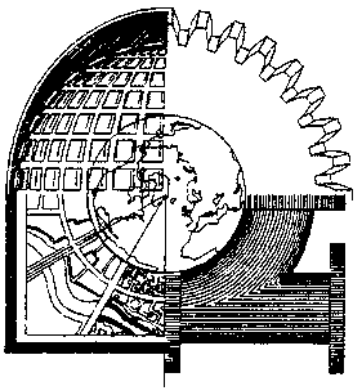
ACTAS
del
CONGRESO INTERNACIONAL DE INGENIERIA DE PROYECTOS

(ACTS OF INTERNATIONAL CONGRESS OF PROJECTS ENGINEERING)



Asturias, 5, 6 y 7 de Octubre de 1994

VOLUMEN II

Universidad de Oviedo

Asociación Española de Ingeniería de Proyectos

Diseño: M.J. VALLHONRAT

RECONOCIMIENTO DE CARACTERES OPTICOS BASADO EN CONOCIMIENTO

Agustín Cernuda del Río
Seresco Asturiana, S.A. (Oviedo)

Benjamín López Pérez, y Juan Manuel Cueva Lovelle
Area de Lenguajes y Sistemas Informáticos.Dto. de Matemáticas. Universidad de Oviedo
cueva@hp400.ccu.uniovi.es

RESUMEN

Este documento presenta un proyecto de Ingeniería Informática que constituye un caso de desarrollo que implica multitud de disciplinas. Es un proyecto piloto, que pretende explorar la viabilidad técnica de un proyecto mayor.

Se trata de un sistema de reconocimiento de caracteres ópticos basado en conocimiento. El reconocimiento de caracteres es un área tradicional de aplicación de disciplinas estadísticas y de modelos matemáticos apoyados en procesos de Visión Artificial y tratamiento de imágenes. Este es el caso de proyectos anteriores desarrollados por alumnos [3][4] y profesores [9] de la Universidad de Oviedo. Aunque estos sistemas ofrecieron buenos resultados, su comportamiento consiste en tomar un patrón determinado, extraer una serie de características (sobre todo cuantitativas) y a continuación efectuar un tratamiento global de las mismas, utilizando técnicas de análisis discriminante y similares. Un sistema realmente efectivo de reconocimiento de caracteres ópticos exige generalmente equipos de gran potencia, con periféricos de captación muy eficaces y automatizados y frecuentemente con componentes hardware adicionales.

Aquí se persigue explorar una vía alternativa. Se trata de llevar a la práctica una idea que consiste en abordar el problema del reconocimiento de caracteres en base a paradigmas arquitecturales procedentes de la Inteligencia Artificial. Se persigue encontrar una arquitectura modular y sencilla, que se comporte de manera "inteligente" y reconozca los patrones no en base a un procesamiento numérico ciego, sino en base a los propios rasgos que los humanos identificamos habitualmente en un carácter.

PALABRAS CLAVE. *Reconocimiento de caracteres ópticos, visión artificial, inteligencia artificial, reglas, gramáticas, bases de conocimiento, aprendizaje automático, librerías de enlace dinámico.*

1 OBJETIVOS

Como se dijo en la presentación anterior, se busca analizar una arquitectura y estudiar la viabilidad técnica de un reconocedor de caracteres basado en conocimiento, y no en mecanismos estadísticos. Las ventajas que podrían obtenerse con la implantación definitiva de tal sistema son:

- La posibilidad de optimizar determinados aspectos del conocimiento contenido en el sistema, de forma que para cada caracter no se extraigan todas las características sino sólo las necesarias para discernir de cuál se trata. Con algoritmos sencillos de detección de rasgos y con bases de conocimiento ponderadas y optimizadas, puede lograrse un aumento de la velocidad.

- No utilizar algoritmos muy complejos y globales (como en los reconocedores estadísticos), sino algoritmos sencillos y específicos utilizados inteligentemente por la base de conocimientos; trasladar la dificultad de la decisión a las etapas superiores del razonamiento, y no a las etapas básicas del tratamiento de imágenes.

- Al ser esos algoritmos sencillos, es posible que muchos de ellos se limiten a utilizar aritmética entera.

- El ruido introducido por los aparatos de captación desviaría mucho menos a un reconocedor basado en conocimiento, siempre que ese ruido no alterase gravemente las características topológicas de las letras.

- Además, el reconocimiento en base a rasgos topológicos y no a criterios cuantitativos haría posible aplicar el reconocedor a conjuntos muchísimo más amplios de tipos de letra, si son morfológicamente similares. (Hay que tener en cuenta que muchos reconocedores estadísticos pueden necesitar entrenamiento separado para las variantes en negrita y normal del mismo font).

- El comportamiento del reconocedor sería, en suma, más "humano"; podrían evitarse errores de los reconocedores estadísticos, y en todo caso muchos de los errores cometidos serían "disculpables".

Anteriormente se ha dicho que un sistema de reconocimiento eficaz suele necesitar instalaciones potentes; en el caso del desarrollo, suele ser necesario un equipo humano, técnico y material del que no puede disponerse en todos los ámbitos, además de una cierta experiencia acumulada en el sector, al tratarse de aplicaciones que involucran un alto componente de I+D. Sin embargo, se enfoca este proyecto de forma que pueda ser abordado por un equipo muy reducido (incluso una sola persona) con medios limitados; las conclusiones y los componentes desarrollados pueden utilizarse para un ataque más frontal al problema.

2 COMPONENTES FUNDAMENTALES DEL SISTEMA

Pasamos a identificar los módulos de mayor interés que se han detectado al describir la arquitectura del sistema.

2.1 Manejo del formato TIFF

Casi todas las actividades del reconocedor están guiadas por la información gráfica de entrada, de una u otra forma; el acceso a bajo nivel al formato TIFF [1] debe garantizarse de forma separada, ofreciendo a las funciones superiores unos servicios de obtención de información que permitan obviar la eventual complejidad de los campos de los ficheros y que permitan centrarse en cada algoritmo concreto a más alto nivel, con la cohesión funcional que esto proporciona.

2.2 Preproceso y transformaciones de imágenes

Estas funciones constituyen también un capítulo separado en los algoritmos a implantar; sin embargo, en un primer momento podemos movernos con casos ideales y en un campo restringido de muestras de entrada, y posponer estas funciones el tiempo que se considere oportuno (no se detentrá el resto del desarrollo). Aun así, hay que decir que el reconocimiento basado en morfología exige una cierta calidad de las imágenes, en el sentido de que se admite cualquier clase de ruido si no destruye características visuales fundamentales del carácter en cuestión.

2.3 Segmentación

Tanto la segmentación en renglones como la segmentación en caracteres constituyen etapas absolutamente fundamentales en el proceso de reconocimiento de caracteres. La segmentación en renglones puede verse influida por factores como la excesiva cercanía de estos entre sí, o sobre todo por la inclinación de las páginas al efectuar la captación de datos [9]. Se ha desarrollado un algoritmo propio de suavizados sucesivos del histograma horizontal, que da buenos resultados en general y que detecta y desprecia con gran sencillez las zonas extensas en blanco.

La segmentación en caracteres es mucho más delicada; la tipografía aconseja en la mayoría de las ocasiones acercar entre sí determinados pares de caracteres, e incluso soldarlos (lo que ocurre a menudo con la **t** y la **i**, por ejemplo). En una gran parte de los fonts, los caracteres invaden la frontera vertical que les separa de sus vecinos. El algoritmo desarrollado efectúa un barrido horizontal del renglón en su parte central, y extrae el caracter por "contagio" de sus pixels.

2.4 Interfaz con usuario

El interfaz con el usuario es una parte que a priori podría considerarse poco importante de cara al desarrollo pero que en realidad puede influir más de lo previsto en la estructura de los programas. Al utilizar el **API** (*Applications Progammer Interface*) de *Windows*, el sistema de menús que rige la aplicación se convierte en un bucle de manejo de eventos, ante los que se responde lanzando distintos procesos.

2.5 Extracción de características

Este es uno de los aspectos en los que queda más por hacer en el sentido de que las características que se elijan, los algoritmos a seguir para ello, su respuesta o tolerancia frente al ruido y el compromiso ya mencionado entre exactitud y eficiencia, constituyen un campo abierto para la experimentación. Las decisiones a tomar dependen en gran medida de criterios empíricos, y no hay que olvidar que esta es la tónica habitual en la disciplina de la visión artifical y el proceso de imágenes.

En este proyecto se han contemplado elementos como la detección de diversos tipos de trazos horizontales y verticales [7], de regiones encerradas, de regiones aisladas (puntos), de muescas de diversas clases, y de otros elementos morfológicos. Este conocimiento algorítmico se encuentra aislado en módulos DLL (librerías de enlace dinámico, con enlace en tiempo de ejecución); esto permite que el paradigma de ampliación de la base de conocimientos [6] sea efectivo también en sentido algorítmico.

2.6 Clasificación de patrones

Al igual que el capítulo anterior, este módulo se presenta a la vez como crítico pero con una definición bastante vaga y libre. Sí cabe suponer que la extracción de características va a estar guiada por las necesidades que plantee el clasificador, es decir, no se extraen todas las características posibles y a continuación se asigna el patrón, sino que se extraen las características que el clasificador estime necesarias [8]. Incluso se pretende contemplar alguna clase de minimización gramatical o de reglas, de forma que para cada patrón se extraiga el mínimo número de características necesario, primando a las características de mayor poder de discriminación [2][10].

Este apartado exigiría una discusión muy detallada, puesto que conlleva un alto fundamento teórico y un gran componente de investigación. El reconocimiento está dirigido por una gramática de reglas [5]; además, en el transcurso del proyecto piloto se han evaluado y propuesto -aunque no desarrollado- alternativas que serían de enorme importancia en el proyecto final, como la aplicación de diversos métodos de ponderación y optimización del conocimiento (árboles Y/O equilibrados bajo criterios de profundidad, árboles Y/O con aplicación ordenada de decisiones, árboles Y/O con hechos ponderados, aplicación de algoritmos de búsqueda, mejora con ponderaciones particularizadas a los ejemplos...)

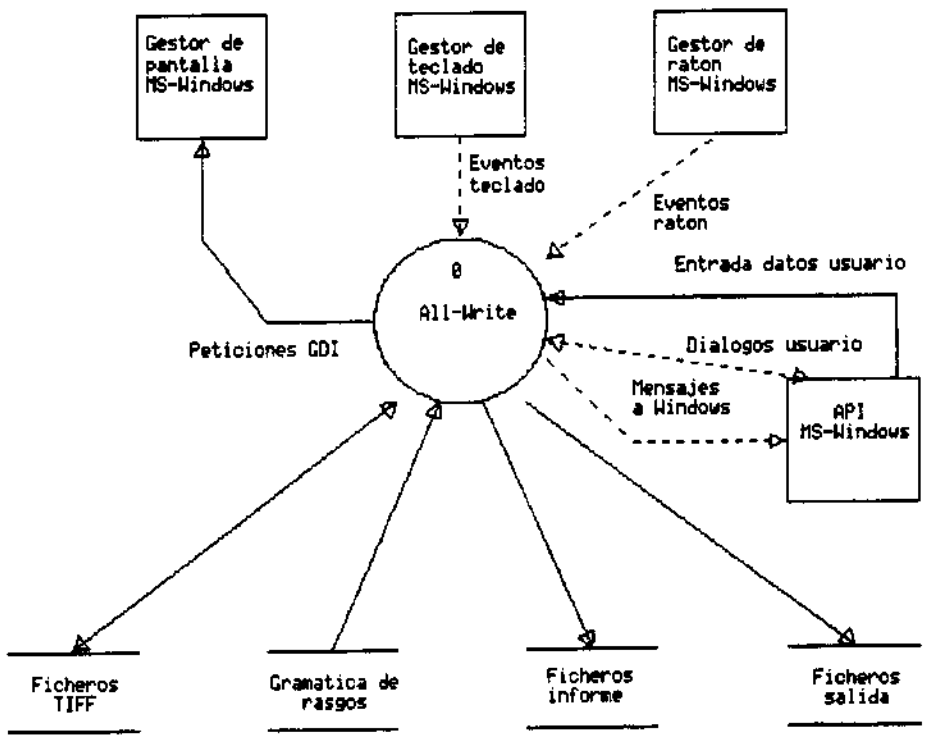
3 PLAN DE DESARROLLO

En esta aplicación existen, como ya se ha visto, ciertas particularidades dadas por el estado de la Visión Artificial y por las características del reconocimiento de patrones y de la Inteligencia Artificial.

Esto nos sitúa en una posición difícil a la hora de elaborar un plan de desarrollo riguroso en cuanto a prestaciones, costes o períodos de tiempo. Uno de los objetivos perseguidos es conocer la accesibilidad o no de determinadas prestaciones, evaluar el coste computacional que suponen, su viabilidad, etc.; en este y otros sentidos, el plan de desarrollo se ciñe a actividades y propósitos más que a las ponderaciones que la gestión de proyectos suele abordar.

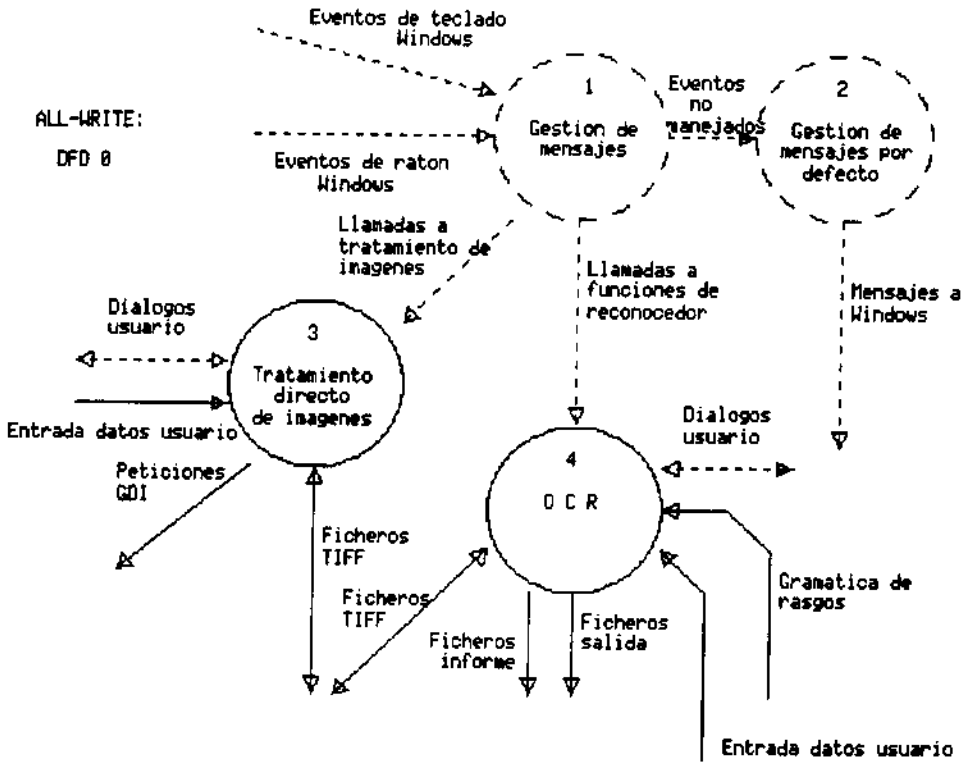
En cuanto a los documentos de análisis y diseño, se incluyen sólo a un nivel muy general de arquitectura, puesto que su detalle ampliaría enormemente el presente documento.

3.1 Diagrama de contexto

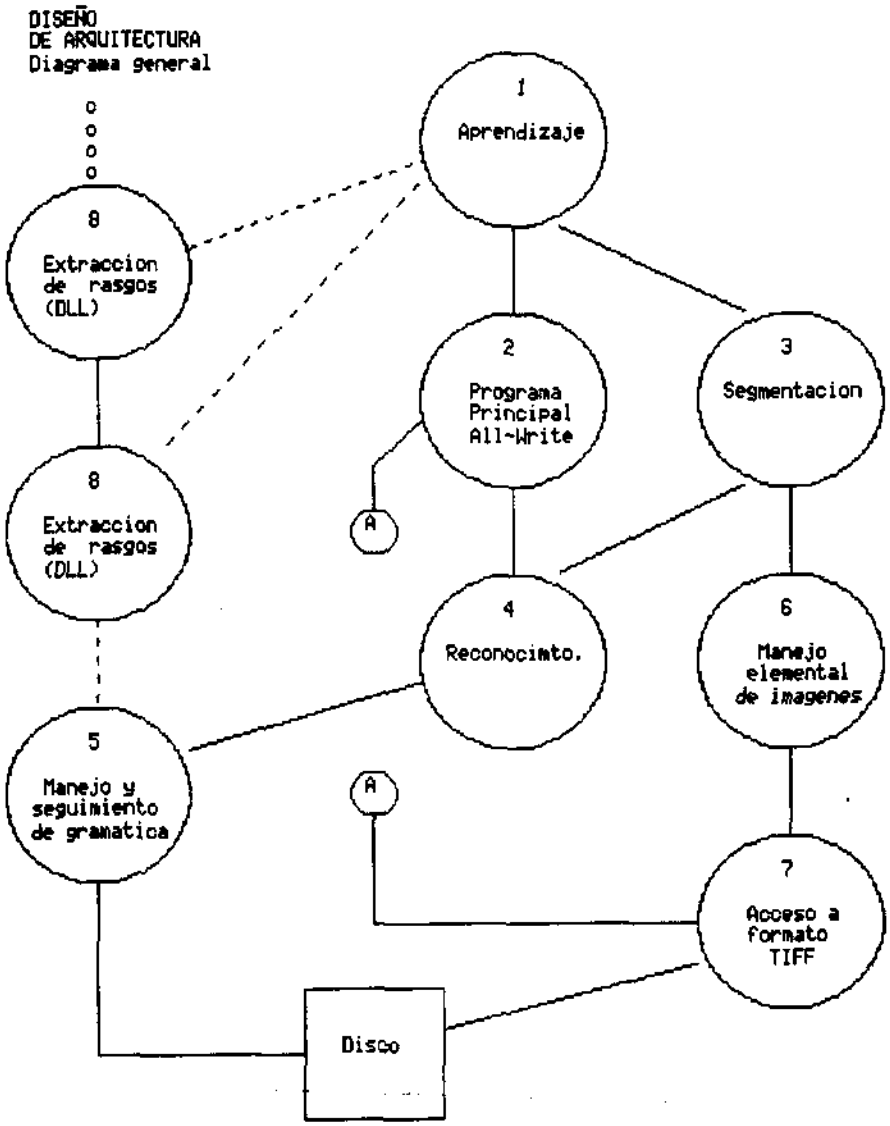


ALL-WRITE: Diagrama de contexto

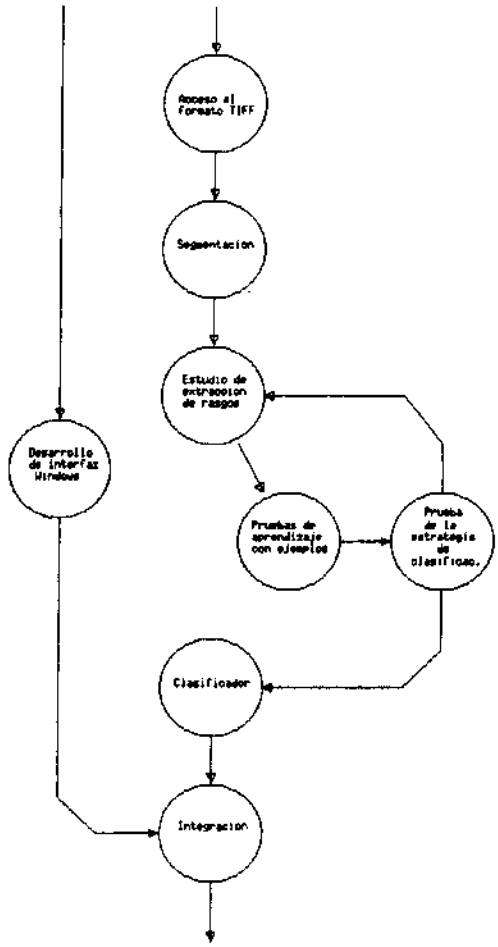
3.2 Primera descomposición



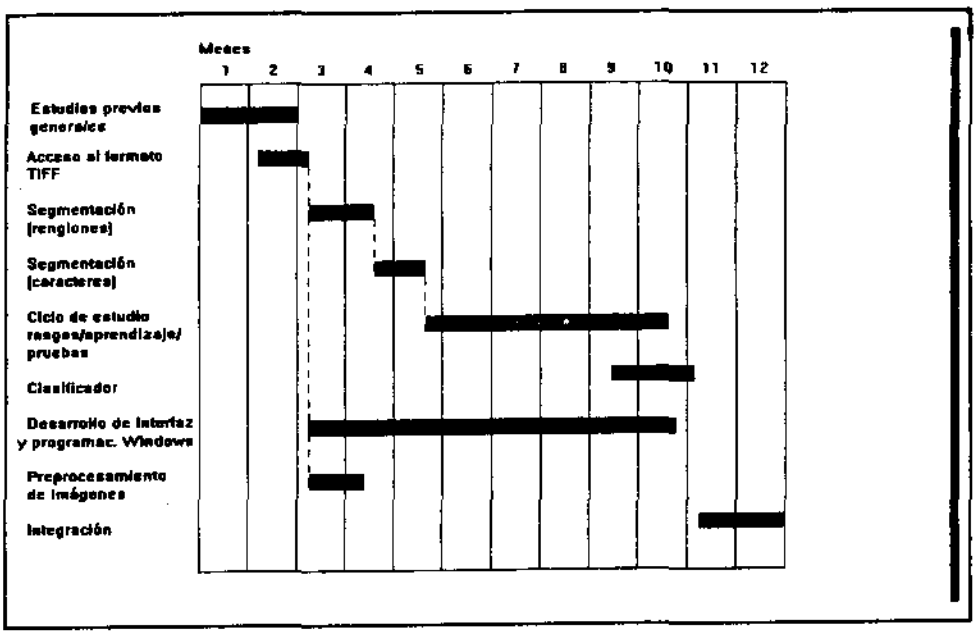
3.3 Diseño de arquitectura



3.4 Precedencias



Sobre este gráfico de precedencias, se muestra un plan de tiempos aproximado, de un año de duración:



4 CONCLUSIONES: CASO PRACTICO

Tras conocer cuáles son las ideas fundamentales que promovieron la realización del proyecto, cuál es el camino seguido para plasmar estas ideas en un sistema real y en qué medida el desarrollo del proyecto aportó nuevas ideas a los dominios en los que se inscribe, podemos conocer un caso práctico de ejecución, con objeto de comprobar cuál es el comportamiento de la aplicación y hasta qué punto responde a la idea inicial.

El fichero a clasificar es una tabla de números, dispuestos en columnas. La razón de elegir una imagen compuesta exclusivamente por números responde a razones de brevedad; un conjunto de 10 patrones proporcionará una regla más legible que una regla para un alfabeto complejo.

Además, con el conjunto de rasgos que se ha desarrollado, suele aparecer algún sinónimo si se trata un conjunto de patrones muy grande, y estos sinónimos son los que se resuelven con nuevos rasgos elegidos específicamente para la ocasión (por ejemplo, cierta clase de y y la v son

topológicamente iguales, y su distinción exigiría un algoritmo sencillo de vectorización [7] para detectar una prolongación vertical en la parte inferior). Para evitar extenderse demasiado en trabajo poco significativo y poder centrarse en el proceso global, que es lo que interesa, se ha acudido a un universo de ejemplos más intuitivo y manejable.

La aplicación real de un reconocedor en estos casos es evidente; por ejemplo, para la transcripción de distribuciones estadísticas, o de cualquier otra información numérica tabulada, en la que existen cientos o miles de dígitos. Conozcamos ahora los pasos que se siguen para construir la base de conocimiento.

4.1 Primer paso del aprendizaje: recopilación de informes

Como ya se ha dicho, esta aplicación no realiza un aprendizaje automático, sino manual. Es el desarrollador el que construye la base de conocimiento. Una inversión mayor de tiempo y recursos permitiría la implantación de un sistema de aprendizaje automático [2][10].

El primer paso es testear las funciones de detección de rasgos sobre una muestra significativa el alfabeto a reconocer. Para ello, se utiliza cierta opción del menú que genera un fichero de informes de aprendizaje.

4.2 Segundo paso: creación de un marco en 1st-CLASS

Con objeto de poder utilizar la información del aprendizaje en un programa externo -el programa 1STCLASS, que permite construir sistemas basados en conocimiento a partir de ejemplos-, se creó en este programa el marco adecuado, introduciendo los atributos equivalentes a los rasgos que se detectan y el abanico de valores posibles.

4.3 Tercer paso: introducción de ejemplos

Una vez que el aprendizaje abarcó una muestra de cada uno de los diez caracteres que se pretendía clasificar, se tomó el fichero de salida y se introdujeron sus datos en el programa **1st-CLASS**.

4.4 Cuarto paso: obtención de una regla de clasificación

Así, el programa dispone de una serie de ejemplos, que examinará para obtener una regla que los clasifique. Ya se ha hablado acerca de criterios de optimización del conocimiento (véase 2.6), pero el programa dispone de medios limitados en ese aspecto. Solicitando la construcción de una regla optimizada, se obtiene lo siguiente:

En primer lugar, la regla no almacena el conocimiento de la forma más parecida posible al conocimiento humano. A pesar de nuestra política de eliminar atributos poco significativos, siguen existiendo vías no muy adecuadas para la clasificación, que no hemos tenido en cuenta -lógicamente- y que no hemos "podado"; y aun examinando a fondo los ejemplos para eliminar todos los rasgos superfluos, la función de detección de rasgos siempre deja un margen para el error.

En segundo lugar, la regla dista mucho de ser óptima en cuanto a tiempo de ejecución. El primer hecho que se interroga siempre es la existencia de regiones encerradas, que es con mucha diferencia el algoritmo más costoso. Evidentemente, la posibilidad de ponderar los atributos o hechos, como se indicó en su momento, daría lugar a un reconocedor muchísimo más rápido en este caso. (El reconocedor con la gramática alfabética es significativamente más rápido, por las características de la regla generada).

4.5 Quinto paso: redacción de la gramática de rasgos

Dado el árbol multicamino de decisión que constituye la regla, queda elaborar la gramática de rasgos correspondiente. Tras transformar la regla en un árbol binario, se redactó la base de conocimientos.

4.6 Resultado de la clasificación

Tras clasificar una imagen completa del mismo tipo que la muestra de diseño, se obtuvo el siguiente resultado:

15217 34783 67391 79710 71014 71014	15217 34783 67391 79710 71014 71014
13420 55844 68831 58442 48052 42857	13420 55844 68831 58442 48052 42857
52841 91667 90322 48940 58016 33333	52841 91667 90322 48940 58016 3*33*
19960 38933 69763 86364 86364 79249	19960 389*3 6976* 86*64 86364 79249
22511 59740 72727 70996 59740 54545	22511 59740 72727 70996 59740 54545
60277 95652 86557 49152 56215 10000	60277 95652 86557 49152 56215 10000
19565 39130 65217 71014 76812 76087	19565 39130 65217 71014 76812 76087
39827 59740 68831 56710 47619 50216	*9827 59740 68831 56710 47619 50216
55303 91667 14041 48530 56429 10000	55303 91667 14041 48530 56429 10000
35498 60606 77489 79221 65368 41991	35498 60606 77489 79221 65*68 41991
11667 46429 58571 49048 45000 43571	11667 46429 58571 49048 45000 43571
50216 95455 03879 47394 57020 50000	50216 95455 03879 47*94 57020 50000
31159 66667 89130 94203 99275 00725	31159 66667 89130 94203 99275 00725
22464 52174 65217 72464 77536 82609	22464 52174 65217 72464 77536 82609
73958 00000 59390 48474 56289 55556	73958 00000 59*90 48474 56289 55556

Muestra de validación (izquierda, clases verdaderas) y resultado de la clasificación con la gramática NUMEROS.GRM. El asterisco corresponde a patrones no clasificados.

Los asteriscos corresponden a caracteres que el reconocedor es incapaz de asignar con el conocimiento que posee (no encajan en ninguna regla de clasificación). Sobre un total de 450 caracteres aparecen 9 errores.

En este caso, la muestra completa consta de 450 patrones, de los que sólo 10 se utilizaron en el aprendizaje. Podemos aplicar la medición más inmediata y habitual, que es el porcentaje de errores respecto a la muestra completa, lo que daría un porcentaje de efectividad de un **98% de aciertos**. Hay varias apreciaciones que hacer.

4.7 Conclusiones

En primer lugar, ese 98% de efectividad -que sería aceptable en la mayoría de los reconocedores numéricos comerciales- corresponde al primer paso, rigurosamente directo y tal como se ha descrito en este capítulo, en el desarrollo de esta base de conocimiento. Con la corrección adecuada -que como veremos sería sencilla- puede llegarse fácilmente al 100% en esta muestra, ya que la imagen tiene la calidad adecuada para no destruír características topológicas de los caracteres.

En segundo lugar, todos los errores corresponden a caracteres que no han podido clasificarse, pero **no hay ningún caracter mal asignado**. Es decir, un reconocedor numérico podría haber arrojado errores de clasificación, pero en este caso el reconocedor basado en conocimiento se ha mostrado más seguro y cauto a la hora de clasificar.

En tercer lugar, **todos los errores de esta muestra corresponden a una misma clase**: el número 3. El conocimiento del clasificador se ha generado de forma automática y sin ningún arreglo posterior. Es posible que el caracter que se eligió para construir la regla fuese poco significativo.

En cuarto lugar, el clasificador resultante de esa regla es relativamente lento. Pero esto no significa en absoluto que el esquema basado en conocimiento dé lugar a reconocedores lentos. En su momento, al examinar visualmente la regla, se predijo que 1st-CLASS no había proporcionado precisamente una regla rápida. De hecho, si se pudiese optimizar el conocimiento, un reconocedor de este tipo emplearía el tiempo estrictamente necesario para clasificar cada carácter.

Por último, merece la pena mostrar un curioso resultado. En el desarrollo de este proyecto se probaron otros casos, lógicamente, y entre ellos se extrajo una gramática para clasificar caracteres alfabéticos de dos tipos distintos. Los resultados fueron también positivos, aunque aparecieron algunos sinónimos perfectamente lógicos. Pero entre otras pruebas, se aplicó la gramática de caracteres a la muestra de números, y el resultado fue:

tsztz adz*a bzagt zg ...

donde los caracteres reales eran:

15217 34783 67391 79 ...

Obsérvese el resultado; el clasificador asignó caracteres -por supuesto, erróneamente- de acuerdo con las siguientes correspondencias:

1 ----> t

2 ----> z

5 ----> s

6 ----> b

7 ----> z

9 ----> g

(g con una sola región encerrada en la parte superior, es decir, la g similar al 9).

Las confusiones del reconocedor son extraordinariamente significativas. La similitud entre las formas asignadas y las reales es demasiado sospechosa. Aunque esto ocurre a veces también en reconocedores estadísticos, el fenómeno no es ni mucho menos tan acusado como en este, en el que la base de conocimientos le ha hecho adoptar un comportamiento prácticamente "inteligente". Este resultado, de cara a los objetivos del proyecto, resulta casi tan importante como el resultado de la clasificación real.

5 PRESUPUESTO APROXIMADO

Se ofrece un extracto del presupuesto aproximado:

Gastos de hardware	542.335
Gastos de software	133.820
Gastos de personal	3.000.000
Gastos generales (20%)	735.231
Total	4.411.386

6 REFERENCIAS BIBLIOGRAFICAS

[1] Aldus Corporation (1988) Tag Image File Format. Specification. Revision 5.0 Final

[2] Bahamonde, A. (1992) An Algorithm to Build Optimal Inheritance Graphs, Centro de Inteligencia Artificial, Universidad de Oviedo (Gijón)

[3] Canal, C. (1990) Reconocimiento de caracteres impresos leidos mediante scanner, Cuaderno Didéctico nº 34 del Departamento de Matemáticas (Universidad de Oviedo)

[4] Cernuda, A. (1991) Reconocimiento de caracteres ópticos. Generación de PostScript (Proyecto fin de carrera de la E. U. de Informática de Oviedo, Universidad de Oviedo), 134 págs.

[5] Cueva, J. M. (1992) Conceptos básicos de traductores, compiladores e intérpretes, Cuaderno Didáctico nº 9 del Departamento de Matemáticas (Universidad de Oviedo), 128 págs.

[6] Dussauchoy, A. y Chatain, J.N. (1988) Sistemas expertos. Métodos y herramientas, Paraninfo (Madrid), 238 págs.

[7] Fernández, A. (1991) Vectorización de imágenes reconocidas mediante scanner, Proyecto Fin de Carrera de la Escuela Universitaria de Informática de Oviedo, Universidad de Oviedo.

[8] Lee, K. H. (1988) Character Recognition Using Attributed Grammar, IEEE

[9] López, B. (1991) Reconocedores de caracteres ópticos, Proyecto Fin de Carrera de la Universidad de Málaga (Málaga)

[10] Quinlan, J.R. (1993) C4.5. Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo (California)