 293–301.

Swonger, C. W. 1972. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. *Frontiers of Pattern Recognition*, 511–519.

Tak, Yoon-Sik, & Hwang, Eenjun. 2007 (oct.). A Leaf Image Retrieval Scheme Based on Partial Dynamic Time Warping and Two-Level Filtering. *Pages 633 –638 of: Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*.

Theodoridis, S., & Koutroumbas, K. 2006. *Pattern Recogniton*. Academic Press.

Tomek, I. 1976a. An experiment with the edit nearest-neighbor rule. *IEEE Trans. on Systems, Man and Cybernetics*, **6**, 448–452.

Tomek, I. 1976b. A generalization of the k-NN rule. *IEEE Trans. on Systems, Man and Cybernetics*, **6**, 121–126.

Ullman, J. R. 1974. Automatic selection of reference data for use in a nearest neighbor method of pattern classification. *IEEE Trans. on Information Theory*, **20**, 541–543.

van Otterloo, Petrus Johannes. 1988. *A Contour-Oriented Approach to Digital Shape Analysis.* Ph.D. thesis, Technische Universiteit Delft.

Vázquez, F., Sánchez, J., & Pla, F. 2005. A stochastic approach to Wilson's editing algorithm. *Pages 35–42 of: Lecture Notes on Computer Science*, vol. 3523.

Wagner, R., & Fischer, M. 1974. The string-to-string correction problem. *Journal of the ACM*, **21**, 168–173.

Wang, L., & Fu, X. 1998. *Data Mining with Computational Intelligence*.

Wang, Lusheng, & Jiang, Tao. 1994. On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology*, 337–348.

Wilfong, G. 1991. Nearest neighbor problems. *7th Annual ACM Symposium on Computational Geometry.*, 224–233.

Wilson, D. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on Systems, Man. and Cybernetics*, **2**, 408–421.

Wilson, D. Randall, & Martínez, Tony R. 2000. Reduction techniques for instance based learning algorithms. *Machine Learning*, **38**, 257–286.

Winkler, W. E. 1999. The state of record linkage and current research problems. *Available in: http://www.census.gov/srd/www/byname.html.*

Yu, Shiqi, Tan, Daoliang, Huang, Kaiqi, & Tan, Tieniu. 2007. Reducing the Effect of Noise on Human Contour in Gait Recognition. *Pages 338–346 of:* Lee, Seong-Whan, & Li, Stan (eds), *Advances in Biometrics*. Lecture Notes in Computer Science, vol. 4642. Springer Berlin Heidelberg.

Zhang, Chunyan, Tang, Jin, & Luo, Bin. 2006. Shape edit distance on contour based shapes. *Pages 310–315 of: 6th Int. Conf. on Intelligent Systems Design and Applications (ISDA)*, vol. 2.

Zhang, D., & Lu, G. 2004. Review of shape representation and description techniques. *Pattern Recognition*, **37**, 310–315.

Zhang, J. 1992. Selecting typical instances in instance-based learning. *9th Int. Conf. on Machine Learning*, 470–479.

Zhou, L., & Zahir, S. 2006. A new efficient context-based relative-directional chain coding. *IEEE Int. Symposium on Signal Processing and Information Technology.*, 787–790.

Apéndices

Tabla 7.1: Distancias *intra* e *inter* clase (NIST-19).

-	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	130.3	191.5	230.0	203.6	297.9	237.4	291.2	246.4	222.0	266.7	247.8	184.4	287.1	260.5	170.6	177.3	172.4	198.9	273.3	234.8	238.9	215.7	305.3	208.6	218.1	293.8
B	191.5	162.5	226.8	182.7	281.8	237.0	295.8	293.5	216.6	242.2	271.5	187.0	335.2	302.5	163.8	180.3	172.7	223.4	235.7	237.5	256.5	240.9	328.2	226.9	228.7	268.6
C	230.0	226.8	149.2	250.7	249.1	221.2	235.7	329.1	251.3	295.1	287.5	178.4	354.6	333.6	209.9	239.0	208.0	252.2	252.3	263.8	265.3	273.2	352.2	267.7	268.8	267.9
D	203.6	182.7	250.7	177.0	305.6	258.7	319.3	305.4	222.9	251.2	292.2	204.8	340.8	310.3	164.3	184.1	187.1	248.7	257.8	251.8	260.7	245.0	333.4	245.0	234.9	287.0
E	297.9	281.8	249.1	305.6	234.8	254.2	281.5	369.3	318.2	321.5	315.2	261.4	393.8	383.8	295.1	292.3	277.6	299.5	283.8	309.1	323.8	341.4	394.0	317.1	334.8	298.4
F	237.4	237.0	221.2	258.7	254.2	192.6	281.6	317.6	262.5	289.8	267.2	216.8	353.3	333.0	242.7	225.2	237.7	237.4	265.3	259.0	294.9	291.2	360.4	257.9	277.8	276.7
G	291.2	295.8	235.7	319.3	281.5	281.6	241.2	360.8	336.6	341.6	328.9	264.1	377.4	365.2	300.8	307.9	279.2	299.3	306.9	323.5	306.1	317.2	380.7	327.5	327.6	321.8
H	246.4	293.5	329.1	305.4	369.3	317.6	360.8	250.8	324.7	332.2	294.0	293.4	318.0	269.3	300.9	281.5	285.9	290.8	348.2	317.9	292.1	280.7	314.3	278.6	278.9	359.1
I	222.0	216.6	251.3	222.9	318.2	262.5	336.6	324.7	201.6	264.4	304.7	201.6	373.5	327.1	180.8	209.1	217.2	266.9	272.0	242.3	279.2	252.1	358.9	246.3	234.7	308.8
J	266.7	242.2	295.1	251.2	321.5	289.8	341.6	332.2	264.4	235.0	314.2	266.8	372.9	334.2	245.8	253.9	259.0	282.2	245.9	261.6	308.2	283.0	356.1	278.9	271.1	309.5
K	247.8	271.5	287.5	292.2	315.2	267.2	328.9	294.0	304.7	314.2	243.3	254.2	327.9	301.6	284.8	266.4	263.7	251.3	309.2	300.3	286.4	286.6	337.5	255.7	277.6	317.5
L	184.4	187.0	178.4	204.8	261.4	216.8	264.1	293.4	201.6	266.8	254.2	120.5	341.2	303.9	165.1	203.7	182.2	225.4	243.4	253.5	229.3	231.2	332.4	215.9	221.7	249.7
M	287.1	335.2	354.6	340.8	393.8	353.3	377.4	318.0	373.5	372.9	327.9	341.2	286.2	314.6	343.1	326.6	318.8	318.9	381.9	353.2	321.7	316.0	350.5	328.4	319.7	391.1
N	260.5	302.5	333.6	310.3	383.8	333.0	365.2	269.3	327.1	334.2	301.6	303.9	314.6	249.1	300.2	282.2	294.8	293.5	351.5	325.0	285.6	264.6	302.7	279.1	274.4	378.2
O	170.6	163.8	209.9	164.3	295.1	242.7	300.8	300.9	180.8	245.8	284.8	165.1	343.1	300.2	105.1	155.1	153.1	232.0	245.6	224.7	238.3	213.2	325.7	221.8	206.4	288.8
P	177.3	180.3	239.0	184.1	292.3	225.2	307.9	281.5	209.1	253.9	266.4	203.7	326.6	282.2	155.1	135.5	177.5	216.6	257.6	212.8	262.4	234.8	319.9	223.2	215.8	289.1
Q	172.4	172.7	208.0	187.1	277.6	237.7	279.2	285.9	217.2	259.0	263.7	182.2	318.8	294.8	153.1	177.5	144.4	211.3	251.4	231.5	249.1	239.4	324.2	221.6	223.5	273.1
R	198.9	223.4	252.2	248.7	299.5	237.4	299.3	290.8	266.9	282.2	251.3	225.4	318.9	293.5	232.0	216.6	211.3	198.8	275.0	262.1	278.2	261.6	332.2	237.3	262.6	295.4
S	273.3	235.7	252.3	257.8	283.8	265.3	306.9	348.2	272.0	245.9	309.2	243.4	381.9	351.5	245.6	257.6	251.4	275.0	186.2	261.8	307.7	295.7	368.3	273.6	280.8	267.0
T	234.8	237.5	263.8	251.8	309.1	259.0	323.5	317.9	242.3	261.6	300.3	253.5	353.2	325.0	224.7	212.8	231.5	262.1	261.8	196.7	299.0	266.8	355.2	255.7	240.7	300.2
U	238.9	256.5	265.3	260.7	323.8	294.9	306.1	292.1	279.2	308.2	286.4	229.3	321.7	285.6	238.3	262.4	249.1	278.2	307.7	299.0	226.5	230.1	302.7	267.7	251.4	329.5
V	215.7	240.9	273.2	245.0	341.4	291.2	317.2	280.7	252.1	283.0	286.6	231.2	316.0	264.6	213.2	234.8	239.4	261.6	295.7	266.8	230.1	190.0	290.8	247.7	213.1	336.9
W	305.3	328.2	352.2	333.4	394.0	360.4	380.7	314.3	358.9	356.1	337.5	332.4	350.5	302.7	325.7	319.9	324.2	332.2	368.3	355.2	302.7	290.8	292.8	321.7	308.6	394.5
X	208.6	226.9	267.7	245.0	317.1	257.9	327.5	278.6	246.3	278.9	255.7	215.9	328.4	279.1	221.8	223.2	221.6	237.3	273.6	255.7	267.7	247.7	321.7	195.8	224.9	298.2
Y	218.1	228.7	268.8	234.9	334.8	277.8	327.6	278.9	234.7	271.1	277.6	221.7	319.7	274.4	206.4	215.8	223.5	262.6	280.8	240.7	251.4	213.1	308.6	224.9	191.7	323.0
Z	293.8	268.6	267.9	287.0	298.4	276.7	321.8	359.1	308.8	309.5	317.5	249.7	391.1	378.2	288.8	289.1	273.1	295.4	267.0	300.2	329.5	336.9	394.5	298.2	323.0	228.0

Tabla 7.2: Distancias *intra* e *inter* clase (USPS).

-	0	1	2	3	4	5	6	7	8	9
0	67.1	91.4	187.6	217.1	208.7	263.9	148.9	164.7	134.3	135.6
1	91.4	59.0	209.1	242.8	208.6	291.6	157.8	160.5	147.6	135.8
2	187.6	209.1	152.3	204.3	234.9	239.4	208.3	194.7	192.9	196.4
3	217.1	242.8	204.3	147.1	255.3	237.7	253.4	208.4	214.1	207.6
4	208.7	208.6	234.9	255.3	184.7	284.9	211.8	216.5	213.7	200.3
5	263.9	291.6	239.4	237.7	284.9	187.9	253.5	257.5	240.3	248.4
6	148.9	157.8	208.3	253.4	211.8	253.5	106.8	200.0	169.8	182.3
7	164.7	160.5	194.7	208.4	216.5	257.5	200.0	126.0	178.3	140.4
8	134.3	147.6	192.9	214.1	213.7	240.3	169.8	178.3	132.9	147.2
9	135.6	135.8	196.4	207.6	200.3	248.4	182.3	140.4	147.2	108.5

Tabla 7.3: Distancias *intra* e *inter* clase (MPEG-7).

-	apple	bell	bone	brick	car	phone	children	cup	face	fountain
apple	582.3	882.0	1408.1	1052.0	801.3	1197.0	1005.5	1404.0	833.4	1139.3
bell	882.0	633.2	1563.5	998.6	658.7	1318.3	858.8	1566.0	820.5	1095.7
bone	1408.1	1563.5	655.1	1917.8	1672.4	1429.6	1706.5	1515.2	1276.1	1976.2
brick	1052.0	998.6	1917.8	197.4	863.1	1601.1	914.3	1710.1	1228.7	625.2
car	801.3	658.7	1672.4	863.1	195.0	1172.1	698.6	1603.4	883.3	975.7
phone	1197.0	1318.3	1429.6	1601.1	1172.1	440.6	1392.6	1431.0	1211.1	1686.8
children	1005.5	858.8	1706.5	914.3	698.6	1392.6	197.0	1758.4	993.2	905.0
cup	1404.0	1566.0	1515.2	1710.1	1603.4	1431.0	1758.4	946.4	1261.4	1797.8
face	833.4	820.5	1276.1	1228.7	883.3	1211.1	993.2	1261.4	311.7	1304.3
fountain	1139.3	1095.7	1976.2	625.2	975.7	1686.8	905.0	1797.8	1304.3	218.4