Universidad de León

Departamento de Ingeniería Eléctrica
y de Sistemas y Automática

SISTEMA EXPERTO PARA EL RECONOCIMIENTO DE DOCUMENTOS MANUSCRITOS BASADO EN EL ANÁLISIS DE TRAZOS

Tesis presentada por
David Álvarez León

para la obtención del Título de Doctor
por la Universidad de León

Directores:
Dr. D. Ramón Ángel Fernández Díaz
Dra. Dña. Lidia Sánchez González

León,
Noviembre 2015

University of León

Department of Electrical Engineering, Systems and Automatic

EXPERT SYSTEM FOR HANDWRITTEN DOCUMENTS RECOGNITION BASED ON STROKE ANALYSIS

A dissertation submitted by
David Álvarez León

in fulfillment of the requirements for the Degree of
Doctor of Philosophy (PhD)
University of León

Supervised by:
Dr. D. Ramón Ángel Fernández Díaz
Dra. Dña. Lidia Sánchez González

León
November, 2015

A Francisco y Carmen, mis padres,

... por ser mis mejores maestros en la vida.

Siempre han dado y darán todo por y para mí.

A Beatriz, mi hermana,

... por su paciencia

siempre infinita conmigo.

A Ana,

... por ser mi compañera de camino.

Por ser TÚ.

Agradecimientos

Esta Tesis, al igual que otras, es el resultado de un largo tiempo de trabajo, esfuerzo y mucha dedicación. Y llega un momento en el que merece la pena detenerse por unos momentos y pensar en todas aquellas personas que han dado su apoyo para que este trabajo pudiera llevarse a cabo. Intentaré en estas líneas representar a todas ellas, aunque mi agradecimiento no sólo se reduce a esta lista.

Al departamento de Ingeniería Eléctrica y de Sistemas y Automática, y con especial afecto, a mis directores Ramón y Lidia por hacer de este proyecto una ilusión y por iniciarme en el mundo de la investigación. Gracias, Ramón, por los cafés con leche mientras revisábamos ideas y esas tardes que sacrificabas conmigo que casi te cuestan *"el divorcio"*... Y a ti Lidia, agradecerte enormemente tu incansables ganas para que todo fuera perfecto y tu dedicación. Pero sobre todo, a ambos, por vuestros ánimos en la cafetería en los momentos más duros.

A Beatriz, Jenny, Marta, Helen, Mary, Iris, Laura, Ana y en especial, a Carlos. Gracias a todos por creer en este proyecto y dedicarle vuestro tiempo a escribir letras en las aburridas plantillas para mis pruebas. En este trabajo me guardo lo mejor de cada uno de vosotros. Prometo devolveros ese tiempo *"robado"*.

A Carlos, por los detalles de revisión e impresión y por supuesto, por su amistad. De igual modo, a todos los componentes de *Carpe Diem* y en especial a Christian, por acogerme como a uno más de la familia. A la *"Tribu Omaha"* por sus ánimos y A José Bernardo, por su infinita paciencia y por escucharme cuando le contaba mis ideas sobre los trazos en el trabajo.

Con un especial cariño, a mis padres, Francisco y Carmen y a mi hermana Beatriz. Ellos han sido el pilar fundamental en mi vida y por supuesto, para la realización de esta Tesis. Vosotros me habéis enseñado que con esfuerzo y perseverancia, uno no tiene límites y puede conseguir todo lo que se proponga. Gracias por señalarme el camino, vuestros sabios consejos y apoyo incondicional.

A Ana, mi novia, mi amiga y mi compañera de camino. Seguramente no encuentre las palabras adecuadas para agradecerte todo lo que has hecho y haces por mí. Sólo puedo pensar que tú haces que todo tenga sentido, incluida esta Tesis. Gracias por encontrar siempre las mejores palabras de ánimo en los momentos más difíciles y por enseñarme que todo esfuerzo tiene su recompensa. Gracias por *"volver a sonreír"* aun siendo momentos duros. Y recuerda que, como siempre te he dicho, lo mejor está por llegar.

A vosotros y a todos los que de alguna forma han sido partícipes en este trabajo

Gracias.

RESUMEN

En este trabajo se propone un sistema experto para el reconocimiento de texto manuscrito a partir de la información obtenida al analizar los trazos que componen el propio texto.

Tras aplicar un serie de operaciones previas a la imagen adquirida, se realiza la segmentación correspondiente para obtener sus letras. Los píxeles de cada letra se clasifican en verticales u horizontales atendiendo a sus características y se agrupan formando trazos verticales u horizontales. Mediante zonificación dinámica se localizan las intersecciones existentes entre los trazos verticales con sus horizontales adyacentes. El uso de una gramática formal permite reducir esta característica a una cadena representativa que es reconocida por un autómata finito para comprobar su validez.

El almacenamiento de los descriptores se realiza mediante una base de conocimiento con estructura de árbol *trie*. En sus nodos, se almacenan cada uno de los elementos de la cadena representativa previamente generada incorporando, además, un nodo hoja con el carácter que identifica.

De esta manera, el motor de inferencia es capaz de realizar búsquedas de nuevos caracteres sobre la base de conocimiento. En un contexto alfabético, el motor de inferencia se ayuda de un corrector ortográfico para componer la palabra de la imagen introducida a reconocer. Por el contrario, en un reconocimiento numérico, toma una decisión estadística en base al entrenamiento.

Se han desarrollado una serie de experimentos tanto para el reconocimiento alfabético como para el reconocimiento numérico. Para el primero de ellos, con el fin de comprobar la eficacia del sistema experto, se ha desarrollado una aplicación, XIRIS, que permite realizar los experimentos en base a cuatro escenarios posibles. En dichos escenarios, se han utilizando palabras sintéticas previamente generadas a partir de una muestra de caracteres reales escrita por dos autores. Los resultados señalan una tasa de acierto en el mejor de los escenarios del 95,36% para las palabras y de 95,46% para las letras que componen esas palabras. Por otro lado, para los experimentos con caracteres numéricos se ha utilizado la base de datos MNIST con números escritos a mano por cerca de 250 escritores alcanzando una tasa de acierto del 88,77%.

ABSTRACT

In this work an Expert System for handwritten text recognition is proposed from the obtained information by analysing the strokes contained in the text.

After applying certain pre-processing operations on previously acquired image, the word segmentation is performed to obtain its characters. Pixels of each character are classified as vertical or horizontal according to its characteristics, and then, they are grouped by orientation as vertical or horizontal strokes. In this way, by dynamic zoning the adjacency regions between vertical and horizontal strokes are located. The use of a formal grammar allows us to extract a representative string from the stroke position. That chain is recognized by a finite automaton to chech its validity.

To store the descriptors, a knowledge base with *trie* tree structure is defined. In their nodes are stored each element of the generated representative string, having a node leaf with the character identification.

Thereby, the inference engine is capable of searching for new characters in the knowledge base. In an alphabetic context, the inference engine uses a spell checker to compose the word of the input image to be recognized. By contrast, in a numerical recognition, a statistical decision is performed based on training knowledge.

The experiments have been developed for both, alphabetic and numeral recognition. For the first one, in order to test the effectiveness of the Expert System, an application, XIRIS, is developed to perform experiments on four designed scenarios. In addition, a sample of real characters written by two authors is used to gener-

ate synthetic words for testing purposes. The results show a final word recognition rate of 95,36%. For the letters of the processed words the hit rate is 95,46%. On the other hand, MNIST database, that contains handwritten numbers written by around 250 writers, is considered for numeric recognition achieving in the experiments a recognition rate of 88,77%.

Índice general

Índice de figuras

Índice de tablas

Listado de Acrónimos

BTASOM Mapas Auto-Organizados en árboles binarios, *Binary Tree Time Adaptative Self-organized Maps*. 40, 131, 132

FFBPNN Red Neuronal con Retropropagación, *Feedforward Back Propagation Neural Network*. 40

GHSOM Mapas Auto-Organizados Jerarquizados, *Growing Hierarchical Self-organized Maps*. 40, 131, 132

HMM Modelos Ocultos de Markov, *Hidden Markov Models*. 30–32, 39, 41

HWAI Interpretación de Direcciones Manuscritas, *HandWritten Address Interpretation*. 32

ICR Reconocimiento Inteligente de Caracteres, *Intelligent Character Recognition*. 23

KNN *k*-Nearest Neighbor, *k-Nearest Neighbor*. 39

LVQ Learning Vector Quantization,*Learning Vector Quantization*. 42

MLP Perceptrón Multicapa, *Multilayer Perceptron*. 30, 40

NN Redes Neuronales, *Neural Network*. 40

OCR Reconocimiento Óptico de Caracteres, *Optical Character Recognition*. 23

RNN Redes Neuronales Recurrentes, *Recurrent Neural Network*. 40

SOM Mapas Auto-Organizados, *Self-organized Maps*. 30, 40, 42

TDNN Redes Neuronales con Retardo en el Tiempo, *Time Delay Neural Networks*. 30, 40

"El razonamiento matemático puede considerarse más bien esquemáticamente como el ejercicio de una combinación de dos instalaciones, que podemos llamar la intuición y el ingenio".

– Alan Mathison Turing

1 Introducción

Este trabajo se centra en la aplicación de la visión artificial en la búsqueda de patrones y más concretamente, en el reconocimiento de texto manuscrito. A partir de una imagen de entrada que representa un texto, se extraen las características que identifican los caracteres de dicho texto. Más tarde, estas características son clasificadas para ser contrastadas con las de nuevos textos desconocidos y estimar así una solución de reconocimiento. Puesto que existen muchas y diferentes formas de escritura, las técnicas para el reconocimiento están fundados en numerosas perspectivas. Los sistemas expertos es una de ellas y, a diferencia de otros, se basan en un conjunto de reglas establecidas a partir de una muestra conocida para crear su propio conocimiento y así poder usarlo en nuevas entradas.

La metodología detallada en los próximos capítulos describe el desarrollo de un sistema experto capaz de reconocer texto manuscrito basándose en un conjunto de reglas generadas a partir de las características extraídas de un texto.

1.1 Motivación

Desarrollar una metodología capaz de obtener un texto como imagen y transformarlo a texto en un ordenador ha sido un problema de gran interés perseguido científicamente desde el origen de las máquinas. Los esfuerzos continuos por resolver este problema son prueba de ello llevándose a cabo investigaciones en todos los ámbitos posibles.

Actualmente, existe todavía la transcripción de documentos en el que los usuarios deben reproducir en un ordenador un documento escrito a mano. Un claro ejemplo de este trabajo es el llevado a cabo en entidades gubernamentales donde todavía se copian documentos antiguos para una posterior consulta o realización de búsquedas en el propio documento. La automatización del proceso de transcripción reduciría el consumo de tiempo excesivo en realizar esta tarea.

Pese a que ya existen estudios que abarcan este problema, se plantea una nueva perspectiva basada en sistemas expertos donde el desarrollo de una metodología capaz de automatizar la transcripción de documentos es el marco donde se ha desarrollado este trabajo.

1.2 Objetivos

El principal objetivo del presente trabajo consiste en desarrollar una metodología que defina un sistema experto para el reconocimiento de texto manuscrito en su variante Off-line, es decir, a partir de texto previamente escrito. Por lo tanto, el problema global se puede dividir en una sucesión de cinco etapas: preprocesamiento, segmentación, extracción de características, clasificación y reconocimiento.

La finalidad de la etapa de preprocesamiento es obtener una imagen normalizada aplicando para ello una serie de algoritmos sobre la imagen previamente escaneada. Esto servirá como entrada a la etapa de segmentación en la cual se debe conseguir una unidad básica lo más significativa posible que pueda definir una unidad léxica. Se parte pues, de las líneas de un texto para obtener las palabras, mecanismo necesario para obtener después las letras. La extracción de las características a partir de las unidades básicas previamente segmentadas es la tercera tarea, cuya definición debe ser única e inequívoca para cada una de las diferentes clases de unidades básicas encontradas. La clasificación de manera eficiente de las características obtenidas con el objeto de ofrecer la mejor solución a nuevas entradas es el propósito de la cuarta etapa. Por último, en una quinta etapa, las nuevas entradas son comparadas por el Motor de Inferencia con las existentes en el entorno de clasificación para obtener, junto con la ayuda de un corrector ortográfico, una solución al problema de reconocimiento.

El sistema diseñado a partir de la metodología descrita, debe ser analizado para evaluar su eficiencia. Por ello, éste debe ser entrenado a partir de una muestra previamente seleccionada con el fin de reconocer el mayor número de palabras posible de un texto. De esta manera se podrán obtener resultados concluyentes sobre el rendimiento del sistema para un posterior estudio y evaluación del mismo.

De una forma más concisa, se especifican los siguientes objetivos a alcanzar:

1. Determinar un método que permita extraer las características de los caracteres segmentados. Se deben utilizar los mecanismos apropiados para recoger aquella información que sea relevante para el sistema de reconocimiento independientemente de la configuración del carácter.

2. Formalizar un mecanismo que transforme las características extraídas de un carácter en unidades representativas. Los descriptores deben definirse de la forma más concisa posible con el fin de reducir al máximo el número de ambigüedades que se pudieran producir.

3. Desarrollar un sistema de clasificación a partir de las características obtenidas. El conocimiento se debe almacenar de forma que pueda ser consultado posteriormente para dar una solución satisfactoria sobre nuevos elementos desconocidos.

4. Especificar un Motor de Inferencia que realice búsquedas en el sistema de clasificación de manera que se obtenga la mejor solución posible al reconocimiento.

1.3 Alcance

En este trabajo se presenta una metodología en la que se describe el desarrollo de un sistema experto capaz de reconocer textos y números manuscritos basado en el análisis de trazos de los caracteres que lo componen. No se trata de focalizar cada sección del capítulo de la metodología de manera independiente, ya que la metodología en sí es un conjunto dependiente de estados en el que el resultado de uno, se convierte en la entrada del siguiente. Por lo tanto, se debe ver el sistema experto como un todo en el que su principal finalidad es la de reconocimiento.

Asimismo, se detallan las contribuciones de este trabajo sobre el campo de investigación y se exponen y analizan los resultados que se han obtenido en los experimentos llevados a cabo. Finalmente, los resultados son comparados con los de otros autores para cuantificar el impacto de este trabajo.

1.4 Estructura del documento

A continuación, en el Capítulo 2, se hace una revisión sobre el estado del arte existente actualmente en el campo de visión artificial. Se examina en mayor detalle el reconocimiento óptico de caracteres con especial interés sobre caracteres manuscritos. Se exponen las dos variantes existentes en este tipo de reconocimiento: On-line y Off-line; y se detallan las diferentes metodologías analizando en mayor medida aquellas basadas en sistemas expertos.

En el Capítulo 3 se plantea una metodología para diseñar un sistema experto capaz de reconocer textos manuscritos en función de sus caracteres ya sean alfabéticos o numéricos.

En una primera etapa de preprocesamiento se preparan los datos para obtener de ellos la mayor información posible. Por tanto, se describen en la primera sección los métodos utilizados que permiten eliminar aquellos elementos no relevantes y que dificultan la tarea de reconocimiento.

Una de las contribuciones de este trabajo se expone en la sección 3.3, donde se describe cómo una vez los datos han sido tratados se extraen las características de los mismos. Para ello, se construye una cadena representativa siguiendo las reglas de una gramática formal que es validada posteriormente por un autómata finito.

En la sección 3.4, se muestra otra aportación de este trabajo donde se detalla la forma en que se realiza la clasificación de las características. Mediante un árbol con estructura XML se almacenan las cadenas representativas generadas en la etapa anterior de tal manera que facilitará el acceso a futuras búsquedas. De esta forma, es posible generar una base de conocimiento con los caracteres introducidos por un usuario.

El diseño de un Motor de Inferencia capaz de obtener resultados de una base de conocimiento previamente generada es el objeto de la sección 3.5. En esta contribución se describe el proceso donde a partir de la búsqueda de una nueva entrada en la base de conocimiento, se analizan los resultados obtenidos estimándose la mejor solución al reconocimiento con la ayuda de un diccionario.

En el Capítulo 4 se presentan los experimentos realizados. Se detallan cuáles son los fundamentos así como el diseño que se ha seguido para llevarlos a cabo. Además, se describe el proceso para la optimización del reconocimiento en los experimentos y se muestra la aplicación que se ha desarrollado para la obtención de los resultados. Se distinguen en este apartado los resultados obtenidos a partir de palabras generadas sintéticamente de los conseguidos a partir de caracteres numéricos. Finalmente, se contrastan los resultados logrados con los de otros autores para analizar la eficacia del sistema experto.

Las conclusiones que se han obtenido analizando los resultados se exponen en el Capítulo 5. Asimismo, también se puntualizan las diferentes líneas futuras que persiguen la continuidad de este trabajo.

Por último, en el Apéndice A se muestra el esquema XSD esencial para la generación de la base de conocimiento con estructura de árbol. El Apéndice B detalla la plantilla utilizada para recoger las muestras utilizadas en los experimentos. En el Apéndice C se señalan de una forma más concreta los resultados obtenidos en

la optimización del reconocimiento mientras que en el Apéndice D se especifican las tablas de resultados obtenidos en los experimentos. El Apéndice E recoge una muestra de las letras utilizadas en los experimentos así como una muestra de números de la base de datos MNIST. Para terminar, el Apéndice F enumera las publicaciones relacionadas con esta Tesis.

"Las ciencias no tratan de explicar, incluso apenas tratan de interpretar, construyen modelos principalmente. Por modelo, se entiende una construcción matemática que, con la adición de ciertas interpretaciones verbales, describe los fenómenos observados. La justificación de tal construcción matemática es sólo y precisamente que se espera que funcione".

– John Von Neumann

2

Antecendentes

En este capítulo se introduce brevemente el reconocimiento óptico de caracteres, uno de los campos con mayor interés dentro de la visión artificial. Dentro de este tipo de reconocimiento se describe en mayor detalle el reconocimiento de caracteres manuscritos, haciendo distinción entre reconocimiento *On-line* y reconocimiento *Off-line* atendiendo a la forma en que se recogen los datos. Se exponen las características que identifican a cada uno de ellos y se analizan de forma exhaustiva los métodos seguidos por los autores destacados en el campo poniendo especial interés en los métodos *Off-line*.

2.1 Introducción

Desde siempre, se ha perseguido el interés por el que las máquinas puedan realizar tareas asignadas por el hombre de manera eficaz y un claro ejemplo de ello es la búsqueda de patrones como lo podría hacer un ojo humano. Hoy en día, y gracias a los avances en las tecnologías, se está más cerca de lograr ese objetivo mediante las técnicas que ofrece la visión artificial. De entre los múltiples ámbitos que abarca este término, quizás el más conocido sea el que aplica sobre la transcripción de documentos en general a lo que se conoce comúnmente como Reconocimiento Óptico de Caracteres, *Optical Character Recognition* (OCR).

Concretamente, cuando se trata de reconocer texto manuscrito, se suele denominar Reconocimiento Inteligente de Caracteres, *Intelligent Character Recognition* (ICR) para distinguirlo de su predecesor. Lo que se persigue en este tipo de reconocimiento es la transformación del lenguaje representado de forma espacial o como marcas gráficas en una representación simbólica como puede ser ASCII o UNICODE para su posterior interpretación.

Esta tarea no es sencilla ya que la forma de escribir está ligada a la persona en sí. Además, si se parte de la definición de que un lenguaje está formado por un conjunto de formas básicas como son los caracteres o letras y un conjunto de reglas para combinar esas formas [36], la tarea de reconocimiento se puede convertir en un verdadero problema si se pretende un reconocimiento independiente del lenguaje.

Existen dos formas de reconocimiento en el ICR atendiendo a la forma de la obtención de escritura manuscrita (Ver Fig. 2.1). Si se utiliza un dispositivo electrónico o un bolígrafo especial para recoger los caracteres introducidos por el usuario, entonces se realizará un reconocimiento *On-line*. Si por el contrario el reconoci-

miento se pretende sobre un papel ya escrito, éste será de tipo *Off-line* [127].

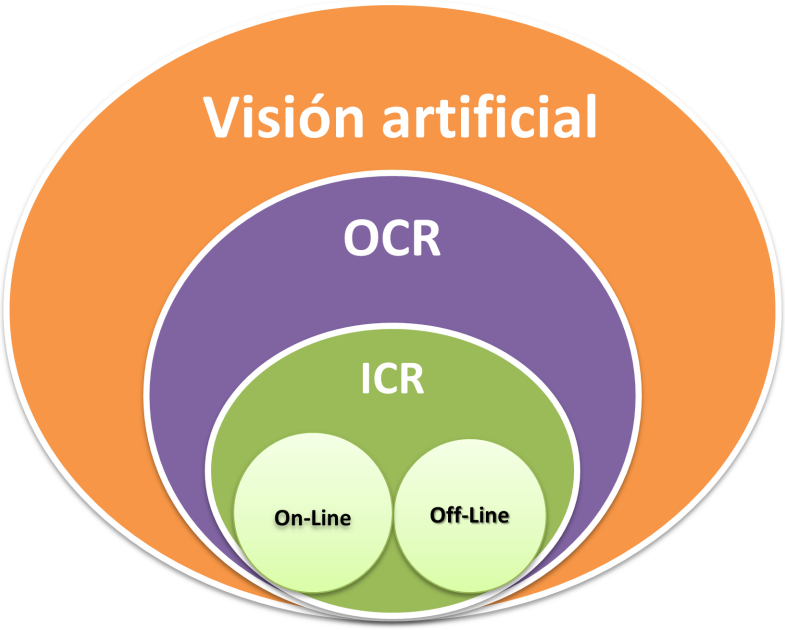


Figura 2.1: Esquema y clasificación general del reconocimiento óptico de caracteres manuscritos en la visión artificial.

2.2 Reconocimiento *On-line*

Este tipo de reconocimiento se lleva a cabo mientras se está realizando la escritura. Para ello, se utiliza un dispositivo específico que recoge parámetros tales como la posición, la velocidad o la aceleración en función del tiempo de escritura (ver figura 2.2).

A lo largo de la historia de este tipo de reconocimiento, se han desarrollado numerosas perspectivas. Así pues, mientras los primeros intentos de reconocimiento se realizaron utilizando solamente letras [44, 46], con el paso de los años, estas téc-

nicas han ido mejorando debido a los avances en la tecnología y prueba de ello son los numerosos estudios realizados [126, 156, 162].

Uno de los sistemas más atractivos desarrollados dentro del reconocimiento *On-line*, es aquel en el que el entorno se va adaptando progresivamente al usuario partiendo de unas características básicas para cada carácter. Después, el propio usuario puede ir añadiendo nuevas formas para ir acomodando el sistema a sus necesidades. La ventaja de esta metodología es que se moldea a cualquier tipo de lenguaje, estilos personales de escritura o puede recoger cambios de hábitos en la escritura [42, 107, 109].

En esta forma de reconocimiento era habitual el uso de dispositivos como tablas digitalizadoras o basados en lápices digitales que recogían los trazos introducidos por el usuario [38, 140]. Hoy en día, esta tecnología ha ido migrando paulatinamente hacia dispositivos móviles en el que las pantallas táctiles hacen el mismo trabajo con altos índices de precisión [78, 154]. En ambos casos, el fin del reconocimiento es el mismo: reconocer mensajes escritos o comandos gestuales.

Otro campo de especial interés es el reconocimiento de firma [58, 123] en el que el objetivo es claro: verificar la identidad de una persona determinada mediante la singularidad y exclusividad de la forma de escribir de la persona en cuestión. Estos sistemas suelen estar fuertemente entrenados con el fin de minimizar la tasa de error ya que dependiendo del ámbito de integración, puede ser un factor crítico [70].

Es también bastante frecuente el uso de este tipo de reconocimiento en ámbitos educativos. Por ejemplo, el reconocimiento de formas matemáticas para realizar cálculos es uno de los campos más interesantes [98, 148]. Pero la aplicación por excelencia dentro de este ámbito, es en el de la enseñanza de escritura. Trata, de forma general, de mostrar en la pantalla de un dispositivo un determinado carácter

para que el alumno intente reproducirlo de la forma más concisa posible con el fin de mejorar sus habilidades caligráficas [8, 29, 137, 171]. Muy cerca de este ámbito también se suele usar en propósitos de rehabilitación por personas que padezcan algún trastorno en el control del sistema motor ya sea por enfermedades como el Parkinson [49, 59, 128], por dislexia o disgrafía [134].

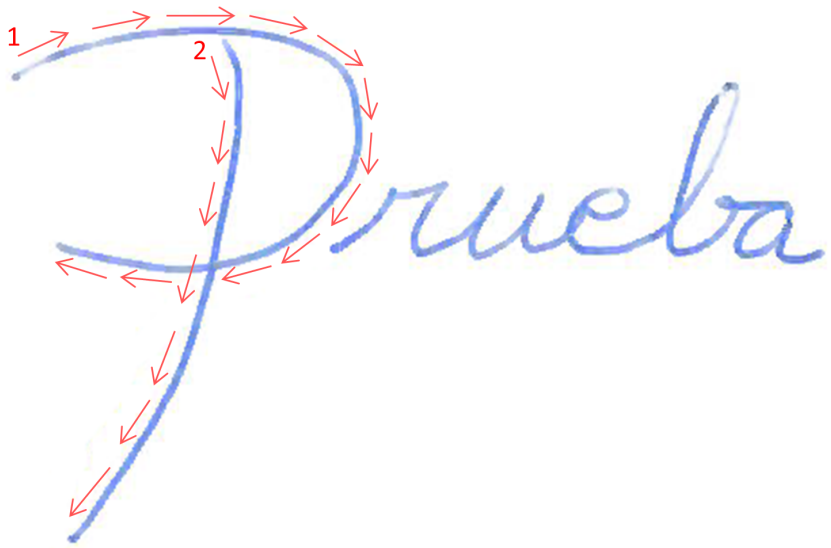


Figura 2.2: Parámetros recogidos durante el proceso de reconocimiento *Online*. Algunos de estos parámetros tales como las coordenadas, la dirección, la presión o la curvatura son determinantes a la hora de reconocer texto mediante esta variante de reconocimiento.

2.2.1 Características

Por lo general, los datos que se recogen con un dispositivo son preprocesados para reducir así cualquier ruido que pueda portar la señal. Después, se normaliza el

trazo y se segmenta en unidades significativas [61]. El ruido puede ser generado por diferentes factores como puede ser un error en el propio dispositivo que digitaliza la señal, errores en la escritura, movimientos en los dedos, etc. Por eso se suelen utilizar todo tipo de filtros para obtener una señal que sea lo más homogénea posible [113].

Otros procesos que se usan habitualmente en la normalización son la corrección de la linea base [22], corrección de la inclinación [25, 101], así como ajustar la letra para que sea del mismo tamaño [113].

Para el proceso de segmentación se llevan a cabo varias operaciones sobre la entrada normalizada con el fin de obtener una representación en forma de unidad básica que más tarde va a ser utilizada por el algoritmo de reconocimiento. Se suelen considerar dos niveles. En un primer nivel, se obtiene el mensaje como un todo y se centra en la detección de la linea base [147, 163], la segmentación de palabras [156] o separar aquello que no tiene un significado textual, pero sí representativo como puede ser un estilo de escritura [161], ecuaciones [40, 148], diagramas [166] o diacríticos [143] en el texto. El objetivo de este primer nivel es definir regiones espaciales o temporales para extraer las unidades básicas. En un segundo nivel, la metodología se centra en la segmentación de los elementos del primer nivel en letras separadas, si se tratara de palabras, o incluso trazos. Este último método suele ser el más común en el reconocimiento manuscrito [126].

Uno de los problemas que se suele encontrar en la segmentación de las palabras en letras es la dificultad para determinar donde empieza y acaba un carácter. La mayoría de las técnicas empiezan escaneando de abajo a arriba buscando trazos que suelen usarse a la hora de escribir un determinado carácter. Estos trazos suelen ocultarse en la señal debido a la anticipación o a efectos de súper imposición [117, 125]. Algunos de los métodos para la segmentación se basan en la observación de características del trazo como por ejemplo, estudio del punto máximo

de curvatura [165], puntos de cruce de velocidad cero [76], mínimos en el eje de las $y(t)$ [74] o el mínimo en valor absoluto de la velocidad [141]. Existen también otros métodos que se basan en la detección de puntos de referencia [7, 93, 124] así como de formas primitivas [14, 19, 33, 92]. Son útiles también métodos basados en el modelo donde se utilizan técnicas de regresión no lineal a partir del modelo que se escribe para obtener las características del trazo que se desea escribir [62]. Es frecuente encontrar técnicas donde se combina la segmentación con el reconocimiento [149, 168].

Existen cuatro factores que pueden afectar en el reconocimiento y que debido a ellos, se pueden generar interpretaciones equivocadas sobre el texto que se desea reconocer [140]:

- *Variaciones geométricas:* Son aquellas que afectan a cambios en la posición, tamaño, línea base e inclinación del texto.
- *Ruido neuromecánico:* Afecta a la calidad con la que se escribe el texto.
- *Variaciones gráficas:* Son las diferentes formas en las que uno o varios escritores pueden representar un mismo carácter.
- *Problemas de secuencia:* El orden en el que se realizan los trazos para representar un mismo carácter.

En cuanto al sistema de clasificación, existen dos tipos claramente diferenciables: métodos basados en reglas y en la forma, y métodos estadísticos

2.2.2 Métodos basados en reglas y en la forma

La idea principal de estos métodos es la descripción de la forma del carácter de manera que sea lo más abstracta posible sin atender a las posibles variaciones que

el carácter puede tener. Los sistemas basados en reglas, que eran bastante habituales en los 60, se abandonan debido a las dificultades encontradas a la hora de crear las reglas o al generarlas de manera automática puesto que cada vez los sistemas eran más complejos debido a la gran cantidad de maneras que existen de escribir la misma palabra. Actualmente, estos métodos se están combinando con lógica difusa que usa algún tipo de información probabilística como por ejemplo la frecuencia de aparición de ciertas características a la hora de escribir [122]. Cabe destacar, que si se usa esta metodología definiendo las reglas correctamente, el sistema se puede simplificar drásticamente ya que no requiere de una muestra extensa ni de un entrenamiento complejo.

2.2.3 Métodos estadísticos

Estos métodos definen cada una de las formas básicas como un número fijo de características para construir un espacio multidimensional donde la probabilidad de que se produzca un acierto es equilibrado en la red durante la fase de entrenamiento. Dentro de los métodos estadísticos, se pueden distinguir tres clases [127]:

Métodos explícitos. Los métodos explícitos siguen modelos matemáticos, concretamente, el análisis discriminante lineal que se utiliza tanto de forma directa como indirecta. Uno de los problemas que se encuentran en estos modelos es que por lo general se basan en hipótesis sobre la forma o los parámetros que describen la distribución estadística además de requerir tiempos de computación elevados y consumos excesivos de memoria.

Métodos implícitos. Estos métodos se caracterizan principalmente por usar redes neuronales donde los grados de probabilidad en la clasificación se determinan a partir de los datos de entrenamiento [17]. Existen también redes neuronales

como Perceptrón Multicapa, *Multilayer Perceptron* (MLP) cuyos errores se propagan desde el nodo final de la red hacia los diferentes nodos que la componen igualando las probabilidades de éstos como modo de aprendizaje. Las redes Kohonen o Mapas Auto-Organizados, *Self-organized Maps* (SOM) [84, 94] y las Redes Neuronales con Retardo en el Tiempo, *Time Delay Neural Networks* (TDNN) suelen ser de especial interés entre los autores que aplican este método.

Procesos de Markov. Los Modelos Ocultos de Markov, *Hidden Markov Models* (HMM), basados en los procesos de Markov, es una de las metodologías más utilizadas [6, 69, 85, 105, 131]. Su modelo se basa en dos capas estocásticas en la que en la primera de ellas oculta un proceso subyacente con estados de transición condicionales bajo una distribución probabilística y que, además, suele depender por lo general del estado inmediatamente anterior. La segunda capa es un proceso visible que se establece por el proceso subyacente y que se caracteriza por la emisión de un símbolo bajo una distribución probabilística del estado en el que se encuentra actualmente. La figura 2.3 muestra un ejemplo. La idea principal es determinar la probabilidad en la que, dada una determinada secuencia observada, permita ir de un estado a otro para obtener un símbolo resultado.

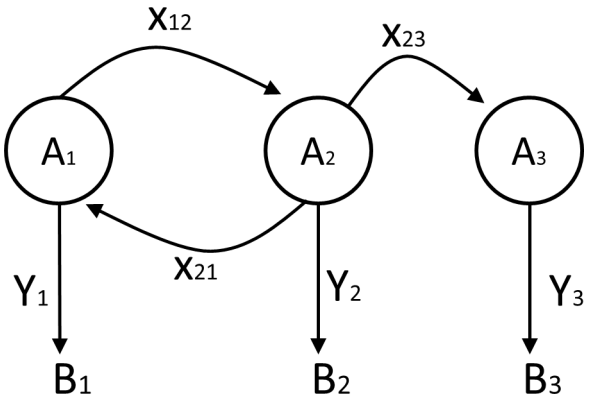


Figura 2.3: Ejemplo de un modelo oculto de Markov con sus transiciones entre los diferentes estados donde x son estados ocultos, y las salidas observables, a son las probabilidades de transición y b son las probabilidades de producir la salida.

Los HMM pueden clasificarse, atendiendo al modo de obtención de los datos en un tiempo *t*, en discretos o continuos. El modelo discreto utiliza un algoritmo que cuantifica los datos para obtener un símbolo en determinados instantes de tiempo *t* en el que se realiza la observación al sistema. Por el contrario, el modelo continuo asume que las distribuciones de los símbolos observables son densidades de probabilidad definidas sobre espacios de observación continuos de tiempo *t* en las que se suelen usar funciones de densidad de tipo Gaussianas.

Algunos autores van un paso más allá y utilizan una combinación de ambos modelos, tanto discreto como continuo [136], o aprovechan las propiedades de los HMM junto con el de las redes neuronales creando así redes híbridas en sus metodologías [167].

2.3 Reconocimiento *Off-line*

El reconocimiento *Off-line* se podría definir como una tarea en la que se transforma una imagen que contiene texto escrito a mano en su correspondiente transcripción en un dispositivo electrónico. Para ello, es necesario previamente realizar una serie de operaciones con el fin de normalizar la imagen de entrada y obtener así un reconocimiento más preciso.

El esfuerzo por reconocer texto manuscrito es una tarea con más de cuarenta años de historia y en la que se han visto implicados muchos grupos de investigación [52, 71] buscando una solución a este problema en los diferentes escenarios posibles.

El reconocimiento de direcciones postales es una de las más interesantes dentro

de esta forma de reconocimiento [34, 41]. La tarea consiste en asignar un texto en una carta a una dirección de envío. Para ello, se necesitan determinar datos como el destinatario, la dirección postal, el código postal, la ciudad, el país, etc. El sistema denominado Interpretación de Direcciones Manuscritas, *HandWritten Address Interpretation* (HWAI) no sólo se centra en el reconocimiento, sino que va un paso más allá e intenta una interpretación en base al conocimiento de las direcciones registradas en su dominio. De esta manera, la clasificación de cartas se realiza igualmente aun faltando datos imprescindibles para que las cartas lleguen a sus destinatarios [152]. Uno de los principales problemas con los que se suele encontrar este sistema es la distinción entre la letra escrita a mano y la impresa ya que a veces puede estar incluso combinada. Por otro lado, cierto es que cuando se reconoce el código postal no es necesario realizar ningún otro reconocimiento sobre la carta. Cuando esto no es posible o existe ambigüedad en la lectura del código postal por posible confusión en los números, es necesario reconocer el nombre de la ciudad y el problema pasa de un reconocimiento numérico a un reconocimiento de palabras manuscritas. Aparte del sistema HWAI, existen otras múltiples implementaciones interesantes con otras perspectivas a este mismo problema [43, 75, 82, 87].

Otra de las perspectivas importantes es también la de reconocimiento de números poniendo especial interés en el reconocimiento de cheques bancarios. Por lo general, los cheques tienen el fondo coloreado cumpliendo patrones complejos. Además, la posición en la que aparecen los campos impresos puede variar drásticamente dependiendo de la compañía [51]. Por norma general, los datos a reconocer de un cheque bancario son: la cantidad escrita en letra, la cantidad escrita en número, la fecha y la firma [39]. En un paso previo, se elimina aquella información no útil aplicando una selección por regiones [95]. Los métodos utilizados en la clasificación y reconocimiento son muy variados, destacando de entre todos ellos aquellos basados en HMM [60, 64, 115] junto con los modelos basados en redes neuronales [63, 118, 164].

2.3.1 Características

De manera global, se podría dividir el reconocimiento *Off-line* en cinco etapas como se muestra en la imagen 2.4. Las características de este tipo de reconocimiento se describen en mayor detalle en cada uno apartados siguientes que se corresponden con cada una de las fases.

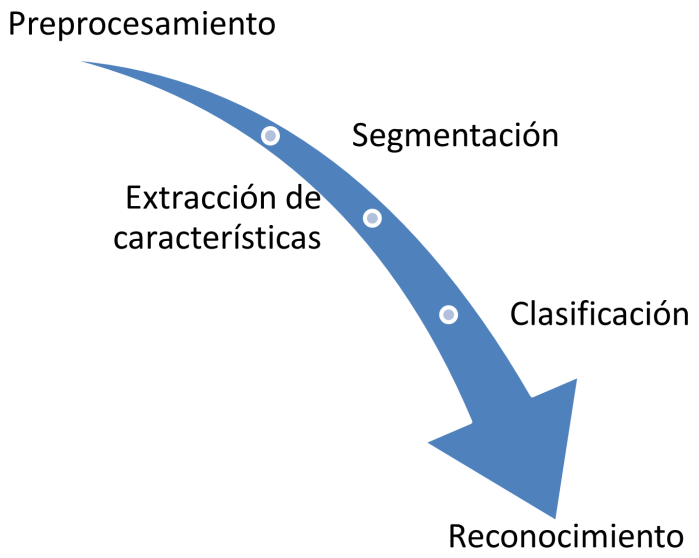


Figura 2.4: Esquema general de las etapas involucradas en el reconocimiento *Off-line*.

2.3.2 Preprocesamiento

El preprocesamiento es la etapa por la cual, una vez obtenida la imagen a través de un escáner o cualquier otro medio óptico, se prepara la imagen para ser tratada posteriormente de forma normalizada. Este paso es independiente del tipo de reconocimiento que se realice ya que los algoritmos usados en el preprocesamiento pueden ser compartidos por otros tipos de reconocimientos.

Algunas de las operaciones que con frecuencia se emplean antes de cualquier reconocimiento se describen a continuación [158]:

Corrección de umbral Este mecanismo convierte una imagen en color o escala de grises en una imagen en blanco y negro. La finalidad es extraer la parte que se desea reconocer del resto de elementos que se encuentran en la imagen [138]. En el histograma de una imagen existen dos puntos importantes: el punto más alto que indica el color claro lo más próximo al blanco posible que corresponde al fondo de la imagen y el punto más bajo que corresponde al color del texto que se desea reconocer lo más oscuro posible cercano al negro. Normalmente se suele buscar el valor óptimo de umbral situado entre estos dos puntos anteriormente mencionados. Existen multitud de métodos para la corrección del umbral [96, 116, 119], pero todos tienen un elemento en común: adaptar y preparar la imagen de entrada para las siguientes etapas.

Reducción de ruido El objetivo es extraer el material principal de la imagen y separarlo de otros elementos que puedan aportar ruido como puede ser la textura de fondo, el granulado o cualquier otro tipo de trazos o puntos que no tengan nada que ver con el texto que se desea reconocer. Como éstos pueden ser causados por el medio que las ha capturado, es habitual el uso de "filtros" para mejorar la calidad de la imagen basados en operaciones morfológicas como puede ser la apertura o cierre u operaciones de adelgazamiento (*Thinning*) [86] o cualquier otro tipo de operaciones [112, 151].

Normalización La corrección de efectos en el texto que surgen de la forma de escritura como por ejemplo, la corrección de la línea base, inclinación de los caracteres, etc, es otra práctica también habitual. Por lo general, el texto se escribe realizando movimientos ascendentes y descendentes respecto a la horizontal, pero rara vez estos movimientos se realizan de forma vertical sin tener un grado de inclinación. Existe por lo tanto, la necesidad de una corrección tanto del ángulo de la

línea base como de la inclinación de la propia palabra o carácter. El algoritmo utilizado para la corrección de la línea base, de entre las múltiples implementaciones [26, 99, 111, 159], suele basarse en girar la imagen hasta conseguir que la línea base del texto quede completamente alineada con la horizontal [20]. Por otro lado, en la corrección de la inclinación de las palabras o de las letras, lo que el algoritmo pretende es eliminar el ángulo existente en el carácter inclinado y modificarlo para que quede lo más vertical posible. Existe también una amplia literatura con la implementación de este tipo de algoritmo [20, 26, 37, 47, 77, 145, 158].

2.3.3 Segmentación

El objetivo de la segmentación es conseguir unidades básicas de información que servirán para el proceso de reconocimiento. Existen tres niveles de segmentación. En un primer nivel, partiendo de un texto, se considera obtener las líneas que lo componen. En un segundo nivel, lo que se trata es de obtener las palabras a partir de la línea y por último, en un tercer nivel, la obtención de letras a partir de las palabras.

Para la segmentación del texto en líneas, lo habitual en un documento impreso es hacer una proyección horizontal [150]; sin embargo, esta tarea es más compleja en los documentos manuscritos ya que los trazos, al escribir, no se definen de una forma uniforme. Existen métodos que se basan en obtener los mínimos locales para cada una de las filas para trazar así, una línea imaginaria por donde poder segmentar [81].

Para el segundo nivel de segmentación, la mayoría de las perspectivas se basan en encontrar los huecos que se dejan entre las palabras [104, 142] ya que suelen ser mayores que los que se dejan entre las propias letras si éstas se escribieran separadas.

Por último, y el más complejo de todos, es el de la segmentación de palabras en letras [31]. En la literatura, se considera la segmentación *explícita* e *implícita* dependiendo de si las unidades básicas se esperan que sean letras o no. En la explícita, no existe un algoritmo en concreto para esta tarea que lo haga con un alto grado de precisión [30, 97]. La paradoja de Sayre [139] describe que una letra no puede ser segmentada antes de que el texto que la contiene sea reconocido y al revés, que un texto no puede ser reconocido hasta que no se reconozca cada una de sus letras. Sin embargo, la segmentación explícita es la más usada ya que si se produce una sobresegmentación, no repercute en el reconocimiento final y admite soluciones para la clasificación. Las características que se buscan en la segmentación de palabras en letras suelen basarse en buscar puntos de ligadura o curvaturas cóncavas [80] así como la distinción de la altura en los trazos.

2.3.4 Extracción de las características

En este proceso se aplican algoritmos en los que se extraen aquellas características que sirven para definir una determinada unidad básica previamente segmentada. Se pueden clasificar en 3 grupos dependiendo si las características se extraen de la palabra (alto nivel), de las letras (nivel medio) o de alguna parte de la letra (nivel bajo).

En cuanto a las características de alto nivel, son aquellas que se extraen de la palabra como un conjunto. Estas características pueden ser bucles, trazos ascendentes o descendentes de las palabras. Este tipo de características se pueden aprovechar para encontrar en la palabra trazos de unión, trazos de finalización, barras de "t" y puntos en el texto. Suelen basarse en la detección de elementos estructurales así que tienen poca dependencia con el estilo de la escritura. Para simplificar la búsqueda, se suele hacer sobre palabras en las que se les ha aplicado ya un proceso para obtener su esqueleto [24, 144, 145].

Las características de nivel medio se extraen a partir de la letra. El principal problema que existe es la gran cantidad de formas posibles [28]. Además, aquellos sistemas que implementan un reconocimiento de caracteres tienen que hacer frente a formas que no son en sí una letra. Uno de los métodos propuestos para este fin se basa en el descarte de aquellos caracteres que tienen demasiados trazos ascendentes o descendentes. Aparte de este método, existen muchos otros que consideran, por ejemplo, la distribución de la transición entre el fondo frente al texto [121], la extracción de características basados en trazos [56] o la vectorización de los contornos [121]. Aunque sin duda, el método por excelencia es el método de zonificación. A grandes rasgos, este método consiste en la superposición de una rejilla sobre el carácter que se desea reconocer y extraer como característica las regiones afectadas tal y como muestra el ejemplo de la figura 2.5. En sus múltiples implementaciones se podrían clasificar en estáticas, si no varían durante el proceso de entrenamiento, o dinámicas si se van adaptando a lo largo del tiempo utilizando algoritmos como el de Voronoi [71].

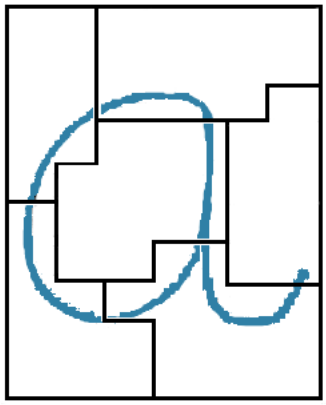


Figura 2.5: Ejemplo zonificación aplicada sobre la letra *a*.

Las características de bajo nivel se extraen de fragmentos de letras. Éstas pueden ser elementos geométricos básicos como líneas, curvas, trazos, etc [20, 45, 120]. En algunos casos, también se suele considerar la posición de los puntos [57, 108] así como las regiones ascendentes y descendentes [23]. Para una visión más genérica

de la forma, se consideran también la curvatura [24], el centro de masas y los histogramas de la dirección de los trazos [83, 144, 145].

2.3.5 Clasificación y reconocimiento

La clasificación y el reconocimiento son procesos estrechamente relacionados. Por ello, en este apartado se detallan ambas etapas de manera conjunta.

El proceso de reconocimiento consiste en encontrar una unidad léxica, ya sea palabra o letra, que mejor se adapte con las características encontradas. Por lo general, se usa el conjunto de características extraídas de la propia palabra que se desea reconocer y se calcula un valor de coincidencia (puede ser una probabilidad, distancia o coste) con la posible interpretación que se le puede dar. Finalmente, el mejor de los valores de entre las coincidencias encontradas, se asume que es la correcta interpretación de la unidad léxica de entrada.

Uno de los modelos que se suele seguir es el de la ***programación dinámica*** [13, 57, 130]. En él, se intenta encontrar la mejor relación entre los M fragmentos obtenidos en la segmentación con las N letras que componen la palabra. Para ello, se construye dinámicamente un grafo dirigido que va creciendo en función de los elementos segmentados. A cada uno de los nodos, mediante un entrenamiento previo, se le asigna un coste de manera que cuando se intenta reconocer una palabra, se busca el mejor camino con el menor coste para estimar así una solución.

Al igual que en el reconocimiento *On-line*, se suelen usar también técnicas basadas en los ***modelos ocultos de Markov*** [3, 72]. En este caso, existe un conjunto finito de estados en el que las transiciones entre ellos se realizan en función de una determinada probabilidad. Además, estas transiciones no dependen ni varían en función de un tiempo t como sucedía en el reconocimiento *On-line*. Se utilizan algo-

ritmos como el de Baum-Welch (o alguna variante) [9–12] en el entrenamiento y de Viterbi para dar forma al modelo [50, 160]. En este modelo, la capa oculta corresponde con las letras de la posible interpretación mientras que la capa visible se construye a partir de los conjuntos de características obtenidos en la segmentación. Finalmente, la interpretación que más se aproxime a un conjunto de características de entrada, corresponderá con la unidad léxica que se está buscando. La principal ventaja de los HMM es que son modelos fácilmente escalables ya que si se desea añadir nuevas entradas, el modelo no necesita ser entrenado de nuevo.

Algunas de las técnicas se centran más en un enfoque ***holístico*** en el que se consideran las características de la palabra como un todo en el reconocimiento. Algunos de estos métodos, convierten la palabra en una cadena que es comparada con el léxico usando la métrica de Levehntein para calcular la unidad léxica más parecida [73, 110]. Esto suele funcionar cuando se hace para palabras pequeñas que tienen un número reducido de características. Sin embargo, cuando el número de éstas crece, el problema se acentúa ya que también crece el número de palabras con las que hay que comparar produciendo una clasificación más compleja y por lo tanto una probabilidad mayor de error. Es por esto que este método de clasificación rara vez se utiliza para el reconocimiento de palabras y se emplea en escenarios más acotados, como por ejemplo en la transcripción de direcciones de correos en cartas (se busca la correcta relación entre el código postal y el nombre de la ciudad) [35, 102, 103].

Existen también otros métodos no paramétricos basados en el algoritmo ***k*-Nearest Neighbor, *k-Nearest Neighbor* (KNN)** [53, 135]. Su uso es bastante frecuente ya que se trata de un algoritmo sencillo de implementar obteniendo a su vez, buen rendimiento. Además, no requiere de un conjunto grande de entrenamiento para conseguir unas tasas de reconocimiento más que aceptables. Es común evaluar las distancias entre cada uno de los elementos adyacentes mediante el algoritmo de Euclides. Para evitar fallos en la clasificación, algunos autores realizan un paso previo de comparación entre la muestra de entrenamiento y el conjunto global para

conseguir así una clasificación por subconjuntos [129] aunque no suele dar buenos resultados si no sigue una distribución normal [5].

Las metodologías basadas en **Redes Neuronales, *Neural Network* (NN)** son también ampliamente aceptadas debido a su flexibilidad y a la multitud de implementaciones que permite. Se muestra un ejemplo de NN en la figura 2.6. En ellas, las características extraídas en los pasos anteriores son utilizadas como fuente de entrada para la fase de entrenamiento de la red. Se dice así que la red tiene la capacidad de *aprender* de los patrones introducidos para aplicarlos en los nuevos que se introduzcan en la fase de reconocimiento. La Red Neuronal con Retropropagación, *Feedforward Back Propagation Neural Network* (FFBPNN) es de las más comunes, en la que los pesos de los nodos de la red son recalculados y compensados para llegar a un reconocimiento más efectivo [65, 87, 90, 114]. Existen otras arquitecturas como las redes neuronales en el que las múltiples capas están conectadas por MLP. Los perceptrones pueden tener una conexión total o parcial a otros nodos configurando así una red balanceada cuyo perceptrón final refleja la solución [21, 106]. Otras arquitecturas basadas en redes neuronales como por ejemplo, las TDNN [27, 91], las Redes Neuronales Recurrentes, *Recurrent Neural Network* (RNN) [5, 145], o incluso las redes Neocognitron [54, 55] se utilizan también con bastante frecuencia en la etapa de clasificación y posterior reconocimiento.

Para el reconocimiento de caracteres numéricos existen otros sistemas de clasificación basados en las redes SOM comentadas anteriormente. Las redes Mapas Auto-Organizados Jerarquizados, *Growing Hierarchical Self-organized Maps* (GH-SOM) son introducidas por L.Bezerra et al. [15] las cuales exponen la posibilidad de determinar la profundidad y el tamaño de la jerarquía de una red SOM durante el proceso de entrenamiento no supervisado. Esto da la capacidad de expansión dinámica de la red en función de los datos de entrada. Basándose en esta teoría, H. Shah-Hosseini [146] plantea una perspectiva con árboles binarios denominados Mapas Auto-Organizados en árboles binarios, *Binary Tree Time Adaptative Self-*

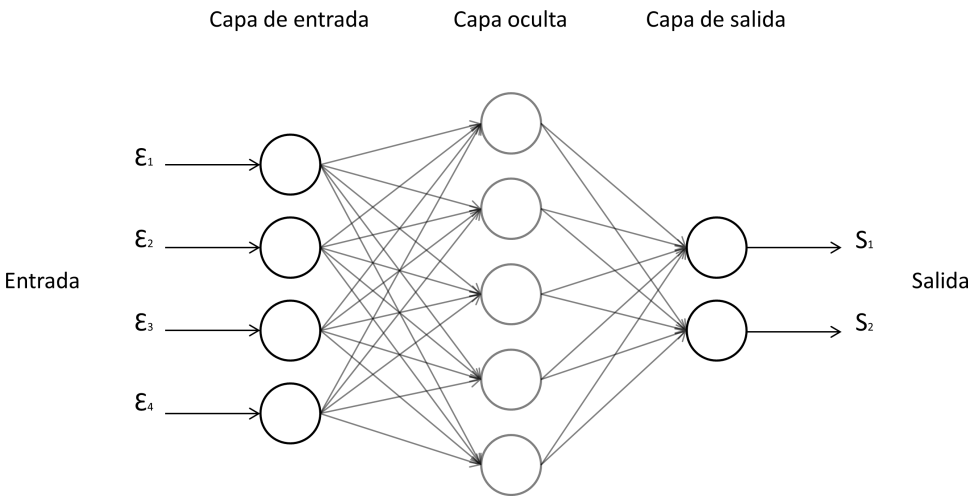


Figura 2.6: Ejemplo de una red neuronal. Normalmente, las redes neuronales se componen de una capa de entrada, una capa oculta y una capa de salida.

organized Maps (BTASOM) donde la profundidad del árbol y el número de hijos se introducen dinámicamente durante la fase de entrenamiento.

Cabe destacar que algunos de estos autores, después de haber realizado un reconocimiento, aplican una fase de postprocesamiento con el objetivo de obtener la mejor interpretación en caso de que se obtengan varias posibles soluciones. El método consiste en dar una puntuación a los posibles candidatos y escogerlo en función del contexto de la frase en la que se encuentre la palabra o de un diccionario en caso de que se trate de un reconocimiento de palabras aisladas. Estos métodos normalmente se usan en los reconocimientos por líneas de texto donde el contexto de la frase se puede utilizar como un valor añadido ya que puede aportar información que no es posible obtener en otros ámbitos [48, 155].

Es habitual encontrar sistemas que utilizan también métodos híbridos como una combinación de los que se han visto, para fortalecer la arquitectura de reconocimiento. Es muy común encontrar sistemas que combinan redes neuronales con HMM [18], HMM con lógica difusa [132] o redes neuronales junto con ló-

gica difusa [66], etc.

Otros enfoques basados en ***gramáticas regulares*** como el que describe Kyoon Ha Lee et al. [89] se usan también con frecuencia. En su estudio consideran que un carácter está formado por una sucesión de trazos, y que dependiendo de la posición y la curvatura de los mismos, pueden generar una serie de atributos dependientes. Después, mediante una gramática regular se puede reconocer un carácter atendiendo a un conjunto de producciones y reglas previamente preestablecidas. Estas metodologías sirven de base para otros sistemas más complejos.

Sin embargo, y desde siempre, existen otros desarrollos enfocados en ***sistemas expertos*** que al igual que muchos modelos, utilizan métodos propios para alcanzar el reconocimiento. Normalmente, utilizan técnicas de preprocesamiento similares a otros modelos pero crean su propia forma de clasificar las características a través de un conjunto de reglas propio. Estas características se extraen de un conjunto de entrenamiento que sirve para crear las reglas propias para el sistema en cuestión.

Subba Reddy et al. [133] utilizan un sistema de zonificación para la extracción de las características basado en una rejilla de $15x15$ que divide en tres regiones horizontales y tres verticales para obtener la posición de los píxeles de la imagen. Después, estas características son clasificadas en una combinación de redes SOM y Learning Vector Quantization,*Learning Vector Quantization* (LVQ) para obtener un sistema de reconocimiento. Si durante el proceso de reconocimiento se produce algún conflicto, interviene el sistema experto diseñado que resuelve el conflicto en base al conocimiento adquirido en la fase de entrenamiento.

Otro trabajo de interés es el presentado por Jiří Šíma [100] en el que utiliza una red neuronal para construir la base de conocimiento. Después, introduce un sistema experto en el que, junto con la base de conocimiento entrenada, utiliza un enfoque heurístico previamente configurable para trabajar con aquella información

no catalogada para alcanzar así la solución al reconocimiento.

Por último, Maher Ahmed et al. [1] diseñan un sistema experto para reconocer tanto caracteres alfabéticos como numéricos o cualquier otro tipo de símbolo manuscrito. Identifica los trazos y la dirección que toman los mismos como característica principal. Después crea una base de conocimiento a partir de símbolos conocidos utilizando esta característica. Para el reconocimiento, simplemente compara los nuevos símbolos con los almacenados en la base de conocimiento y así poder estimar una posible solución.

"Primero tienes que aprender las reglas del juego, y después jugar mejor que nadie".

– Albert Einstein

3

Metodología empleada

En este capítulo se describe la metodología seguida para desarrollar este trabajo: diseñar e implementar un sistema experto para el reconocimiento *Off-line* de textos manuscritos. Para ello, esta metodología se divide en una serie de etapas en la que el resultado de cada una de ellas sirve como entrada a la siguiente. En cada sección de este capítulo se detalla en mayor medida cuáles son estas etapas y cómo interactúan entre sí para alcanzar finalmente el reconocimiento.

3.1 Introducción

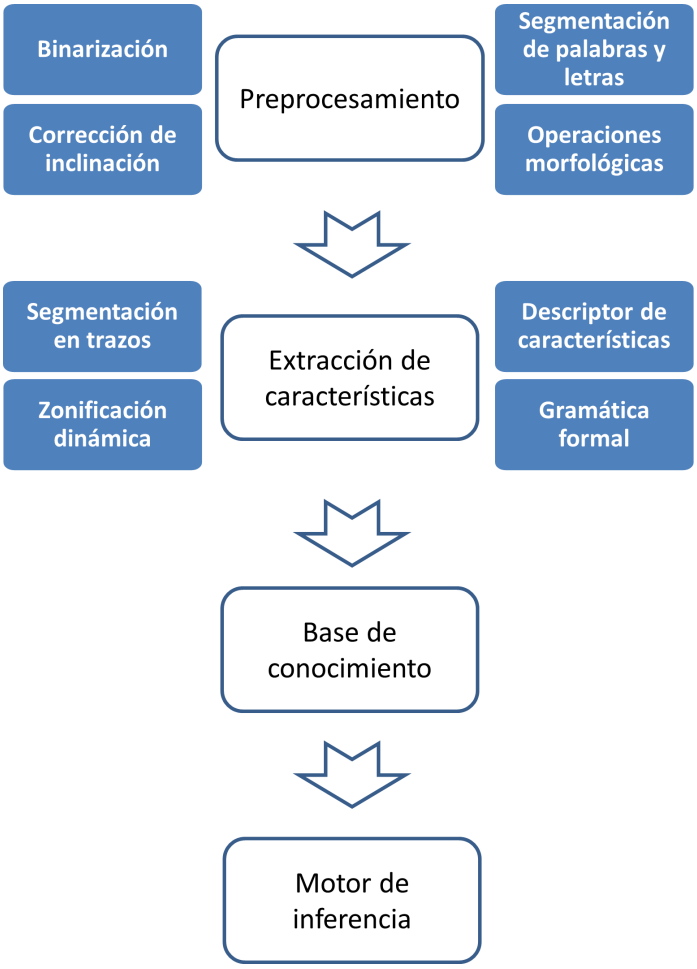


Figura 3.1: Esquema de las etapas que intervienen en el reconocimiento.

La figura 3.1 muestra de forma general el conjunto de etapas que intervienen en el proceso de reconocimiento. Partiendo de las etapas de preprocesamiento y segmentación de palabras y letras, que suelen ser común en todos los sistemas de reconocimiento, este trabajo también se centra en la extracción de las características y en el sistema de clasificación de las mismas. Para ello, se parte de la idea en la

que las letras se pueden descomponer en trazos verticales y horizontales. De esta manera, aplicando zonificación dinámica – que consiste en aplicar una rejilla en cada uno de los trazos verticales que componen una letra– se puede identificar la posición donde los trazos verticales están unidos con los trazos horizontales tal y como proponen A. Alonso et al. [4]. Por lo tanto, el descriptor de características de dicha letra estará formado por una cadena en la que se identifican los trazos verticales que la componen junto con las posiciones en las que se adhiere uno o más trazos horizontales. Esta cadena sigue las reglas de una gramática regular y se valida mediante un autómata finito siendo esta la primera de las aportaciones de esta tesis. Más tarde, en otra de las contribuciones de este trabajo se detalla como estas cadenas son guardadas en una base de conocimiento con una estructura basada en árbol cuyos nodos son cada uno de los elementos que componen la cadena de descriptores y las hojas, la representación de la propia letra en sí. Finalmente, otra de las contribuciones se señala en la etapa de reconocimiento donde el Motor de Inferencia busca nuevas letras en la base de conocimiento considerando los pesos de las hojas del árbol y los resultados obtenidos se comparan frente a un diccionario para estimar la mejor solución.

Así pues, se parte de un documento manuscrito previamente escaneado con el fin de conseguir una imagen que representa el texto que se pretende reconocer. Después de aplicar filtros y correcciones en la etapa de preprocesamiento como se describe en 3.2, se segmenta el texto siguiendo el algoritmo expuesto en la sección 3.2.3 con el fin de obtener inicialmente las palabras y después las letras. Acto seguido, en la sección 3.3 se detalla cómo se realiza la extracción de las características con el propósito de obtener una representación única para cada tipo de letra y poder así realizar una clasificación eficiente en una base de conocimiento como se analiza en la sección 3.4. Asimismo, en la sección 3.5 se describe cómo el Motor de Inferencia realiza búsquedas de nuevas letras sobre los patrones de características que se han almacenado previamente en la base de conocimiento mediante entrenamiento para, finalmente, estimar la solución más óptima al proceso de reconocimiento.

3.2 Preprocesamiento

Esta etapa tiene por objetivo reducir las variaciones que se puedan producir en la escritura y obtener una entrada lo mas uniforme y normalizada posible. Estas variaciones suelen estar ligadas a la forma de escribir del propio autor y suelen interferir de forma negativa en el reconocimiento. Una adecuada fase de preprocesamiento puede mejorar considerablemente los resultados [16]. Por eso, se realizan las operaciones que se describen en los siguientes apartados con el fin de minimizar el efecto de factores externos en la fase de reconocimiento.

3.2.1 Binarización

Tras obtener una representación del texto en formato de imagen, se realiza un filtrado de umbral siguiendo el método de *Otsu* [116] en el que se convierte la imagen en color en una imagen binaria en la que sólo se tienen en cuenta el color blanco y el negro. Si se considera que el texto está escrito de un color más oscuro que el fondo del papel, esta operación permite separar el texto del resto de elementos. Por lo tanto, dada la ecuación de umbral 3.1, se debe buscar un valor *thres* que optimice dicha ecuación para obtener la mejor imagen binarizada. Nótese que $Im(x,y)$ corresponde al píxel x,y de la imagen de entrada.

$$Im(x,y) = \begin{cases} 1 & (x,y) \geq thres \\ 0 & \text{en otro caso} \end{cases} \tag{3.1}$$

Una vez realizada esta operación, es posible representar la imagen como una matriz de dos dimensiones donde el tamaño de dicha matriz corresponde con las dimensiones de la propia imagen. De esta manera, los elementos de la matriz $M(x,y)$

toman los valores obtenidos en el proceso de binarización a partir de la ecuación anterior.

3.2.2 Operación de apertura

Aquellos pequeños poros o líneas recogidos por el dispositivo en el proceso de obtención de la imagen que no son objeto de reconocimiento, se eliminan utilizando una operación morfológica de *apertura*. Dicha operación combina la operación de *erosión* seguida de la operación de *dilatación* para conseguir eliminar aquellos pequeños puntos que sean más pequeños que el elemento estructurante 3.2.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.2}$$

La operación de erosión consigue un efecto de reducción sobre la imagen de entrada. Consiste en una transformación morfológica que combina dos vectores (en este caso la matriz original y el elemento estructurante) aplicando sobre ellos una operación de sustracción de elementos [67]. Si se denota como A la matriz que contiene la representación de la imagen original, SE el elemento estructurante 3.2 y B como la matriz resultante de aplicar SE sobre A dentro de un espacio de 2 dimensiones Z^2, se podría definir la operación de erosión como:

$$A \ominus B = \{x \in Z^2 \mid \quad para\ cada \quad b \in B \quad \exists \quad a \in A \quad tal\ que \quad x = a - b\} \tag{3.3}$$

A efectos prácticos, esto se consigue desplazando el elemento estructurante *SE* a lo largo de la matriz *A* aplicando en el punto *x* la operación de sustracción obteniendo como resultado la operación de erosión en la matriz *B*.

En la operación de dilatación, se consigue el efecto contrario al de erosión, es decir, un efecto de expansión sobre la imagen de entrada. Realiza una transformación morfológica que nuevamente combina dos vectores aplicando sobre ellos una operación de adición. Por lo tanto, se podría definir la operación de dilatación como:

$$A \oplus B = \{c \in Z^2 \mid \quad c = a + b \quad \textit{para cada} \quad a \in A \quad y \quad b \in B\} \qquad (3.4)$$

Como en el caso de la operación de erosión, esto se consigue desplazando el elemento estructurante *SE* y aplicando la adición a lo largo de la matriz *A* en el punto *c* obteniendo una matriz *B* como resultado de la operación de dilatación.

3.2.3 Segmentación del texto en sus líneas

Para obtener las unidades básicas de reconocimiento que se necesita en las fases posteriores, es necesario segmentar el texto en cada una de las líneas que la componen. De cada una de ellas se obtienen las palabras que finalmente serán divididas tomando como unidad básica las letras que la forman.

Para la segmentación de texto en cada una de las líneas, se utiliza un algoritmo basado en proyecciones verticales como el que propone A. Zahour et al. [169]. En este caso, para adaptar al texto, se realiza una proyección horizontal (a lo largo del eje y o lo que es lo mismo, de la altura de la imagen) donde la superficie de cada barra es proporcional a la frecuencia de aparición representadas de forma acumulada. Sólo se tendrán en consideración para hacer esta acumulación aquellos píxeles categorizados como 1 en el proceso de binarización.

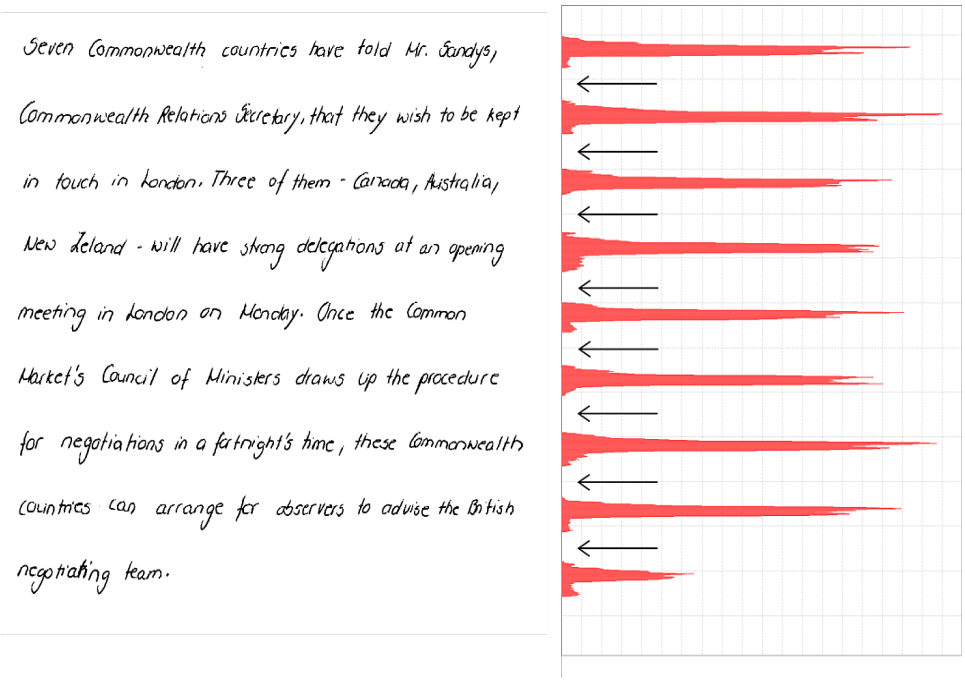


Figura 3.2: Proyección horizontal de un texto manuscrito. Las flechas indican las zonas de segmentación que corresponden con los huecos existentes entre las líneas.

Por último, se realiza la segmentación del texto en líneas usando aquellas zonas donde existan huecos en la proyección ya que la acumulación es 0 y son regiones en las que no se encuentra ningún posible elemento a reconocer como muestra la figura 3.2.

3.2.4 Corrección de inclinación

La corrección de inclinación es también un proceso común en la etapa de preprocesamiento. En este caso, se realiza a nivel de líneas de texto y se aplica haciendo uso del algoritmo de Changming Sun et al. [153].

Nuevamente, se parte de la matriz $M(x,y)$ conteniendo la representación de la línea de texto. Entonces, se trata de buscar un ángulo a en dicha matriz que logre deshacer la inclinación en base a las siguientes reglas:

1. El ángulo a considerado debe estar dentro de un rango de valores a acotado por $a \in [-\frac{\pi}{4}, \frac{\pi}{4}]$.
2. Para cada píxel de la matriz se calcula su valor h.
3. Se calcula la distancia a mover cada píxel de la matriz M en base a la ecuación 3.5 tal y como se indica en la figura 3.3.

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} 1 & -tg(a) \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{3.5}$$

4. El píxel en la posición h toma el valor 0 y el píxel en la posición $h \pm M$ toma el valor 1.
5. Se calcula la proyección vertical de la matriz resultante (ver figura 3.4).

Para encontrar el ángulo que mejor se ajusta al proceso de corrección de inclinación, se calcula la proyección vertical de cada matriz M_a obtenida para cada uno de los ángulos en el rango. Aquella proyección vertical que contenga el pico más

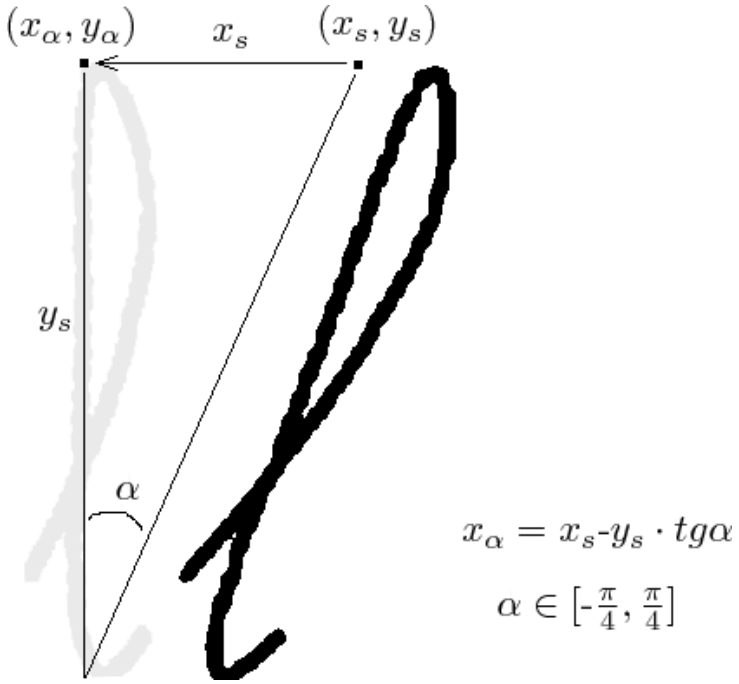


Figura 3.3: Corrección de inclinación para la letra 'l'. El proceso lleva a que todos los píxeles de la letra original cambien de posición aplicando la ecuación señalada.

alto, corresponderá con la matriz con el ángulo óptimo para la corrección de inclinación y, por consiguiente, la matriz con la línea de texto con el ángulo corregido. Se muestra un ejemplo en la figura 3.5.

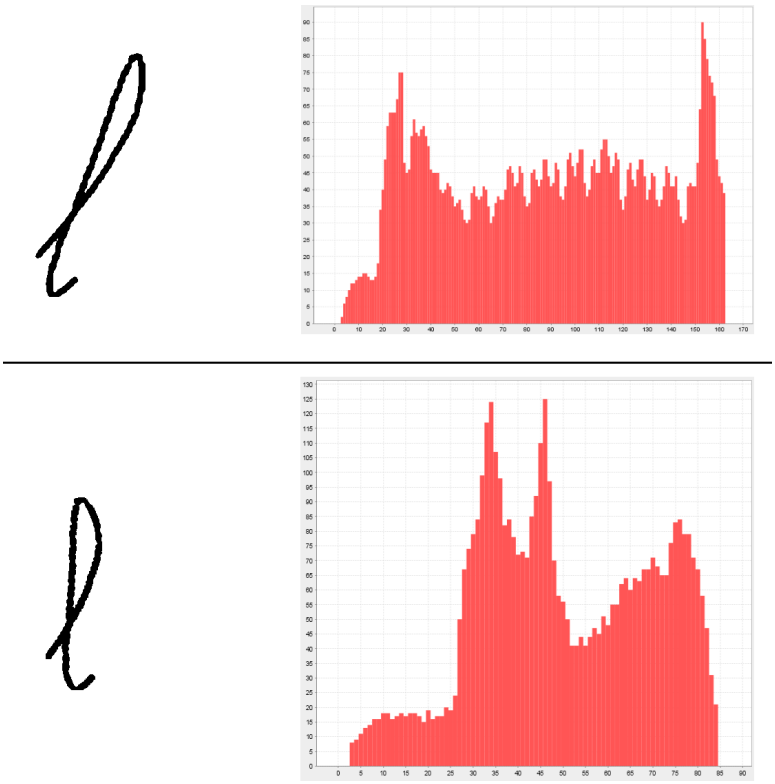


Figura 3.4: Proyecciones verticales de la letra 'l'. En la parte superior, se representa la proyección vertical para la letra en la que se aprecia que el pico más alto corresponde a una acumulación de 90 píxeles. Sin embargo, en la parte inferior, después del proceso de corrección, se puede apreciar que para la misma letra se obtiene que el pico más alto corresponde con la acumulación de 125 píxeles.

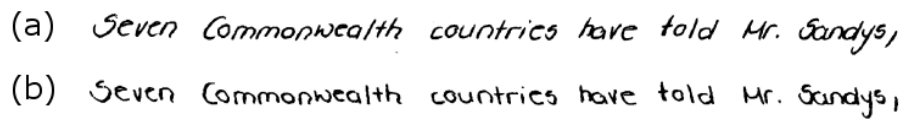


Figura 3.5: (a): Línea de texto original. (b): Línea de texto tras aplicar el proceso de corrección de inclinación.

3.2.5 Operación de cierre

Como consecuencia del movimiento de píxeles en el proceso de corrección de inclinación, aparece un efecto de granulado en el texto tal y como se puede apreciar en la figura 3.6.

Figura 3.6: Efecto de granulado tras la corrección de inclinación.

Para corregir estas imperfecciones, se aplica la operación morfológica de cierre que consiste en realizar primero una operación de dilatación seguida de una operación de erosión. Esto permite que se rellenen los pequeños agujeros así como estrechas fisuras que hayan podido aparecer durante el proceso de corrección de inclinación. Si a la imagen inicial A se le aplica el elemento estructurante SE 3.2 para obtener la imagen resultante B, se podría definir la operación de cierre como:

$$(A \oplus B) \ominus B \tag{3.6}$$

El resultado de aplicar la operación de cierre sobre la imagen anterior 3.6 daría como resultado la imagen 3.7.

Figura 3.7: Imagen a la que se le ha aplicado la operación de cierre.

3.2.6 Operación de adelgazamiento

El proceso de adelgazamiento permite reducir los trazos que componen una imagen en líneas que sirven como base para definir el propio trazo. Esto permite que el algoritmo pueda utilizarse para cualquier escritor ya que sea cual sea el grosor de su trazo, con este proceso se obtiene siempre la línea de definición del trazo con el mismo grosor, es decir, de 1 píxel. Para ello, se aplica el algoritmo de Zhang T Y, Suen C Y. [170] en el que se alcanza el adelgazamiento mediante un conjunto de iteraciones.

El algoritmo parte de una ventana de tres filas por tres columnas $(3x3)$ en la que cada píxel tiene 8 píxeles adyacentes. Dado un píxel (i,j), la siguiente tabla 3.1 denota la distribución de los píxeles colindantes:

Tabla 3.1: Distribución de píxeles adyacentes a un píxel dado.

P9 = (i-1 , j-1)	P2 = (i-1 , j)	P3 = (i-1 , j+1)
P8 = (i , j-1)	P1 = (i , j)	P4 = (i , j+1)
P7 = (i+1 , j-1)	P6 = (i+1 , j)	P5 = (i+1 , j+1)

Así pues, en cada una de estas iteraciones se realizan dos subiteraciones en las que se evalúa cada uno de los píxeles con sus adyacentes en base a una serie de condiciones. Si éstas se cumplen, entonces se marca el píxel en cuestión para ser borrado ya que no se trata de un píxel candidato para formar parte de la línea des-

criptiva del trazo.

En la primera subiteración, para cada píxel se evalúan las siguientes cuatro condiciones:

1. $2 \leq B(P1) \leq 6$
2. $A(P1) = 1$
3. $P2 * P4 * P6 = 0$
4. $P4 * P6 * P8 = 0$

Donde:

- $A(P1)$ es el número de transiciones de puntos no marcados a puntos marcados (transiciones "01") en la secuencia $P2$,$P3$,$P4$,$P5$,$P6$,$P7$,$P8$,$P9$,$P2$ (Se añade un $P2$ extra por tratarse de una secuencia circular).
- $B(P1)$ es el número de vecinos marcados de P1.

Si se cumpliera alguna de ellas, se marcaría dicho píxel para ser eliminado al finalizar la subiteración.

En la segunda subiteración, se toma como entrada el resultado de la primera subiteración evaluando para cada uno de los píxeles las siguientes cuatro condiciones:

1. $2 \leq B(P1) \leq 6$

2. $A(P1) = 1$

3. $P2 * P4 * P8 = 0$

4. $P2 * P6 * P8 = 0$

Al igual que en la primera subiteración, si se cumpliera alguna de las condiciones, se marcaría dicho píxel para ser eliminado al finalizar la subiteración.

El proceso continúa hasta que no se realiza ninguna modificación en alguna de las subiteraciones.

La figura 3.8 muestra el proceso de adelgazamiento tomando como entrada la imagen obtenida en el proceso anterior.

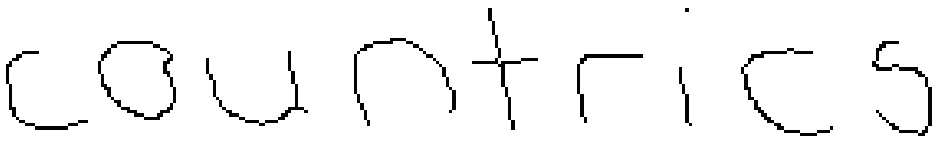


Figura 3.8: Imagen a la que se le ha aplicado la operación adelgazamiento.

3.2.7 Operación de dilatación

El siguiente paso consiste en realizar una operación de dilatación sobre las líneas obtenidas en el proceso de adelgazamiento. Esto hará que los trazos obtenidos a partir de cualquier escritor sea cual sea su forma de escribir, tengan el mismo grosor. Para el siguiente proceso de extracción de características, se requiere que el grosor del trazo sea de al menos 3 píxeles de ancho. Por lo tanto, la operación que se aplica es la definida ya en la ecuación 3.4 con el elemento estructurante *SE* 3.2.

La figura 3.9 muestra el resultado de aplicar la operación de dilatación sobre el proceso de adelgazamiento.

Figura 3.9: Imagen a la que se le ha aplicado la operación de dilatación.

3.2.8 Segmentación de palabras

La segmentación de palabras permite obtener a partir de las líneas de texto las palabras de manera unitaria. El proceso es similar al visto en el apartado 3.2.3 pero en este caso, se realiza una proyección vertical para identificar los huecos que existen entre las palabras. Puesto que las palabras también contienen huecos que separan a las letras, estos huecos también son identificados. Para diferenciar unos huecos de otros, se utiliza como discriminante la media del tamaño de los huecos ya que es más frecuente que el tamaño entre palabras sea notablemente mayor que el existente entre las letras de una palabra. De esta manera, sólo se tendrán en cuenta aquellos huecos que sean mayores que los de la media y como consecuencia, los huecos entre palabras.

La figura 3.10 muestra el proceso sobre una línea de texto.

Una vez identificados los huecos, se realiza la obtención de las palabras a partir de la línea de texto realizando secciones en los huecos encontrados.

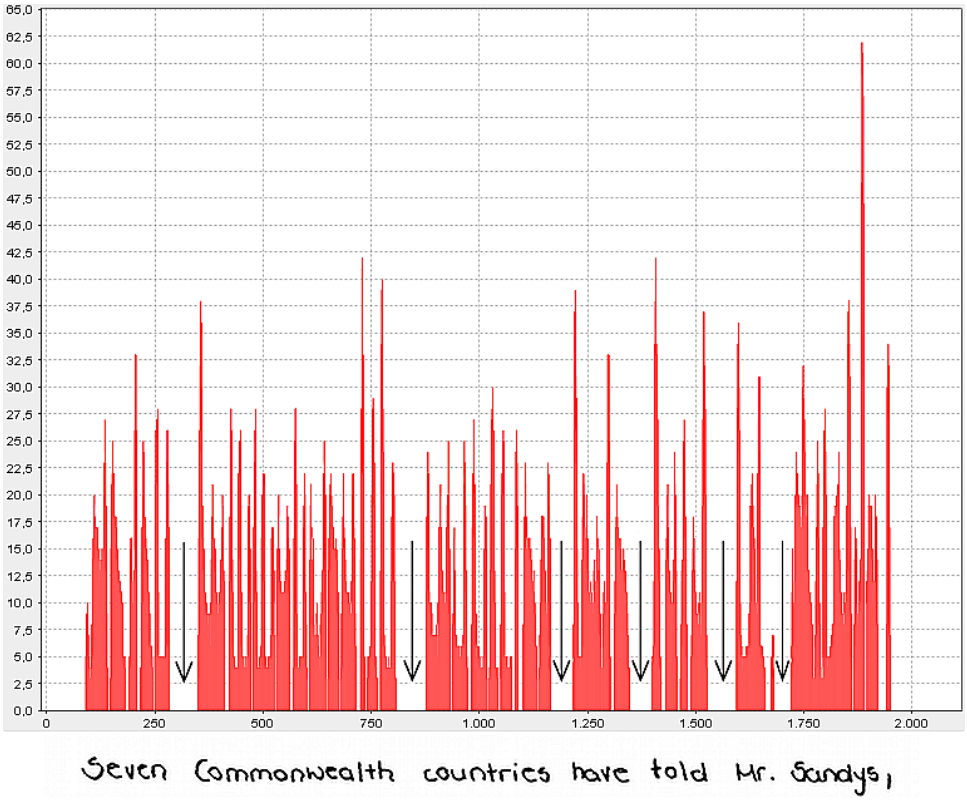


Figura 3.10: Obtención de las palabras a partir de las líneas usando su proyección vertical para encontrar los huecos que separan las palabras.

3.2.9 Extracción de las letras de las palabras

Una vez más es de utilidad el algoritmo de A. Zahour et al. [169] para obtener las letras a partir de las palabras. Las letras van a ser consideradas como la unidad básica de la que se van a extraer las características que va a usar la base de conocimiento para realizar el reconocimiento.

Al igual que en el paso anterior, se buscan los huecos que existen entre las letras que componen una palabra y que sirven como referencia para realizar las secciones para obtener así cada una de las letras por separado como muestra el ejemplo de la

figura 3.11.

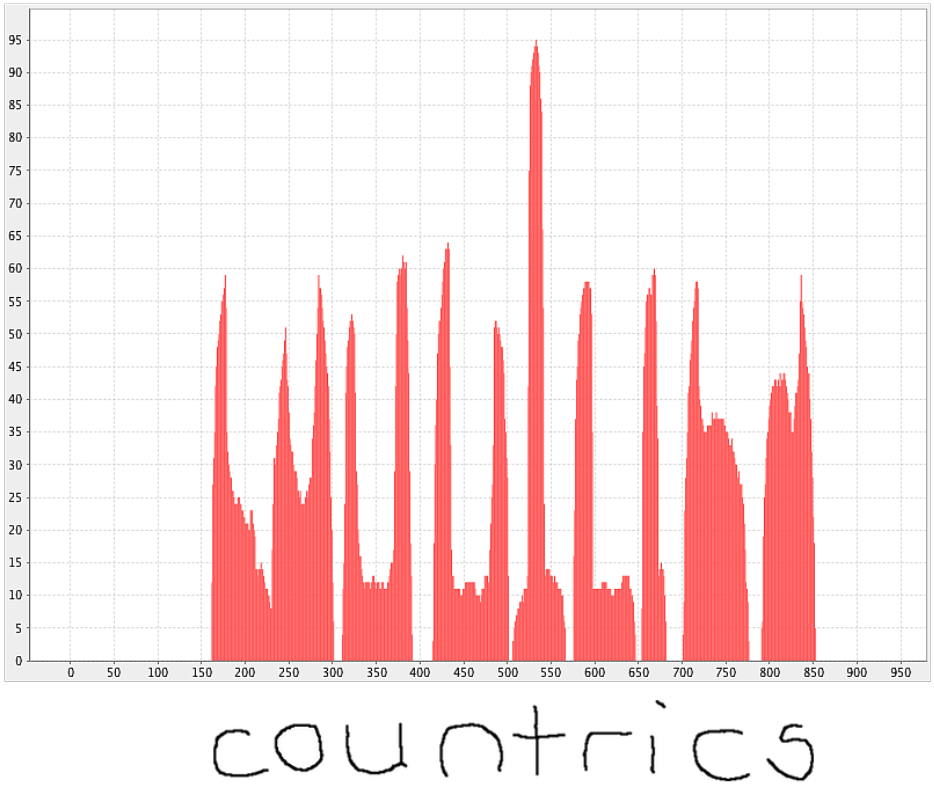


Figura 3.11: Obtención de las letras a partir de una palabra usando su proyección vertical para encontrar los huecos que separan cada una de las letras.

3.2.10 Operaciones finales

Tras aplicar los pasos anteriores, se obtiene una imagen de la letra sin considerar los márgenes en blanco. Por ello, se realiza un proceso de ajuste de manera que la imagen quede sin estos márgenes mediante una operación de recorte.

Por último, debido a que a partir de la imagen original se han ido obteniendo cada vez secciones más pequeñas, se ha ido perdiendo precisión en cuanto al número de píxeles que finalmente componen una letra. Por ello, con el fin de aumentar el índice de precisión, se realiza un reescalado de la imagen de la letra a un tamaño 2 veces mayor respecto a la original.

Se muestra en la figura 3.12 un ejemplo de reescalado sobre una letra.

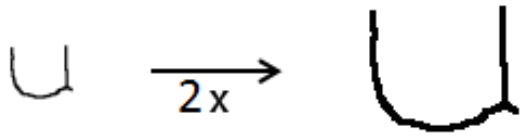


Figura 3.12: Reescalado aplicado en la letra 'u'.

3.3 Extracción de características

En la etapa de extracción de características se da una definición formal e inequívoca a cada uno de los tipos de unidades básicas, o letras, obtenidas a partir del proceso anterior de preprocesamiento.

Así pues, una vez obtenidas las letras, se parte de la idea de que éstas se pueden descomponer en trazos verticales y horizontales como propone A. Alonso et al. [4]. Considerándose los trazos verticales como los principales, se buscan las conexiones existentes entre cada uno de los trazos verticales con sus trazos horizontales adyacentes. Para ello, se utiliza un método de zonificación dinámica en el cual se

desplaza una rejilla por cada uno de los trazos verticales obteniendo así las posiciones de conexión con los trazos horizontales. Estas posiciones son guardadas como descriptor principal dentro de un vector de características. Por último, se utiliza un lenguaje formal que, mediante un conjunto de reglas ya preestablecido, comprueba que dicho vector está bien formado.

3.3.1 Segmentación de las letras en trazos

Se parte de la idea de que una letra se puede descomponer en trazos verticales y horizontales. Para la segmentación de una letra en sus trazos, se consideran las siguientes 4 hipótesis [4]:

1. Cada píxel pertenece a un y solo un trazo.
2. Sólo se consideran dos tipos de trazos: verticales y horizontales.
3. Los trazos verticales son los principales y por consiguiente, ningún trazo horizontal puede estar superpuesto sobre un trazo vertical.
4. Los trazos verticales sólo pueden conectar con trazos horizontales y no pueden hacerlo con otros trazos verticales.

El primer paso consiste en la asignación de cada uno de los píxeles que componen la imagen a una de las dos categorías posibles: vertical u horizontal. Para ello, se calcula la distancia a 0^o y 90^o en base a la anchura y la altura del trazo como muestra la figura 3.13. Siguiendo la ecuación 3.7, si la distancia en el eje horizontal a 0^o (dh) es mayor que en el eje vertical a 90^o (dv), dicho píxel se etiqueta como horizontal h; en caso contrario el píxel se etiqueta como vertical v.

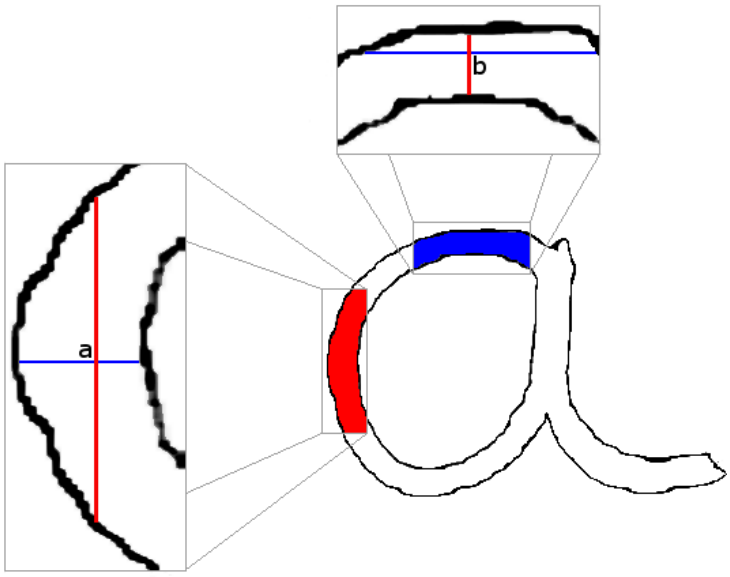


Figura 3.13: Etiquetado de los píxeles en un trazo. El píxel situado en el punto a es marcado como vertical v. Por el contrario, el píxel en la posición b es etiquetado como horizontal h.

$$P(x,y) = \begin{cases} v & dv \geq dh \\ h & \text{en cualquier otro caso} \end{cases} \tag{3.7}$$

Una vez etiquetados todos los píxeles, éstos se reagrupan por regiones en trazos verticales y horizontales.

Para verificar la tercera hipótesis, los trazos verticales se extienden sobre los trazos horizontales. Por lo tanto, si hubiera alguna región horizontal superpuesta a una región vertical, los píxeles implicados son etiquetados como verticales y se añaden al trazo vertical. Por otro lado, el trazo horizontal afectado, queda dividido en dos nuevos trazos horizontales como se puede apreciar en la imagen 3.14.

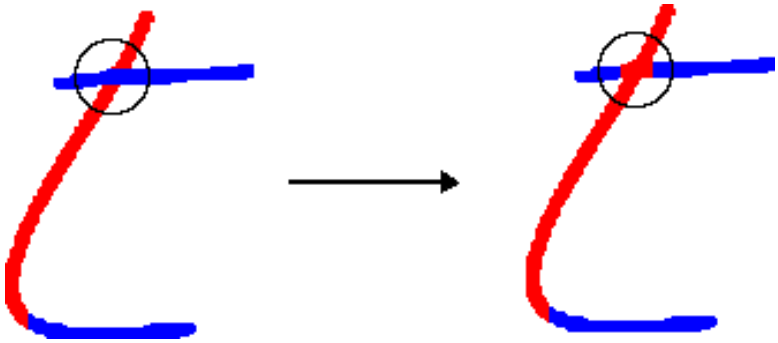


Figura 3.14: Corrección de superposición de trazos. Cuando un trazo horizontal superpone a un trazo vertical, los píxeles implicados se etiquetan como verticales.

Puede ocurrir que alguna imagen que contenga por ejemplo una letra "v", "w","x" o incluso una "y", esté compuesto sólo por la unión de varios trazos verticales. Para que se cumpla la cuarta hipótesis, es necesario introducir un trazo horizontal auxiliar entre los dos trazos verticales conectados de manera que queden claramente diferenciados.

El proceso de inserción de un trazo horizontal entre dos verticales requiere de la búsqueda de mínimos relativos dentro de la imagen en cuestión. Esta es otra de las aportaciones de esta tesis. Para ello, se realiza un recorrido de los píxeles de la imagen de izquierda a derecha y de arriba a abajo considerando que:

1. Sólo se tiene en cuenta la parte interior del trazo.
2. Se debe encontrar al menos un píxel etiquetado como v o h hacia la derecha.
3. Debe existir al menos un píxel etiquetado como v o h en el siguiente píxel hacia abajo o en un rango horizontal de $\pm$ 4 píxeles.

Este proceso se repite hasta que se encuentre un píxel en el que no se pueda continuar en ninguna de las direcciones, con lo que se marca dicho píxel como mí-

nimo local. El proceso comenzaría de nuevo, pero esta vez desde la parte superior del último mínimo relativo encontrado y hacia la derecha hasta finalizar la imagen.

Se muestra un ejemplo con la imagen de la letra ”w” en la figura 3.15. En ella, se aprecia la ruta que va tomando el algoritmo hasta encontrar los mínimos locales.

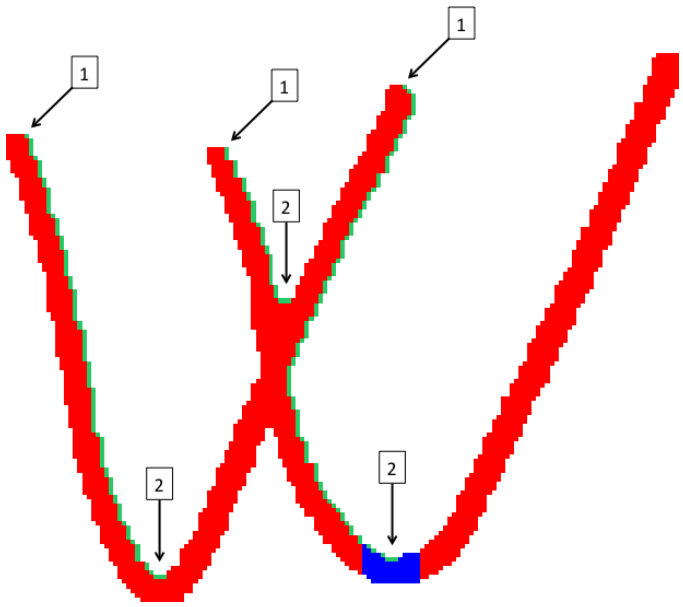


Figura 3.15: Algoritmo de búsqueda de mínimos. Los puntos 1 indican dónde comienza la búsqueda dando por finalizado el proceso en el punto 2 y por lo tanto considerando dicho punto como mínimo relativo.

En algunas ocasiones, la búsqueda de mínimos relativos puede realizarse en letras cuyos trazos están ”cerrados” como pasa por ejemplo con la letra ”a”, ”b”, ”d”, etc. Así que no basta con encontrar un mínimo relativo, sino que se debe verificar que no está dentro de un trazo ”cerrado”. Por tanto, el siguiente paso consiste en buscar desde el píxel marcado como mínimo, la existencia de una ruta desde dicho píxel hasta alguno de los bordes de la imagen siendo normalmente el borde superior. En la figura 3.16 se muestra un ejemplo de cómo el algoritmo encuentra una ruta válida al borde superior. Por el contrario, la figura 3.17 muestra un ejemplo de una letra ”cerrada” en la que el algoritmo de búsqueda no encuentra una solución.

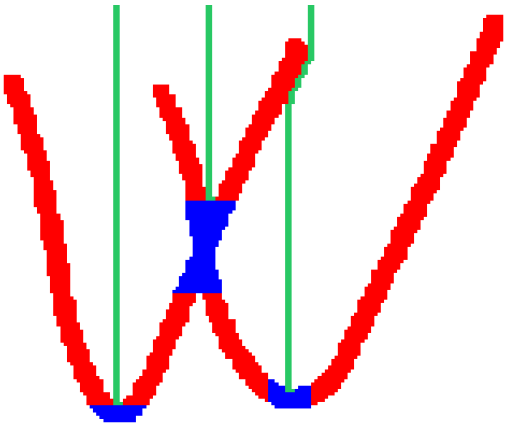


Figura 3.16: Algoritmo de búsqueda de apertura. Se busca la salida desde el mínimo hasta alguno de los bordes de la imagen, normalmente el superior. Si se encuentra una salida entonces se inserta un trazo horizontal en la zona donde está situado el mínimo.

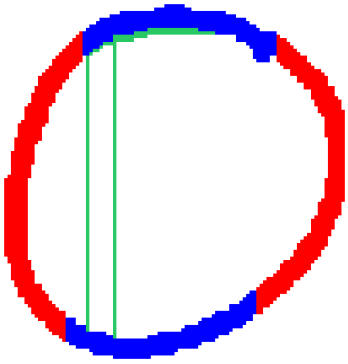


Figura 3.17: Algoritmo de búsqueda de apertura sin éxito en encontrar una salida a alguno de los bordes de la imagen en la letra "o".

Si se ha encontrado una ruta hacia alguno de los bordes desde el píxel marcado como mínimo, se procede a la inserción de un trazo horizontal auxiliar cuyo tamaño es igual al ancho del trazo siempre y cuando el píxel marcado como mínimo no esté etiquetado como h.

Existe con frecuencia la aparición de trazos verticales muy pequeños. Esto es debido normalmente a alguna forma poco común de los trazos evaluados y como consecuencia puede hacer que se introduzcan imperfecciones no deseadas en las siguientes etapas. Para solventar esta dificultad, se evalúa cuál es el mayor trazo vertical existente v_{max} en la imagen contando los píxeles que contiene. Después, siguiendo la ecuación 3.8 se toman aquellos trazos cuyo tamaño sea inferior del

tamaño del trazo mayor v_{max} en el rango δ y se etiquetan y añaden a los trazos horizontales colindantes, produciéndose en la mayoría de las ocasiones uniones entre trazos horizontales.

$$Trazo(x) = \begin{cases} Aceptado & tam(x) \geq \delta \,\%\, tam(v_{max}) \\ Excluido & en\ otro\ caso \end{cases} \tag{3.8}$$

Se puede ver un ejemplo en la figura 3.18 Este proceso sólo afecta a trazos verticales, los trazos horizontales son todos considerados, independientemente de su tamaño.

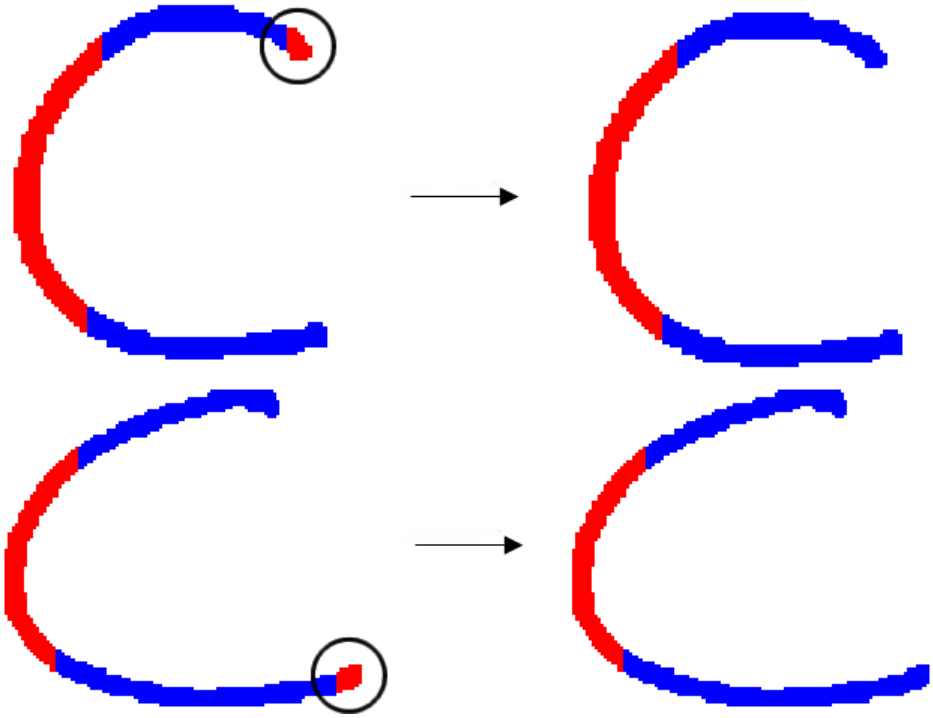


Figura 3.18: Descarte de trazos verticales que son inferiores a un porcentaje del tamaño del trazo más grande.

Una vez etiquetados como *v* o *h* todos los píxeles que componen los trazos en la imagen, se procede a la evaluación de las conexiones existentes para cada trazo vertical con cada uno de sus trazos horizontales adyacentes.

Este proceso se realiza mediante el uso de una rejilla de dimensión 4*x*3 cuyo tamaño se va adaptando a la altura del trazo vertical que se está evaluando. De esta manera, se obtienen 8 regiones posibles de adyacencia: 4 en la parte izquierda del trazo y 4 en la parte derecha. Como se puede apreciar en la figura 3.19, el etiquetado de las 8 regiones de adyacencia están identificadas del 1 al 8 comenzando por la esquina superior izquierda con el 1 y se continúa numerando en sentido contrario a las agujas del reloj hasta finalizar en la esquina superior derecha con el 8. Siguiendo esta convención, se tiene un sistema en el que es posible identificar dónde los trazos verticales conectan con los horizontales.

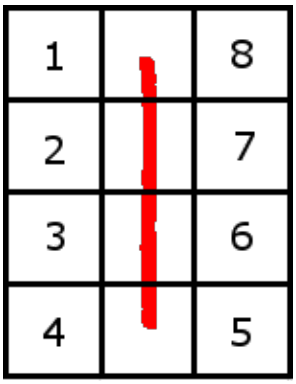


Figura 3.19: Rejilla para el proceso de zonificación dinámica. La rejilla se compone de 8 regiones, 4 en la parte izquierda y 4 en la parte derecha pertenecientes a las zonas donde los trazos horizontales pueden adherirse.

Es posible que algún trazo horizontal conecte en dos o más regiones de adyacencia a un trazo vertical. En este caso, se evalúa en qué región de las implicadas conectan más píxeles horizontales con los verticales y se le asigna dicha región

como de adyacencia al trazo vertical, excluyendo así el resto de regiones. En caso de que sean el mismo número de píxeles en las regiones superiores, prevalecerá la la zona más superior. Esto es, si por ejemplo existe el mismo número en la región 7 y 8, prevalece la región 8 y se descarta la región 7. La misma norma se aplica a las regiones inferiores.

3.3.3 Descriptor de características

Puesto que las conexiones existentes entre un trazo vertical y sus adyacentes van a ser la característica principal, el conjunto de estas características obtenidas de cada uno de los trazos encontrados en una imagen va a ser considerado como el descriptor de características de la letra que se está analizando.

Para poder dar una definición formal, se construye una cadena representativa alfanumérica en la que se inserta una *V* que identifica al trazo vertical encontrado en la imagen, seguido del número de la posición o posiciones donde el trazo vertical en cuestión tiene trazos horizontales adheridos. Se muestra un ejemplo en la figura 3.20.

La imagen se recorre de izquierda a derecha repitiendo el proceso con todos los trazos verticales hallados y concatenando a la cadena descriptora principal. Como resultado, se obtiene una cadena que contiene la unión de todas las representaciones de cada uno de los trazos y por ende, una descripción de la letra como se muestra por ejemplo, con la letra "h" en la figura 3.21.

Como los trazos verticales se analizan de forma independiente, puede pasar que

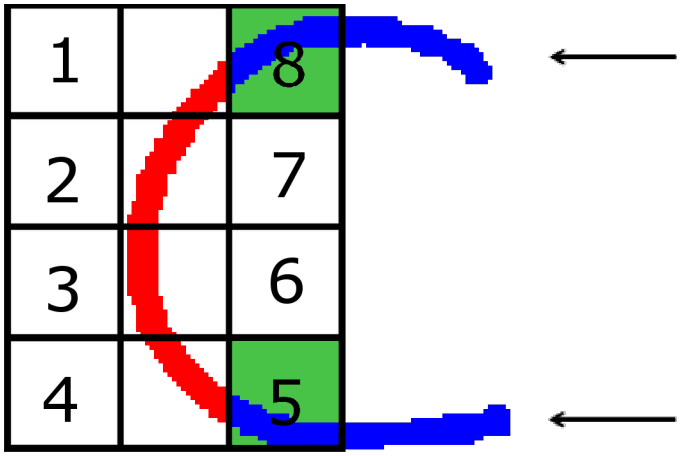


Figura 3.20: Obtención de características de la letra "c". Las regiones de adyacencia son la 5 y la 8, con lo que la cadena representativa es: *V*58.

diferentes letras produzcan las mismas cadenas descriptoras. Si se estudian, por ejemplo, de forma independiente las letras "o" y "s", se puede observar que ambas letras generan la misma cadena representativa (ver figura 3.22-a). La diferencia principal radica en, que a parte de la situación y el tamaño de los trazos, el número de trazos horizontales que interconectan dos trazos verticales es diferente. Mientras que en la letra "o" existen dos trazos horizontales interconectados a los trazos verticales, en la letra "s" solamente existe uno. En la figura 3.22 se puede ver con detalle este caso.

Para resolver esta incongruencia, basta con repetir el valor numérico de la posición donde conecta el trazo horizontal al vertical en aquellos trazos horizontales que estén conectados a dos trazos verticales. De esta manera se logra diferenciar aquellos trazos horizontales que están conectados a un trazo vertical de aquellos que están conectados a dos trazos verticales (ver figura 3.22-b).

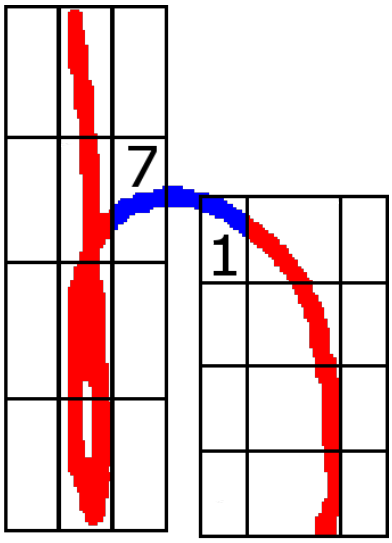


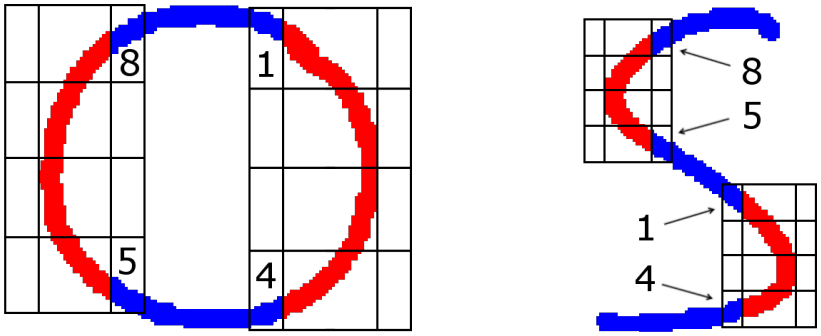
Figura 3.21: Definición de características para la letra "h". Se recorren los trazos de la imagen procesándolos de izquierda a derecha. La zonificación dinámica se ajusta a la dimensión del trazo vertical y permite obtener las regiones de adyacencia de cada trazo vertical con sus horizontales obteniendo de esta manera la cadena representativa "V7V1".

3.3.4 Definición de la gramática formal

La cadena representativa que contiene los descriptores de la letra debe estar bien formada y para ello, debe de existir un conjunto de reglas que permita su correcta construcción.

La producción de cadenas con una correcta estructura se realiza mediante una gramática formal. De esta manera, se dice que una gramática formal es una definición matemática que establece un conjunto de reglas que permiten escribir cadenas de caracteres válidas usando un lenguaje formal preestablecido [68].

Más en concreto, dentro de las gramáticas formales, Noam Chomsky establece una clasificación jerárquica en la que existen cuatro niveles:



a)	V58V14	=	V58V14
b)	V5588V1144	≠	V558V114

Figura 3.22: a) Las letras "o" y "s" producen la misma cadena representativa. **b)** La ambigüedad se resuelve repitiendo aquellos valores de la posición de los trazos horizontales interconectados.

- Gramáticas sin restricciones. Producen lenguajes recursivamente enumerables. Estos lenguajes pueden ser reconocidos por máquinas de Turing.
- Gramáticas dependientes del contexto. Producen Lenguajes dependientes de contexto que son reconocidos por autómatas linealmente acotados.
- Gramáticas independientes del contexto. Producen lenguajes independientes del contexto que son reconocidos por autómatas de pila.
- Gramáticas regulares. Producen lenguajes regulares y son reconocidos por autómatas finitos.

Dentro de esta clasificación, la gramática regular es el tipo más simple de gramática formal [32] y es la que se va a utilizar para generar la cadena representativa de la letra a analizar.

Una gramática regular se representa como $G = (\Sigma_T, \Sigma_N, S, P)$ donde:

Σ_T : Alfabeto de símbolos terminales.

Σ_N : Alfabeto de símbolos no terminales.

$S \in \Sigma_N$: Es el axioma o símbolo inicial.

P : Conjunto de reglas o producciones del tipo $\alpha \rightarrow \beta$

De donde se deduce que el alfabeto Σ se forma como:

$\Sigma = \Sigma_T \cup \Sigma_N$

Si se parte de esta definición para crear la gramática regular que va a producir una cadena representativa bien formada, ésta debe además verificar las siguientes reglas:

1. Cada trazo vertical se representará en un bloque de caracteres.
2. Cada bloque comienza por el carácter V seguido de uno o más dígitos que representan las posiciones de conexión de los trazos horizontales con los verticales.
3. Los dígitos tienen que ir en sucesión creciente.
4. Para poder representar las ambigüedades, no puede haber más de dos dígitos iguales seguidos.

Y por tanto, la definición formal de la gramática regular $G_{CR} = (\Sigma_T, \Sigma_N, S, P)$ es:

$\Sigma_T = \{V,1,2,3,4,5,6,7,8\}$

$\Sigma_N = \{V,1,2,3,4,5,6,7,8\}$

$S = \{V\}$

$P = \{ V \rightarrow V|1|2|3|4|5|6|7|8,$

$$1 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$2 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$3 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$4 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$5 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$6 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$7 \rightarrow V|1|2|3|4|5|6|7|8,$$
$$8 \rightarrow V|1|2|3|4|5|6|7|8\}$$

Una vez definida la gramática regular que permite la producción de cadenas representativas basadas en las características de los trazos, se requiere de la validación de la misma. Un autómata finito es capaz de realizar esta tarea permitiendo reconocer aquellas cadenas generadas por gramáticas regulares que cumplen las reglas predefinidas.

La definición formal de un autómata finito, viene expresado en una 5-upla $(Q, \Sigma, q_0, \delta, F)$ donde:

Q : Es un conjunto finito de estados.

Σ : Alfabeto de símbolos.

q_0 : Es el estado inicial donde $q_0 \in Q$

δ : Función de transición $Qx\Sigma \rightarrow Q$

F : Estados finales o de aceptación $F \subseteq Q$

Partiendo de esta definición y usando como gramática regular la diseñada anteriormente, se define el autómata finito que valida las cadenas representativas como sigue:

$$Q = \{ q_0, q_v, q_1, \ldots, q_8, q_{11}, \ldots, q_{88}, q_{err} \}$$

$$\Sigma = \{ V, 1, 2, 3, 4, 5, 6, 7, 8 \}$$

$$q_0 = \{ q_0 \}$$

$$F = \{ q_v, q_1, \ldots, q_8, q_{11}, \ldots, q_{88}, q_{err} \}$$

Funciones de transición δ

$$\delta(q_0, x) = \begin{cases} q_v & \text{x = v} \\ q_{err} & \text{otro elemento} \qquad (\text{x dígito}) \end{cases}$$

$$\delta(q_v, x) = \begin{cases} q_i & \text{x = i} \qquad (\text{i dígito}) \\ q_v & \text{x = v} \end{cases}$$

$$\delta(q_i, x) = \begin{cases} q_v & \text{x = v} \\ q_{ii} & \text{x = i} \\ q_j & \text{x = j > i} \\ q_{err} & \text{otro elemento} \qquad (\text{j} < \text{i}) \end{cases}$$

$$\delta(q_{ii}, x) = \begin{cases} q_v & \text{x = v} \\ q_j & \text{x = j > i} \\ q_{err} & \text{otro elemento} \qquad (\text{j} \leq \text{i}) \end{cases}$$

De esta manera, el proceso de validación de la cadena representativa se lleva a cabo por el autómata finito anteriormente definido. Éste parte del estado inicial y va procesando cada uno de los símbolos de la cadena cambiando de estado en el autómata de acuerdo a las producciones establecidas en las funciones de transición. Una vez finalizada la lectura de la cadena se comprueba su estado. Si dicho estado es un estado final o de aceptación, entonces se dice que la cadena leída pertenece al lenguaje reconocido por el autómata y por tanto, se trata de una cadena válida bien formada. Por el contrario, si el autómata no ha llegado a un estado de aceptación, se trata de una cadena que no pertenece al lenguaje y por tanto, la cadena es rechazada.

3.4 Base de conocimiento

Una vez las cadenas representativas han sido generadas en base a la gramática regular y validadas por un autómata finito, deben ser almacenadas como conocimiento ya que poseen las características extraídas de la letra analizada en cuestión.

Para este fin, se crea una base de conocimiento la cual posee una estructura de árbol.

Por definición, se dice que un árbol es una colección de elementos llamados nodos, de entre los cuales, se distingue un nodo raíz que junto con una relación de *paternidad* se crea una estructura jerárquica sobre los nodos [2]. Formalmente, un árbol se puede definir como sigue:

1. Un único nodo es por si mismo un árbol. Dicho nodo es raíz de dicho árbol.
2. Si se considera que n es un nodo y que $A_1, A_2, \ldots, A_k$ son árboles con raíces $n_1, n_2, \ldots, n_k$, se podría construir un nuevo árbol de manera que n es la raíz del nuevo árbol y $n_1, n_2, \ldots, n_k$ son los *subárboles* de la raíz. De este modo, los nodos $n_1, n_2, \ldots, n_k$ reciben el nombre de *hijos* del nodo n.

Para una sucesión de nodos de un árbol $n_1, n_2, \ldots, n_k$ tal que n_i es el padre de $n_i + 1$ para $1 \leq i < k$ se denomina *camino* a la secuencia desde el nodo n_1 al nodo n_k. Por otro lado, la *longitud* del camino es el número de nodos del camino menos 1. Se puede decir de esta manera, que existe un camino de longitud cero desde cualquier nodo a sí mismo.

Más concretamente, dentro del amplio conjunto de estructuras que componen los árboles (binarios, AVL, B, etc.), existe una implementación diseñada especial-

mente para la recuperación de información. Esta estructura se denomina *Trie* que proviene de la palabra inglesa re*trie*val.

Normalmente, la información que almacena un *trie* es un conjunto de claves que están definidas en un espacio o alfabeto. Estas claves suelen estar alojadas en las hojas del árbol y los nodos encontrados en el camino para llegar a la hoja son pasarelas para guiar la búsqueda. De esta manera, el árbol está compuesto por unidades estructurales en la que cada clave tiene un número de hijos acotado los cuales representan cada uno de los posibles símbolos que pueden continuar al símbolo representado por su nodo padre como muestra el ejemplo de la figura 3.23.

La implementación que se ha considerado, como aportación en este trabajo, reduce el alfabeto del *trie* al mismo alfabeto que el que se ha usado anteriormente en la gramática regular para construir las cadenas representativas: $\Sigma = \{V,1,2,3,4,5,6,7,8\}$. De esta manera, el árbol sólo se construirá con símbolos que están dentro de la gramática regular. Además, en los nodos hojas, en vez de almacenar la cadena obtenida al recorrer el camino, se va a almacenar el carácter que identifica dicho camino.

El árbol que aloja la base de conocimiento, funcionalmente, sigue las reglas definidas en un esquema XSD, expuesto en el apéndice A, con lo que finalmente se generará un archivo XML con todas las entradas procesadas.

Por tanto, para crear la base de conocimiento mediante la estrategia detallada anteriormente, se descompone la cadena representativa en cada uno de los elementos que la componen y se construye un árbol cuyo nodo raíz siempre empieza por V (las cadenas representativas siempre empiezan por V) y se añade un nodo por cada elemento de la cadena representativa en cuestión como muestra la figura 3.24. Una vez agregados todos los nodos, se añade un nodo hoja que contendrá la letra que representa el camino construido. De esta manera, se tiene una ruta

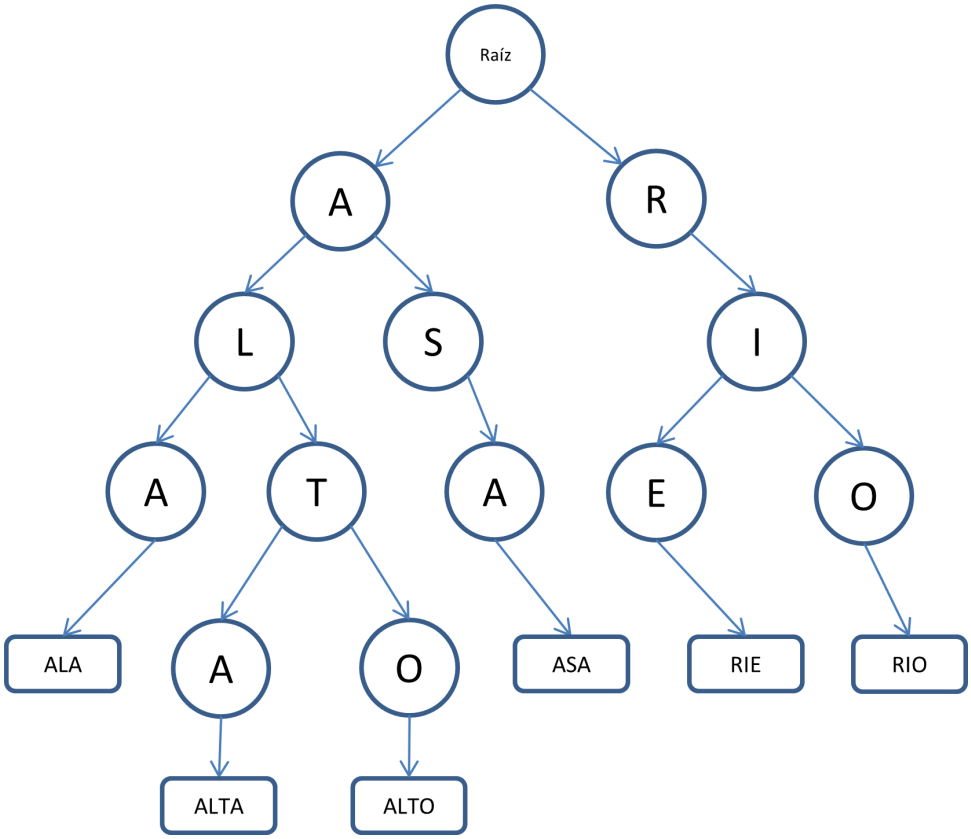


Figura 3.23: Ejemplo de un árbol *Trie*. Cada uno de los nodos hijo al raíz representa un camino válido. Las hojas contienen la clave que identifica el camino recorrido.

específica para cada cadena representativa.

Puede suceder que dos letras diferentes tengan la misma cadena representativa (ya sea por una segmentación errónea o por una identificación equivocada de los trazos verticales y horizontales que la componen) y por consiguiente, tengan la misma ruta pero se identifican bajo diferentes nodos hoja. Por ello, en cada nodo hoja, además de la letra que identifica esa ruta, se almacena también el número de veces que se repite ese camino para una letra en cuestión (Ver ejemplo de la figura

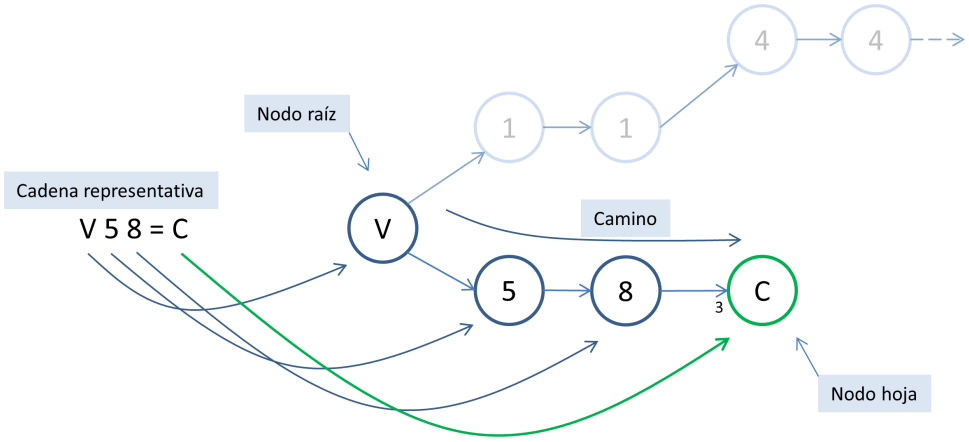


Figura 3.24: Construcción de la base de conocimiento. Cada cadena representativa se incluye en el árbol como un nuevo camino. Los elementos de la cadena representativa son añadidos al árbol a partir del nodo raíz generando una nueva secuencia en la que finalmente, se añade un nodo hoja con la solución a dicho camino.

3.25).

De esta manera se puede diferenciar aquellas letras que tienen un mismo camino consiguiendo además, un peso para cada uno de los nodos hoja. El objetivo de este sistema es tener anotado cuál es el nodo hoja más adecuado como solución al reconocimiento sin obviar otras posibles soluciones para el mismo camino.

El árbol *trie* se rellena con los datos introducidos mediante un proceso de entrenamiento previo generando así una base de conocimiento específica para los datos introducidos. La figura 3.26 muestra un ejemplo de cómo sería una base de conocimiento sencilla.

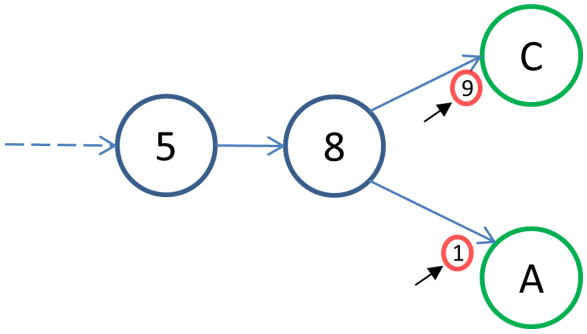


Figura 3.25: Representación de la repetición de un camino. Además de la solución al camino, se anota el número de veces que se ha reproducido el camino para llegar a la misma solución.

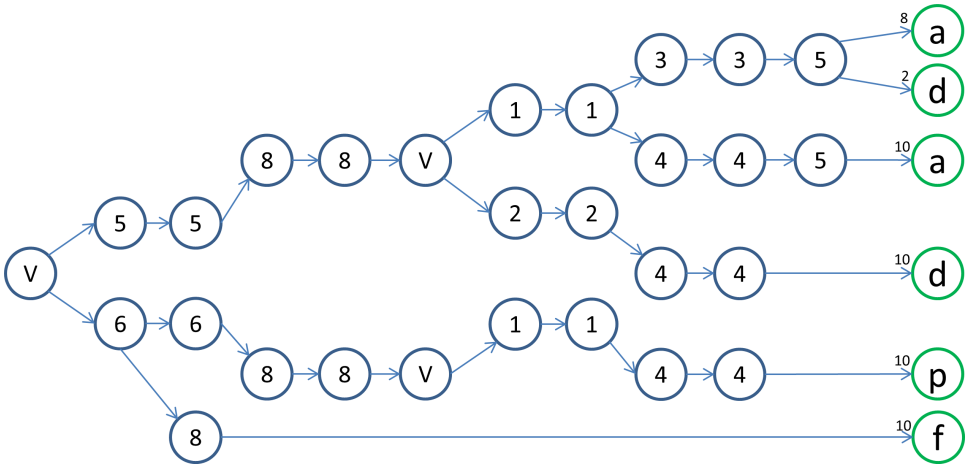


Figura 3.26: Ejemplo de Base de Conocimiento con la estructura generada para algunas letras.

3.5 Motor de inferencia

La base de conocimiento no es de por sí suficiente para el sistema reconocedor. Se necesita de un Motor de Inferencia que sea capaz de tomar decisiones sobre las nuevas entradas en función de los valores almacenados en la propia base de

conocimiento.

Por tanto, el Motor de Inferencia toma los descriptores (las cadenas representativas) obtenidos de caracteres desconocidos que se pretende reconocer y realiza el proceso de búsqueda en el árbol *trie* previamente generado en el proceso de entrenamiento.

Para ello, en un primer paso, se segmenta la palabra que se desea reconocer en las respectivas letras que la componen como se ha descrito en la etapa de preprocesamiento de la sección 3.2. Después, se generan los descriptores de cada una de las letras por separado y se obtienen las respectivas cadenas representativas utilizando el método de extracción de características de la sección 3.3 como se muestra en la figura 3.31 (1).

Una vez en este punto, se descompone la cadena representativa en cada uno de los elementos y se compara cada uno de estos elementos con los nodos existentes en la base de conocimiento como se describe a continuación: el proceso comienza comparando el primer elemento de la cadena representativa con el nodo raíz. Si el símbolo es hallado, entonces se busca el siguiente elemento de la cadena en el subárbol asociado al símbolo encontrado. Se sigue de forma análoga hasta que se finaliza la cadena representativa. Si existen nodos hoja asociados al camino recorrido entonces la búsqueda ha terminado con éxito y se devuelven los nodos con sus pesos para una evaluación posterior. En caso contrario, si no hubiera nodos hoja, se produciría un rechazo ya que la ruta es desconocida para la base de conocimiento y como consecuencia se devuelve un elemento vacío.

En las siguientes imágenes 3.27, 3.28, 3.29 y 3.30 se muestra un ejemplo de cómo sería el proceso de búsqueda de la solución en la base de conocimiento a partir de la cadena representativa obtenida en la etapa de extracción de características. En este ejemplo, se devolvería la letra "a" como solución a la cadena $V5588V11445$.

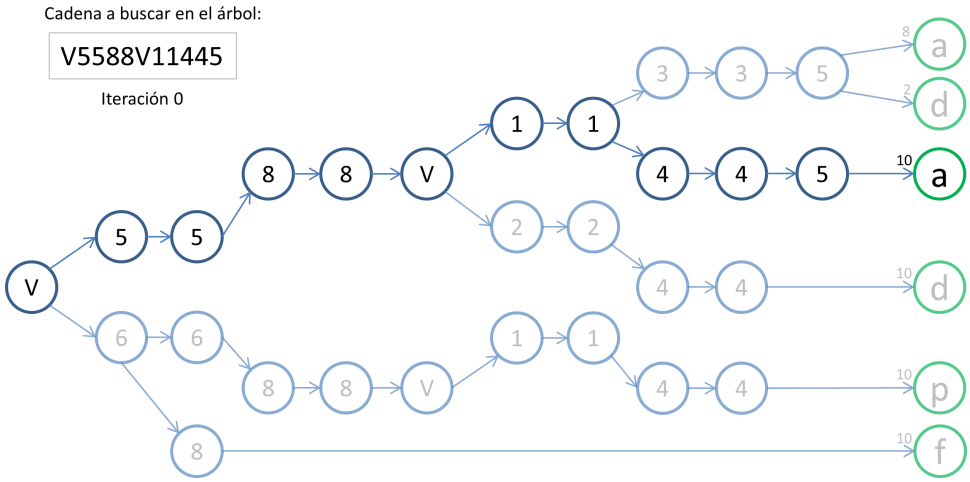


Figura 3.27: Recorrido del árbol buscando una solución (1): El proceso comienza buscando el primer elemento de la cadena representativa en la raiz del árbol.

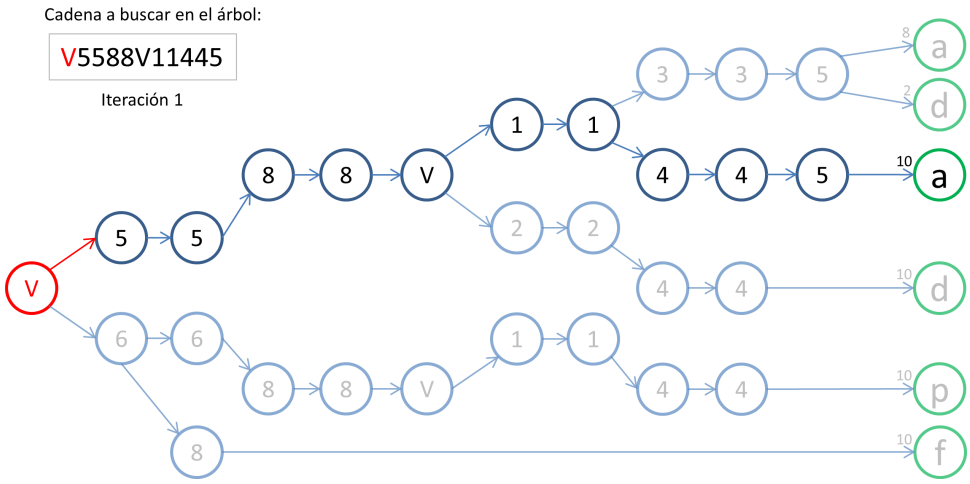


Figura 3.28: Recorrido del árbol buscando una solución (2): Una vez encontrado el nodo padre, se busca entre los hijos el siguiente elemento de la cadena representativa.

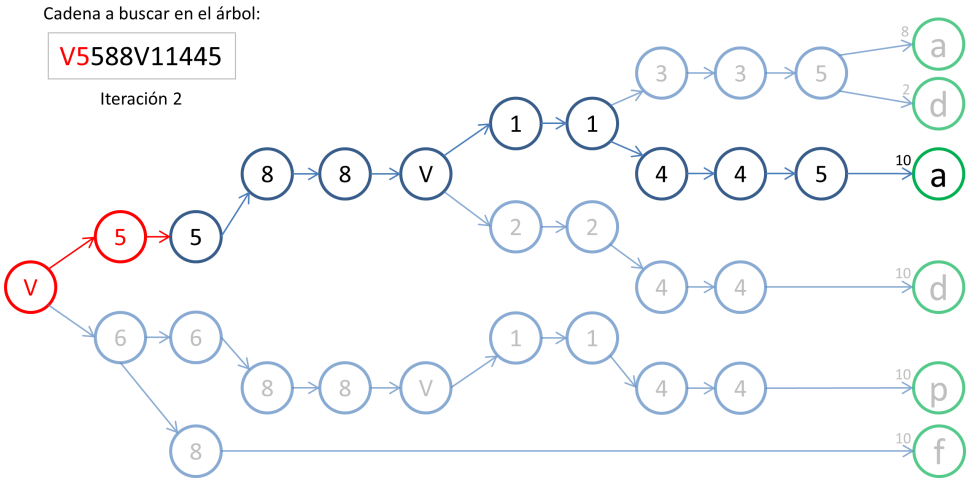


Figura 3.29: Recorrido del árbol buscando una solución (3): El proceso se repite hasta completar la cadena representativa o hasta que no se pueda continuar.

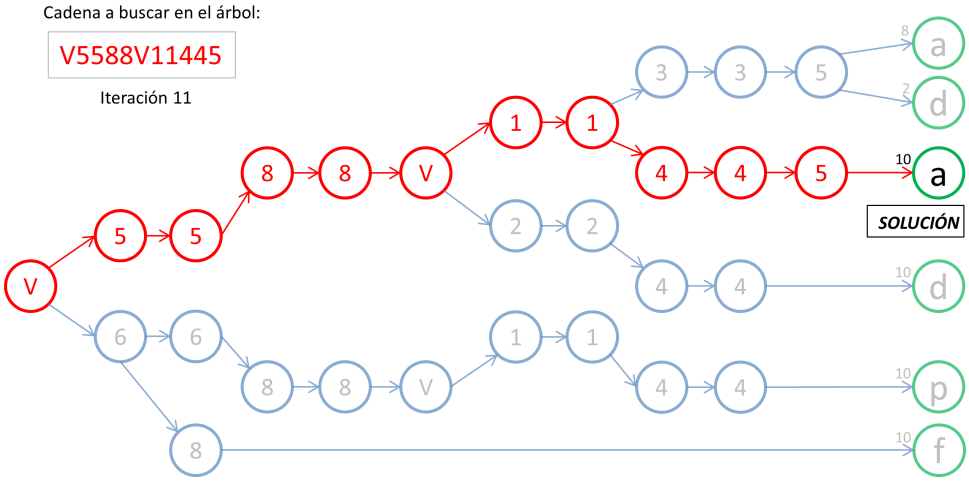


Figura 3.30: Recorrido del árbol buscando una solución (4): Una vez recorrida toda la cadena representativa, se devuelve el símbolo almacenado en la hoja para el camino recorrido como solución. En caso de que no exista un nodo hoja para el camino recorrido, se devuelve un elemento vacío.

El proceso se repite para cada una de las letras que componen la palabra que se desea reconocer obteniendo para cada letra los posibles candidatos para ser anali-

zados posteriormente como muestra la figura 3.31 (2).

Una vez obtenidos los posibles candidatos para cada una de las letras, el Motor de Inferencia construye una lista con todas las posibles soluciones en orden de probabilidad descendente. Para ello, se suman los números de aciertos de cada candidato como muestra la figura 3.31 (3). A continuación, se usa un corrector ortográfico para analizar la lista generada y escoge como solución al reconocimiento aquella palabra de la lista que tenga mayor valor de probabilidad y sea ortográficamente correcta (ver figura 3.31 (4)). Si todas las palabras de la lista no cumplen con el criterio ortográfico, se puede considerar que es una palabra que no está en el diccionario y por tanto, la solución sería la palabra con el índice más alto de probabilidad.

La figura 3.31 muestra un ejemplo con la palabra *pala*. En ella se puede ver todo el proceso de reconocimiento llevado a cabo por el Motor de Inferencia.

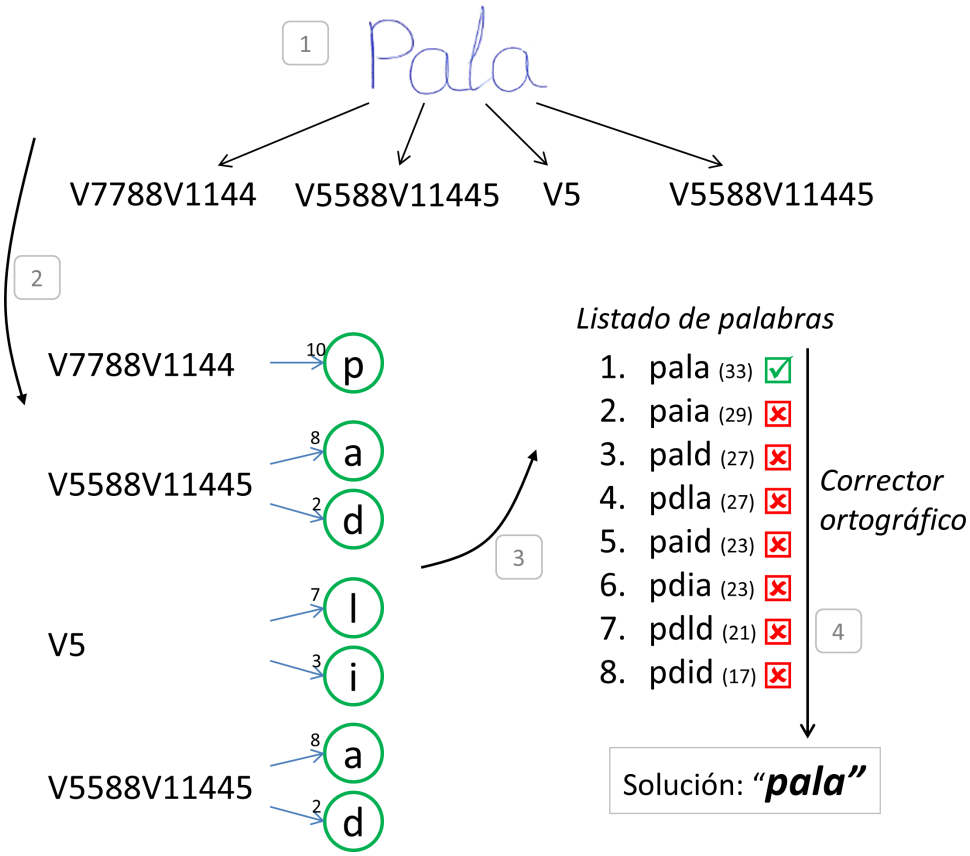


Figura 3.31: Ejemplo del proceso de reconocimiento.(1) La palabra *pala* se segmenta en sus letras y se obtienen los descriptores de cada una de las letras de forma independiente. Para el sistema, la palabra es desconocida y no es más que un conjunto de trazos. (2) Cada una de las cadenas representativas son analizadas independientemente y se obtienen los posibles candidatos usando la base de conocimiento generada previamente. (3) Se genera una lista de posibles soluciones a partir de los candidatos de cada una de las cadenas representativas. Esta lista se ordena en base a las probabilidades de cada candidato. (4) El corrector ortográfico evalúa la lista de palabras candidatas y selecciona aquella cuya probabilidad sea mayor y sea ortográficamente correcta.

"La diferencia entre la teoría y la práctica es que, en teoría, no hay diferencia entre la teoría y la práctica".

– Richard Moore

4

Experimentos y resultados

Este capítulo muestra los resultados y cómo han sido obtenidos. En la introducción se puntualizan los fundamentos de los experimentos y se detalla el diseño de los mismos. Se configuran también aquellos parámetros que son necesarios optimizar y obtener así los mejores resultados. El desarrollo de una aplicación para medir la eficacia de la metodología descrita es esencial. Por eso, se muestra el funcionamiento básico de la aplicación desarrollada así como las opciones disponibles para la realización de los experimentos. Se exponen los resultados obtenidos y son comparados con los de otros autores.

4.1 Introducción

Una vez expuesta con detalle una metodología para el reconocimiento de caracteres manuscritos en el capítulo anterior, se requiere de la comprobación de su eficacia.

Normalmente, cuando se desea evaluar un sistema de reconocimiento existen varios aspectos que se deben considerar:

1. ***Las imágenes de entrada.*** Se debe evaluar que el conjunto de datos de entrada tanto en la fase de entrenamiento como en la de validación sea el adecuado. Por otro lado, las imágenes deben tener unas características mínimas que garanticen que el reconocimiento se puede llevar a cabo.

2. ***Preprocesamiento previo.*** Un correcto procesamiento de las imágenes hace que el sistema sea menos sensible a errores a la hora de extraer las características de dicha imagen. Este procesamiento conlleva el empleo de técnicas habituales como son la binarización, corrección de inclinación, operaciones morfológicas, etc.

3. ***Extracción de características.*** La extracción de las características de un carácter implica la transformación de un modelo de datos gráfico en un modelo de datos simbólico para ser almacenado por el clasificador o en este caso, por una base de conocimiento.

4. ***Clasificación de las características.*** El modelo de datos simbólico se analiza y se agrupa por diferentes clases al cual puede pertenecer dentro de la base de conocimiento. El sistema experto usará junto con la base de conocimiento un sistema de corrección de errores basado en el uso de un diccionario.

5. ***Medidas de rendimiento.*** En cuanto al análisis del rendimiento, se han utilizado los términos de *Acierto*, *Fallo* y *Rechazo* que permiten categorizar los

resultados obtenidos.

Así pues, en el diseño de los experimentos se han seguido estas cinco pautas con el objetivo de poder medir la calidad de la metodología expuesta en el presente trabajo.

4.2 Reconocimiento alfabético

4.2.1 Diseño de los experimentos

Para la realización de los experimentos alfabéticos primeramente se recogieron las muestras de dos escritores diferentes como se detalla a continuación.

Cada uno de los escritores escribió 20 muestras de cada tipo de letra del alfabeto, comenzando por la letra *a* y finalizando en la letra *z*, utilizando para ello la plantilla del anexo B. Sólo se han considerado las letras minúsculas. Después, cada una de las hojas se escaneó a una resolución de 300 p.p.p. obteniendo las plantillas completas en formato digital a una resolución de 2530 x 1240 . Finalmente, mediante un sencillo proceso, se extrajeron cada una de las muestras de cada una de las regiones de escritura de la plantilla a un fichero de imagen independiente. En total se constituye una muestra de 520 imágenes por escritor.

Para el diseño de las pruebas, se ha hecho una separación del conjunto de imágenes de cada autor en dos grupos por letra:

- El 80% se reservó para la fase de entrenamiento y para la construcción de la base de conocimiento. Este conjunto se compone de un total de 416 imágenes (16 muestras por letra).

- El 20% restante se reservó para la realización de las pruebas. Este conjunto se compone de 104 imágenes (4 muestras por letra).

La figura 4.1 muestra la distribución de la muestra en los conjuntos de imágenes para entrenamiento y pruebas.

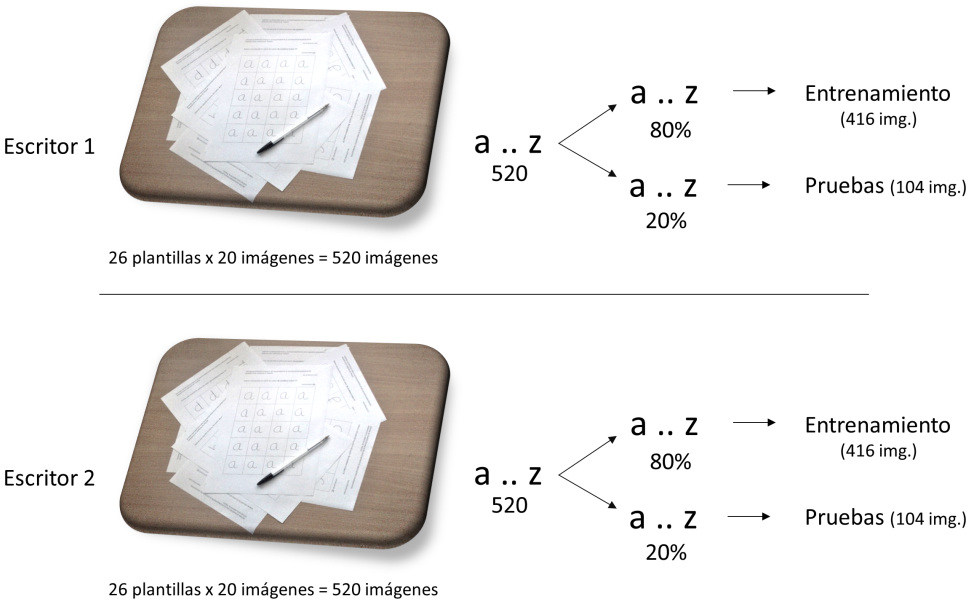


Figura 4.1: Distribución de la muestra alfabética en los conjuntos de entrenamiento y pruebas para cada uno de los escritores.

Por otro lado, en las pruebas se han considerado cuatro posibles escenarios para poder estimar cuál de los escenarios ofrece mejor rendimiento:

- Escenario 1 (N): Construcción de una única base de conocimiento sin usar el algoritmo de corrección de inclinación. Las búsquedas de nuevos elementos se realizan sobre esta base de conocimiento. A este escenario se denotará como "N" (ver figura 4.2).

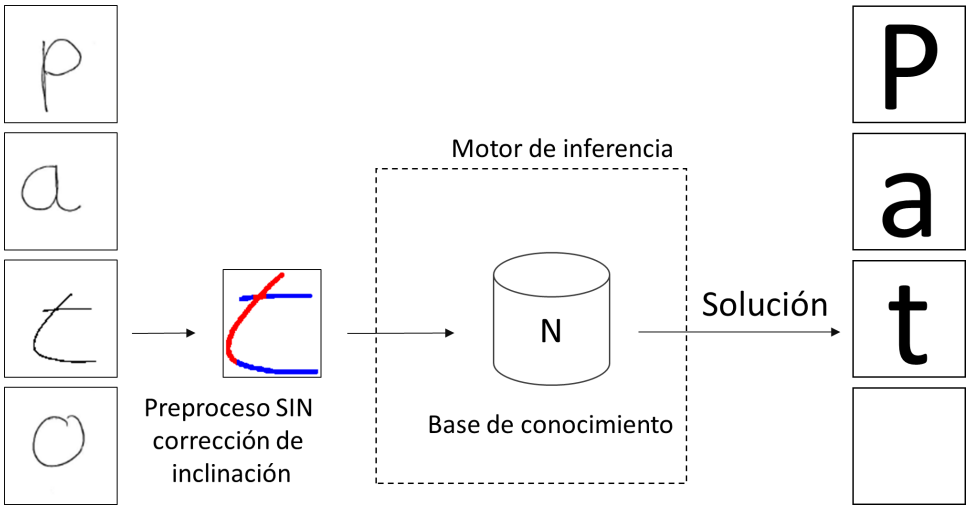


Figura 4.2: Esquema general del escenario "N".

- Escenario 2 (S): Construcción de una única base de conocimiento usando el algoritmo de corrección de inclinación. Las búsquedas de nuevos elementos se realizan sobre esta base de conocimiento. A este escenario se denotará como "S" (ver figura 4.3).

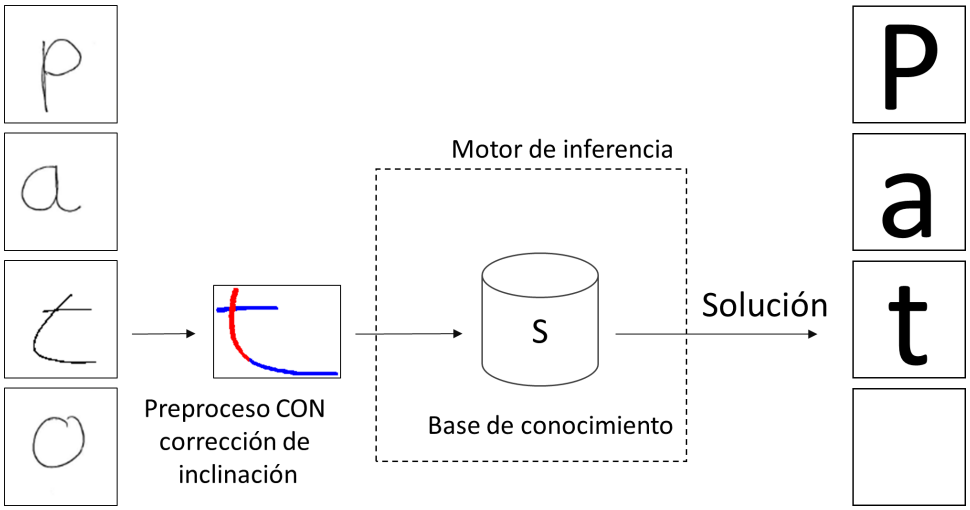


Figura 4.3: Esquema general del escenario "S".

- Escenario 3 (NS): Construcción de dos bases de conocimiento. La primera de ellas no usa el algoritmo de corrección de inclinación mientras que en la segunda sí. Las búsquedas de nuevos elementos se realizan sobre la primera base de conocimiento de manera que si se produce un rechazo, se realiza una segunda búsqueda sobre la segunda base de conocimiento cuyos elementos han sido registrados con la corrección de inclinación aplicada. A este escenario se denotará como "NS" (ver figura 4.4).

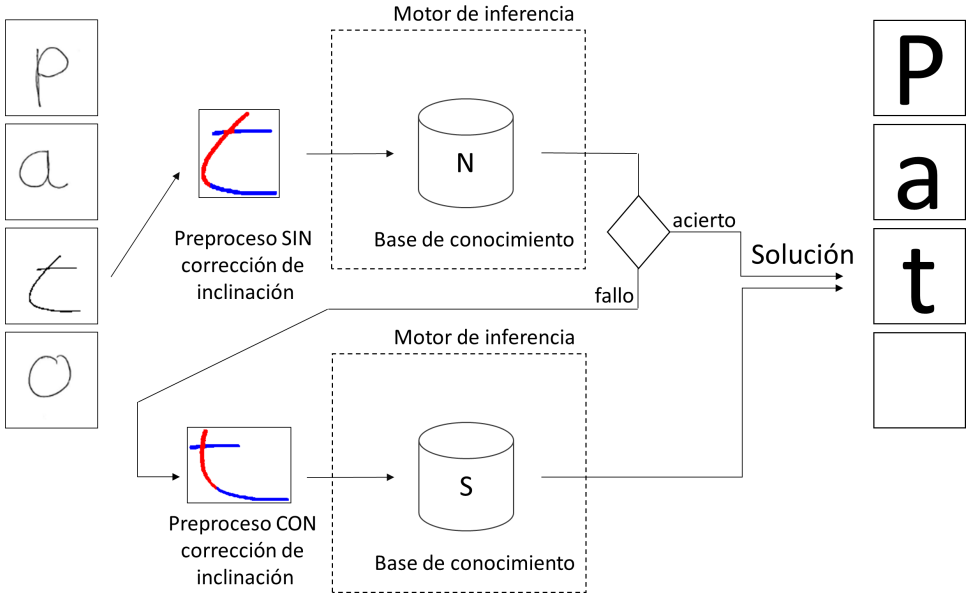


Figura 4.4: Esquema general del escenario "NS".

- Escenario 4 (SN): Construcción de dos bases de conocimiento. La primera de ellas usa el algoritmo de corrección de inclinación mientras que en la segunda no. Las búsquedas de nuevos elementos se realizan sobre la primera base de conocimiento de manera que si se produce un rechazo, se realiza una segunda búsqueda sobre la segunda base de conocimiento cuyos elementos han sido registrados sin la corrección de inclinación aplicada. A este escenario se denotará como "SN" (ver figura 4.5).

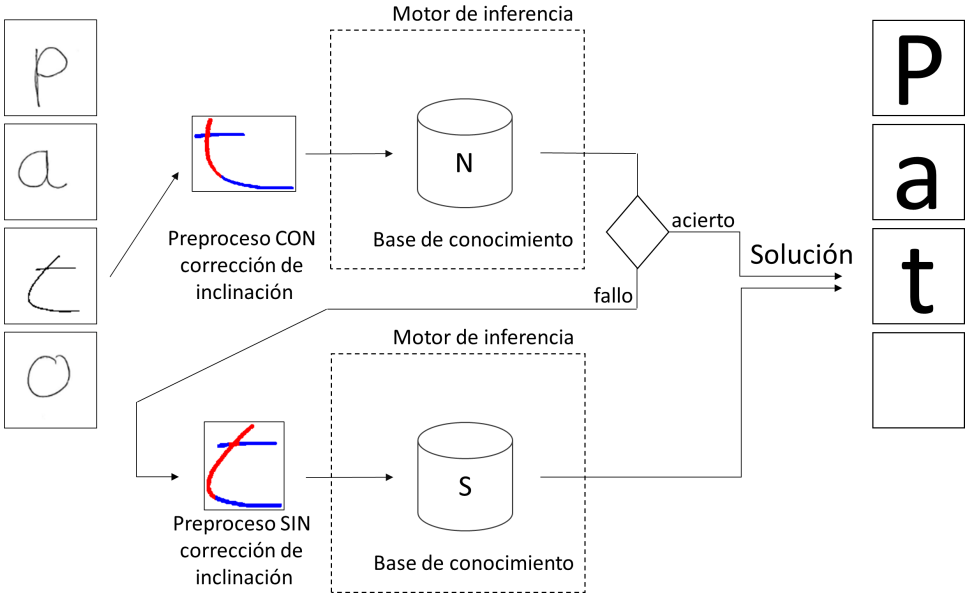


Figura 4.5: Esquema general del escenario "SN".

4.2.2 Optimización del reconocimiento

Uno de los factores que se debe considerar en el diseño de los experimentos es el tamaño del trazo vertical que se debe excluir en la etapa de segmentación. Por un lado, si se consideran todos los trazos verticales, por muy pequeños que sean, pueden añadir ruido y hacer que se construyan cadenas de descriptores complejas

cuando la información que añaden no es relevante. Por otro lado, si se consideran sólo trazos verticales grandes y se obvian aquellos trazos pequeños es posible que se pierda información y en consecuencia, se generen cadenas de descriptores que no sean lo suficientemente concisas para una letra en cuestión.

Por lo tanto, la primera de las pruebas que se ha realizado consiste en obtener el tamaño de trazo vertical a excluir que más se ajusta a obtener la cadena representativa óptima. Para ello, teniendo como referencia el mayor de los trazos verticales hallados para el carácter que se desea reconocer, se excluyen el resto de trazos verticales que no alcancen un porcentaje de tamaño mínimo en comparación con el mayor de los trazos hallados tal y como se muestra en la ecuación 3.8. Para estos experimentos, se ha considerado δ en el rango $\delta \in [6\%,12\%]$.

Siguiendo la idea expuesta, se han generado las bases de conocimiento (una con corrección de inclinación y otra sin dicha corrección) para cada escritor a partir de su propio conjunto de entrenamiento. Después, se han seleccionado 1000 palabras aleatorias de un diccionario y se han generado palabras sintéticas a partir del conjunto de pruebas de cada escritor seleccionando aleatoriamente las letras que componen dicha palabra. Estas palabras sintéticas se diferencian de las palabras reales en que se construyen a partir de letras a las que se le añade algún tipo de modificación, como por ejemplo una pequeña alteración del tamaño, un giro o una modificación de la inclinación. Por último, se pasa al reconocimiento de las palabras generadas usando la metodología hasta ahora explicada. La figura 4.6 muestra un esquema de cómo se han realizado estos experimentos.

Los resultados expresados muestran la tasa de acierto y fallo en palabras (no se consideran los rechazos en las palabras porque siempre se da una palabra como solución); y la tasa de acierto, fallo y rechazo en las letras que componen esas palabras para cada uno de los porcentajes de exclusión de trazos verticales en los cuatro tipos de escenarios posibles. Como se puede apreciar, se ha hecho diferenciación

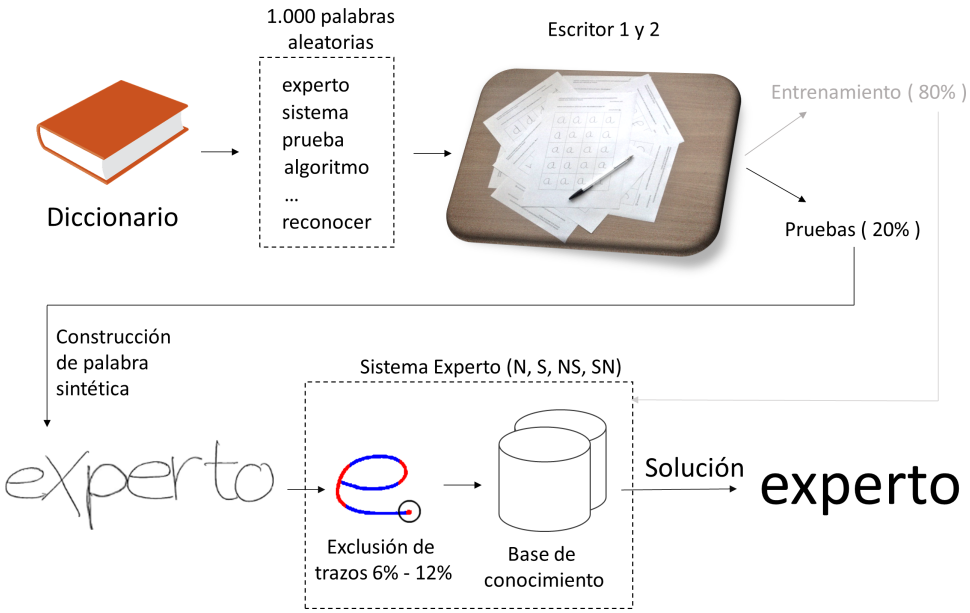


Figura 4.6: Esquema de los ensayos de exclusión de trazos. Se han generado 1000 palabras sintéticas con las letras de cada uno de los escritores y para cada uno de los escenarios posibles.

entre el reconocimiento por palabras y el reconocimiento por letras ya que es posible que se produzca un fallo en el reconocimiento de una palabra pero el reconocimiento de las letras que la componen sea acertado.

A continuación en la tabla 4.1 se muestran las tasas de reconocimiento obtenidas para la exclusión de trazos verticales inferiores al 6%, 7%, 8%, 9%, 10%, 11% y 12%. Cabe reseñar que en esta tabla sólo se muestran las tasas de acierto para el escenario "NS" que es en la que se obtiene el mejor rendimiento. Para ver los resultados con más detalle, se recomienda ver el anexo C.

Tabla 4.1: Tabla resumen de tasas de acierto en palabras para el estudio de exclusión de trazos verticales.

PALABRAS - Con corrector ortográfico								
		Porcentaje de exclusión						
		6%	7%	8%	9%	10%	11%	12%
NS	Escritor 1	**96,30**	**95,90**	**94,10**	**93,50**	**92,40**	**93,20**	**90,20**
	Escritor 2	**93,60**	**96,10**	**94,90**	**95,30**	**96,60**	**96,70**	**96,30**

Como se puede apreciar en la tabla de resultados, la tasa de reconocimiento alcanzado en la exclusión de trazos al 11% para el escritor 2 ofrecen la mejor tasa de reconocimiento en cuanto al reconocimiento de palabras: 96,70%. Para una visión global del conjunto, la figura 4.7 muestra las tasas de acierto en palabras de cada uno de los escritores así como la media para cada uno de los valores de exclusión.

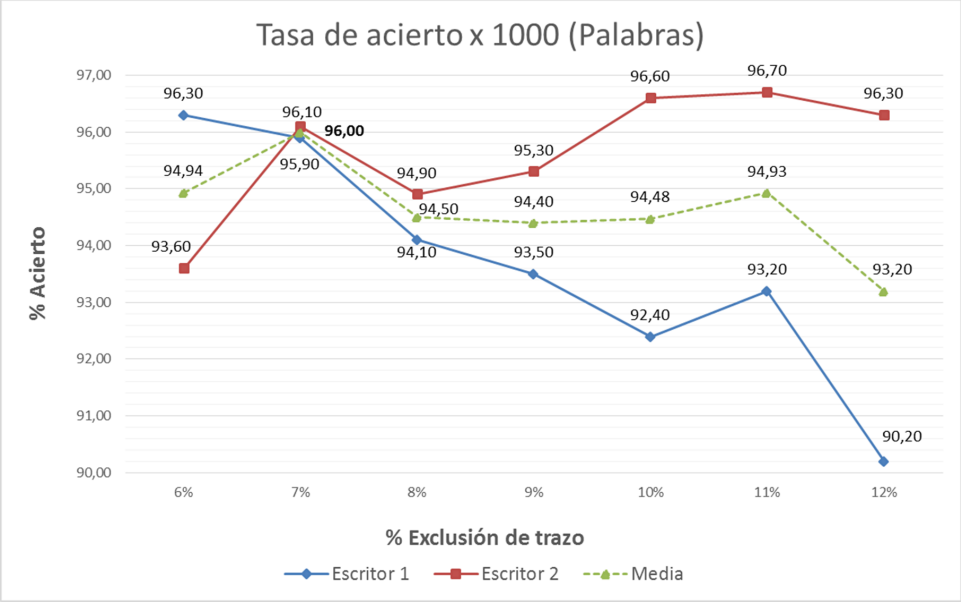


Figura 4.7: Gráfica de las tasas de reconocimiento por palabras. Se representan las tasas de acierto para cada porcentaje de exclusión de cada uno de los escritores así como la media de las tasas.

Por otro lado, también se analizan los resultados obtenidos en las letras de las palabras reconocidas. Se muestra en la tabla 4.2 las tasas de reconocimiento obtenidos en el mismo escenario que en el reconocimiento por palabras "NS" y para el mismo rango de porcentajes de exclusión. Al igual que en el caso de las palabras, también se detallan los resultados en el anexo C.

Tabla 4.2: Tasas de acierto en letras para el estudio de exclusión de trazos verticales.

LETRAS								
		Porcentaje de exclusión						
		6%	7%	8%	9%	10%	11%	12%
NS	Escritor 1	**96,28**	**95,83**	**93,72**	**93,92**	**92,73**	**92,84**	**91,17**
	Escritor 2	**94,26**	**95,16**	**93,96**	**94,10**	**94,30**	**94,51**	**93,41**

En el reconocimiento de letras, la tasa de reconocimiento óptima se encuentran en la exclusión de trazos al 6% para el escritor 1: 96,28%. La figura 4.8 muestra los resultados para las letras también con la media de los resultados para los escritores por cada uno de los valores de exclusión.

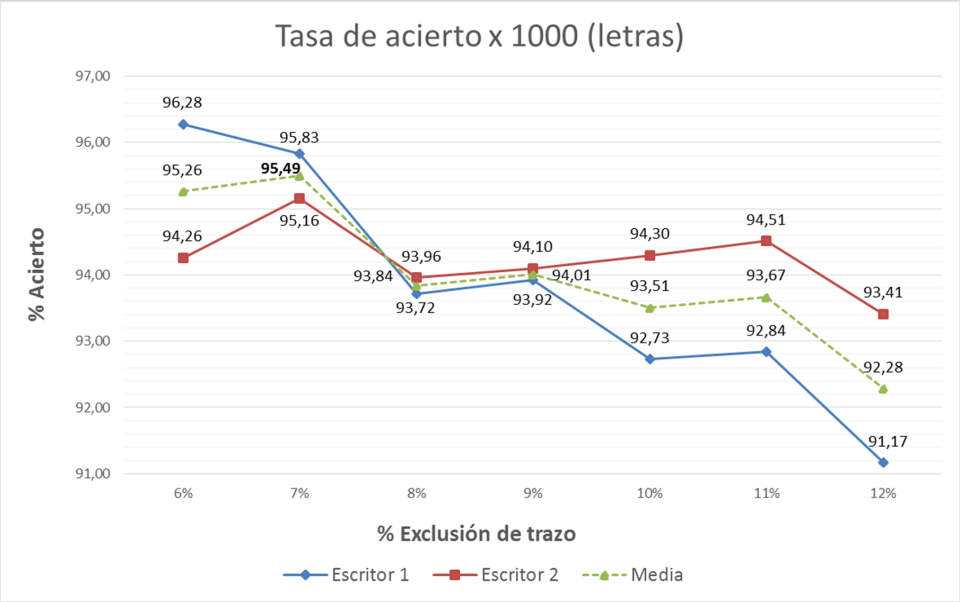


Figura 4.8: Gráfica de las tasas de reconocimiento por letras. Se representan las tasas de acierto para cada porcentaje de exclusión de cada uno de los escritores así como la media de las tasas.

Con el cálculo de las medias en las gráficas, se trata de encontrar un punto donde la tasa de reconocimiento en letras sea óptimo. Cuanto mayor sea esta tasa, los resultados obtenidos en el reconocimiento por palabras serán también óptimos debido a que el corrector ortográfico deberá inferir en menor medida para estimar la solución al reconocimiento de palabras.

Volviendo a la figura 4.7, este punto óptimo se encuentra en el 7% en el estudio con palabras (96,00%), que coincide con el 7% del estudio con letras (95,49%)

de la figura 4.8. Por lo tanto, se establece como valor de exclusión el 7% para la realización de los experimentos alfabéticos.

4.2.3 Aplicación desarrollada: XIRIS

Para la realización de las pruebas y obtención de los resultados se ha desarrollado la aplicación "XIRIS" utilizando el lenguaje JAVA. Esta aplicación permite integrar en una única interfaz gráfica la metodología anteriormente expuesta pasando por el procesamiento previo, segmentación, corrección de inclinación, clasificación, generación de las bases de conocimiento, y el proceso final de reconocimiento mediante el Motor de Inferencia.

Para una mejor visualización, la aplicación consta de dos bloques principales: modo palabra y modo proceso. El *modo palabra* se centra en la generación de una única palabra, mientras que el *modo proceso*, realiza el proceso de reconocimiento a un número de palabras asignado por el usuario.

Modo palabra

Mediante este modo, se permite la generación de una única palabra sintética que el usuario deberá introducir en el cuadro de texto. Tras pulsar el botón "Generar", se crea una imagen con la palabra sintética usando las letras guardadas para pruebas del usuario.

Una vez hecho esto, el usuario puede pulsar el botón "Reconocer palabra" para iniciar el proceso de reconocimiento. Una vez finalizado, en la pestaña de "Detalles" (ver figura 4.9) aparece el resultado previo del reconocimiento sin usar el corrector ortográfico. También aparecen las sugerencias del corrector ortográfico

en una lista (si las hubiere) y finalmente, la palabra candidata seleccionada como solución al problema de reconocimiento.

Figura 4.9: Interfaz gráfica de la aplicación XIRIS: modo "Palabra". Permite generar una palabra sintética aleatoria y reconocerla. En la pestaña de "Detalles" se puede apreciar el resultado obtenido sin pasar por el corrector, las sugerencias del corrector y la palabra finalmente devuelta por el sistema experto.

En esta pantalla es también posible ver el proceso en detalle, observando en la pestaña de "Letras segmentadas" (ver figura 4.10) el proceso aplicado a cada una de

las letras de manera individual. De esta manera, se muestra una lista con la imagen de la letra en cuestión así como el descriptor que la identifica.

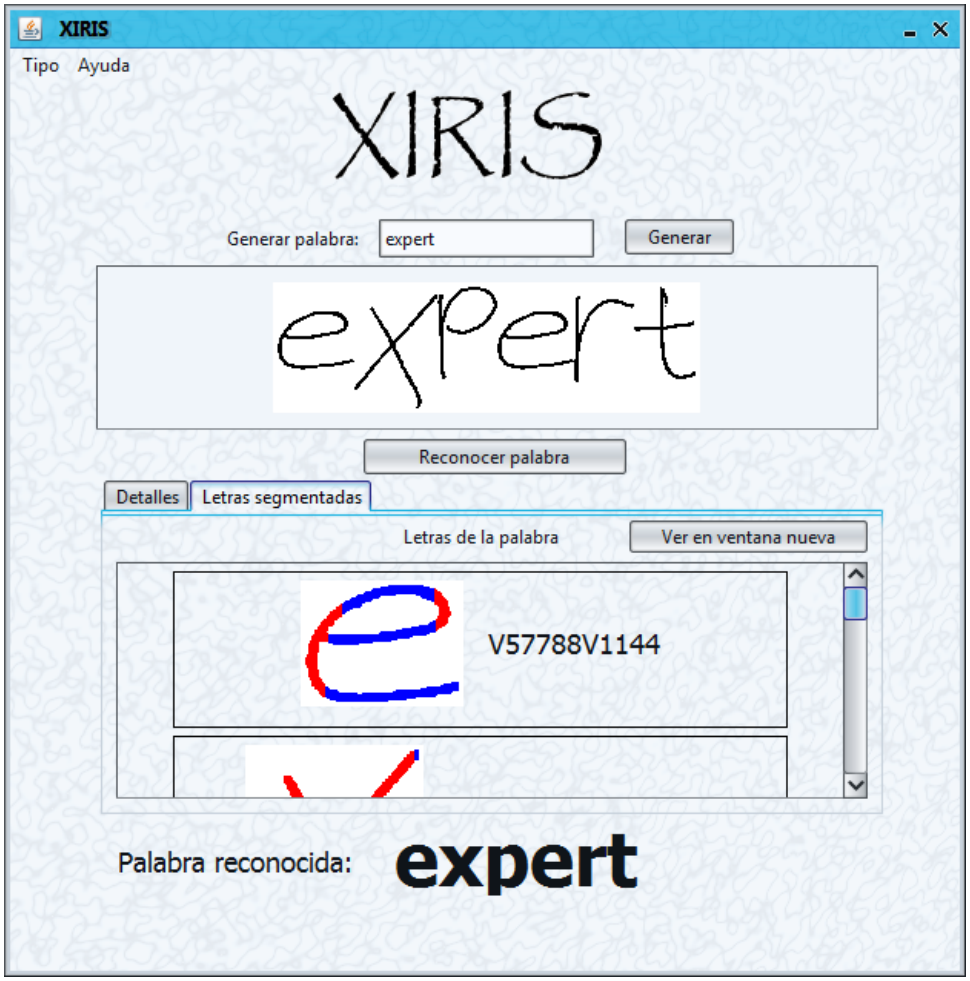


Figura 4.10: Interfaz gráfica de la aplicación XIRIS: modo "Palabra". Permite generar una palabra sintética aleatoria y reconocerla. En la pestaña de "Letras segmentadas" se puede apreciar el resultado de forma unitaria por cada una de las letras así como la cadena representativa. También se muestra el resultado final devuelto por el sistema experto.

Modo proceso

Este modo permite la generación automática de palabras sintéticas aleatorias basadas en un diccionario. Para ello, el usuario debe seleccionar un fichero de texto donde se ubiquen aquellas palabras que el programa elegirá aleatoriamente y que el programa debe reconocer. A continuación se debe introducir el número de palabras que se desean generar usando el fichero anteriormente seleccionado y pulsar sobre el botón "Generar" para iniciar el proceso.

Entre las opciones disponibles se encuentran las siguientes:

- **Check BC-N**: Permite la generación de la base de conocimiento sin la corrección de inclinación para su posterior uso. Solo es necesario generarlo una única vez.
- **Check BC-S**: Permite la generación de la base de conocimiento con la corrección de inclinación para su posterior uso. Sólo es necesario generarlo una única vez.
- **Test** (Sólo se puede seleccionar uno):
  - **N**: Usa sólo la base de conocimiento sin corrección de inclinación.
  - **S**: Usa sólo la base de conocimiento con corrección de inclinación.
  - **NS**: Usa la base de conocimiento sin corrección de inclinación y si se produce un rechazo, utiliza la base de conocimiento con corrección de la inclinación.
  - **SN**: Usa la base de conocimiento con corrección de inclinación y si se produce un rechazo, utiliza la base de conocimiento sin corrección de la inclinación.

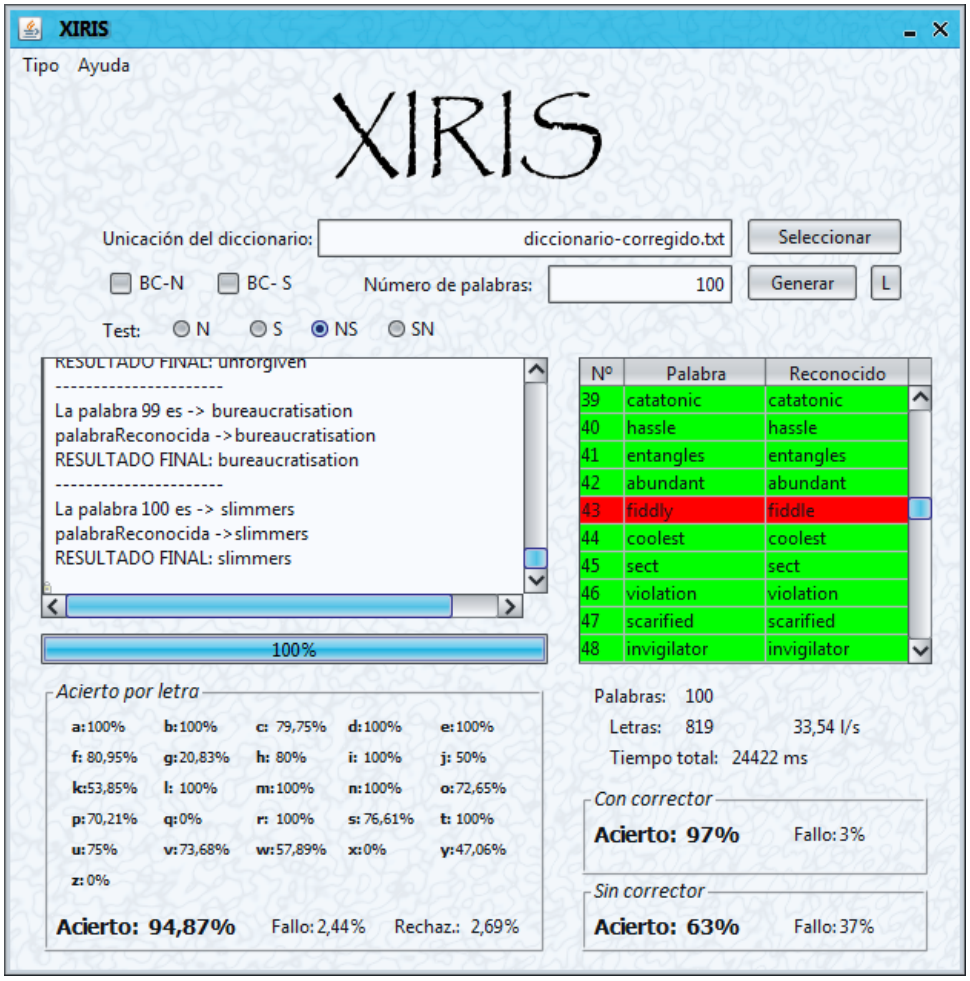


Figura 4.11: Interfaz gráfica de la aplicación XIRIS: modo "Proceso" que muestra toda la información necesaria para ser analizada: acierto con corrector, acierto sin corrector, acierto por letras, etc; para una muestra con 100 palabras.

Una vez finalizado el proceso de generación de palabras sintéticas aleatorias y su posterior reconocimiento, se muestran los diferentes resultados.

Como se puede apreciar en la figura 4.11, los resultados están agrupados por regiones, donde se ve con detalle las distintas operaciones que realiza el proceso.

Se muestra un listado de las palabras generadas junto con su resultado. Si el resultado coincide con la palabra generada, se muestra en color verde. En caso contrario, se muestra en rojo. Por otro lado, se muestra la tasa de acierto existente por cada una de las letras de forma individual así como las tasas de acierto, fallo y rechazo de forma conjunta.

Se muestran también las palabras procesadas de la muestra, el número de letras procesadas correspondientes a las palabras, la velocidad de reconocimiento en letras por segundo y el tiempo necesario para la realización del proceso completo.

Por último, se muestran las tasas de acierto y de fallo usando tanto corrector ortográfico como sin la utilización del mismo.

El programa XIRIS ha sido utilizado para obtener los resultados que se detallan en el apartado siguiente.

4.2.4 Resultados

Para la realización de las pruebas, se ha seleccionado aleatoriamente un conjunto de 10.000 palabras de un diccionario.

Después, se ha dividido cada palabra en las letras que la compone. Por cada una de las letras, se ha elegido al azar una letra con el mismo significado dentro del conjunto reservado para las pruebas de cada uno de los escritores y además, se ha alterado alguna de sus características como por ejemplo, el tamaño, la inclinación o se ha dado un giro; para finalmente construir una palabra sintética con el mismo significado que la seleccionada originalmente.

Este proceso se ha realizado cinco veces. Finalmente, se han procesado un total

de 400.000 palabras, en las que han sido tratadas 3.343.384 letras. La figura 4.12 muestra un esquema general del plan de pruebas llevado a cabo.

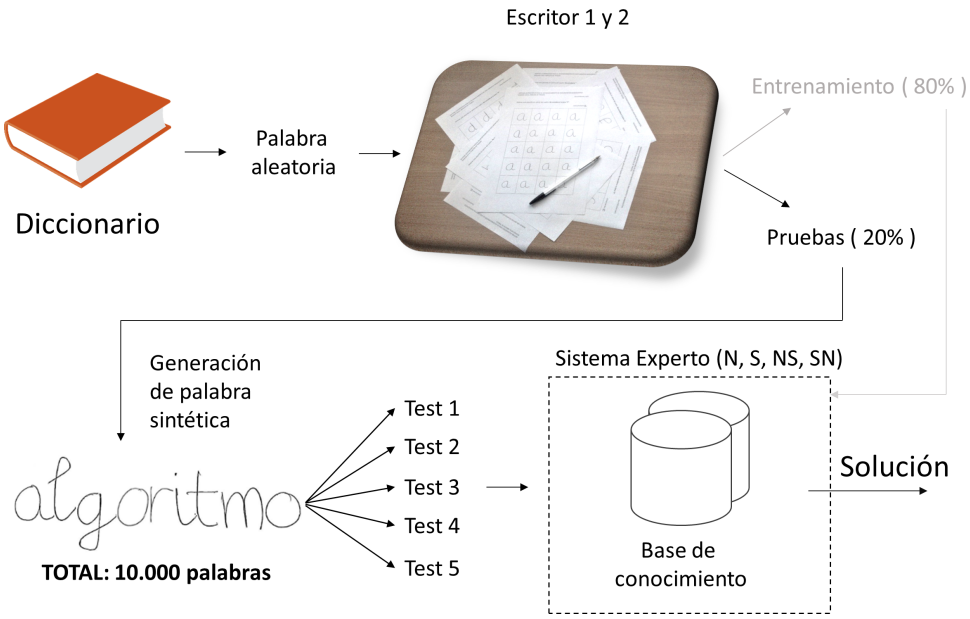


Figura 4.12: Esquema general de las pruebas realizadas. Se realizan 5 test con cada uno de los escritores utilizando en cada test 10.000 palabras sintéticas.

Así pues, para cada una de las pruebas realizadas se han obtenido las tasas de acierto y fallo en el reconocimiento de palabras. Además, también se han recogido las tasas de acierto, fallo y rechazo para las letras que componen esas palabras. Se recomienda ver el anexo F para un análisis más exhaustivo de los resultados.

Las gráficas que se muestran en las figuras siguientes reflejan las tasas de acierto de forma resumida para los resultados obtenidos en los cinco test en el reconocimiento de palabras. En la primera de ellas, la figura 4.13, se muestra la media de las tasas de acierto de los dos escritores agrupados por test usando el corrector ortográfico. Por el contrario, en la segunda gráfica que se muestra en la figura 4.14, se exponen los resultados sin aplicar dicho corrector ortográfico. El objetivo es

poder comprobar la eficacia del uso de un mecanismo de corrección comparando los resultados obtenidos.

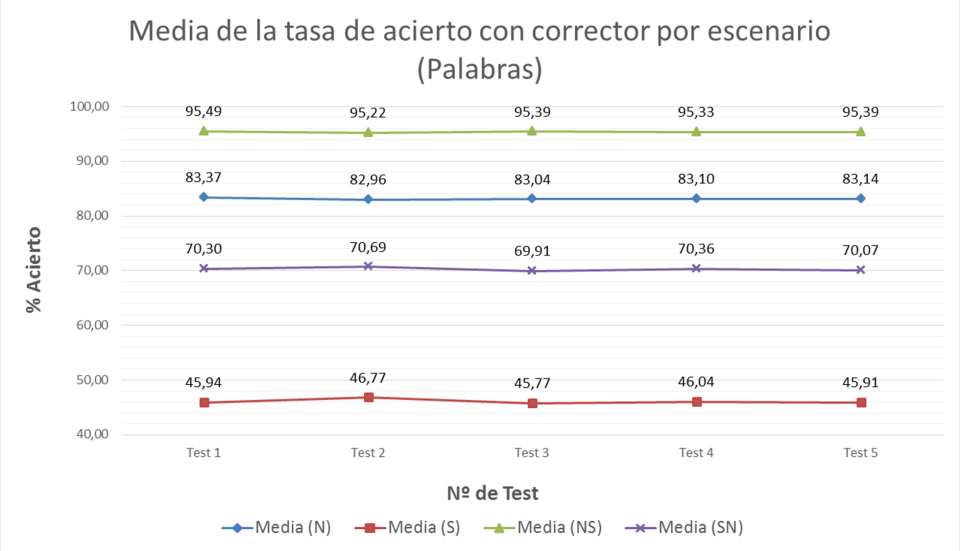


Figura 4.13: Gráfica en el que se representan las medias para cada test en cada uno de los escenarios para el reconocimiento de palabras usando corrector ortográfico.

De igual modo, se expresan en la tabla 4.3 los resultados finales de los rendimientos obtenidos en el reconocimiento por palabras en cada uno de los escenarios.

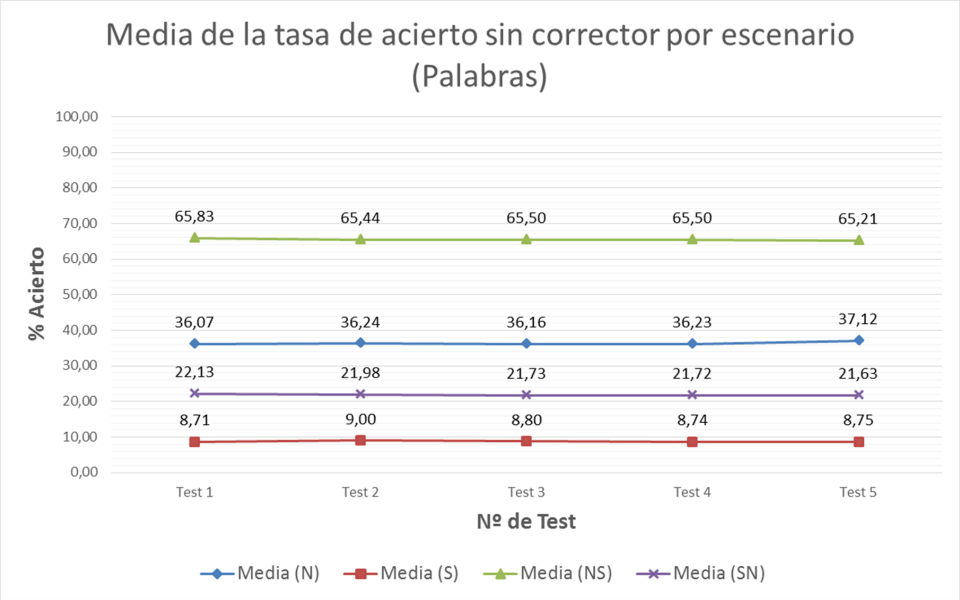


Figura 4.14: Gráfica en el que se representan las medias para cada test en cada uno de los escenarios para el reconocimiento de palabras sin usar un mecanismo de corrección.

Tabla 4.3: Media de las tasas de reconocimiento de palabra para cada uno de los escenarios.

MEDIA POR ESCENARIO - PALABRAS						
	Con corrector			Sin corrector		
	acierto	fallo	rechazo	acierto	fallo	rechazo
N	83,12	16,88	x	36,36	63,64	x
S	46,09	53,95	x	8,80	91,20	x
NS	**95,36**	4,64	x	65,50	34,51	x
SN	70,26	29,84	x	21,84	77,96	x

Cabe destacar que en el reconocimiento de palabras no se han considerado los

rechazos debido a que, al ser una palabra, si no se identifica un carácter de la misma se asume como un fallo, ya que no pueden existir palabras con huecos en las mismas. Por lo tanto, dependiendo del escenario en el que se esté realizando el test, se realizan las siguientes acciones: Si es un escenario "N" o "S" (aquellos que se componen sólo de una base de conocimiento) se sustituye el rechazo por una letra al azar. Por el contrario, si se trata de un escenario "NS" o "SN" (aquellos que se componen de dos bases de conocimiento), se intenta reconocer con la primera base de conocimiento y si se produce un rechazo, se hace un nuevo intento de reconocimiento sobre la segunda base de conocimiento. Si aun así, no es posible dar una solución al carácter introducido, se anota como un rechazo a nivel de letra y al igual que en el caso anterior, se sustituye el rechazo por una letra al azar. Esto permite realizar un último intento de reconocimiento a nivel global de palabra mediante el corrector ortográfico.

Si se observa en detalle los resultados obtenidos, se puede apreciar claramente que el uso de un corrector ortográfico influye de manera notable en las tasas de reconocimiento independientemente del escenario en el que se aplique. Además, entre los resultados obtenidos con el uso de un corrector ortográfico, existen a su vez diferencias significativas entre las tasas de acierto de los cuatro escenarios posibles, siendo el escenario "NS" el que mejor resultados registra (aunque no son excesivamente elevados). Por el contrario, el que peor resultados ofrece es aquel escenario cuya base de conocimiento almacena las cadenas representativas de los caracteres con corrección de inclinación aplicado: el escenario "S".

Con el objeto de poder ampliar la discusión de estos resultados, al igual que en el caso de las palabras, también se muestra en la figura 4.15 la media de los resultados de los cinco test por escenario pero en este caso, para el reconocimiento de letras que componen las palabras de las pruebas realizadas. Además, en la tabla 4.4 se detallan los resultados finales de las tasas obtenidas en el reconocimiento de las letras en cada uno de los escenarios. A diferencia de los resultados mostrados en el reconocimiento de palabras, en este caso sí se muestran los rechazos producidos

cuyo fin no es más que con propósitos estadísticos.

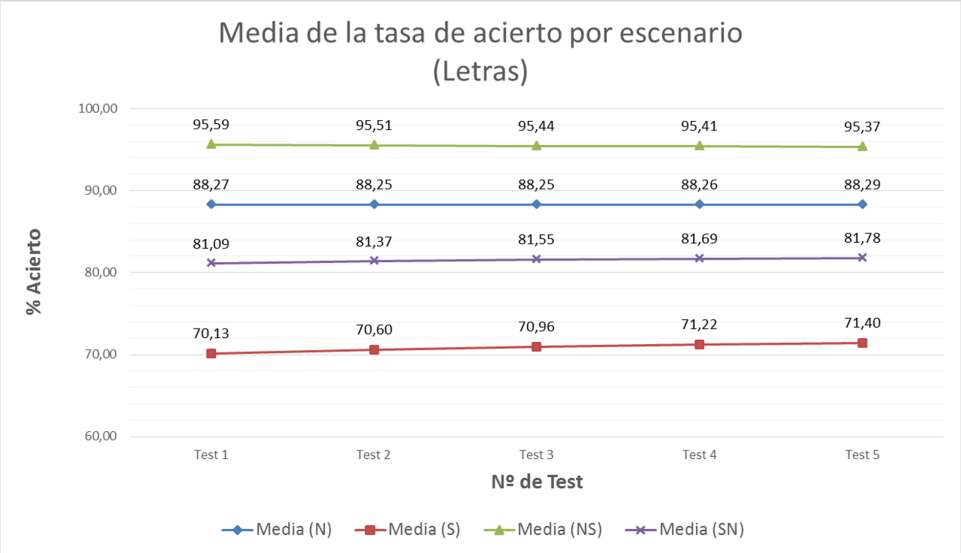


Figura 4.15: Gráfica en el que se representan las medias para cada test en cada uno de los escenarios para el reconocimiento de letras.

Tabla 4.4: Media de las tasas de reconocimiento de letras para cada uno de los escenarios.

MEDIA POR ESCENARIO - LETRAS			
	acierto	fallo	rechazo
N	88,26	3,01	8,72
S	70,86	5,67	23,47
NS	**95,46**	2,33	2,20
SN	81,49	6,27	12,23

De esta manera, si se comparan los resultados obtenidos entre las tasas de reconocimiento de palabras (tabla 4.3) y las tasas de reconocimiento de letras (tabla

4.4), se puede apreciar que existe una estrecha relación entre la tasa de acierto en palabras y en las letras. Cuanto mayor es el índice de aciertos en las letras, mejor resultado se obtendrá en el reconocimiento en palabras. Por lo tanto, si el índice de acierto es elevado en letras, a nivel de reconocimiento de palabra, el corrector ortográfico tendrá que influir en menor medida y será más probable obtener un acierto. Esto ocurre, por ejemplo, en el escenario "NS". Por el contrario, si la tasa de acierto en letras no es demasiado elevada como ocurre en el escenario "S", el sistema deberá sustituir las letras no encontradas, es decir, los rechazos producidos por letras aleatorias. De esta manera, a nivel de reconocimiento de palabra, es más difícil para el corrector ortográfico estimar una solución debido a que, cuantas más letras hayan sido sustituidas aleatoriamente, más variantes de la solución a buscar existen.

Si se realiza una clasificación descendente de los escenarios según su tasa de acierto ("NS", "N", "SN", "S"), se puede observar fácilmente que aquellos escenarios en los que sólo se usa o que usan inicialmente una base de conocimiento cuyos caracteres han experimentado una corrección de inclinación, son los que ofrecen peores resultados. Ante esta situación, sólo cabe mencionar que se necesita realizar un análisis más profundo sobre el algoritmo de corrección de inclinación. De esta manera, se deberían probar otros métodos más complejos que realicen otro tipo de transformaciones a los expuestos en la metodología.

En cuanto a los errores detectados, se producen en la mayoría de los casos en rotaciones entre letras parecidas como puede ser por ejemplo la "u" y la "v", entre la "i" y la "j" o en caso de la "x" y la "k". Pero estos errores son fácilmente identificados a nivel de palabra por el corrector ortográfico y es posible estimar la solución adecuada.

Otro factor no menos importante es la forma en que los diferentes caracteres han sido escritos por sus correspondientes escritores ya que esto influye en gran

medida en el rendimiento del sistema experto. Los resultados obtenidos por ambos escritores, ya sean en palabras o letras, que se muestran en los anexos C y F, denotan que existen diferencias significativas.

Otro factor interesante a tener en cuenta es el estudio de los tiempos de reconocimiento. En este caso, se ha utilizado un equipo con las características que se muestran a continuación en la tabla 4.5.

Tabla 4.5: Características del equipo utilizado en las pruebas.

CARACTERÍSTICAS	
Procesador	Intel(R) Core(TM) *i*5-2400 CPU @ 3.10Ghz
Memoria RAM	4,00 GB
Tipo de sistema	64 bits
Versión JAVA	1.7.0_55

Para el equipo con las características arriba señaladas, los tiempos de reconocimiento obtenidos en cada uno de los test y por cada uno de los escritores en los diferentes escenarios posibles, vienen reflejados en la tabla 4.6. Los resultados se muestran en letras por segundo.

Aquellos escenarios que ofrecen mejores resultados son los que realizan búsquedas en una sola base de conocimiento lo que puede ser hasta cierto punto lógico ya que estos sistemas expertos son más sencillos frente a los que se componen de dos bases de conocimiento.

También existen diferencias reseñables entre los tiempos de los dos escritores y esto es debido a la forma en que se escriben los caracteres, ya sea por las curvaturas, el tamaño, la inclinación, etc; todos estos factores influyen en el tiempo que se debe invertir en realizar el proceso de reconocimiento (sobre todo en las etapas previas a la clasificación).

Tabla 4.6: Tiempos de ejecución para cada uno de los escritores por escenario expresado en letras por segundo.

TIEMPOS (Letras/seg.)							
		Test 1	Test 2	Test 3	Test 4	Test 5	Media
N	Escritor 1	40,37	40,39	40,22	40,39	40,53	40,38
	Escritor 2	42,65	41,97	42,35	42,10	42,50	42,31
S	Escritor 1	40,26	40,32	40,44	40,53	40,39	40,39
	Escritor 2	42,62	42,37	42,22	42,27	42,38	**42,37**
NS	Escritor 1	34,70	34,71	34,27	34,57	34,44	34,54
	Escritor 2	38,57	38,44	38,44	38,10	38,23	38,36
SN	Escritor 1	28,60	28,45	28,26	28,49	29,17	28,59
	Escritor 2	34,31	34,22	33,88	33,99	34,27	34,13

Así pues, para cada uno de los escenarios existe una diferencia de tiempos en los rendimientos de los escritores que se expresan en la tabla 4.7.

Tabla 4.7: Tabla de diferencia de tiempos entre los rendimientos de los escritores.

RENDIMIENTOS		
	Equiparación de tiempos (Escritor 2 / Escritor 1)	Porcentaje
N	42,32/40,38 = 1,0478	4,78%
S	42,37/40,39 = 1,0490	4,90%
NS	38,36/34,54 = 1,1106	11,06%
SN	34,13 / 28,59 = 1,2064	20,64%

4.3 Reconocimiento numérico

4.3.1 Diseño de experimentos

Para la realización de experimentos numéricos muchos autores utilizan la base de datos MNIST de números escritos a mano. Esta es una de las razones de mayor peso para su uso ya que los resultados obtenidos en este trabajo pueden ser comparados con los resultados de otros autores. Además esta base de datos es libre y viene ofrecida por LeCun et al. [88] [1] de forma gratuita.

La base de datos MNIST consta de 70.000 imágenes de caracteres numéricos recogidas a partir de la muestra de aproximadamente 250 autores. Se encuentra dividida en dos grandes grupos: 60.000 imágenes con fines de entrenamiento y 10.000 para la realización de pruebas. Las imágenes que contiene están normalizadas a escala de grises usando técnicas de anti-aliasing. Además, han sido centradas en un área de $28x28$ píxeles tomando como referencia para el punto medio el cálculo del centro de masas de la propia imagen.

La figura 4.16 muestra la distribución de la muestra en los conjuntos de imágenes para entrenamiento y pruebas. La ventaja que ofrece la base de datos MNIST es que sus datos ya vienen separados y preparados para el desarrollo de las pruebas.

[1] http://yann.lecun.com/exdb/mnist/

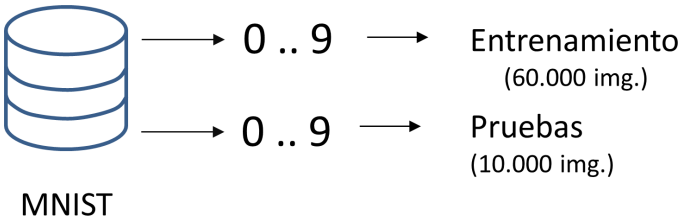


Figura 4.16: Distribución de la muestra en los conjuntos de entrenamiento y pruebas sacados de las base de datos MNIST.

Antes de realizar los experimentos concluyentes, se debe realizar una serie de operaciones para alcanzar la optimización en el reconocimiento. Entre estas operaciones se encuentra la búsqueda del tamaño óptimo de la imagen mediante el reescalado de la misma, la búsqueda del porcentaje óptimo de exclusión de trazos al igual que se ha hecho en el reconocimiento alfabético y la búsqueda del mejor conjunto de datos para el entrenamiento de la base de conocimiento.

Por otro lado, en los experimentos con números también se han considerado los cuatro escenarios presentados en el diseño de los experimentos alfabéticos: *Escenario N*, *Escenario S*, *Escenario NS* y *escenario SN*. De esta manera, se permite probar definitivamente ante datos numéricos de prueba reales cuál es el escenario que ofrece mejores resultados.

4.3.2 Optimización del reconocimiento

Las imágenes que ofrece la base de datos MNIST no son del mismo tamaño que las imágenes recogidas de los dos escritores en la parte alfabética de los experimentos. Por ello, se ha precisado del reescalado de las imágenes numéricas buscando el tamaño óptimo.

Para esta prueba, se han seleccionado 1000 imágenes numéricas aleatorias y

se les ha aplicado un reescalado a la imagen de escala x considerando el rango $x \in [1-6]$. Después, utilizando el tipo de escenario más simple, el *escenario N*, y utilizando siempre la misma muestra de 1000 imágenes numéricas, se ha realizado el reconocimiento para cada escala obteniéndose los resultados que se detallan en la figura 4.17.

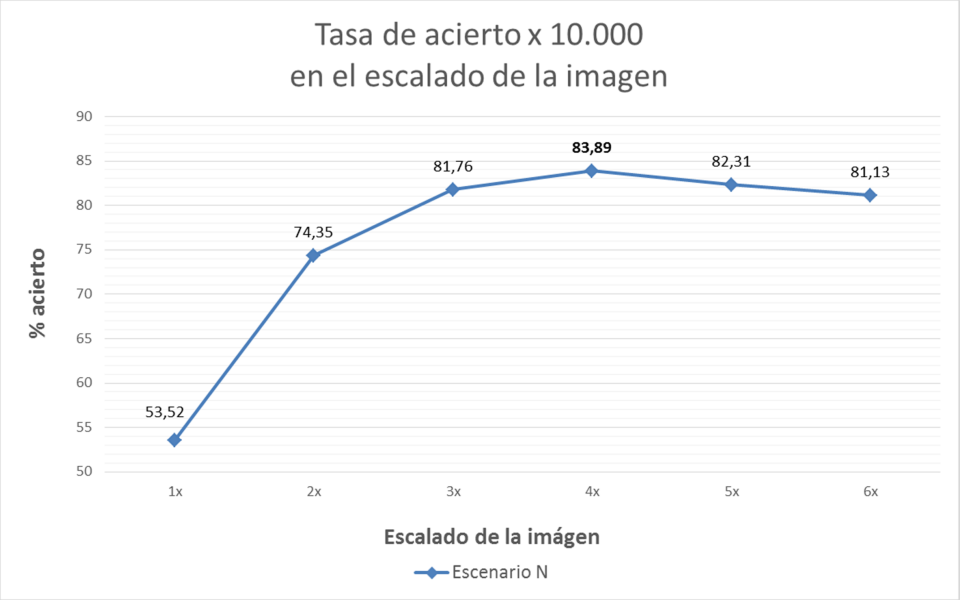


Figura 4.17: Gráfico de la evolución de las tasas de acierto en el escalado de la imagen.

Tabla 4.8: Tasas de acierto, fallo y rechazo obtenidos en el *escenario N* al aplicar el reescalado en las imágenes.

	Escala de la imagen					
	$1x$	$2x$	$3x$	$4x$	$5x$	$6x$
Acierto	53,52	74,35	81,76	**83,89**	82,31	81,13
Fallo	45,99	24,01	14,52	10,82	10,44	10,45
Rechazo	0,49	1,64	3,72	5,29	7,25	8,42

Como se puede apreciar en la tabla 4.8, existe un compromiso entre Acierto, Fallo y Rechazo entre las escalas: mientras que la tasa de fallo va disminuyendo

según se aplica una escala mayor, el índice de rechazo va aumentando. En la escala $4x$, la tasa de fallo es muy similar al producido en escalas posteriores (sólo existe una diferencia del 0,42% aproximadamente); sin embargo, la diferencia en la tasa de acierto se encuentra alrededor de un 2% respecto a las escalas $5x$ y $6x$. Por otro lado, en cuanto a la tasa de rechazo, mientras que en la escala $3x$ se logra mejorar aproximadamente 1,5% la tasa respecto a la escala $4x$, no compensa la pérdida producida de casi un 4% en la tasa de fallo. Por tanto, se ha aplicado a las imágenes originales de $28x28$ píxeles un escalado de $4x$, lo que implica que las imágenes óptimas para el método de reconocimiento de este trabajo tienen una dimensión de $112x112$ píxeles.

Como ya se ha comentado anteriormente, las imágenes de MNIST son de diferente naturaleza que las obtenidas en los experimentos alfabéticos y por tanto, también requieren de la búsqueda del porcentaje óptimo de exclusión de trazos. Por ello, se debe buscar el tamaño de trazo vertical de descarte en función del mayor de los trazos verticales hallados para el carácter numérico analizado en cuestión. Este proceso se realiza de la misma manera que se ha hecho en la optimización del reconocimiento alfabético siguiendo la ecuación 3.8. En concreto, para los experimentos con imágenes numéricas se ha considerado δ en el rango $\delta \in [10\%,16\%]$ ya que se trata de imágenes de diferente dimensión a las imágenes alfabéticas.

Para la realización de estas pruebas, se han creado las bases de conocimiento para los diferentes escenarios: *escenario N*, *escenario S*, *escenario NS* y *escenario SN*. Después se han seleccionado 1.000 imágenes del carácter 0, 1.000 imágenes del carácter 1, y así sucesivamente hasta el carácter 9 creando una muestra final de 10.000 imágenes extraídas de la base de datos MNIST para ser reconocidas por las bases de conocimiento previamente generadas. La figura 4.18 muestra un esquema de cómo se han realizado estos experimentos.

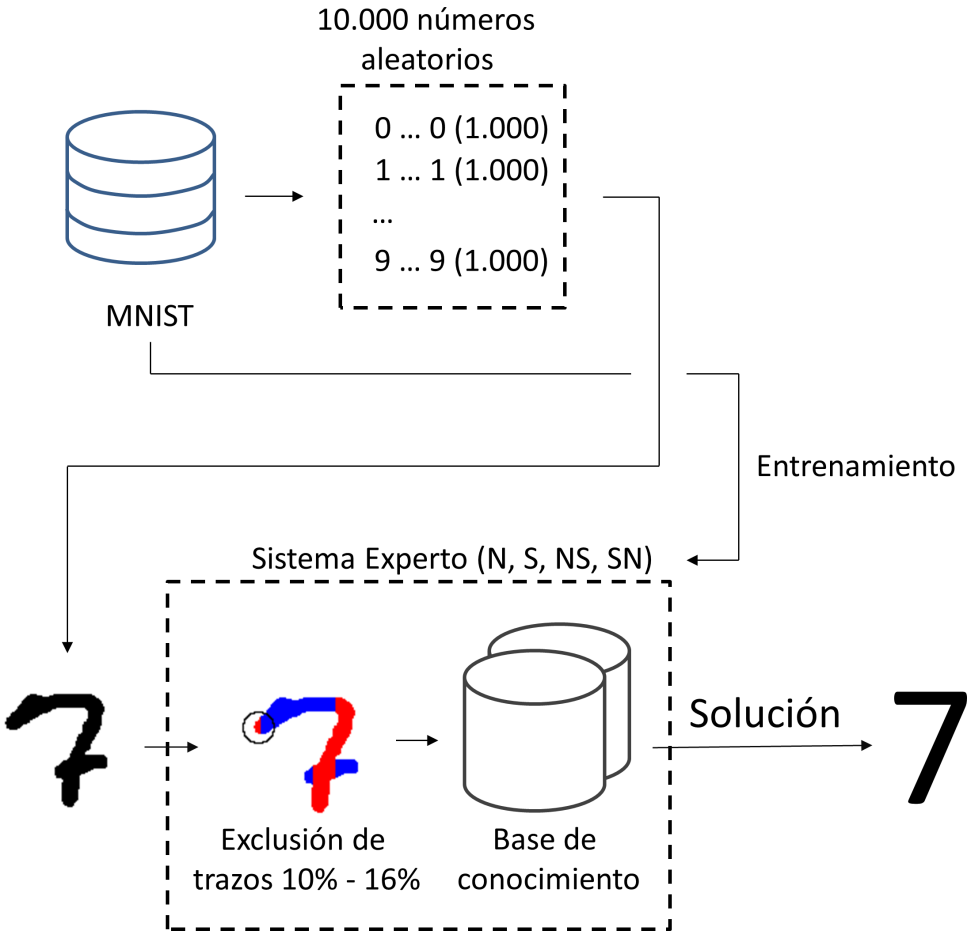


Figura 4.18: Esquema de los ensayos de exclusión de trazos. Se han generado 1.000 palabras sintéticas con las letras de cada uno de los escritores y para cada uno de los escenarios posibles.

En la tabla 4.9 se detallan solamente los índices de acierto obtenidos en los escenarios descritos anteriormente pero en este caso para el rango de exclusión [10%,16%]. Para una observación detallada de los resultados, se recomienda ver el anexo D.

Ante los resultados expuestos en la tabla 4.9, puede verificarse que el mejor re-

Tabla 4.9: Tasas de acierto en letras para el estudio de exclusión de trazos verticales

	Porcentaje de exclusión						
	10%	11%	12%	13%	14%	15%	16%
Escenario N	83,14	83,30	83,58	83,75	83,89	83,86	83,85
Escenario S	80,63	80,66	80,80	80,87	80,95	80,79	80,64
Escenario NS	85,96	86,06	86,29	86,36	**86,47**	86,39	86,39
Escenario SN	84,75	84,76	84,90	84,93	84,93	84,80	84,66

sultado se obtiene en el *escenario NS* con un índice de exclusión del 14%, obteniéndose una tasa de reconocimiento del 86,47%. Es por este motivo, por el que se establece como valor de exclusión de trazos verticales el 14% para la realización de los experimentos numéricos.

Otro de los procesos involucrados en la optimización del reconocimiento es la búsqueda del mejor conjunto de datos dentro de la base de datos MNIST para el entrenamiento de la base de conocimiento.

Para lograr esto, de las 60.000 imágenes que MNIST tiene destinadas para el entrenamiento, se han seleccionado aleatoriamente 10.000 como conjunto de validación. Después, de las 50.000 imágenes restantes, se han seleccionado primeramente conjuntos de 10.000 imágenes aleatorias para generar las bases de conocimiento y ser probadas posteriormente contra la muestra de validación. Este proceso se realiza 10 veces. Una vez finalizado la serie de pruebas, se incrementa la muestra en 10.000 imágenes y se repite de nuevo la serie de 10 pruebas. El procedimiento se reitera hasta alcanzar la muestra completa de 50.000 imágenes. La figura 4.19 detalla el esquema para la selección del mejor conjunto de pruebas tal y como se ha mencionado.

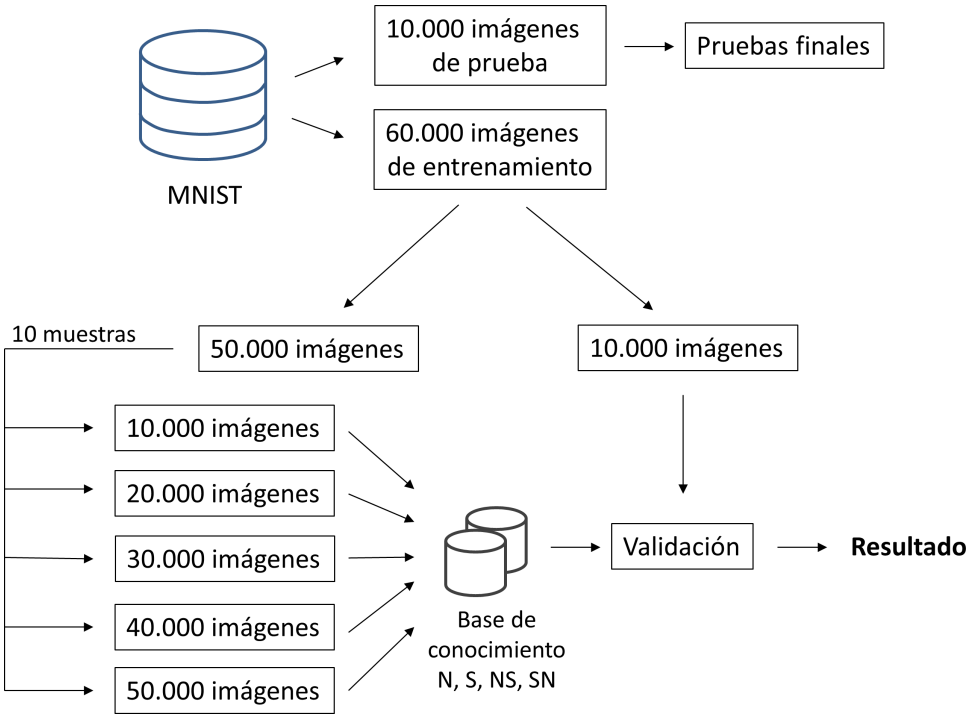


Figura 4.19: Esquema para obtener el mejor conjunto de pruebas numéricas a partir de la base de datos MNIST.

Los resultados obtenidos tras la realización de estas pruebas se reflejan de forma resumida en la figura 4.20. En esta figura sólo se señalan las máximas tasas de acierto obtenidas para cada tamaño de muestra y por escenario. Para ver en más detalle estos resultados, se recomienda ver el anexo E.

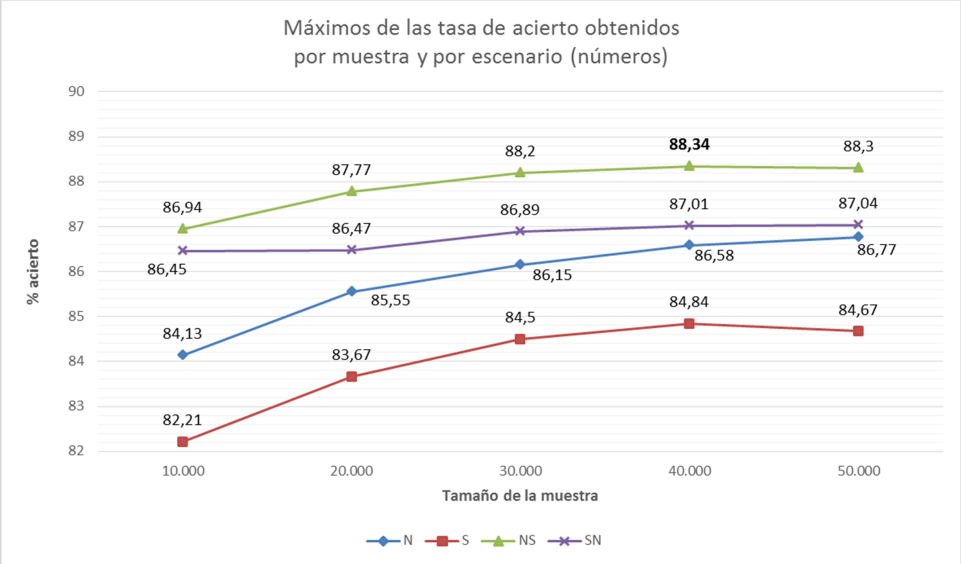


Figura 4.20: Gráfico que expresa las máximas tasas de acierto en la búsqueda del mejor conjunto de pruebas. Los resultados se detallan por tamaño de muestra y por escenario.

Si se entra más en detalle sobre los resultados obtenidos en el *escenario NS* de la gráfica anterior, se puede apreciar que la mejor tasa de reconocimiento, 88,34%, se obtiene en concreto sobre el Nº de test 0 en el tamaño de muestra 40.000 tal como señala la tabla 4.10. Esta es por tanto, la muestra que se va a utilizar para la realización de las pruebas definitivas.

Tabla 4.10: Tasas de acierto en el escenario NS en la búsqueda del mejor conjunto de pruebas.

Nº de Test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	86,46	87,73	88,04	**88,34**	88,30
1	86,90	87,76	87,87	88,18	
2	86,49	87,62	87,83	88,13	
3	86,78	87,69	88,13	88,05	
4	86,45	87,49	88,04	88,07	
5	86,70	87,54	88,04	88,20	
6	86,94	87,62	88,20	88,26	
7	86,80	87,64	88,05	88,16	
8	86,62	87,56	87,65	88,19	
9	86,78	87,77	87,95	88,21	

4.3.3 Aplicación desarrollada: XIRIS

Como se ha comentado anteriormente en la sección 4.2.3, XIRIS permite el reconocimiento de caracteres alfabéticos. Pero además, mediante el *modo Numérico* permite realizar pruebas sobre caracteres numéricos de la base de datos MNIST expresando los resultados obtenidos de forma visual.

Modo Numérico

Este modo está diseñado para probar la eficacia de la metodología expuesta en este trabajo sobre caracteres numéricos usando la base de datos MNIST.

Para la generación de una nueva base de conocimiento, el usuario debe indicar el tamaño de la muestra empleada para el entrenamiento de la misma usando como elementos los catalogados para entrenamiento en la base de datos MNIST. Después deberá seleccionar qué tipo de base de conocimiento desea crear seleccionando *BC-N* para una base de conocimiento normal, sin aplicar el proceso de corrección de inclinación en sus entradas, o *BC-S* para una base de conocimiento con corrección de inclinación aplicada a sus elementos. También se pueden seleccionar la generación de ambas si se desea hacer pruebas sobre los dos tipos.

De igual modo, se debe introducir el número de elementos de prueba indicando cuántos elementos de la base de datos MNIST destinados a pruebas se deben seleccionar.

Entre las opciones disponibles se encuentran las siguientes:

- Tipo de test (sólo se puede seleccionar uno):
  - **N**: Usa sólo la base de conocimiento sin corrección de inclinación.
  - **S**: Usa sólo la base de conocimiento con corrección de inclinación.
  - **NS**: Usa la base de conocimiento sin corrección de inclinación y si se produce un rechazo, utiliza la base de conocimiento con corrección de la inclinación.
  - **SN**: Usa la base de conocimiento con corrección de inclinación y si se produce un rechazo, utiliza la base de conocimiento sin corrección de la inclinación.
- Borrar la muestra existente.

Para lanzar la prueba, el usuario debe pulsar sobre el botón "Ejecutar". Si pulsa sobre el botón "L" limpiará todos los datos existentes en la pantalla. La figura 4.21 muestra un ejemplo de la ejecución de una prueba.

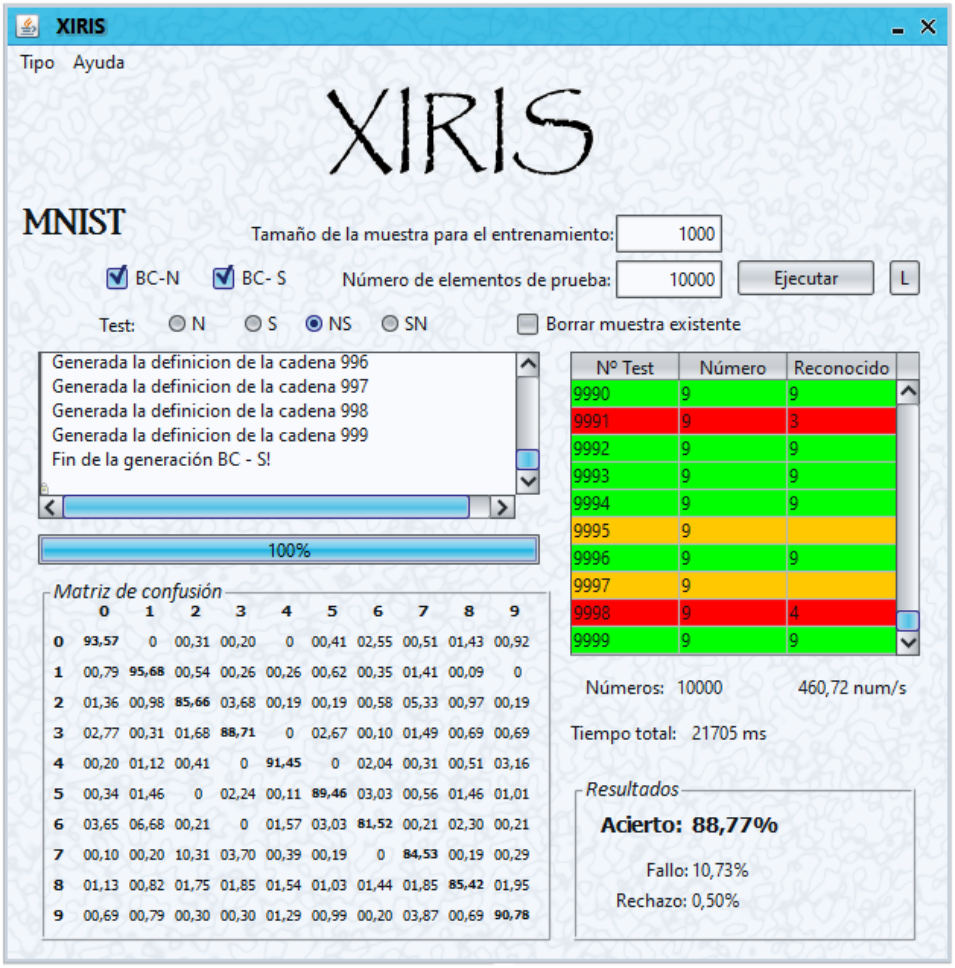


Figura 4.21: Interfaz gráfica de la aplicación XIRIS: modo "Numérico" que muestra toda la información después de ser procesada. Se muestra además la matriz de confusión de los experimentos realizados para una muestra con 10.000 números.

Una vez finalizada la ejecución del proceso, los resultados son mostrados en las diferentes secciones del programa. En un área de texto, se muestra la salida que el proceso va lanzando según se va ejecutando. Asimismo, la barra de progreso indica el porcentaje de ejecución de la prueba.

Por otro lado, se muestra también el listado de los números seleccionados junto con el resultado obtenido en el reconocimiento. Si el resultado coincide con el elemento seleccionado, se muestra en color verde (Acierto). Por el contrario, si no coincide, se muestra en color rojo (Fallo). En el caso de que no se identifique al carácter introducido, se muestra como naranja (Rechazo).

Se indica también el número de elementos procesados, la tasa de reconocimiento expresada en números por segundo así como el tiempo total de ejecución del proceso.

Otra región detalla la matriz de confusión en la que se señala la tasa de acierto por número así como la tasa de intercambio con otros números.

Finalmente, se muestran las tasas finales de acierto, fallo y rechazo alcanzadas en la prueba.

4.3.4 Resultados

Para la realización de los ensayos finales, se ha utilizado nuevamente la base de datos MNIST pero en este caso, se han considerado no sólo los datos para entrenamiento sino también los existentes para pruebas. Para la generación de la base de conocimiento, se ha seleccionado el *escenario NS* ya que durante las tres pruebas realizadas en el proceso de optimización ha sido el escenario que mejores resultados ha aportado. De igual modo, se ha seleccionado el conjunto de datos de muestra de 40.000 imágenes correspondiente al test 0 del proceso de optimización como elementos a ser introducidos en la base de conocimiento. Por otro lado, se ha especificado una escala de $4x$ para redimensionar las imágenes de entrada en la etapa de preprocesamiento y se ha establecido una tasa de exclusión de trazos verticales del 14%. Se presenta un esquema de los elementos que intervienen para

las pruebas finales en la figura 4.22.

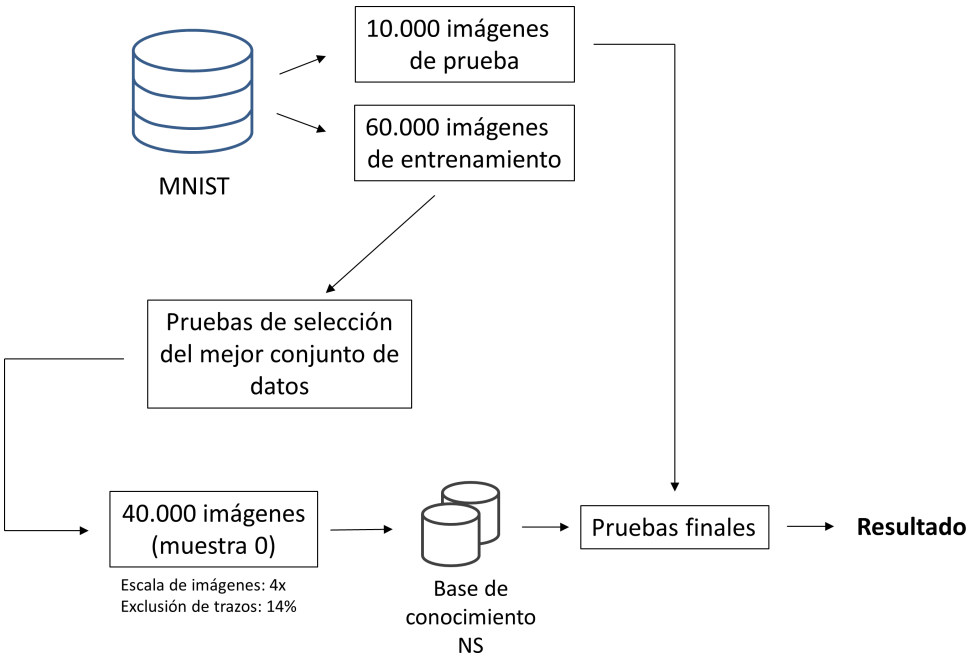


Figura 4.22: Esquema para la realización de las pruebas finales. Se seleccionan los parámetros obtenidos en los estudios anteriores para ajustar el proceso de manera que se obtengan los mejores resultados.

Una vez la base de conocimiento ha sido generada, se utiliza el conjunto de las 10.000 imágenes de pruebas de la base de datos MNIST para obtener los resultados finales. Las tasas de acierto, fallo y rechazo se muestran en la tabla 4.11 tanto para el conjunto total como para cada clase numérica. Asimismo, se muestra la matriz de confusión desglosada por cada clase de número indicando el porcentaje de intercambios con otras clases en los fallos producidos.

Tabla 4.11: Matriz de confusión obtenida en los experimentos numéricos. También se muestran las tasas de acierto, fallo y rechazo alcanzados durante la ejecución de las pruebas.

	% Acierto	% Fallo										% Rechazo
Dígito		0	1	2	3	4	5	6	7	8	9	
0	93,57	0	0	0,31	0,20	0	0,41	2,55	0,51	1,43	0,92	0,10
1	95,68	0,79	0	0,54	0,26	0,26	0,62	0,35	1,41	0,09	0	0
2	85,66	1,36	0,98	0	3,68	0,19	0,19	0,58	5,33	0,97	0,19	0,87
3	88,71	2,77	0,31	1,68	0	0	2,67	0,10	1,49	0,69	0,69	0,89
4	91,45	0,20	1,12	0,41	0	0	0	2,04	0,31	0,51	3,16	0,80
5	89,46	0,34	1,46	0	2,24	0,11	0	3,03	0,56	1,46	1,01	0,33
6	81,52	3,65	6,68	0,21	0	1,57	3,03	0	0,21	2,30	0,21	0,62
7	84,53	0,10	0,20	10,31	3,70	0,39	0,19	0	0	0,19	0,29	0,10
8	85,42	1,13	0,82	1,75	1,85	1,54	1,03	1,44	1,85	0	1,95	1,22
9	90,78	0,69	0,79	0,30	0,30	1,29	0,99	0,20	3,87	0,69	0	0,10
Total	**88,77%**					**10,73%**						**0,50%**

La tasa final de acierto en el reconocimiento es de un 88,77%. Sin embargo, si se realiza una evaluación por número se puede observar que la mejor tasa, con un 95,68%, se produce al identificar el número 1. Por el contrario, los peores índices se registran al intentar identificar el número 6 con un 81.52% puesto que se producen confusiones con el número 1 en mayor medida (6,68%) y con el 0 (3,65%) y 5 (3,03%) de forma más moderada. No menos importante es la confusión producida entre los números 2 y 7 con un 5,33% y entre el 7 y 2 más severo con un 10,31%. Otra confusión importante es la producida entre el número 9 y el número 7 siendo un 3,87% del total de los fallos producidos ante este número. La figura 4.23 algunas de las imágenes de los números en los que se han producido intercambios.

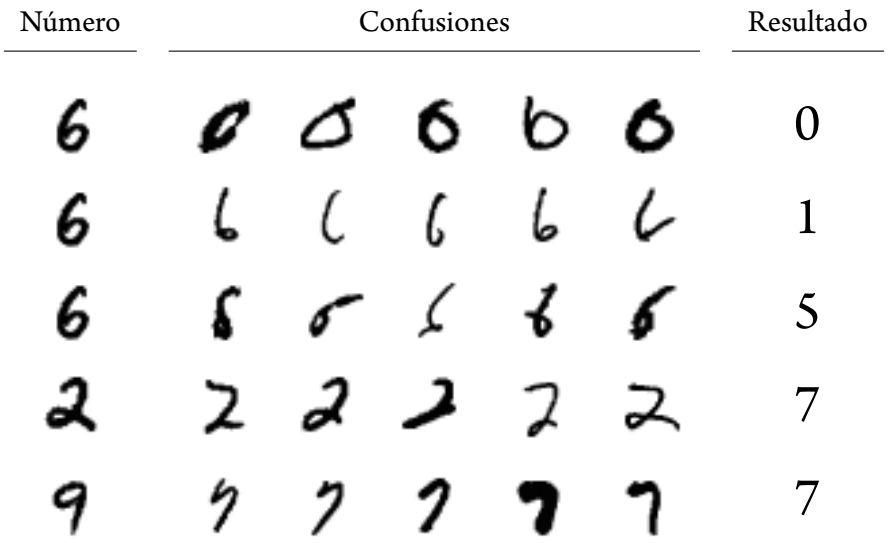


Figura 4.23: Muestra de números en los que se ha producido un intercambio en el reconocimiento.

Una vez más, se pueden atribuir algunos de estos errores al algoritmo de corrección de inclinación. Por otro lado, como se ha comentado ya anteriormente, la forma en que los autores han escrito los números ha influido en gran medida sobre el sistema experto en la etapa de clasificación.

4.4 Comparación con otros autores

Los resultados que se han obtenido en este trabajo son comparables con los de otros autores en similares líneas de investigación dentro del ámbito de reconocimiento de caracteres manuscritos.

En uno de los trabajos presentado por Kessentini et al. [79], propone un sistema de clasificación basado en el paradigma de los Modelos Ocultos de Markov pero con un enfoque multihilo. Para la extracción de las características utiliza dos ventanas deslizantes en las que obtiene los contornos superiores e inferiores de los caracteres y son procesados independientemente por hilos diferentes. En sus experimentos usa 20.898 palabras como muestra para entrenar su sistema y 10.448 palabras para probar el mismo. Estas palabras fueron seleccionadas de la base de datos de palabras reales IRONOFF. Sin embargo, para nuestros experimentos sólo se han necesitado 1.040 imágenes de caracteres de los dos escritores divididas en dos grupos: 832 para generar la base de conocimiento y 208 para realizar los test. Con todo ello, se han generado 400.000 palabras sintéticas para probar el sistema. Los resultados obtenidos en su estudio logran una tasa de acierto de 89,8% mientras que el sistema experto analizado en este trabajo alcanza un 95,36%. No obstante, cabe señalar que mientras la metodología de Kessentini et al. utiliza una base de datos de palabras reales en sus pruebas, las palabras utilizadas en los experimentos de este trabajo han sido generadas previamente.

Respecto al reconocimiento a nivel de caracteres, cabe resaltar el trabajo realizado por Vamvakas et al. [157]. En su metodología propone un sistema basado en divisiones recursivas de la imagen del carácter. De esta manera, se obtienen un conjunto de subimagenes que deben tener un número más o menos proporcional de píxeles entre sí. Para la de extracción de características, obtiene de cada subimagen los niveles y la granularidad de los píxeles para finalmente realizar una clasificación usando una máquina SVM. Los experimentos se realizan tanto en

números usando la base de datos MNIST, como en caracteres usando la base de datos CEDAR. Si se entra en detalle en este último, en sus experimentos usa 19.145 caracteres para el entrenamiento y 2.183 para realizar las pruebas. La tasa de acierto que obtiene es de 94,73% que se aproximan al obtenido en las letras en el presente trabajo: 95,46%. La diferencia radica de nuevo en el tamaño de la muestra utilizada así como en la procedencia de la misma. En cuanto a las pruebas numéricas, utiliza una muestra de 24.270 imágenes de la base de datos CEDAR en la fase de entrenamiento y 5.631 imágenes para realizar las pruebas obteniendo una tasa de acierto de 98,66%. En las pruebas realizadas con la base de datos MNIST, utiliza 60.000 imágenes para el entrenamiento y 10.000 para la realización de las pruebas. En este caso, obtiene una tasa de acierto de 99,03%. Llegado a este punto, se puede comprobar que las tasas de acierto obtenidas por el autor se alejan de las tasas alcanzadas siguiendo la metodología de este trabajo: 88,77%.

Otro de los trabajos que requiere de una especial atención es el presentado por Ahmed et al. [1] ya que se trata también de un sistema experto pero en este caso, para el reconocimiento de símbolos en general. El sistema utiliza una técnica basado en patrones con una determinada estructura formado por unas líneas de referencia. Después, la imagen se escala, se rota y se aplica un adelgazamiento para extraer los trazos de los símbolos y obtener así un modelo. Los modelos son almacenados en una base de conocimiento para ser usados en reconocimientos futuros. En sus experimentos almacena 97 modelos de símbolos diferentes en la base de conocimiento y utiliza 5.726 caracteres manuscritos extraídos de la base de datos CEDAR para las pruebas. En sus resultados, cuando utiliza un umbral = 15, obtiene un índice de acierto del 79,70%. Pero cuando utiliza un umbral = 100, consigue que no existan rechazos con una tasa de acierto del 87,6%. Los resultados obtenidos por el sistema experto presentados en este trabajo no llega a suprimir los rechazos pero sin embargo obtiene una tasa de acierto superior (95,46%) si sólo se considera el reconocimiento de caracteres. Por otro lado, cabe destacar que el sistema experto desarrollado por Ahmed et al. puede reconocer cualquier tipo de símbolo, objeto de interés que podría aplicarse a este trabajo en sus líneas

futuras.

Tabla 4.12: Tabla comparativa de los resultados obtenidos en las redes BTASOM, GHSOM y el método propuesto en este trabajo.

Dígito	H. Shah-Hosseini (BTASOM)	L. Bezerra et al. (GHSOM)	Este trabajo
0	93,16	96,63	93,57
1	96,65	96,24	95,68
2	85,27	87,18	85,66
3	80,69	77,06	88,71
4	74,03	67,62	91,45
5	83,30	79,24	89,46
6	93,53	89,65	81,52
7	87,45	78,55	84,53
8	78,34	79,73	85,42
9	79,98	83,44	90,78
Media	85,24%	83,53%	**88,77%**

Autores como L. Bezerra et al. [15] o H. Shah-Hosseini [146] utilizan también la base de datos MNIST para realizar sus experimentos son caracteres numéricos. El primero de ellos utiliza una red GHSOM donde el tamaño y la profundidad de la jerarquía se modifican durante el proceso de entrenamiento no supervisado. Después de escalar las imágenes originales desde una resolución de 28x28 píxeles a 16x16 píxeles, realiza una transformación de la imagen a escala de grises. Después, selecciona 2.000 imágenes para realizar el entrenamiento de la red. Por último, utiliza 3.000 nuevas imágenes para realizar las pruebas. Por otro lado, H. Shah-Hosseini propone una estrategia basada en una red BTASOM donde la profundidad del árbol y el número de nodos vienen definidos por el usuario. Los experimentos llevados por H. Shah-Hosseini también requieren de una fase inicial de

escalado de las imágenes, transformándolas a una resolución final de 14x14 píxeles. Con estas imágenes se entrena el árbol BTASOM y se utiliza el conjunto de 10.000 imágenes de pruebas que ofrece MNIST.

Se puede apreciar claramente en la tabla 4.12 que la metodología propuesta obtiene mejores resultados alcanzando una tasa de acierto de 88,77%, frente a los resultados obtenidos por los modelos BTASOM y GHSOM cuyas tasas de acierto son de 85,24% y 83,53% respectivamente.

"Lo mejor de los booleanos es que si te equivocas estás a un sólo bit de la solución correcta".

– Anónimo

5

Conclusiones

5.1 Conclusiones

Las conclusiones más importantes que se pueden deducir de este trabajo se enumeran en los seis puntos siguientes. De igual modo, en cada uno de estos puntos se expresan las contribuciones originales aportadas en este trabajo:

1. Se propone un modelo de descriptor para caracteres manuscritos con las siguientes características:

- Las interconexiones existentes entre los trazos verticales y los trazos horizontales adyacentes se analizan como característica principal por una gramática formal cuyo fin es generar una cadena representativa que identifique al carácter.
- Esta cadena representativa es validada por un autómata finito cuyas reglas vienen definidas por una gramática regular propuesta.

2. Se introduce un sistema de clasificación de los descriptores donde el almacenamiento se realiza en una base de conocimiento con estructura de árbol *trie* que permite:
    - Almacenar los descriptores en función de la probabilidad de uso.
    - Una capacidad de búsqueda altamente eficiente dentro de un conjunto elevado de datos.
3. Se presenta un Motor de Inferencia el cual realiza las búsquedas de las nuevas entradas en cuatro escenarios diferentes con el fin de evaluar el rendimiento del sistema de reconocimiento:
    - *Escenario N*: Búsquedas en una base de conocimiento sin corrección de inclinación.
    - *Escenario S*: Búsquedas en una base de conocimiento con corrección de inclinación.
    - *Escenario NS*: Búsquedas en una base de conocimiento sin corrección de inclinación y en el caso de que se produzcan rechazos, se realiza una nueva búsqueda sobre la base de conocimiento con corrección de inclinación.
    - *Escenario SN*: Búsquedas en una base de conocimiento con corrección de inclinación y en el caso de que se produzcan rechazos, se realiza una nueva búsqueda sobre la base de conocimiento sin corrección de inclinación.

4. En un contexto alfabético, el Motor de Inferencia diseñado selecciona los caracteres candidatos para el reconocimiento en base a la probabilidad de su uso. Para dar una solución más acertada, se propone la utilización de un corrector ortográfico que identifica las posibles respuestas y selecciona la más adecuada incrementando de forma significativa la tasa de reconocimiento en palabras.

5. Por otro lado, para el reconocimiento de caracteres numéricos, se ha optimizado el proceso seleccionando la mejor muestra de la base de datos MNIST. A esta muestra se le ha aplicado una escala y un factor de exclusión de trazos verticales de manera que permite al Motor de Inferencia obtener los mejores resultados posibles para este contexto.

6. Se ha desarrollado una aplicación utilizando la metodología detallada en este trabajo. Esta aplicación es capaz de analizar y reconocer texto manuscrito tanto de una palabra como de un conjunto de palabras generadas de forma aleatoria a partir de las muestras introducidas por un autor. Además, es también capaz de reconocer números de la base de datos MNIST.

5.2 Trabajo futuro

Para dar continuidad al esfuerzo dedicado en este trabajo se muestra a continuación el trabajo futuro para seguir avanzando en el conocimiento sobre el reconocimiento de caracteres manuscritos.

Las nuevas líneas de trabajo se identifican en los siguientes puntos:

1. Reducción del número de caracteres mal segmentados. La mejora del algoritmo que identifica los trazos verticales y horizontales permitirá que se produzcan menos ambigüedades y como consecuencia, un incremento de la tasa de reconocimiento.

2. Optimización del proceso de corrección de inclinación. Mejora del algoritmo ya existente o búsqueda de otros algoritmos que se puedan adaptar a la metodología de este trabajo para lograr una corrección de inclinación de los caracteres más eficaz.

3. El reconocimiento de caracteres unidos a partir de secuencias de cadenas representativas puede llevar al reconocimiento de palabras completas sin necesidad de realizar una segmentación de las palabras en sus correspondientes letras.

4. Realizar un proceso de optimización personalizado para cada escritor. En el proceso de entrenamiento, se debe buscar la escala de imagen más apropiada a considerar por el algoritmo así como el cálculo del tamaño de descarte de trazos verticales para cada uno de los escritores de forma independiente.

5. Realización de pruebas alfabéticas sobre bases de datos de palabras reales como por ejemplo IRONOFF, IAM o CEDAR.

6. Utilización de una combinación de sistemas de clasificación por el Motor de Inferencia que permita dar una solución más certera ante casos de ambigüedad o ausencia de respuesta. Estos sistemas de clasificación pueden estar basados en redes neuronales, modelos ocultos de Markov u otros sistemas ya detallados en la introducción.

7. Desarrollo de un sistema experto sin necesidad de entrenamiento previo.

Referencias

[1] M. Ahmed and R. K. Ward. An expert system for general symbol recognition. *Pattern Recognition*, 33(12):1975 – 1988, 2000.

[2] J. E. H. y. U. Aho, Alfred V. Estructura de datos y algoritmos (2ª edición). *Addison Wesley Iberoamericana*, 1998.

[3] J. H. AlKhateeb, O. Pauplin, J. Ren, and J. Jiang. Performance of hidden Markov model and dynamic Bayesian network classifiers on handwritten Arabic word recognition. *Knowledge-Based Systems*, 24(5):680–688, July 2011.

[4] A. Alonso, R. A. Fernández, and I. García. Recognition of merged characters based on vertical strokes and adjacency regions. *V Congress of Hispalinux*, pages 1–23, 2005.

[5] J. Alonso-Weber, M. Sesmero, and a. Sanchis. Combining additive input noise annealing and pattern transformations for improved handwritten character recognition. *Expert Systems with Applications*, 41(18):8180–8188, 2014.

[6] E. Anquetil and G. Lorette. On-line cursive handwritten character recognition using hidden markov models. *Traitement du Signal*, 12(6):575–583, 1995.

[7] E. Anquetil and G. Lorette. Perceptual model of handwriting drawing. application to the handwriting segmentation problem. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 1, pages 112–117 vol.1, Aug 1997.

[8] F. Bara and E. Gentaz. Haptics in teaching handwriting: The role of perceptual and visuo-motor skills. *Human Movement Science*, 30(4):745 – 759, 2011.

[9] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73(3):360–363, 05 1967.

[10] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[11] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 41(1):164–171, 02 1970.

[12] L. E. Baum and G. R. Sell. Growth transformations for functions on manifolds. *Pacific J. Math.*, 27(2):211–227, 1968.

[13] R. Bellman. Adaptive control processes. a guided tour. *Princeton University Press*, 1961.

[14] S. Bercu and G. Lorette. On-line handwritten word recognition: An approach based on hidden markov models. *Proc. Third Int'l Workshop on Frontiers in Handwriting Recognition*, pages 385–390, May 1993.

[15] L. Bezerra Batista, H. Martins Gomes, and R. Fernandes Herbster. Application of Growing Hierarchical Self-Organizing Map in Handwritten Digit Recognition. In *Brazilian Symposium on Computer Graphics and Image Processing*, 2003.

[16] W. Bieniecki, S. Grabowski, and W. Rozenberg. Image preprocessing for improving ocr accuracy. In *Perspective Technologies and Methods in MEMS Design, 2007. MEMSTECH 2007. International Conference on*, pages 75–80, May 2007.

[17] C. Bishop. Neural networks for pattern recognition. *Clarendon Press*, 1995.

[18] T. Bluche, H. Ney, and C. Kermorvant. Tandem hmm with convolutional neural network for handwritten word recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2390–2394, May 2013.

[19] B. Bontempi and A. Marcelli. On-line segmentation of cursive script using an arclength representation. *Handwriting and Drawing Research: Basic and Applied Issues, M.L. Simner, C.G. Leedham, and A.J.W.M. Thomassen, eds.*, pages 315–327, 1996.

[20] R. Bozinovic and S. Srihari. Off-line cursive script word recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(1):68–83, Jan 1989.

[21] A. Brakensiek, A. Kosmala, D. Willett, W. Wang, and G. Rigoll. Performance evaluation of a new hybrid modeling technique for handwriting recognition using identical on-line and off-line data. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 446–449, Sep 1999.

[22] M. Brown and S. Ganapathy. Preprocessing techniques for cursive script word recognition. *Pattern Recognition*, 16(5):447–458, 1983.

[23] H. Bunke, M. Roth, and E. Schukat-Talamazzini. Off-line recognition of cursive script produced by a cooperative writer. *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, 2, 1994.

[24] H. Bunke, M. Roth, and E. Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern Recognition*, 28(9):1399 – 1413, 1995.

[25] D. Burr. Designing a handwriting reader. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-5(5):554–559, Sept 1983.

[26] J. CAI and Z.-Q. LIU. Off-line unconstrained handwritten word recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(03):259–280, 2000.

[27] E. Caillault, C. Viard-Gaudin, and A. Ahmad. Ms-tdnn with global discriminant trainings. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 856–860 Vol. 2, Aug 2005.

[28] F. Camastra and A. Vinciarelli. Cursive character recognition by learning vector quantization. *Pattern Recognition Letters*, 22(6–7):625 – 629, 2001.

[29] P. Carrières and R. Plamondon. An interactive handwriting teaching aid. *Advances in Handwriting and Drawing: A Multidisciplinary Approach. C. Faure, G. Lorette, A. Vinter, and P. Keuss, eds.*, pages 207–239, 1994.

[30] R. Casey and E. Lecolinet. Strategies in character segmentation: a survey. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 1028–1033 vol.2, Aug 1995.

[31] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706, Jul 1996.

[32] S. Chakraborty. Formal languages and automata theory- regular expressions and finite automata- contents, 2003.

[33] K.-F. Chan and D.-Y. Yeung. Elastic structural matching for online handwritten alphanumeric character recognition. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1508–1511 vol.2, Aug 1998.

[34] E. Cohen, J. J. Hull, and S. N. Srihari. Undestanding handritten text in a structured environment: Determining zip codes from addresses. *International Journal of Pattern Recognition and Artificial Intelligence*, 05(01n02):221–264, 1991.

[35] E. Cohen, J. J. Hull, and S. N. Srihari. Control structure for interpreting handwritten addresses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1049–1055, 1994.

[36] F. Coulmas. What Writing Is all About. 1989.

[37] M. Y. D. Yimei, K. Fumitika and S. Malayappan. Slant estimation for handwritten words by directionally refined chain code. *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 53–62, 2000.

[38] C. Davidson. The man (alan kay) who made computers personal. *New Scientist*, pages 32–35, Jun 1993.

[39] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. A multi-expert signature verification system for bankcheck processing. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(05):827–844, 1997.

[40] Y. a. Dimitriadis and J. L. Coronado. Towards an art based mathematical editor, that uses on-line handwritten symbol recognition. *Pattern Recognition*, 28(6):807–822, 1995.

[41] A. C. Downton, R. W. S. Tregidgo, and E. Kabir. Recognition and verification of handwritten and hand-printer british postal adresses. *International Journal of Pattern Recognition and Artificial Intelligence*, 05(01n02):265–291, 1991.

[42] L. Duneau and B. Dorizzi. On-line cursive script recognition: A user-adaptive system for word identification. *Pattern Recognition*, 29(12):1981 – 1994, 1996.

[43] G. Dzuba, A. Filatov, and A. Volgunin. Handwritten zip code recognition. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 2, pages 766–770 vol.2, Aug 1997.

[44] L. Earnest. Machine reading of cursive script. *IFIP Congress*, pages 462–466, 1963.

[45] S. Edelman, T. Flash, and S. Ullman. Reading cursive handwriting by alignment of letter prototypes. *International Journal of Computer Vision*, 5(3):303–331, 1990.

[46] M. Eden. Handwriting and pattern recognition. *Information Theory, IRE Transactions on*, 8(2):160–166, February 1962.

[47] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[48] F. Farooq, D. Jose, and V. Govindaraju. Phrase-based correction model for improving handwriting recognition accuracies. *Pattern Recognition*, 42(12):3271 – 3277, 2009.

[49] B. Ficarra. Handwriting in health and disease. *Legal Medicine*, pages 261–278, 1995.

[50] G. Forney Jr. Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[51] K. Franke and M. Köppen. Towards an universal approach to background removal in images of bankchecks. In *in Proceedings 6th International Workshop on Frontiers in Handwriting Recognition (IWFHR), Tajon, Korea*, pages 91–100, 1998.

[52] H. Fujisawa. Forty years of research in character and document recognition—an industrial perspective. *Pattern Recognition*, 41(8):2435–2446, Aug. 2008.

[53] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic press, Boston, 2 edition, 1990.

[54] K. Fukushima. Training multi-layered neural network neocognitron. *Neural Networks*, 40(0):18 – 31, 2013.

[55] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognitron. *Neural Networks, IEEE Transactions on*, 2(3), May 1991.

[56] P. Gader, M. Mohamed, and J. H. Chiang. Comparison of crisp and fuzzy character neural networks in handwritten word recognition. *IEEE Transactions on Fuzzy Systems*, 3:357–363, 1995.

[57] P. Gader, M. Mohamed, and J.-H. Chiang. Handwritten word recognition with character and inter-character neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(1):158–164, Feb 1997.

[58] J. Galbally, M. Diaz-Cabrera, M. A. Ferrer, M. Gomez-Barrero, A. Morales, and J. Fierrez. On-line signature recognition through the combination of real dynamic data and synthetically generated static data. *Pattern Recognition*, (0):–, 2015.

[59] G. Gangadhar, D. Joseph, and V. S. Chakravarthy. Understanding Parkinsonian handwriting through a computational model of basal ganglia., 2008.

[60] M. Gilloux, M. Leroux, and J.-M. Bertille. Strategies for cursive script recognition using hidden markov models. *Machine Vision and Applications*, 8(4):197–205, 1995.

[61] W. Guerfali and R. Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, 26(3):419–431, 1993.

[62] W. Guerfali and R. Plamondon. A new method for the analysis of simple and complex planar rapid movements. *Journal of Neuroscience Methods*, 82(1):35 – 45, 1998.

[63] D. Guillevic and C. Suen. HMM word recognition engine. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 2, 1997.

[64] D. Guillevic and C. Suen. Hmm-knn word recognition engine for bank cheque processing. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1526–1529 vol.2, Aug 1998.

[65] A. Gupta, M. Srivastava, and C. Mahanta. Offline handwritten character recognition using neural network. In *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on*, pages 102–107, Dec 2011.

[66] M. Hanmandlu, K. Murali Mohan, S. Chakraborty, S. Goyal, and D. Choudhury. Unconstrained handwritten character recognition based on fuzzy logic. *Pattern Recognition*, 36(3):603–623, Mar. 2003.

[67] R. Haralick and L. Shaphiro. *Computer and Robot Vision: Volume 1*. 1992.

[68] R. R. Hausser. Foundations of computational linguistics. *Springer-Verlag*, 1999.

[69] J. Hu, M. Brown, and W. Turin. Hmm based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10):1039–1045, Oct 1996.

[70] D. Impedovo and G. Pirlo. Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 38(5):609–635, 2008.

[71] D. Impedovo and G. Pirlo. Zoning methods for handwritten character recognition: A survey. *Pattern Recognition*, 47(3):969–981, Mar. 2014.

[72] S. Impedovo. *Fundamentals in Handwriting Recognition (Nato ASI Series (Closed) / Nato ASI Subseries F*. Springer Publishing Company, Incorporated, 1st edition, 2012.

[73] J. B. J. Salome, M. Leroux. Recognition of cursive script words in a small lexicon. *Proceedings of the International Conference on Document Analysis and Recognition*, page 774–782, 1991.

[74] K. C. H. Jr. Reading handwritten words using hierarchical relaxation. *Computer Graphics and Image Processing*, 14(4):344 – 364, 1980.

[75] Z. Kamranian, S. A. Monadjemi, and N. Nematbakhsh. A novel free format persian/arabic handwritten zip code recognition system. *Computers & Electrical Engineering*, 39(7):1970 – 1979, 2013.

[76] I. Karls, G. Maderlechner, V. Pflug, S. Baumann, A. Weigel, and A. Dengel. Segmentation and recognition of cursive handwriting with improved structured lexica. In *Proceedings 3rd International Workshop on Frontiers in Handwriting Recognition*, pages 437–442. o.A., 5 1993.

[77] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. A slant removal algorithm. *Pattern Recognition*, 33(7):1261 – 1262, 2000.

[78] G. Keerthi Prasad, I. Khan, N. Chanukotimath, and F. Khan. On-line handwritten character recognition system for kannada using principal component analysis approach: For handheld devices. In *Information and Communication Technologies (WICT), 2012 World Congress on*, pages 675–678, Oct 2012.

[79] Y. Kessentini, T. Paquet, and A. B. Hamadou. Off-line handwritten word recognition using multi-stream hidden markov models. *Pattern Recognition Letters*, 31(1):60 – 70, 2010.

[80] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):366–379, Apr 1997.

[81] G. Kim, V. Govindaraju, and S. N. Srihari. An architecture for handwritten text recognition systems. *International Journal on Document Analysis and Recognition*, 2(1):37–44, 1999.

[82] F. Kimura, Y. Miyake, and M. Shridhar. Handwritten zip code recognition using lexicon free word recognition algorithm. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 906–910 vol.2, Aug 1995.

[83] S. Knerr, E. Augustin, O. Baret, and D. Price. Hidden markov model based word recognition and its application to legal amount reading on french checks. *Computer Vision and Image Understanding*, 70(3):404 – 419, 1998.

[84] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[85] J. S. G. C. K.S. Nathan, H.S.M. Beigi and H. Maruyama. Realtime on-line unconstrained handwriting recognition using statistical methods. In *Proc. Int'l Conf. Accoustics, Speech, and Signal Processing*, pages 2,619–2,622, 1995.

[86] L. Lam, S.-W. Lee, and C. Suen. Thinning methodologies-a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(9):869–885, Sep 1992.

[87] Y. Le Cun, O. Matan, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jacket, and H. Baird. Handwritten zip code recognition with multilayer networks. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume ii, pages 35–40 vol.2, Jun 1990.

[88] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[89] K. H. Lee, K.-B. Eom, and R. Kashyap. Character recognition using attributed grammar. pages 418–423, Jun 1988.

[90] Y. Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3(3):440–449, Sept 1991.

[91] E. Lethelier, M. Leroux, and M. Gilloux. An automatic reading system for handwritten numeral amounts on french checks. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 92–97 vol.1, Aug 1995.

[92] X. Li, M. Parizeau, and R. Plamondon. Segmentation and reconstruction of on-line handwritten scripts. *Pattern Recognition*, 31(6):675 – 684, 1998.

[93] X. Li and D.-Y. Yeung. On-line handwritten alphanumeric character recognition using dominant points in strokes. *Pattern Recognition*, 30(1):31 – 44, 1997.

[94] C.-Y. Liou and H.-C. Yang. Handprinted character recognition based on spatial topology distance measurement. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):941–945, Sep 1996.

[95] K. Liu, C. Y. Suen, M. Cheriet, J. N. Said, C. Nadal, and Y. Y. Tang. Automatic extraction of baselines and data from check images. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(04):675–697, 1997.

[96] Y. Liu and S. Srihari. Document image binarization based on texture features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):540–544, May 1997.

[97] Y. Lu and M. Shridhar. Character segmentation in handwritten words - An overview, 1996.

[98] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35(0):58 – 67, 2014.

[99] M. C. C. S. M. Cote', E. Lecolinet. Automatic reading of cursive scripts using a reading model and perceptual concepts the perpecto system. *International Journal on Document Analysis and Recognition (IJDAR)*, 1(1):3–17, 1998.

[100] J. Šíma. Neural expert systems. *Neural Networks*, 8(2):261 – 271, 1995.

[101] F. J. Maarse and J. W. M. Thomassen. Produced and Perceived Writing Slant Difference Between Up and Down Strokes. 54:131–147, 1983.

[102] S. Madhvanath and G. Kim. Chaincode contour processing for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):928–932, 1999.

[103] S. Madhvanath, E. Kleinberg, V. Govindaraju, and S. Srihari. The HOVER system for rapid holistic verification of off-line handwritten phrases. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 2, 1997.

[104] U. Mahadevan and R. Nagabushnam. Gap metrics for word separation in handwritten lines. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 124–127 vol.1, Aug 1995.

[105] J. Makhoul, T. Starner, R. Schwartz, and G. Chou. On-line cursive handwriting recognition using speech recognition methods. In *In ICASSP*, pages 125–128, 1994.

[106] S. Marukatat, T. Artières, P. Gallinari, and B. Dorizzi. Sentence recognition through hybrid neuro-markovian modelling. In *International Conference on Document Analysis and Recognition ICDAR' 01*, 2001.

[107] A. Meyer. Pen computing: A technology overview and a vision. *SIGCHI Bull.*, 27(3):46–90, July 1995.

[108] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):548–554, 1996.

[109] P. Morasso, M. Limoncelli, and M. Morchio. Incremental learning experiments with scriptor: an engine for on-line recognition of cursive handwriting. *Machine Vision and Applications*, 8(4):206–214, 1995.

[110] J. Moreau. A new system for automatic reading of postal checks. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, page 121–132, 1991.

[111] M. Morita, J. Facon, F. Bortolozzi, S. Garnes, and R. Sabourin. Mathematical morphology and weighted least squares to correct handwriting baseline skew. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 430–433, Sep 1999.

[112] Y. Nakajima, S. Mori, S. Takegami, and S. Sato. Global methods for stroke segmentation. *International Journal on Document Analysis and Recognition*, 2(1):19–23, 1999.

[113] F. Nouboud and R. Plamondon. On-line recognition of handprinted characters: Survey and beta tests. *Pattern Recognition*, 23(9):1031–1044, 1990.

[114] I.-S. Oh and C. Y. Suen. A class-modular feedforward neural network for handwriting recognition. *Pattern Recognition*, 35(1):229–244, Jan. 2002.

[115] C. Olivier, T. Paquet, M. Avila, and Y. Lecourtier. Optimal order of markov models applied to bankchecks. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(05):789–800, 1997.

[116] N. Otsu. A threshold selection method from gray-scale histogram. *IEEE Trans. Systems, Man, and Cybernetics*, 8:62–66, 1978.

[117] V. S. P. Morasso and T. Tsuji. A model for the generation of virtual targets in trajectory formation. *Advances in Handwriting and Drawing: A Multidisciplinary Approach*, pages 333 – 348, 1994.

[118] R. Palacios and A. Gupta. A system for processing handwritten bank checks automatically. *Image and Vision Computing*, 26(10):1297 – 1313, 2008.

[119] P. W. Palumbo, P. Swaminathan, and S. N. Srihari. Document image binarization: Evaluation of algorithms, 1986.

[120] T. Paquet and Y. Lecourtier. Recognition of handwritten sentences using a restricted lexicon. *Pattern Recognition*, 26(3):391 – 407, 1993.

[121] C. Parisse. Global word shape processing in off-line recognition of handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):460–464, 1996.

[122] M. Parizeau and R. Plamondon. A fuzzy-syntactic approach to allograph modeling for cursive script recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(7):702–712, Jul 1995.

[123] M. Parodi and J. C. Gómez. Legendre polynomials based feature extraction for online signature verification. consistency analysis of feature combinations. *Pattern Recognition*, 47(1):128 – 140, 2014.

[124] I. Pavlidis, R. Singh, and N. P. Papanikolopoulos. On-line handwriting recognition using physics-based shape metamorphosis. *Pattern Recognition*, 31(11):1589 – 1600, 1998.

[125] R. Plamondon and W. Guerfali. Why handwriting segmentation can be misleading? In *Pattern Recognition, Proceedings of the 13th International Conference on*, volume 4, pages 396–400, Aug 1996.

[126] R. Plamondon, D. P. Lopresti, L. R. B. Schomaker, and R. Srihari. *Online Handwriting Recognition*, volume 15, pages 123–146. John Wiley - Sons, Inc., 1999.

[127] R. Plamondon and S. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, Jan 2000.

[128] J. L. Poole and C. M. Schneck. Developmental differences in praxis in learning-disabled and normal children and adults. *Perceptual and motor skills*, 78(3 Pt 2):1219–1228, 1994.

[129] M. Prakash and M. Murty. Extended subspace methods of pattern recognition. *Pattern Recognition Letters*, 17(11):1131 – 1139, 1996.

[130] S. D. R. Bellman. Applied dynamic programming. *Princeton University Press*, 1962.

[131] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[132] M. I. Razzak, F. Anwar, S. Husain, A. Belaid, and M. Sher. {HMM} and fuzzy logic: A hybrid approach for online urdu script-based languages' character recognition. *Knowledge-Based Systems*, 23(8):914 – 923, 2010.

[133] N. S. Reddy and P. Nagabhushan. A connectionist expert system model for conflict resolution in unconstrained handwritten numeral recognition. *Pattern Recognition Letters*, 19(2):161 – 169, 1998.

[134] H. A. Reinders-Messelink, M. M. Schoemaker, M. Hofte, L. N. Goeken, A. Kingma, M. M. van den Briel, and W. A. Kamps. Fine motor and handwriting problems after treatment for childhood acute lymphoblastic leukemia. *Med Pediatr Oncol*, 27(6):551–555, 1996.

[135] D. G. S. Richard O. Duda, Peter E. Hart. *Pattern classification (2nd edition)*. John Wiley and Sons, 2001.

[136] G. Rigoll, A. Kosmala, and D. Willett. A new hybrid approach to large vocabulary cursive handwriting recognition. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1512–1514 vol.2, Aug 1998.

[137] R. P. S. Djeziri and J. Robert. A letter model generator to assist in teaching handwriting. *Proc. Ninth Biennial Conf. Int'l Graphonomics Soc.*, pages 181–185, 1999.

[138] P. Sahoo, S. Soltani, and A. Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233 – 260, 1988.

[139] K. M. Sayre. Machine recognition of handwritten words: A project report. *Pattern Recognition*, 5(3):213 – 228, 1973.

[140] L. Schomaker. From handwriting analysis to pen-computer applications. *Electronics Communication Engineering Journal*, 10(3):93–102, Jun 1998.

[141] L. Schomaker and H.-L. Teulings. *Stroke- versus Character-based Recognition of On-line, Connected Cursive Script*, pages 265–277. 1991.

[142] G. Seni and E. Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27(1):41 – 52, 1994.

[143] G. Seni and J. Seybold. Diacritical processing for unconstrained online handwriting recognition using a forward search. *International Journal on Document Analysis and Recognition*, 2(1):24–29, 1999.

[144] A. W. Senior. Off-line cursive handwriting recognition using recurrent neural networks, 1994.

[145] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321, 1998.

[146] H. Shah-Hosseini. Binary tree time adaptive self-organizing map. *Neurocomputing*, 74(11):1823 – 1839, 2011.

[147] A. H. N. Sherkat and R. Whitrow. Zone-estimation for multiple lines of handwriting using approximate spline functions. *Proc. Fifth Int'l Workshop Frontiers in Handwriting Recognition (IWFHRV)*, pages 325–328, 1996.

[148] F. Simistira, V. Katsouros, and G. Carayannis. Recognition of online handwritten mathematical formulas using probabilistic {SVMs} and stochastic context free grammars. *Pattern Recognition Letters*, 53(0):85 – 92, 2015.

[149] B.-K. Sin, J.-Y. Ha, S.-C. Oh, and J. Kim. Network-based approach to online cursive script recognition. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(2):321–328, Apr 1999.

[150] S. Srihari and V. Govindaraju. Analysis of textual images using the hough transform. *Machine Vision and Applications*, 2(3):141–153, 1989.

[151] S. Srihari and V. Govindaraju. Separating handwritten text from overlapping contextual contours. *Proc. Second Int'l Workshop on Handwriting Recognition (IWFHRII)*, pages 111–119, 1991.

[152] S. Srihari and E. Kuebert. Integration of hand-written address interpretation technology into the united states postal service remote computer reader system. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 2, pages 892–896 vol.2, Aug 1997.

[153] C. Sun and D. Si. Skew and slant correction for document images using gradient direction. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 1, pages 142–146 vol.1, Aug 1997.

[154] T. Sylverberg, P. Kristensson, O. Leifler, and E. Berglund. Drawing on paper maps: Reliable on-line symbol recognition of handwritten symbols using a digital pen and a mobile phone. In *Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on*, pages 515–520, July 2007.

[155] H. Takahashi, N. Itoh, T. Amano, and a. Yamashita. A spelling correction method and its application to an OCR system. *Pattern Recognition*, 23(3):363–377, 1990.

[156] C. Tappert, C. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(8):787–808, Aug 1990.

[157] G. Vamvakas, B. Gatos, and S. J. J. Perantonis. Handwritten character recognition through two-stage foreground sub-sampling. *Pattern Recognition*, Vol. 43(8):2807–2816, 2010.

[158] A. Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35(7):1433 – 1446, 2002.

[159] A. Vinciarelli and J. Luettin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043 – 1050, 2001.

[160] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, April 1967.

[161] L. Vuurpijl and L. Schomaker. Coarse Writing-Style Clustering Based on Simple Stroke-Related Features. *Proc. Fifth Int. Workshop Frontiers in Handwriting Recognition*, pages 29–32, 1996.

[162] T. Wakahara, H. Murase, and K. Odaka. On-line handwriting recognition. *Proceedings of the IEEE*, 80(7):1181–1194, Jul 1992.

[163] J. Wang, M. K. H. Leung, and S. I. U. C. Hui. (Revised form 25 October 1995; received for publication 28 May 1996). 30(3), 1997.

[164] W. Wang, A. Brakensiek, A. Kosmala, and G. RIGOLL. Hmm based high accuracy off-line cursive handwriting recognition by a baseline detection error tolerant feature extraction approach. In *In Proc. Int. Workshop on Frontiers in Handwriting Recognition*, pages 209–218, 2000.

[165] H. Wehbi, H. Oulhadj, J. Lemoine, and E. Petit. Numeral characters and capital letters segmentation recognition in mixed handwriting context. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 878–881, Aug 1995.

[166] L. K. Welbourn and R. J. Whitrow. *A Gesture Based Text and Diagram Editor*, chapter 11, pages 221–234. 1990.

[167] C. N. Y. Bengio, Y. LeCun and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. volume 7, pages 1,289–1,303, 1995.

[168] L. Yaeger, B. Webb, and R. Lyon. Combining neural networks and context-driven search for online, printed handwriting recognition in the newton. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 275–298. Springer Berlin Heidelberg, 1998.

[169] A. Zahour, B. Taconet, P. Mercy, and S. Ramdane. Arabic hand-written text-line extraction. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 281–285, 2001.

[170] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239, Mar. 1984.

[171] H. Zhao. A framework of Chinese handwriting learning, evaluating and research system based on real-time handwriting information collection. In

2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, AIMSEC 2011 - Proceedings, pages 23–26, 2011.

Apéndices

A

Esquema XSD de la base de conocimiento

Esquema XSD para la construcción de la estructura de la base de conocimiento en formato XML.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns
    ="gramatica.dtd" xmlns:wmh="http://" elementFormDefault=
    "qualified" targetNamespace="gramatica.dtd">
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="nodo"/>
        <xs:element ref="caracter"/>
        <xs:element ref="trazo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="nodo">
    <xs:complexType>
```

```
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="trazo"/>
        <xs:element ref="nodo"/>
        <xs:element ref="caracter"/>
      </xs:choice>
      <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="caracter">
    <xs:complexType>
      <xs:attribute name="aciertos" type="xs:string"/>
      <xs:attribute name="caracter" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="trazo">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="nodo"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

B

Plantilla para la obtención de letras

Plantilla usada por los escritores para la obtención de letras que sirven de muestra para los experimentos.

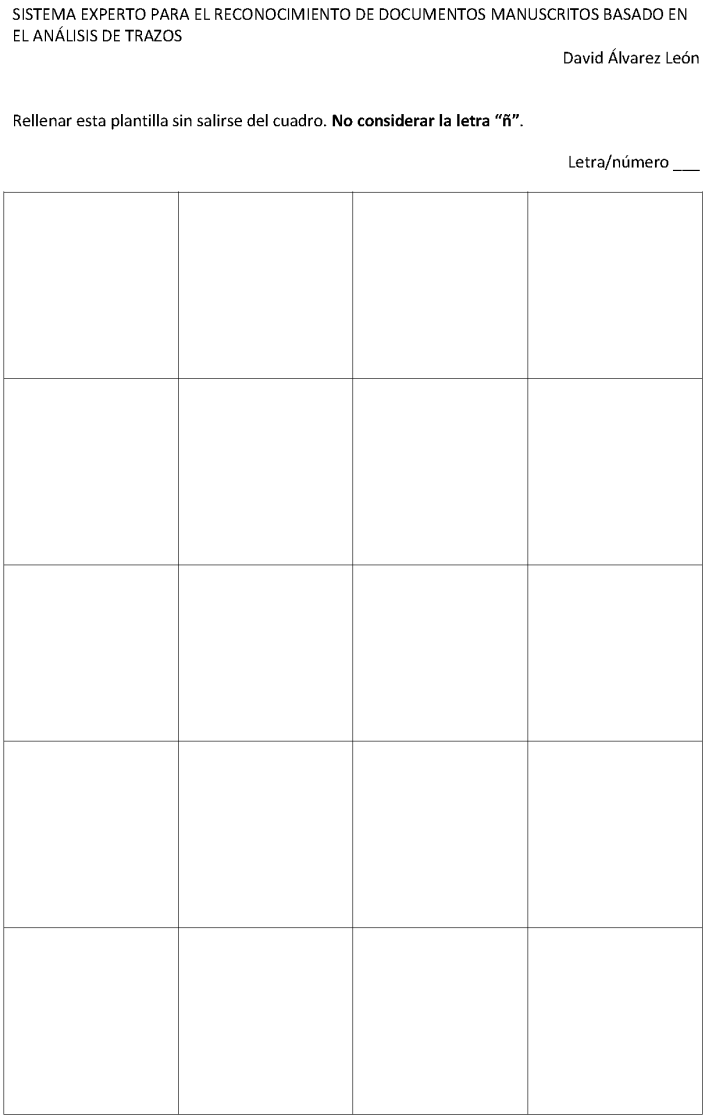


Figura B.1: Plantilla para la obtención de letras.

C

Tablas de exclusión de trazos verticales alfabéticos

Se muestran en las tablas siguientes las tasas de reconocimiento obtenidos, a mayor detalle, para la exclusión de trazos verticales inferiores al 6%, 7%, 8%, 9%, 10%, 11% y 12% en los ensayos realizados para 8000 palabras compuestas por 135.024 letras. Las tablas detallan la tasa de acierto y fallo en palabras (no se consideran los rechazos en las palabras porque siempre se da una palabra como solución); y la tasa de acierto, fallo y rechazo en las letras que componen esas palabras para cada uno de los cuatro tipos de escenario posible.

Resultados para el 6%:

Tabla C.1: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 6% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	78,10	21,90	x	29,90	70,10	x
	Escritor 2	72,10	27,90	x	23,80	76,20	x
S	Escritor 1	29,70	70,30	x	4,50	95,50	x
	Escritor 2	48,70	51,30	x	9,30	90,70	x
NS	Escritor 1	**96,30**	3,70	x	73,20	26,80	x
	Escritor 2	**93,60**	6,40	x	51,30	48,70	x
SN	Escritor 1	60,70	39,30	x	15,90	84,10	x
	Escritor 2	70,20	29,80	x	20,50	79,50	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	86,56	0,91	12,52
	Escritor 2	85,05	4,39	10,56
S	Escritor 1	66,50	7,48	26,02
	Escritor 2	70,55	8,08	21,37
NS	Escritor 1	**96,28**	0,73	2,99
	Escritor 2	**94,26**	4,24	1,50
SN	Escritor 1	78,82	8,23	12,94
	Escritor 2	80,76	8,43	10,80

Resultados para el 7%:

Tabla C.2: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 7% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	84,00	16,00	x	35,70	64,30	x
	Escritor 2	81,00	19,00	x	34,50	65,50	x
S	Escritor 1	37,10	62,90	x	6,60	93,40	x
	Escritor 2	55,00	45,00	x	10,90	89,10	x
NS	Escritor 1	**95,90**	4,10	x	70,20	29,80	x
	Escritor 2	**96,10**	3,90	x	61,20	38,80	x
SN	Escritor 1	65,40	34,60	x	18,10	81,90	x
	Escritor 2	74,40	25,60	x	23,80	76,20	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,77	0,85	10,38
	Escritor 2	88,05	3,29	8,66
S	Escritor 1	69,33	5,52	25,14
	Escritor 2	73,19	6,01	20,79
NS	Escritor 1	**95,83**	1,41	2,76
	Escritor 2	**95,16**	3,44	1,39
SN	Escritor 1	80,83	6,02	13,15
	Escritor 2	82,72	6,29	10,99

Resultados para el 8%:

Tabla C.3: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 8% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	79,30	20,70	x	33,90	66,10	x
	Escritor 2	82,00	18,00	x	32,70	67,30	x
S	Escritor 1	37,50	62,50	x	6,40	93,60	x
	Escritor 2	53,00	47,00	x	11,30	88,70	x
NS	Escritor 1	**94,10**	5,90	x	59,30	40,70	x
	Escritor 2	**94,90**	5,10	x	60,70	39,30	x
SN	Escritor 1	81,60	18,40	x	35,00	65,00	x
	Escritor 2	73,00	27,00	x	24,20	75,80	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	87,25	3,15	9,60
	Escritor 2	87,46	4,34	8,20
S	Escritor 1	69,77	5,34	24,88
	Escritor 2	73,33	5,87	20,79
NS	Escritor 1	**93,72**	3,84	2,44
	Escritor 2	**93,96**	4,79	1,24
SN	Escritor 1	87,70	6,16	6,14
	Escritor 2	85,97	6,44	7,58

Resultados para el 9%:

Tabla C.4: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 9% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,10	17,90	x	40,20	59,80	x
	Escritor 2	81,80	18,20	x	36,50	63,50	x
S	Escritor 1	36,30	63,70	x	4,90	95,10	x
	Escritor 2	58,00	42,00	x	12,20	87,80	x
NS	Escritor 1	**93,50**	6,50	x	60,00	40,00	x
	Escritor 2	**95,30**	4,70	x	60,90	39,10	x
SN	Escritor 1	83,40	16,60	x	34,50	65,50	x
	Escritor 2	76,30	23,70	x	24,80	75,20	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,49	2,95	8,55
	Escritor 2	88,44	4,41	7,14
S	Escritor 1	69,39	5,52	25,08
	Escritor 2	73,82	6,00	20,18
NS	Escritor 1	**93,92**	3,88	2,20
	Escritor 2	**94,10**	4,80	1,10
SN	Escritor 1	87,63	6,29	6,08
	Escritor 2	86,22	6,39	7,39

Resultados para el 10%:

Tabla C.5: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 10% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	81,90	18,10	x	37,20	62,80	x
	Escritor 2	88,50	11,50	x	43,30	56,70	x
S	Escritor 1	40,80	59,20	x	7,10	92,90	x
	Escritor 2	56,90	43,10	x	11,10	88,90	x
NS	Escritor 1	**92,40**	7,60	x	54,50	45,50	x
	Escritor 2	**96,60**	3,40	x	70,20	29,80	x
SN	Escritor 1	81,90	18,10	x	36,70	63,30	x
	Escritor 2	76,50	23,50	x	24,60	75,40	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,15	3,53	8,32
	Escritor 2	89,40	3,66	6,94
S	Escritor 1	71,17	5,11	23,72
	Escritor 2	74,63	5,98	19,38
NS	Escritor 1	**92,73**	3,86	3,39
	Escritor 2	**94,30**	4,00	1,70
SN	Escritor 1	87,67	6,33	6,00
	Escritor 2	86,35	6,53	7,11

Resultados para el 11%:

Tabla C.6: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 11% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	81,80	18,20	x	37,20	62,80	x
	Escritor 2	84,90	15,10	x	44,40	44,60	x
S	Escritor 1	39,10	60,90	x	7,10	92,90	x
	Escritor 2	57,20	42,80	x	12,40	87,60	x
NS	Escritor 1	**93,20**	6,80	x	54,20	45,80	x
	Escritor 2	**96,70**	3,30	x	72,30	27,70	x
SN	Escritor 1	79,10	20,90	x	33,30	66,70	x
	Escritor 2	73,40	26,60	x	25,10	74,90	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,24	2,96	8,79
	Escritor 2	89,29	3,50	7,21
S	Escritor 1	70,46	5,78	23,76
	Escritor 2	74,36	6.04	19,60
NS	Escritor 1	**92,84**	3,93	3,22
	Escritor 2	**94,51**	3,87	1,61
SN	Escritor 1	86,66	7,34	6,00
	Escritor 2	85,83	6,83	7,34

Resultados para el 12%:

Tabla C.7: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para la exclusión de trazos verticales inferiores al 12% del mayor de los trazos verticales hallados.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	74,80	25,20	x	28,20	71,80	x
	Escritor 2	87,70	12,30	x	39,90	60,10	x
S	Escritor 1	40,00	60,00	x	6,50	93,50	x
	Escritor 2	55,70	44,30	x	12,80	87,20	x
NS	Escritor 1	**90,20**	9,80	x	47,50	52,50	x
	Escritor 2	**96,30**	3,70	x	69,10	30,90	x
SN	Escritor 1	81,90	18,10	x	34,90	65,10	x
	Escritor 2	75,10	24,90	x	24,30	75,70	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	84,68	6,21	9,11
	Escritor 2	87,19	5,31	7,50
S	Escritor 1	71,17	5,51	23,32
	Escritor 2	74,65	5,85	19,49
NS	Escritor 1	**91,17**	7,17	1,66
	Escritor 2	**93,41**	5,76	0,83
SN	Escritor 1	86,92	8,36	4,72
	Escritor 2	85,90	7,59	6,49

D

Tablas de exclusión de trazos verticales numéricos

En las tablas siguientes se detallan las tasas de reconocimiento obtenidos en mayor detalle para la exclusión de trazos verticales inferiores al 10%, 11%, 12%, 13%, 14%, 15% y 16% en los ensayos realizados para 10.000 caracteres numéricos de la base de datos MNIST. Las tablas detallan la tasa de acierto, fallo y rechazo alcanzados para cada uno de los cuatro tipos de escenario posible.

Resultados para el *escenario N*:

Tabla D.1: Tasas de reconocimiento de caracteres numéricos aplicando exclusión de trazos verticales en el escenario N.

	Porcentaje de exclusión						
	10%	11%	12%	13%	14%	15%	16%
Acierto	83,14	83,30	83,58	83,75	83,89	83,86	83,85
Fallo	10,97	10,97	10,86	10,84	10,82	11,00	11,04
Rechazo	5,89	5,73	5,56	5,41	5,29	5,14	5,11

Resultados para el *escenario S*:

Tabla D.2: Tasas de reconocimiento de caracteres numéricos aplicando exclusión de trazos verticales en el escenario S.

	Porcentaje de exclusión						
	10%	11%	12%	13%	14%	15%	16%
Acierto	80,63	80,66	80,80	80,87	80,95	80,79	80,64
Fallo	12,36	12,39	12,28	12,29	12,34	12,52	12,60
Rechazo	7,01	6,95	6,92	6,84	6,71	6,69	6,67

Resultados para el *escenario NS*:

Tabla D.3: Tasas de reconocimiento de caracteres numéricos aplicando exclusión de trazos verticales en el escenario NS.

	Porcentaje de exclusión						
	10%	11%	12%	13%	14%	15%	16%
Acierto	85,96	86,06	86,29	86,36	86,47	86,39	86,39
Fallo	12,17	12,10	11,91	11,88	11,82	12,00	12,05
Rechazo	1,87	1,84	1,80	1,76	1,71	1,61	1,56

Resultados para el *escenario SN*:

Tabla D.4: Tasas de reconocimiento de caracteres numéricos aplicando exclusión de trazos verticales en el escenario SN.

	Porcentaje de exclusión						
	10%	11%	12%	13%	14%	15%	16%
Acierto	84,75	84,76	84,90	84,93	84,93	84,80	84,66
Fallo	13,38	13,40	13,30	13,31	13,36	13,59	13,78
Rechazo	1,87	1,84	1,80	1,76	1,71	1,61	1,56

E

Tablas de búsqueda del mejor conjunto de datos numérico

Las siguientes tablas señalan las tasas de reconocimiento obtenidos en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST. Para ello, del conjunto de 60.000 datos que MNIST tiene para pruebas, primeramente se selecciona un conjunto de 10.000 imágenes como conjunto de validación. Después, de los 50.000 restantes, se realizan 10 pruebas en cada uno de los conjuntos comenzando por 10.000 números seleccionados de forma aleatoria e incrementando la muestra de 10.000 en 10.000 hasta alcanzar el total de 50.000. Los resultados que se muestran a continuación, reflejan las tasas de acierto, fallo y rechazo producidos en los diferentes escenarios.

E.1 Escenario N

Tasas de Acierto:

Tabla E.1: Tasas de acierto en el escenario N en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	84,05	**85,55**	85,89	86,37	**86,77**
1	83,49	85,36	85,87	**86,58**	
2	83,80	85,20	86,01	86,51	
3	**84,13**	85,13	85,87	86,50	
4	83,90	84,91	85,93	86,29	
5	83,66	85,53	85,90	86,50	
6	83,25	85,14	85,94	86,38	
7	83,44	85,38	86,07	86,46	
8	83,71	85,23	86,07	86,54	
9	83,80	85,17	**86,15**	86,48	

Tasas de Fallo:

Tabla E.2: Tasas de Fallo en el escenario N en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	10,65	10,78	10,89	10,82	10,82
1	11,12	10,81	10,91	10,81	
2	10,80	11,18	10,83	10,84	
3	11,04	10,98	10,85	10,73	
4	11,10	11,14	11,07	10,99	
5	11,06	10,57	10,94	10,75	
6	11,35	11,20	10,88	10,86	
7	11,16	10,88	10,79	10,83	
8	11,12	10,82	10,82	10,75	
9	11,04	11,10	10,92	10,76	

Tasas de Rechazo:

Tabla E.3: Tasas de Rechazo en el escenario N en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	10,65	10,78	10,89	10,82	10,82
1	11,12	10,81	10,91	10,81	
2	10,80	11,18	10,83	10,84	
3	11,04	10,98	10,85	10,73	
4	11,10	11,14	11,07	10,99	
5	11,06	10,57	10,94	10,75	
6	11,35	11,20	10,88	10,86	
7	11,16	10,88	10,79	10,83	
8	11,12	10,82	10,82	10,75	
9	11,04	11,10	10,92	10,76	

E.2 Escenario S

Tasas de Acierto:

Tabla E.4: Tasas de acierto en el escenario S en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	81,63	82,95	83,95	84,11	**84,67**
1	**82,21**	83,35	84,06	84,53	
2	81,52	83,07	83,96	84,57	
3	81,10	83,09	84,28	**84,84**	
4	81,73	**83,67**	84,35	84,33	
5	81,33	83,07	83,97	83,93	
6	81,72	82,74	**84,50**	84,35	
7	81,73	83,54	84,12	84,47	
8	81,46	83,61	84,32	84,46	
9	82,03	83,53	84,34	84,22	

Tasas de Fallo:

Tabla E.5: Tasas de Fallo en el escenario S en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	11,65	11,33	11,66	11,60	11,36
1	11,60	11,59	11,29	11,30	
2	12,32	12,12	11,44	11,35	
3	12,36	11,47	11,27	11,21	
4	12,34	11,19	11,24	11,49	
5	11,47	11,80	11,50	11,87	
6	11,01	11,53	11,20	11,26	
7	12,21	11,70	11,31	11,34	
8	12,18	11,74	11,43	11,49	
9	11,67	11,41	11,20	11,48	

Tasas de Rechazo:

Tabla E.6: Tasas de Rechazo en el escenario S en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	6,72	5,72	4,39	4,29	3,97
1	6,19	5,06	4,65	4,17	
2	6,16	4,81	4,60	4,08	
3	6,54	5,44	4,45	3,95	
4	5,93	5,14	4,41	4,18	
5	7,20	5,13	4,53	4,20	
6	7,27	5,73	4,30	4,39	
7	6,06	4,76	4,57	4,19	
8	6,26	4,65	4,25	4,05	
9	6,30	5,06	4,46	4,30	

E.3 Escenario NS

Tasas de Acierto:

Tabla E.7: Tasas de acierto en el escenario NS en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	86,46	87,73	88,04	**88,34**	**88,30**
1	86,90	87,76	87,87	88,18	
2	86,49	87,62	87,83	88,13	
3	86,78	87,69	88,13	88,05	
4	86,45	87,49	88,04	88,07	
5	86,70	87,54	88,04	88,20	
6	**86,94**	87,62	**88,20**	88,26	
7	86,80	87,64	88,05	88,16	
8	86,62	87,56	87,65	88,19	
9	86,78	**87,77**	87,95	88,21	

Tasas de Fallo:

Tabla E.8: Tasas de Fallo en el escenario NS en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	12,39	11,60	11,44	11,29	11,34
1	11,91	11,49	11,59	11,36	
2	12,36	11,62	11,64	11,42	
3	12,22	11,50	11,33	11,51	
4	12,32	11,69	11,32	11,52	
5	11,99	11,73	11,39	11,39	
6	11,96	11,61	11,19	11,33	
7	12,21	11,60	11,42	11,41	
8	12,43	11,66	11,66	11,40	
9	12,13	11,49	11,40	11,35	

Tasas de Rechazo:

Tabla E.9: Tasas de Rechazo en el escenario NS en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	1,15	0,67	0,52	0,37	0,36
1	1,19	0,75	0,54	0,46	
2	1,15	0,76	0,53	0,45	
3	1,00	0,81	0,54	0,44	
4	1,23	0,82	0,64	0,41	
5	1,31	0,73	0,57	0,41	
6	1,10	0,77	0,61	0,41	
7	0,99	0,76	0,53	0,43	
8	0,95	0,78	0,69	0,41	
9	1,09	0,74	0,65	0,44	

E.4 Escenario SN

Tasas de Acierto:

Tabla E.10: Tasas de acierto en el escenario SN en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	85,03	86,45	86,79	86,95	**87,04**
1	85,70	86,36	86,64	86,49	
2	85,19	86,42	86,69	86,82	
3	85,58	**86,47**	86,63	**87,01**	
4	**86,45**	86,13	86,87	86,83	
5	85,61	86,15	86,79	86,92	
6	85,56	86,33	**86,89**	86,92	
7	85,55	86,15	86,66	86,88	
8	85,44	86,04	86,71	86,92	
9	85,08	86,39	86,66	86,89	

Tasas de Fallo:

Tabla E.11: Tasas de Fallo en el escenario S en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	13,82	12,88	12,69	12,68	12,60
1	13,11	12,89	12,82	12,75	
2	13,66	12,82	12,78	12,73	
3	13,42	12,72	12,83	12,55	
4	13,66	13,05	12,49	12,76	
5	13,08	13,12	12,64	12,67	
6	13,34	12,90	12,50	12,67	
7	13,46	13,09	12,81	12,69	
8	13,61	13,18	12,60	12,67	
9	13,83	12,87	12,69	12,67	

Tasas de Rechazo:

Tabla E.12: Tasas de Rechazo en el escenario SN en la búsqueda del mejor conjunto de datos numérico de la base de datos MNIST.

Nº de test	Tamaño de la muestra				
	10.000	20.000	30.000	40.000	50.000
0	1,15	0,67	0,52	0,37	0,36
1	1,19	0,75	0,54	0,46	
2	1,15	0,76	0,53	0,45	
3	1,00	0,81	0,54	0,44	
4	1,23	0,82	0,64	0,41	
5	1,31	0,73	0,57	0,41	
6	1,10	0,77	0,61	0,41	
7	0,99	0,76	0,53	0,43	
8	0,95	0,78	0,69	0,41	
9	1,09	0,74	0,65	0,44	

F

Tablas de resultados de las pruebas alfabéticas

Se muestran en este anexo los detalles de los resultados de los cinco test realizados. En total, han sido generadas y analizadas 400.000 palabras sintéticas en las que han sido tratadas 3.343.384 letras. A continuación, se expresan en las tablas las tasas de acierto y fallo en las palabras usadas para el test en cuestión así como la tasa de acierto, fallo y rechazo en las letras analizadas de las palabras. Los resultados están ordenados por escritor y por escenario. En el caso de las palabras, se muestra también las tasas agrupadas por el uso de corrector ortográfico o sin el uso del mismo.

Resultados para el Test 1:

Tabla F.1: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para el Test 1.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,95	17,05	x	36,47	63,53	x
	Escritor 2	83,78	16,22	x	35,66	64,34	x
S	Escritor 1	37,05	62,95	x	6,06	93,94	x
	Escritor 2	54,83	45,17	x	11,36	88,64	x
NS	Escritor 1	**95,07**	4,93	x	69,38	30,62	x
	Escritor 2	**95,91**	4,09	x	62,28	37,72	x
SN	Escritor 1	65,74	34,26	x	17,94	82,06	x
	Escritor 2	74,85	25,15	x	26,32	73,68	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,66	0,88	10,46
	Escritor 2	87,88	5,07	7,04
S	Escritor 1	69,54	5,48	24,97
	Escritor 2	70,72	5,66	23,62
NS	Escritor 1	**95,70**	1,61	2,68
	Escritor 2	**95,48**	2,28	2,23
SN	Escritor 1	80,75	6,26	13,00
	Escritor 2	81,42	6,27	12,31

Resultados para el Test 2:

Tabla F.2: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para el Test 2.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,52	17,48	x	36,93	63,07	x
	Escritor 2	83,4	16,6	x	35,54	64,46	x
S	Escritor 1	37,39	62,61	x	6,2	93,8	x
	Escritor 2	56,15	43,85	x	11,8	88,2	x
NS	Escritor 1	**95,15**	4,85	x	69,11	30,89	x
	Escritor 2	**95,29**	4,71	x	61,77	38,23	x
SN	Escritor 1	65,97	34,03	x	18,16	81,84	x
	Escritor 2	75,4	24,6	x	25,8	74,2	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,69	0,88	10,43
	Escritor 2	87,80	5,16	7,02
S	Escritor 1	69,50	5,54	24,96
	Escritor 2	71,69	5,76	22,55
NS	Escritor 1	**95,69**	1,61	2,70
	Escritor 2	**95,32**	2,75	1,92
SN	Escritor 1	80,80	6,23	12,98
	Escritor 2	81,93	6,32	11,75

Resultados para el Test 3:

Tabla F.3: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para el Test 3.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,61	17,39	x	36,86	63,14	x
	Escritor 2	83,46	16,54	x	35,45	64,52	x
S	Escritor 1	36,63	63,37	x	6,33	93,67	x
	Escritor 2	54,91	45,09	x	11,26	88,74	x
NS	Escritor 1	**95,09**	4,94	x	68,8	31,2	x
	Escritor 2	**95,69**	4,31	x	62,19	37,81	x
SN	Escritor 1	64,78	35,22	x	17,92	80,02	x
	Escritor 2	75,03	24,97	x	25,54	74,46	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,71	0,92	10,37
	Escritor 2	87,79	5,15	7,06
S	Escritor 1	69,53	5,52	24,95
	Escritor 2	72,38	5,83	21,78
NS	Escritor 1	**95,67**	1,64	2,68
	Escritor 2	**95,21**	3,10	1,68
SN	Escritor 1	80,79	6,14	13,06
	Escritor 2	82,31	6,36	11,33

Resultados para el Test 4:

Tabla F.4: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para el Test 4.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,86	17,14	x	36,67	63,33	x
	Escritor 2	83,34	16,66	x	35,79	64,21	x
S	Escritor 1	37,22	62,78	x	5,96	94,04	x
	Escritor 2	54,85	45,45	x	11,51	88,49	x
NS	Escritor 1	**95,38**	4,62	x	69,42	30,58	x
	Escritor 2	**95,27**	4,73	x	61,58	38,42	x
SN	Escritor 1	65,64	34,36	x	17,75	82,25	x
	Escritor 2	75,08	24,92	x	25,68	74,32	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,70	0,91	10,38
	Escritor 2	87,81	5,14	7,04
S	Escritor 1	69,52	5,51	24,97
	Escritor 2	72,91	5,90	21,19
NS	Escritor 1	**95,70**	1,65	2,65
	Escritor 2	**95,12**	3,37	1,45
SN	Escritor 1	80,78	6,16	13,05
	Escritor 2	82,60	6,39	11,01

Resultados para el Test 5:

Tabla F.5: Tasas de reconocimiento de palabras y letras en cada uno de los escenarios para el Test 5.

PALABRAS							
		Con Corrector			Sin Corrector		
		acierto	fallo	rechazo	acierto	fallo	rechazo
N	Escritor 1	82,55	17,45	x	37,27	62,75	x
	Escritor 2	83,72	16,28	x	36,97	63,03	x
S	Escritor 1	36,55	63,45	x	5,74	94,26	x
	Escritor 2	55,27	44,73	x	11,75	88,25	x
NS	Escritor 1	**95,07**	4,93	x	68,37	31,63	x
	Escritor 2	**95,7**	4,3	x	62,05	37,95	x
SN	Escritor 1	65,58	35,42	x	17,47	82,53	x
	Escritor 2	74,55	25,45	x	25,78	74,22	x

LETRAS				
		acierto	fallo	rechazo
N	Escritor 1	88,70	0,91	10,38
	Escritor 2	87,87	5,12	7,00
S	Escritor 1	69,45	5,52	25,04
	Escritor 2	73,34	5,94	20,71
NS	Escritor 1	**95,67**	1,65	2,67
	Escritor 2	**95,06**	3,59	1,35
SN	Escritor 1	80,74	6,18	13,08
	Escritor 2	82,82	6,43	10,75

G

Muestra de las imágenes

En este capítulo se exponen algunas de las imágenes usadas en los experimentos. Dentro de esta muestra, se pueden hallar las siguientes imágenes:

- Imágenes de letras empleadas para la generación de palabras sintéticas clasificadas por los escritores.
- Imágenes de números de la base de datos MNIST.

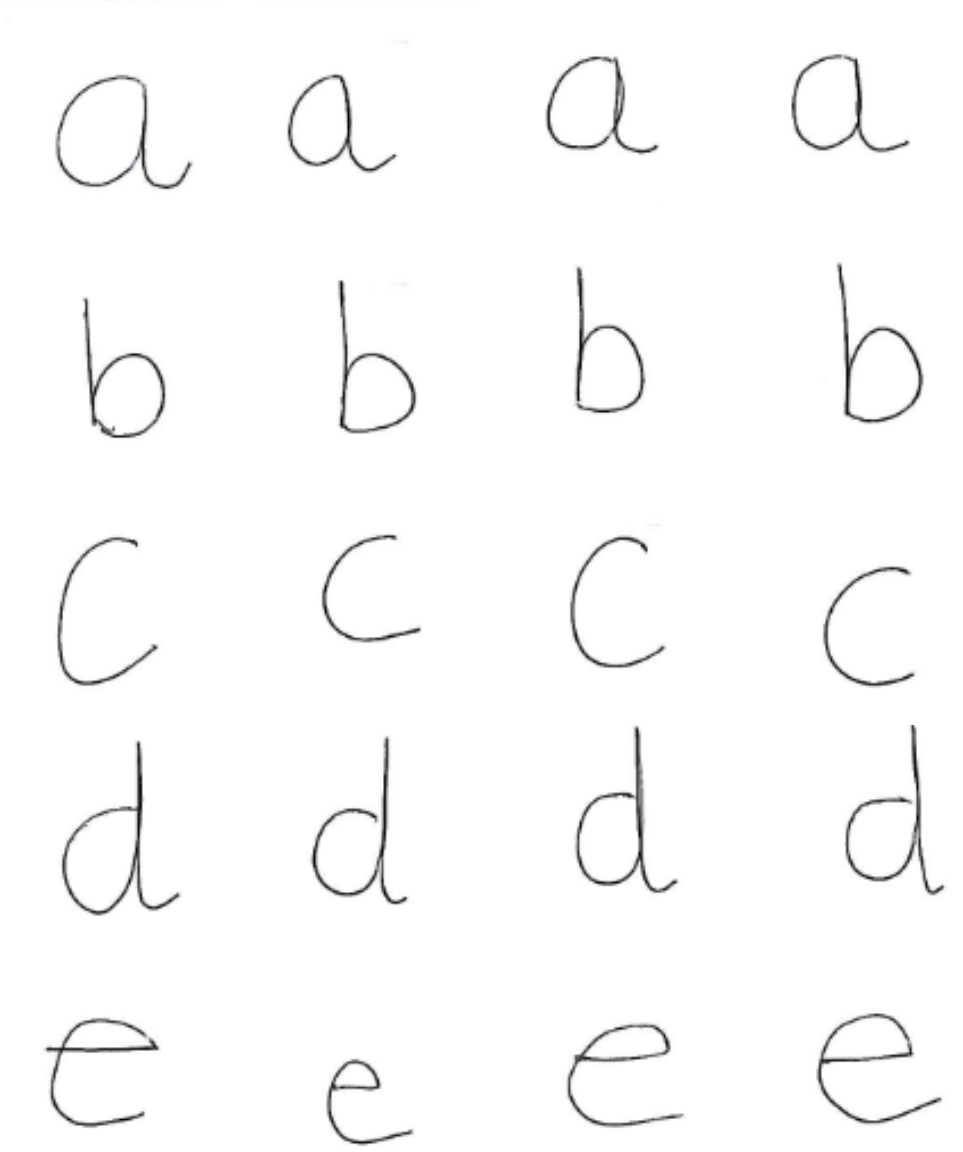


Figura G.1: Muestra de letras escritas por el escritor 1

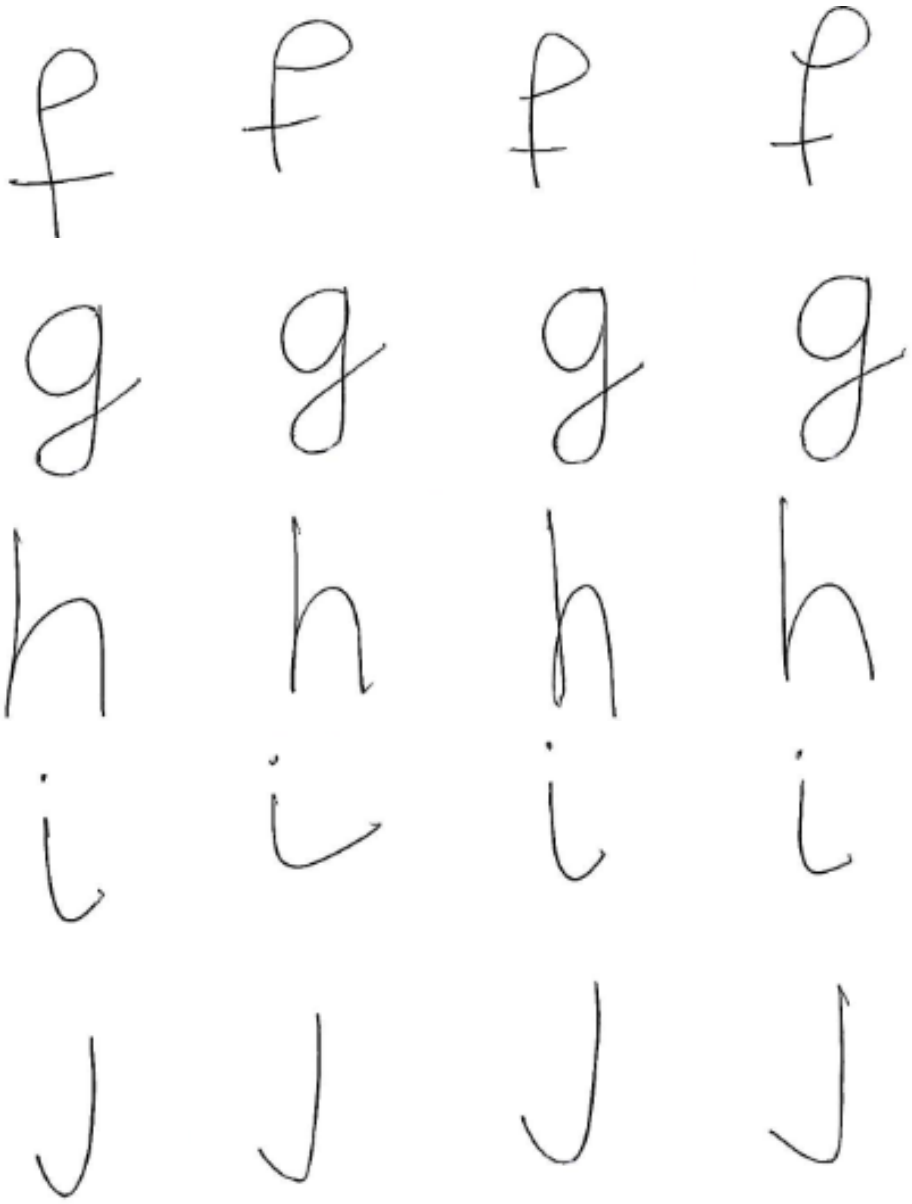


Figura G.2: Muestra de letras escritas por el escritor 1 (cont.)

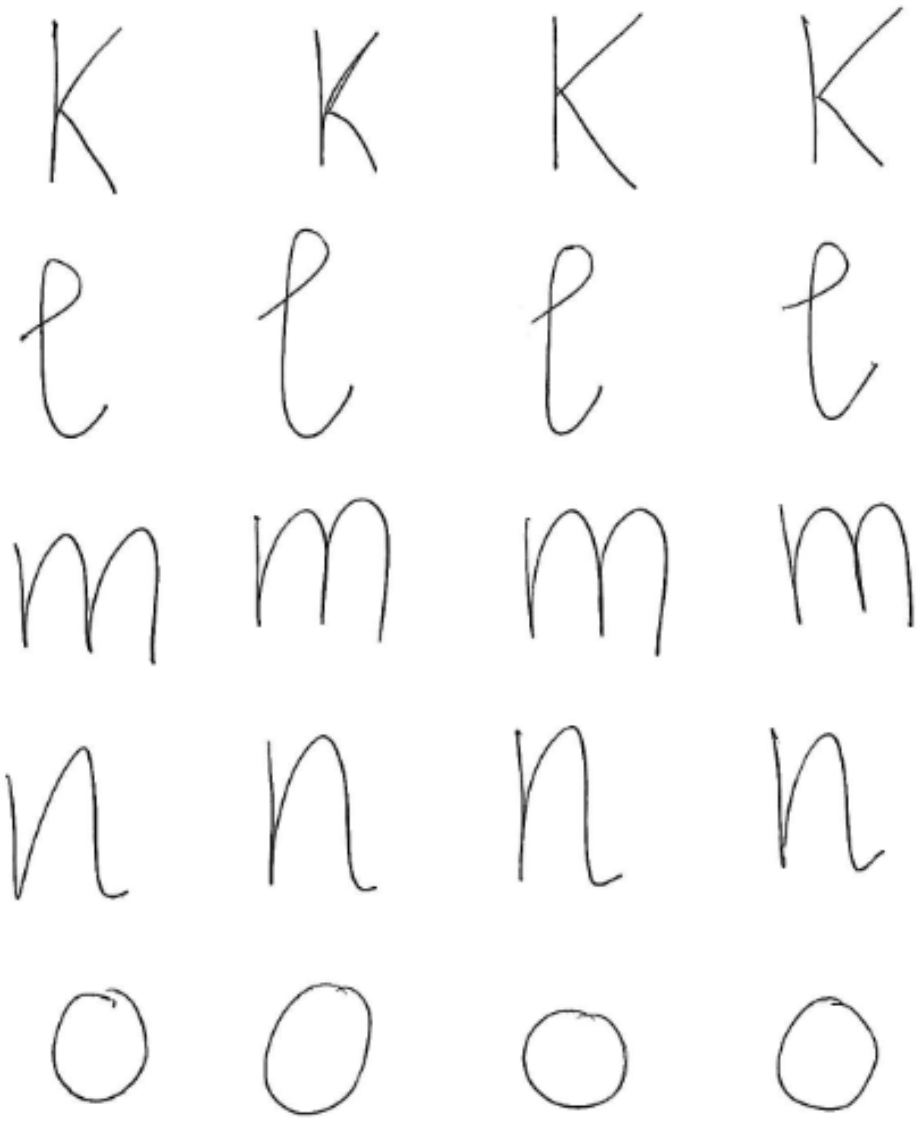


Figura G.3: Muestra de letras escritas por el escritor 1 (cont.)

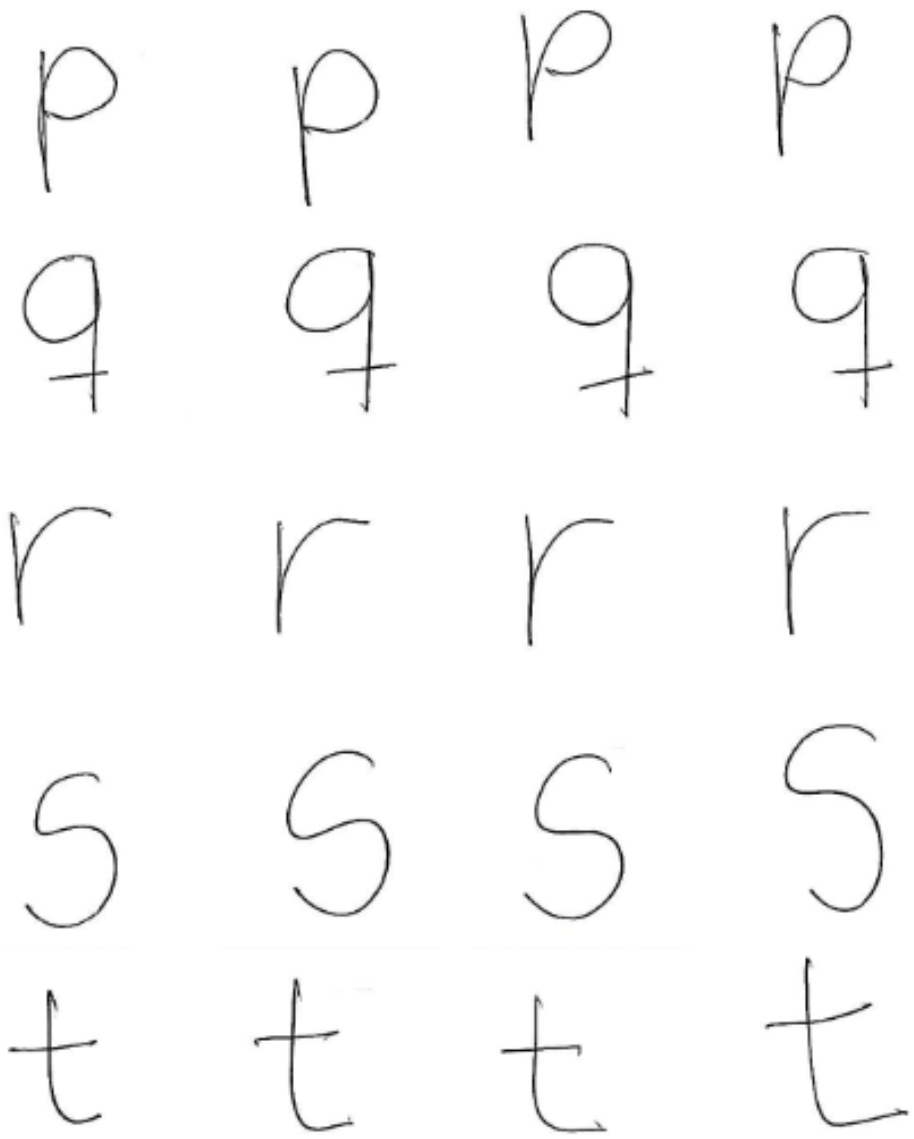


Figura G.4: Muestra de letras escritas por el escritor 1 (cont.)

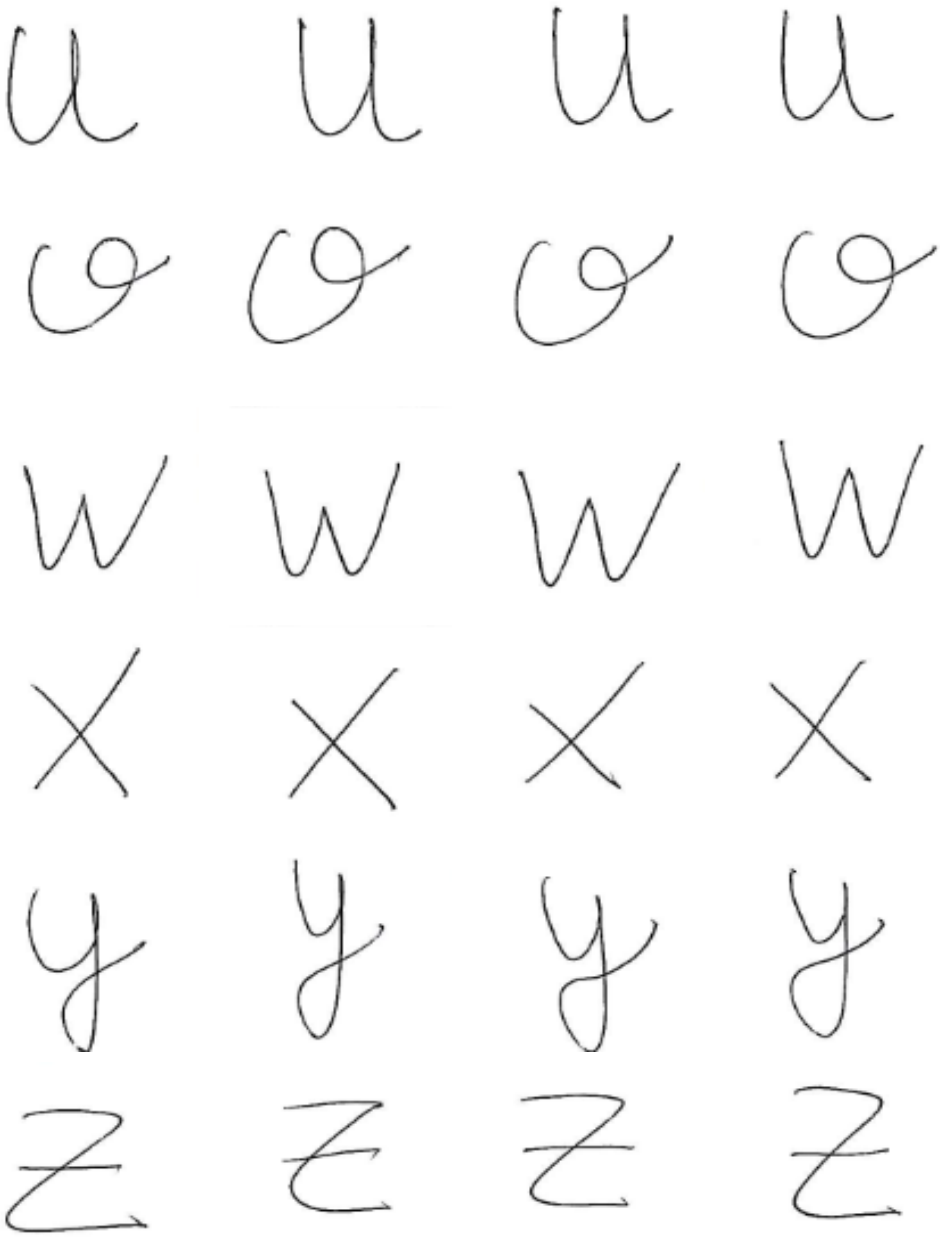


Figura G.5: Muestra de letras escritas por el escritor 1 (cont.)

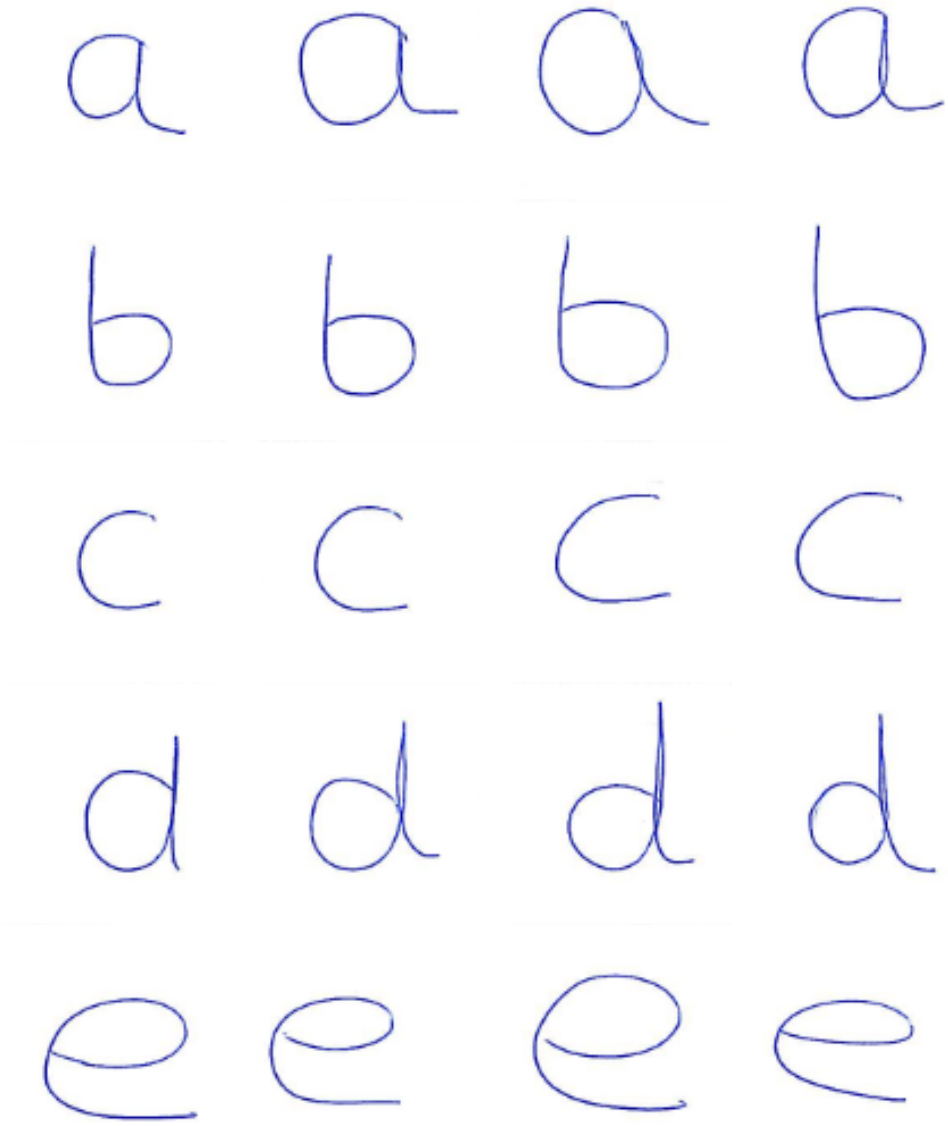


Figura G.6: Muestra de letras escritas por el escritor 2

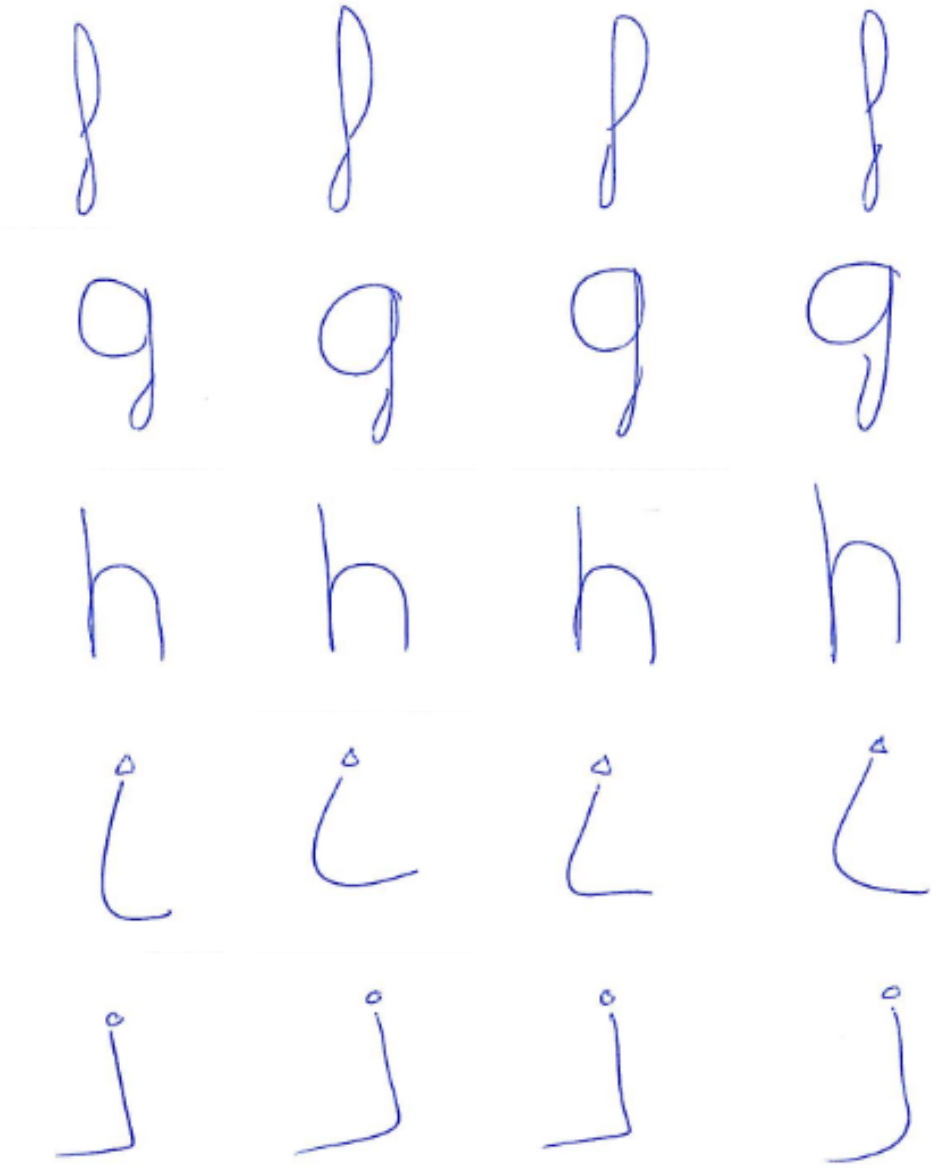


Figura G.7: Muestra de letras escritas por el escritor 2 (cont.)

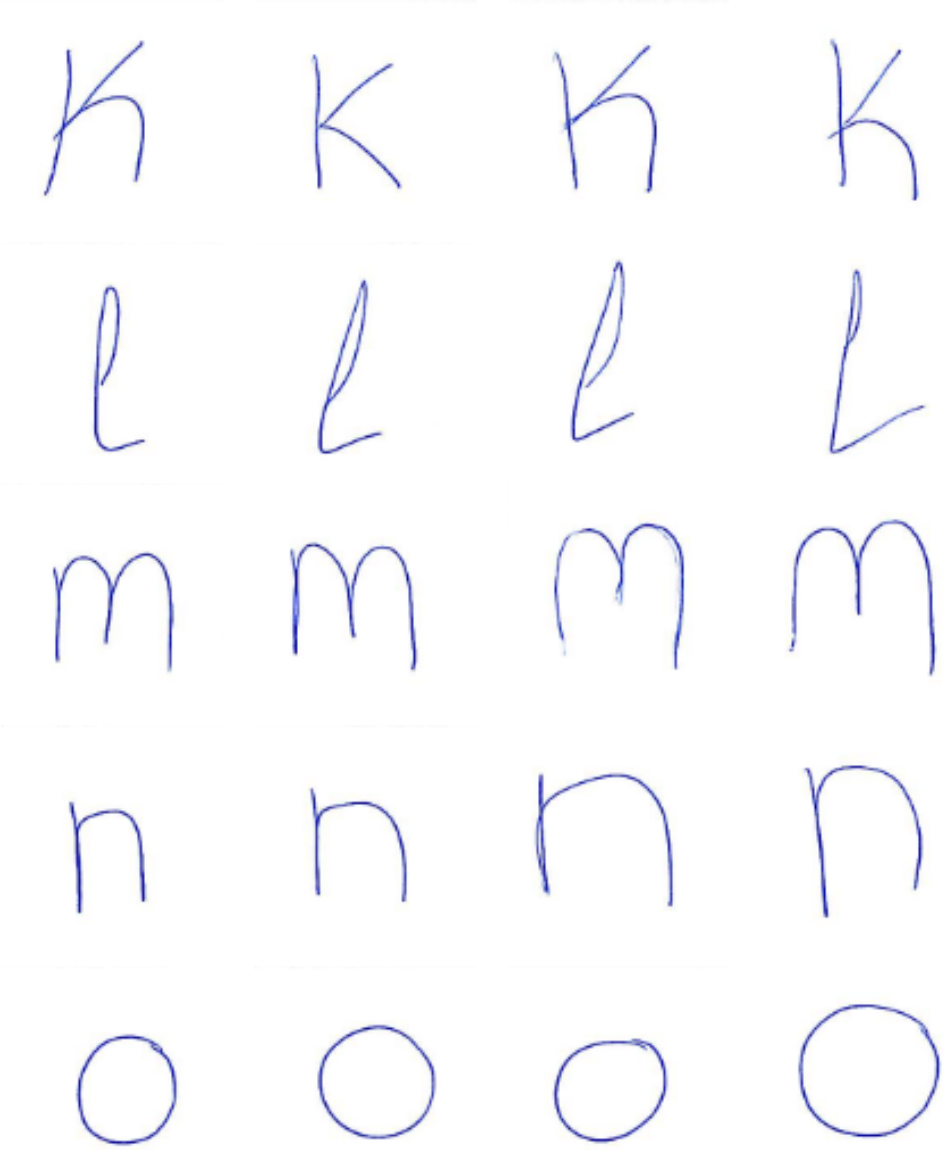


Figura G.8: Muestra de letras escritas por el escritor 2 (cont.)

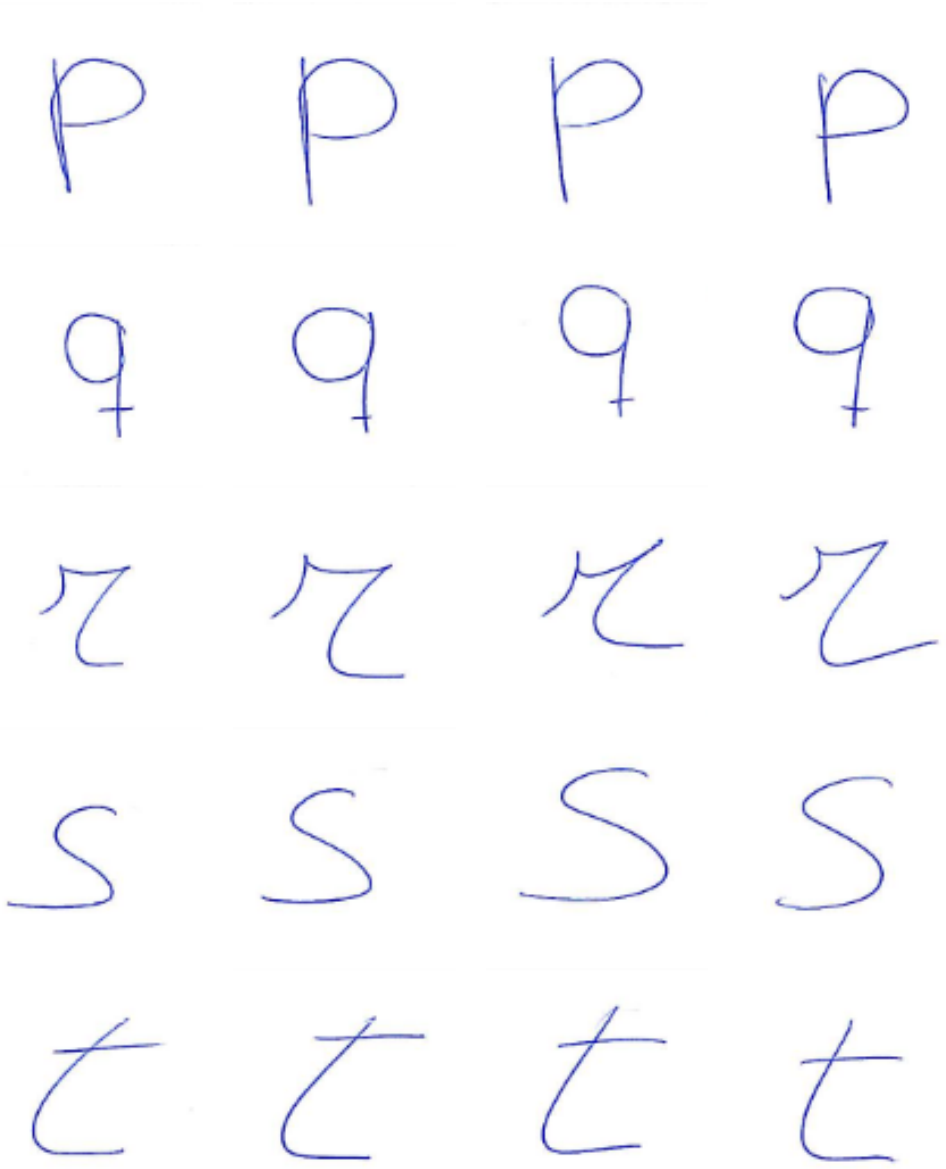


Figura G.9: Muestra de letras escritas por el escritor 2 (cont.)

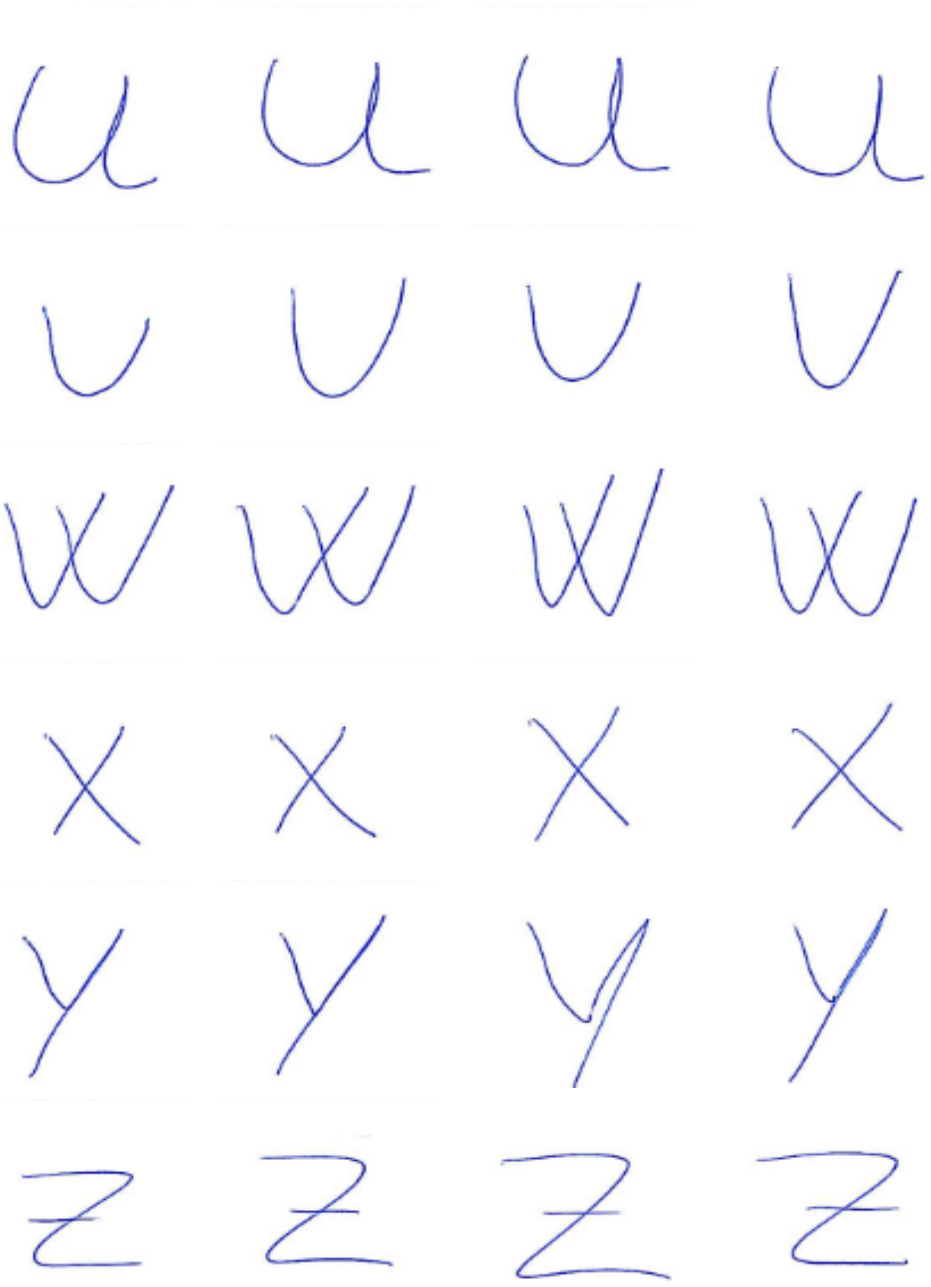


Figura G.10: Muestra de letras escritas por el escritor 2 (cont.)

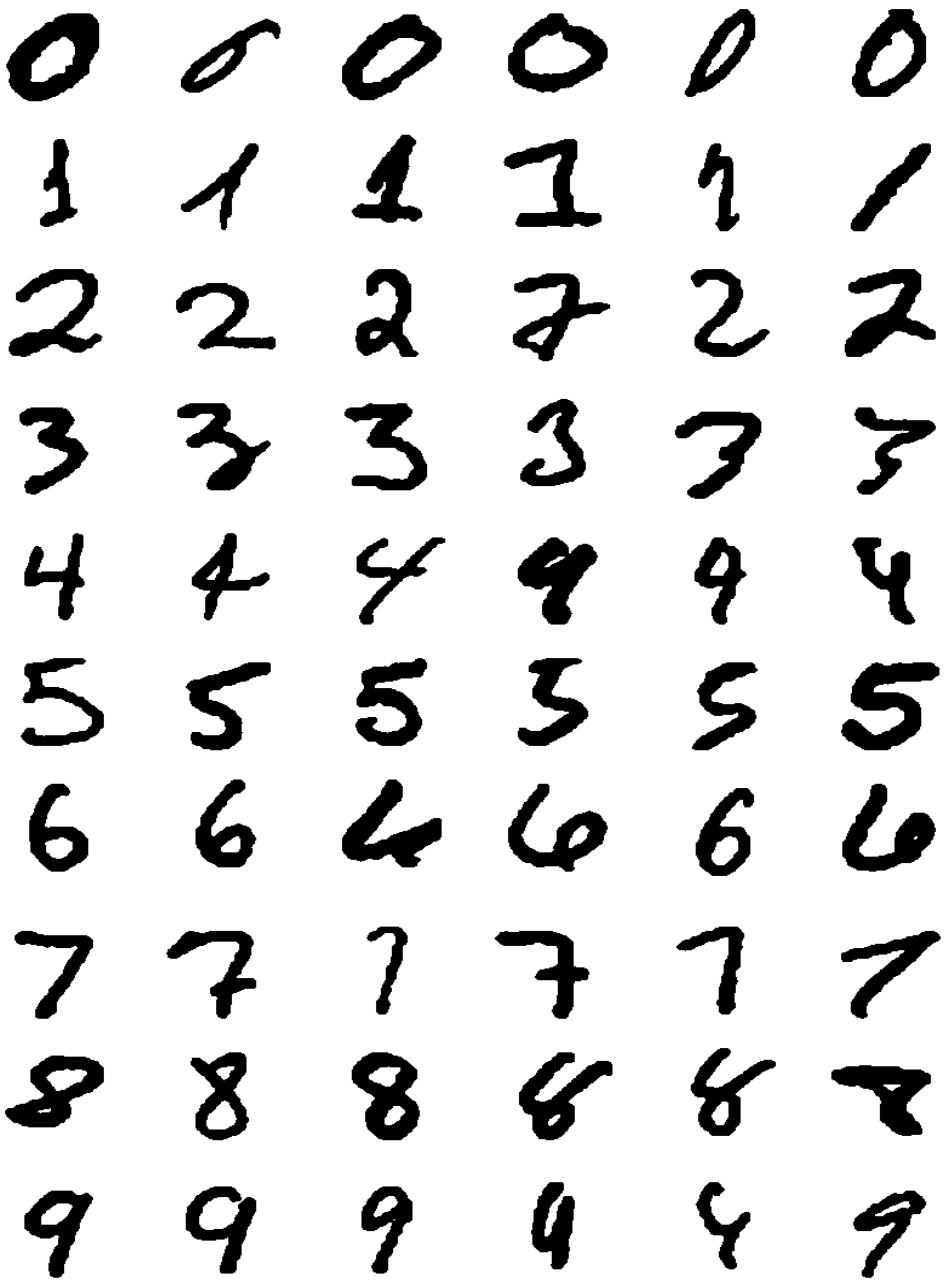


Figura G.11: Muestra de números extraídos de la base de datos MNIST

H

Publicaciones

1. D. Álvarez, R. Fernández, and L. Sánchez *Stroke-based Intelligent Character Recognition using a Deterministic Finite Automaton*. Logic Jnl IGPL 2015 23: 463-471.

2. D. Álvarez, R. Fernández, and L. Sánchez. *Stroke Based Handwritten Character Recognition*. Hybrid Artificial Intelligent Systems, volume 7208 of Lecture Notes in Computer Science, pages 343–351. Springer Berlin Heidelberg, 2012.

3. D. Álvarez, R. Fernández, and L. Sánchez. *Stroke-based Intelligent Word Recognition using a Formal Language*. 10th International Conference on Soft Computing Models in Industrial and Environmental Applications, volume 368 of Advances in Intelligent Systems and Computing, pages 101–110. Springer International Publishing, 2015.

4. D. Álvarez, R. Fernández, L. Sánchez, and J. Alija. *Expert System for Handwritten Numeral Recognition using Dynamic Zoning*. Hybrid Artificial Intelligent Systems, volume 9121 of Lecture Notes in Computer Science, pages 125–135. Springer International Publishing, 2015.