UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Reconocimiento De Escritura No Basado en Líneas

Proyecto Final de Carrera

Ingeniería informática

Autor: Andreu Escribà del Arco

Director: Carlos David Martínez Hinarejos

26 de septiembre de 2013

Agradecimientos

Quiero agradecer a todas las personas que directa o indirectamente me han ayudado con este proyecto.

A mi padres por aguantarme tantos años, a mi hermano por escuchar todos los problemas que me daba el proyecto.

También quiero dar las gracias a mis compañeros y amigos que han recorrido conmigo este camino.

Sobre todo también doy las gracias a mi director, Carlos David Martínez Hinarejos, por toda la ayuda que me ha dado.

Por ultimo quiero agradecer a todos los profesores que he tenido durante la carrera por ayudarme a ser como soy ahora.

Resumen

Este trabajo tiene como objetivo comprobar cómo la detección y fusión de palabras partidas mejoran la tarea de reconocimiento de escritura manuscrita $off - line$. Para ello se probarán distintos clasificadores con el objetivo de detectar aquellas líneas acabadas en palabras partidas para fusionarlas después con el resto de la palabra. Y por último comprobará el beneficio que esto representa en la tarea de reconocimiento.

Palabras clave: iatros,palabras partidas,clasificador, reconocimento de escritura manuscrita

Índice general

1. Introducción **3**
1.1. Generalidades sobre las tareas de clasificación 3
1.2. Reconocimiento de escritura manuscrita 5
1.3. Objetivos 5

2. Detección de palabras partidas **6**
2.1. Descripción del corpus 6
2.2. Preparación de las imágenes 6
2.3. Extracción de características 7
2.4. Modelos ocultos de Markov 8
2.5. Vecinos más cercanos 9
2.6. Perceptrón multicapa 9
2.7. Máquinas de soporte vectorial 10
2.8. Comparación de resultados 11

3. Reconocimiento de la escritura **13**
3.1. Iatros 13
3.2. Fusionar líneas 13
3.3. Experimentos 14

4. Conclusiones y líneas futuras **16**
4.1. Conclusiones 16
4.2. Líneas futuras 17

Capítulo 1

Introducción

El reconocimiento óptico de caracteres es un problema que se lleva tratando en la informática desde hace años. Consiste en identificar de forma automática a partir de una imagen un conjunto de símbolos pertenecientes a un alfabeto, para luego poder trabajar con ellos.

Debido a la gran cantidad de información escrita que se genera cada día se puede ver la utilidad de automatizar su digitalización para así evitar que se tenga que hacer a mano, permitiendo así un importante ahorro de recursos y mejorando la calidad de muchos servicios.

Toda técnica de OCR se divide en unos pasos básicos. El primer paso consiste en binarizar la imagen puesto que la mayoría de algoritmos de reconocimiento parte de una imagen binaria. Después procederemos a segmentar la imagen. Después se procede a adelgazar cada componente de manera que no se distorsione la forma. En penúltimo lugar se normaliza el tamaño y por último se aplica el método de reconocimiento elegido.

El reconocimiento de caracteres puede ser aplicado en diversas áreas con mejor y peor resultado. Por ejemplo el reconocimiento de caracteres impresos se considera ya un problema resuelto. En cambio el reconocimiento de escritura manuscrita es un problema donde todavía queda mucho por mejorar [1].

1.1. Generalidades sobre las tareas de clasificación

Uno de los problemas que se nos puede presentar en el reconocimiento de escritura manuscrita es la aparición de palabras partidas al final de cada línea, como vemos en la Figura 1.1, pues esto complica el reconocimiento semántico del texto debido a que al procesar una línea puede ocurrir que empiece con

un trozo de palabra o con una palabra completa, cosa que dificulta la tarea de reconocimiento aumentado la complejidad del reconocimiento semántico.

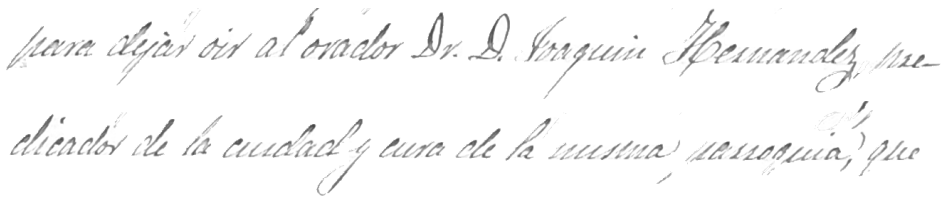

Figura 1.1: Ejemplos de líneas

Toda tarea de clasificación puede dividirse en tres pasos: adquisición de los datos, extracción de características, selección de variables y clasificación[2].

El primer paso consiste en extraer los datos del medio; para ello se utilizará el método más conveniente para cada tarea en particular.

En la etapa de extracción de características aplicaremos primero un preproceso de los datos para corregir posibles errores en su obtención. Luego se aplica el método de extracción de características elegido, como por ejemplo, para formas geométricas: los códigos de contorno, aproximaciones poligonales, signaras... El método de extracción de características depende mucho del tipo de datos del que disponemos, pero siempre ha de cumplir las siguientes condiciones: ser insensibles al ruido de captura, ser lo suficientemente discriminante para la tarea y poseer algún tipo de invariancia relacionada con la tarea (traslación, rotación, o transformaciones no lineales).

Los clasificadores pueden dividirse en dos grandes grupos: clasificadores supervisados y clasificadores no supervisados.

Los clasificadores supervisados cosiste en, mediante un entrenamiento con ejemplos, clasificar nuestros elementos en una de las clases disponibles. Entre los clasificadores supervisados podemos encontrar: clasificadores estadísticos, como los bayesianos; clasificadores basados en funciones discriminares lineales, como el algoritmo de perceptrón; clasificadores basado en distancias, como el de vecinos más cercanos; o clasificadores sintáctico-estadísticos, como los modelos ocultos de Markov.

Los clasificadores no supervisados consisten en agrupar los elementos por sus características similares, como por ejemplo el algoritmo de C-medias. Estos clasificadores no requieren de un entrenamiento, pues no se conoce a priori las clases disponibles.

1.2. Reconocimiento de escritura manuscrita

En la escritura manuscrita no solo se busca reconocer cada letra, sino que se intenta obtener su significado semántico [3]. El proceso es bastante complejo y no siempre se compone de los mismos pasos. Una de las muchas soluciones aplicadas es la que se compone de los siguientes pasos [3]: preproceso, extracción de características y reconocimiento e interpretación integrada a través de modelos de estados finitos y transductores estocásticos de estados finitos.

El primer paso consiste en dividir el fichero en líneas de texto y limpiar la imagen de ruido. Una vez extraídas las líneas y eliminado el ruido se procede corregir el *skew* (inclinación de las palabras) y el *slant* (inclinación de las letras); con esto conseguimos que las palabras estén alineadas horizontalmente y las letras verticalmente.

Después procederemos a eliminar la mayor parte de la variabilidad vertical de la imagen mediante un escalado. A continuación crearemos vectores de características mediante el normalizado del nivel de gris y las derivadas horizontales y verticales.

Una vez tengamos los vectores de características utilizaremos un clasificador basado en modelos ocultos de Markov junto con el modelo del lenguaje modelado mediante N-gramas, con los que obtendremos la transcripción más probable del mensaje. Por tanto, para poder simplificar el modelo del lenguaje intentaremos eliminar las palabras partidas.

1.3. Objetivos

- Detección de palabras partidas. Determinar un método de extracción de características. Estudio de diversos clasificadores y ajuste de parámetros.
- Técnicas de unión de las líneas partidas. Desarrollo de una técnica para unir dos líneas con palabras partidas.
- Efecto en la decodificación. Comprobación mediante tres experimentos de la mejora producida al juntar las líneas.

Capítulo 2

Detección de palabras partidas

En este capítulo vamos a tratar el tema de la detección de líneas acabadas en palabras partidas. Para ello primero estableceremos qué es una palabra partida. Tras ello se establecerá el proceso de la imagen: recortado del final de cada imagen y extracción de características. Por último, desarrollaremos una metodología de experimentación para cada clasificador.

2.1. Descripción del corpus

Para este trabajo se ha utilizado el corpus conocido como "Cristo-Salvador" [4]. El manuscrito, titulado ""Noticia histórica de las fiestas con que Valencia celebró el siglo sexto de la venida a esta capital de la milagrosa imagen del Salvador" por Vicente Boix, es un pequeño documento compuesto por 53 imágenes en color de las páginas de texto, escaneado a 300 dpi y escrito completamente por un solo escritor.

Para este trabajo se ha utilizado de la página 4 a la 32 como entrenamiento y de la pagina 33 a la 53 como test. Posteriormente se han extraído las líneas del texto y el conjunto de entrenamiento ha sido dividido en dos, dejando 400 líneas para entrenamiento y 275 como conjunto de validación. El conjunto de test se compone de 497 líneas.

2.2. Preparación de las imágenes

Como no es necesaria la imagen con la línea completa para detectar si acaba en una palabra partida, se ha ideado un programa que recorta el final de esa imagen. Para ello primero se binariza la imagen y después se recorta el trozo final que contenga píxeles negros, dando lugar a una imagen de tamaño fijo con la que después trabajaremos. Como se puede ver en las imágenes de

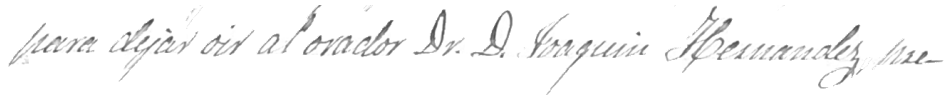

Figura 2.1: Palabra partida

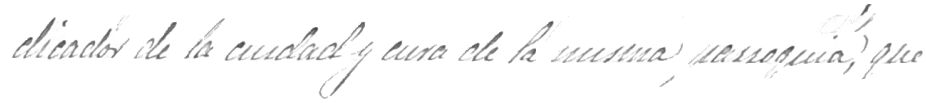

Figura 2.2: Palabra no partida

las Figuras 2.1 y 2.2 contamos ya con suficiente información para empezar a trabajar, puesto que se ve claramente el guión al final de la palabra, que es lo que nos indica si ha sido partida o no.

2.3. Extracción de características

En lo relacionado a la extracción de características vamos a utilizar dos técnicas distintas y diferenciadas. Para el clasificador que utiliza modelos ocultos de Markov calcularemos los códigos de contorno [5] de las imágenes y para el resto utilizaremos PCA (Principal Component Analysis).

Para el cálculo de los códigos de contorno hemos desarrollado el siguiente procedimiento: partiendo de la imagen recortada anteriormente procedemos a eliminar todos los pixeles que tengan 8 pixeles adyacentes, de manera que nos quede solo el contorno de la imagen; después se busca la esquina inferior izquierda de la imagen y se recorre su contorno en sentido horario; durante este recorrido vamos guardando la dirección del siguiente pixel que recorremos de manera que al final tendremos una cadena que describe el contorno de la imagen.

Para el resto de experimentos aplicaremos la técnica del PCA(Principal Component Analysis) que nos permite reducir la dimensionalidad de nuestros datos quedándonos solo con los N más representativos. Para ello utilizaremos un programa implementado en Octave[6].

Con estos datos construiremos tres conjuntos distintos: el conjunto de

entrenamiento con 400 líneas, el conjunto de validación con 277 líneas y el conjunto de test con 490 líneas.

2.4. Modelos ocultos de Markov

Los modelos ocultos de Markov (HMM) son modelos probabilistas empleados en el reconocimiento de texto manuscrito, lenguaje hablado, etiquetado gramatical ... En nuestro caso los vamos a utilizar junto con los códigos de contorno calculados anteriormente. Una vez tengamos los códigos de contorno de todas la imágenes dividiremos el conjunto de entrenamiento en dos y usaremos los 400 primeros elementos para entrenamiento y el resto para validación. En este caso iremos variando el número de estados del modelo desde 1 hasta 80.

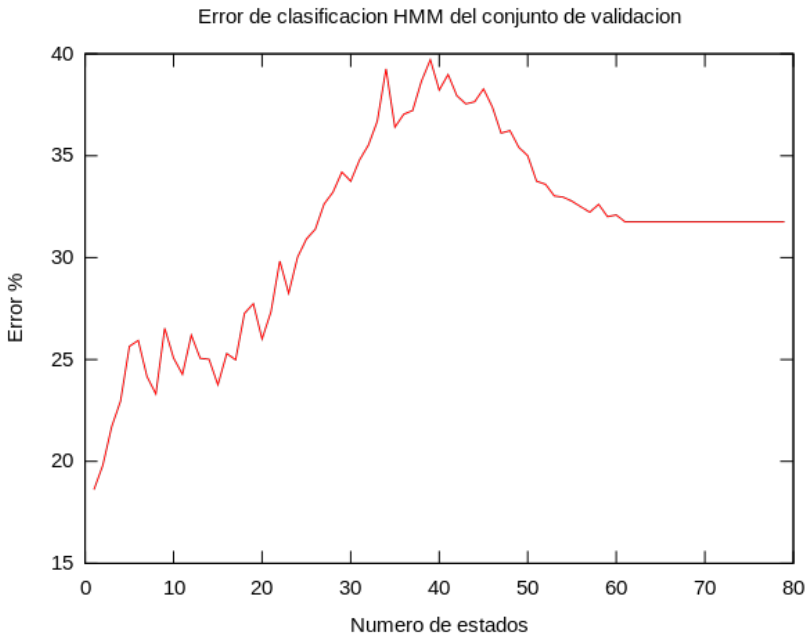

Figura 2.3: Error de clasificación en HMM del conjunto de validación

Como podemos observar en la Figura 2.3 el error menor lo obtenemos con un modelo de un solo estado. Este error es del 18.61 % en el conjunto de validación y de un 18.91 % en el conjunto de test.

2.5. Vecinos más cercanos

Este clasificador es uno de los clasificadores basados en distancias más sencillos. Consiste en calcular la distancia de cada elemento con los elementos del conjunto de entrenamiento y quedarse con la clase más común en los k elementos más cercanos a éste. En nuestro caso utilizaremos distancia euclidiana. En este experimento variaremos el número de vecinos así como el número de componentes de los datos.

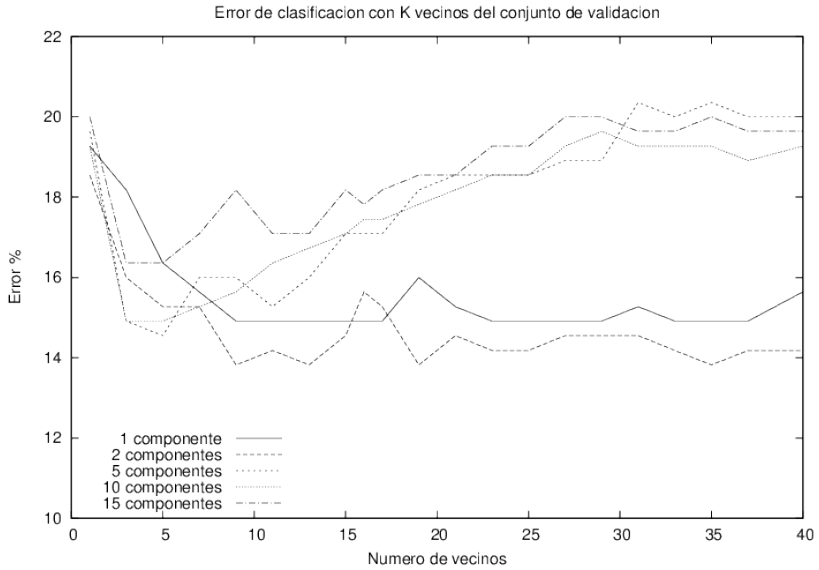

Figura 2.4: Error de clasificación en k-vecinos del conjunto de validación

En este caso observamos en la gráfica de la Figura 2.4 que con vectores de características de dos componentes y k=9 obtenemos en menor error: 13.82 % en el conjunto de validación y 14.49 % en el conjunto de test.

2.6. Perceptrón multicapa

El perceptrón multicapa es un tipo de red neuronal que nos permite tratar resolver problemas que no son linealmente separables, principal limitación del algoritmo de perceptrón. En nuestro caso utilizaremos Stuttgart Neural Network Simulator [7] para simular nuestra red y lo entrenaremos con el algoritmo de *backpropagation* y un factor de aprendizaje de 0.1. Utilizaremos una red con una capa de entrada, una capa oculta y una capa de salida. Variaremos el número de neuronas de la capa oculta y las dimensiones de los datos.

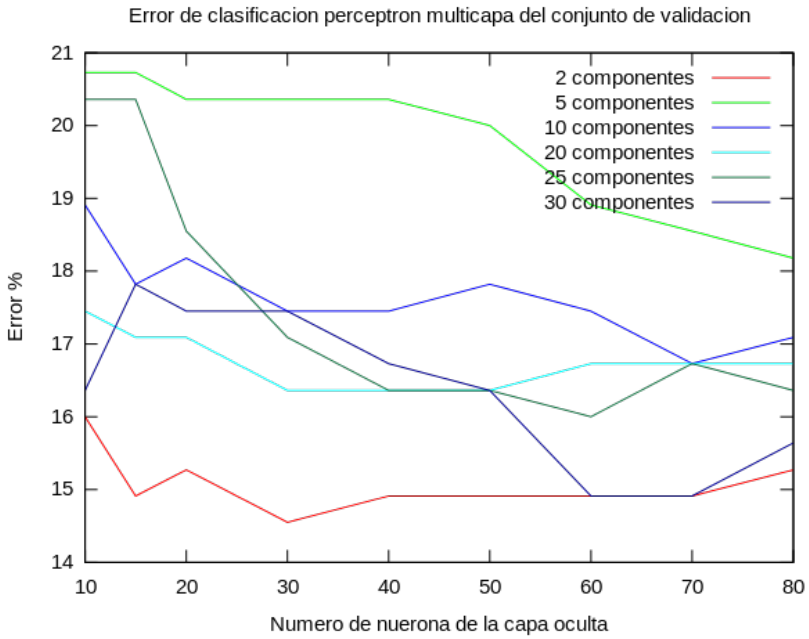

Figura 2.5: Error de clasificación perceptrón multicapa del conjunto de validación

Podemos observar en la gráfica de la Figura 2.5 que el error, salvo casos extremos, se mantiene entre el 16 % y el 18 %, excepto en el caso de utilizar 2 componentes y una capa oculta de 30 elementos. Con ello obtenemos el mínimo error que en este caso es de 14.55 % en validación y 14.89 % en test.

2.7. Máquinas de soporte vectorial

Las máquinas de soporte vectorial (SVM) son clasificadores que utilizan un conjunto de algoritmos desarrollados por Vladimir Vapnik. Funcionan construyendo un hiperplano separador que permita separar las distintas clases. En nuestro caso utilizaremos la herramiento libsvm [8]. En nuestros experimentos siempre probaremos con los tres kernels implementados en nuestra herramienta y variaremos el número de dimensiones de los datos.

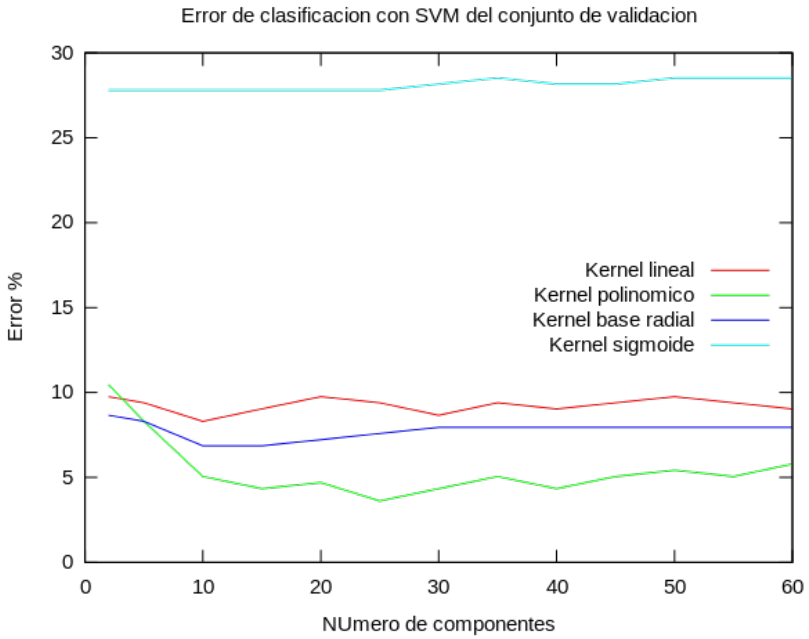

Figura 2.6: Error de clasificación en máquinas de soporte vectorial del conjunto de validación

Aquí vemos claramente que el error se ha reducido drásticamente respecto a los anteriores, como se puede observar en la gráfica de la Figura 2.6. Utilizando vectores de características de 25 componentes obtenemos un error de 3.61 % en validación y 4.63 % en test.

2.8. Comparación de resultados

Tabla 2.1: Resultado Clasificadores

Clasificador	Validación	Test
HMM	18.61 %	18.91 %
K vecinos	13.82 %	14.49 %
Perceptrón multicapa	14.55 %	14.89 %
SVM	3.61 %	4.63 %

Tabla 2.2: Matriz de confusión del mejor resultado

	no partidas	partida	error relativo %
no partidas	342	8	2.3 %
partida	15	132	10.2 %

Como podemos ver en la Tabla 2.1 el mejor resultado se obtiene utilizando las máquinas de soporte vectorial. Este error del 4.63 % supone casi una reducción del 80 % respecto al error que obtendríamos si todas las líneas se clasificaran como no partidas. También se puede observar en la Tabla 2.2 como solo un 2.3 % de las líneas no partidas están mal clasificadas. En cambio, las líneas partidas presentan un 10.2 % de error de clasificación. El primer error no es preocupante, pues no afecta demasiado a la tarea que nos hemos propuesto; en cambio, el segundo error sí que puede afectar más al no juntar palabras que deberían ir juntas.

Capítulo 3

Reconocimiento de la escritura

En este capítulo vamos a comprobar cómo la unión de líneas puede afectar al reconocimiento de la escritura manuscrita. Para ello usaremos el reconocedor iatros. Después de juntar las líneas mediante unos experimentos comprobaremos las mejoras producida.

3.1. Iatros

Para la parte del reconocimiento de la escritura utilizaremos iatros[9]. iatros es una herramienta que nos permite trabajar con texto manuscrito y con sonidos. Esta aplicación utiliza modelos ocultos de Markov para identificar las letras y después mediante transductores estocásticos de estados finitos obtiene las palabras que forman las líneas de texto. Además ofrece unas serie de herramientas para preprocesar nuestros datos y extraer características.

3.2. Fusionar líneas

Para fusionar las líneas con palabras partidas aplicaremos todos los pasos de preparación de las imágenes para el reconocedor de escritura manuscrita (iatros) [3] excepto la extracción de características. Este preproceso consistirá en: binarizar la imagen, corregir la inclinación horizontal de las palabras, corregir la inclinación vertical de las letras y normalizar la altura de los descendientes y ascendientes. Esto nos facilitará el proceso pues nos facilitará la tarea de alinear las palabras.

A la hora de fusionar dos imágenes nos aseguraremos que ambas tengan la misma altura; para ello escalaremos la segunda imagen manteniendo el ratio de aspecto. Si no se corrige la altura podemos encontrar casos en que las palabras de distintas líneas presentan un tamaño muy diferente.

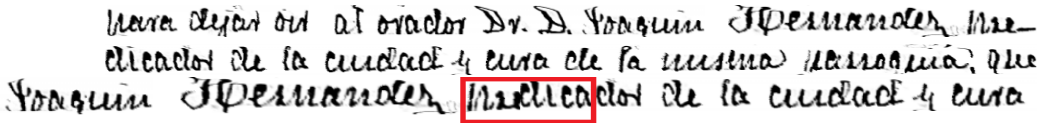

Figura 3.1: Linea fusionada

Tabla 3.1: Resultado iatros

Palabras partidas	29.929653
Sin palabras partidas perfectas	29.888084
Sin palabras partidas con clasificador	31.390728

Como podemos ver en la Figura 3.1 el guión desaparece por completo y el resultado final parece bastante prometedor.

Una vez fusionadas las líneas correspondientes procedemos a extraer las características mediante la herramienta facilitada por el reconocedor iatros.

3.3. Experimentos

Para comprobar como ha afectado la fusión de las palabras partidas a la tarea de reconocimiento del texto probaremos tres experimentos distintos. El primer experimento consistirá en utilizar las líneas con palabras partida, el segundo consistirá en fusionar las líneas de forma perfecta; para el tercero fusionaremos las líneas según nos dicte nuestro clasificador.

Para los tres experimentos simplificaremos el problema utilizando solo letras mayúsculas y eliminado los signos de puntuación. Los modelos morfológicos se componen de HMM continuos de 14 estados por símbolo, 16 gaussianas por estado, 48 características por gaussiana, topología estrictamente lineal, matriz de covarianza diagonal, entrenados con la herramienta HTK [10].

Los modelos del lenguaje son brigramas suavizadas obtenidas a partir de la herramienta SLM [11], tomando como conjunto de entrenamiento las transcripciones de la parte de entrenamiento e incluyendo las palabras fuera de vocabulario (vocabulario cerrado).

La herramienta se configurará con los siguientes parámetros: beam 3000, grammar-scale-factor 80, word-insertion-penalty -80, histogram-pruning 100000 y sin símbolos inicial ni final.

Los resultados para los tres experimentos, con esta configuración del reconocedor y con los modelos descritos, se presentan en la Tabla 3.1. Como se

puede observar, la fusión de líneas perfecta no aporta apenas beneficio, y la basada en clasificador es, evidentemente, un poco peor al no fusionar líneas que deberían estarlo

Capítulo 4

Conclusiones y líneas futuras

4.1. Conclusiones

En este trabajo hemos desarrollado una metodología para comprobar cómo las palabras partidas pueden afectar a la tarea de clasificación. Para ello hemos ideado un método de detección de palabras partidas probando diferentes clasificadores.

Hemos podido observar que es muy sencillo recortar el final de la imagen para saber si es una línea partida o no, como vemos en las Figuras 2.1 y 2.2.

En cuanto a los distintos clasificadores probados, hemos podido observar grandes diferencias entre unos y otros. Hemos visto que el mejor resultado lo obteníamos con SVM, como vemos en la Tabla 2.1. El principal inconveniente es que la mayor parte del error provenía de clasificar líneas partidas como no partidas, como se ve en la Tabla 2.2. Estas líneas son las más problemáticas, pues fusionar dos líneas no partidas no perjudica la tarea de clasificación. Aun así el resultado es más que aceptable.

En cuanto a la fusión de líneas se ha ideado un método sencillo apoyándose en las herramientas facilitadas por iatros. Aparentemente esto nos ofrecía resultados aceptables, como se ve en la Figura 3.1.

Finalmente al comprobar cómo mejoraba la no inclusión de líneas partidas a la tarea de reconocimiento se ha visto claramente que su influencia ha sido prácticamente nula, como se ve en la Tabla 3.1.

Parta concluir podemos, decir que la tarea de detección de líneas partidas es sencilla, pero en cambio no aporta ninguna mejora aparente al reconocimiento de caracteres con iatros. Esto puede deberse a que la fusión de las lineas es muy sensible a los cambios de escala de las imágenes y en muchas ocasiones no da buenas fusiones.

4.2. Líneas futuras

En futuros trabajos habría que comprobar nuevos métodos de fusión de líneas para ver cómo afectan a la tarea de reconocimiento y también probar distintos reconocedores distintos a iatros, con el fin de determinar si alguno es más sensible a la no inclusión de líneas partidas.

Índice de figuras

1.1. Ejemplos de líneas 4

2.1. Palabra partida 7
2.2. Palabra no partida 7
2.3. Error de clasificación en HMM del conjunto de validación . . . 8
2.4. Error de clasificación en k-vecinos del conjunto de validación . 9
2.5. Error de clasificación perceptrón multicapa del conjunto de validación 10
2.6. Error de clasificación en máquinas de soporte vectorial del conjunto de validación 11

3.1. Linea fusionada 14

Índice de tablas

2.1. Resultado Clasificadores 11
2.2. Matriz de confusión del mejor resultado 12

3.1. Resultado iatros 14

Bibliografía

[1] Y. J. . S. C. . Y. K. Mori, S. ; Ricoh Co. Ltd., "Historical review of ocr research and development," *Proceedings of the IEEE (Volume:80 , Issue: 7)*.

[2] V. S. D. M. Narasimha Murty, *Pattern Recognition: An Algorithmic Approach*.

[3] A. Juan, A. H. Toselli, J. Domnech, J. González, I. Salvador, E. Vidal, and F. Casacuberta, "Integrated handwriting recognition and interpretation via finite-state models," *Int. J. Patt. Recognition and Artifial Intelligence*, vol. 2004, 2001.

[4] "Cristo-salvador." `https://prhlt.iti.upv.es/page/projects/multimodal/cs/index`.

[5] G. Wilson, "Properties of contour codes," *Vision, Image and Signal Processing, IEE Proceedings*.

[6] "Gnu octave." `http://www.gnu.org/software/octave/`.

[7] U. of Tübingen, "Stuttgart neural network simulator." `http://www.ra.cs.uni-tuebingen.de/SNNS/`.

[8] C.-C. Chang and C.-J. Lin, "Libsvm." `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[9] "iatros." `https://prhlt.iti.upv.es/page/projects/multimodal/idoc/iatros`.

[10] M. J. F. G. T. H. D. K. G. M. J. O. D. O. D. P. V. V. . J. Young, G. Evermann and P. C. Woodland, *The HTK Book*. UK: Cambridge University Engineering Department, 3.4 ed.

[11] R. Rosenfeld *The cmu-cambridge statistical language modelling toolkit v2*, 1998.