E.T.S. de Ingeniería Industrial,
Informática y Telecomunicación

Evaluación de procesos de reconocimiento óptico de caracteres y detección de tablas para la clasificación automática de documentos y su integración en un gestor documental

Grado en Ingeniería Informática

Trabajo Fin de Grado

Leyre Ayllón Lafuente

Mikel Galar Idoate, Mikel Sesma Sara

Pamplona, 11 de Junio de 2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

En este trabajo de fin de grado pretendemos desarrollar un algoritmo de detección de tablas en imágenes de documentos, complementando las funcionalidades de una librería de código abierto de reconocimiento de texto en imágenes, con el fin de realizar una clasificación automática de los documentos.

Para facilitar las tareas de detección de tablas y reconocimiento de texto aplicaremos una fase previa de preprocesamiento de imágenes. El algoritmo desarrollado será aplicado a documentos escaneados para obtener características (palabras, número de páginas, número de tablas y estructura de las mismas…) que permitan crear un modelo de clasificación de documentos no estructurados mediante redes neuronales.

La red neuronal se entrenará con documentos previamente etiquetados y permitirá la predicción de nuevos documentos en el momento de la digitalización de estos.

El algoritmo desarrollado completo de preprocesamiento, detección de tablas y de texto y clasificación de imágenes escaneadas se integrará en una plataforma de gestión documental.

PALABRAS CLAVE

Listado de palabras clave: Clasificación de documentos, redes neuronales, procesamiento de imagen, OCR, detección de tablas.

ÍNDICE

INTRODUCCIÓN

La Inteligencia Artificial está cada vez más presente en nuestro día a día. En su objetivo de emular capacidades del intelecto humano, esta serie de tecnologías ha conseguido abarcar numerosos campos: Automatización del hogar, asistentes de voz, detección de enfermedades, asistentes de atención al cliente, predicción, conducción autónoma, etc. [1, pp. 31–33].

Una de las principales aplicaciones de la Inteligencia Artificial es la clasificación. Este es un proceso de aprendizaje automático o Machine Learning supervisado, es decir, aprendizaje a partir de datos de los que se conoce la clase (datos etiquetados). Esta clasificación se puede llevar a cabo a través de varios métodos, entre los que destaca el conocido como Deep Learning, que utiliza redes neuronales artificiales con niveles jerárquicos. Por nuestra necesidad en particular, vamos a centrarnos en la clasificación de documentos, que supone actualmente un gran campo de aplicación debido a la digitalización que instituciones y empresas se ven en la necesidad de llevar a cabo.

Para realizar este tipo de clasificación es necesario realizar un procesamiento de lenguaje natural (NLP) para poder realizar la interacción entre lenguaje humano y computador. Uno de los procesadores de lenguaje más utilizados en documentos es el conocido como Bag of Words, que representa un documento básicamente por las palabras que contiene sin tener en cuenta el orden de estas.

Por otro lado, otra de las grandes aplicaciones de la Inteligencia Artificial es la visión por computador o visión artificial. Este subcampo incluye métodos para analizar, procesar y entender las imágenes con el fin de obtener información de ellas. En nuestro caso la utilizaremos para obtener información de imágenes de documentos.

Entorno de trabajo

El Trabajo de Fin de Grado realizado resulta de una idea surgida en la empresa en la que se han realizado las prácticas curriculares desde septiembre de 2019, Informática El Corte Inglés S.A. Esta empresa se especializa en soluciones de transformación digital, y por tanto cuenta con un gran número de puestos de digitalización de documentos.

En esta empresa he formado parte de un equipo de desarrollo de soluciones digitales, dirigido por mi tutor de prácticas, y se me han proporcionado todas las herramientas necesarias para el desarrollo de este trabajo.

Motivación

Como hemos mencionado anteriormente, en la empresa hay una gran cantidad de puestos de digitalización. Generalmente, en estos puestos el trabajo consiste principalmente en el escaneo de documentos para su posterior archivado en forma de imagen o archivo PDF en una aplicación de registro de documentos, con opción de indicar manualmente la categoría a la que pertenece el documento.

En este ámbito, en el equipo de trabajo del que yo he formado parte durante las prácticas propusimos una idea de innovación tecnológica que facilitara la tarea de estos puestos de digitalización automatizando la clasificación. Para desarrollar esta idea, decidimos utilizar tecnologías de Inteligencia Artificial tales como Deep Learning.

Para realizar la clasificación documental es preciso obtener determinadas características de los documentos, como las palabras que contienen. Esto se puede realizar mediante lo que se conoce como Reconocimiento Óptico de Caracteres (OCR). Durante el desarrollo, utilicé librerías de código abierto que realizaban esta tarea, pero ninguna de ellas obtenía otra información importante para la clasificación además del texto, como el número y la estructura de las tablas del documento. Es por esto que se vio la necesidad de desarrollar un algoritmo de preprocesamiento de imágenes y detección de tablas que, añadido a la detección de texto de las librerías de código abierto, facilitara y optimizara la tarea del algoritmo de clasificación.

Objetivos

El objetivo principal de este trabajo es desarrollar un algoritmo capaz de procesar imágenes de documentos, detectar tablas y reconocer texto en ellas para poder clasificar los documentos posteriormente con una red neuronal.

Para la consecución de este objetivo hay que alcanzar los siguientes objetivos específicos:

- Diseñar un algoritmo de procesamiento de imagen para mejorar su calidad y permitir que las tareas posteriores sean más precisas.
- Diseñar un algoritmo de detección de tablas y su estructura en imágenes.
- Obtener el texto presente en las imágenes de documentos (OCR).
- Realizar un procesado del lenguaje natural para obtener características de los documentos.
- Diseñar un algoritmo de clasificación de documentos con redes neuronales.
- Entrenar modelo de clasificación.
- Integrar algoritmo completo en aplicación de registro.

PRELIMINARES

Visión por Computador

La Visión por Computador o Visión Artificial se define como el conjunto de técnicas que permiten mejorar la calidad o facilitar la búsqueda de información en una imagen por medio de una computadora. Su objetivo es emular la visión humana utilizando técnicas de conocimiento para la toma de decisiones, por lo que es un área de la Inteligencia Artificial.

El procesamiento digital de imagen consta de varios procesos como son el realce de la imagen, la restauración, la segmentación, los procesos morfológicos y el reconocimiento entre otros [2, pp. 1–30]. En este trabajo vamos a hacer uso principalmente de técnicas de segmentación, restauración y morfología matemática sobre imágenes.

La segmentación se utiliza para extraer los objetos de una imagen para su posterior análisis. Esta se realiza mediante el particionamiento de píxeles en función de características comunes. En este trabajo utilizaremos una técnica de segmentación conocida como binarización.

La binarización tiene como objetivo principal separar elementos en una imagen. Generalmente, consiste en, sobre una imagen en escala de grises, determinar un umbral de intensidad a partir del cual transformar las intensidades de los píxeles de la imagen. Si la intensidad de un píxel es mayor o igual que el umbral establecido, dicha intensidad se transforma a 255 (color blanco). En cambio, si la intensidad es menor, se asigna el valor 0 al píxel (color negro). De esta forma se transforma la imagen original en escala de grises a una imagen en blanco y negro.

Existen variaciones de esta binarización clásica, como la conocida como binarización de Sauvola [3]. Este método calcula varios umbrales para cada píxel en lugar de un único umbral global teniendo en cuenta la media y la desviación estándar de sus píxeles vecinos, lo que se conoce como binarización adaptativa. Para un vecindario de píxeles $w(x, y)$, siendo $\mu(x, y)$ su media, $\sigma(x, y)$ su varianza, k una constante utilizada para decidir qué cantidad de texto se encuentra alrededor de cada vecindario y R una constante que representa el rango dinámico de la desviación estándar en el vecindario, el umbral $T(x, y)$ se calcula como:

$$T(x,y) = \mu(x,y) * \left[1 + k * \left(\frac{\sigma(x,y)}{R} - 1\right)\right].$$

Este método es más efectivo que la binarización clásica en imágenes degradadas con cambios de iluminación, y sirve también para eliminar el ruido.

El ruido de la imagen también puede tratarse con otros métodos. Uno de los que utilizaremos es la detección de componentes conexas en la imagen para eliminar aquellas cuyo tamaño esté por debajo de un determinado umbral, con el objetivo de eliminar manchas o ruido presente en la imagen del documento. Las componentes conexas de una imagen son conjuntos de píxeles con el mismo valor que estén conectados, es decir, sean adyacentes. Para esta detección se pueden considerar como adyacentes 4 u 8 píxeles vecinos de un píxel.

Por otro lado, la morfología matemática es un conjunto de herramientas usadas para extraer aquellos componentes de una imagen que son útiles para la representación y descripción de formas. Por ejemplo, la erosión y la dilatación son dos herramientas de este tipo que sirven para disminuir los puntos más claros o expandirlos con la ayuda de elementos estructurales de diferentes formas según el fin que se quiera conseguir. En nuestro caso las utilizaremos para detectar líneas de tablas en la imagen.

Dado un elemento estructural B con la forma deseada y siendo x cualquier píxel de color blanco de la imagen (podría realizarse con el color negro, o considerar que x es todo píxel perteneciente a un objeto), la erosión es el conjunto de todos los elementos x para los cuales B trasladado por x está presente en la imagen. Es decir, esta operación morfológica disminuye el tamaño de los objetos eliminando puntos que pueden no ser parte de ellos.

La dilatación, en cambio, es la unión de las traslaciones de B por todo elemento x. Es decir, la dilatación aumenta el tamaño de los objetos.

Si se realizan las dos operaciones conjuntamente, el tamaño de los objetos se mantiene, pero se obtienen diferentes efectos. Por ejemplo, si se aplica primero erosión y luego dilatación, lo que se conoce como apertura, con un elemento estructural rectangular horizontal, pueden obtenerse las líneas horizontales de una imagen, ya que primero se eliminan todos los píxeles que no sigan una línea horizontal y después se restaura el tamaño de las líneas [2, pp. 523–532]. Un ejemplo de esta operación de apertura con un elemento estructural rectangular horizontal sobre la imagen de la Figura 1, en la que se

muestra también el elemento estructural utilizado, puede observarse en la Figura 2, en la que se observa la erosión, y en la Figura 3, en la que se aplica la dilatación.

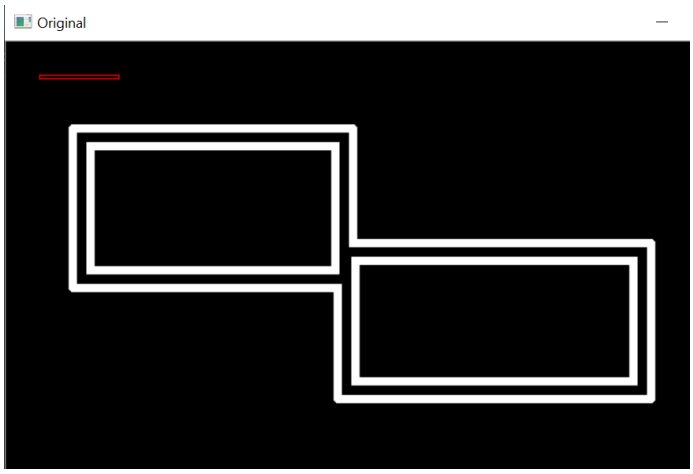


Figura 1. Imagen de entrada con el elemento estructural (rojo)

Figura 2. Procesamiento de la imagen en la Figura 1 con erosión

Figura 3. Procesamiento de la imagen en la Figura 2 con dilatación

Para detectar estas líneas utilizaremos también otra técnica conocida como Transformada de Hough, que se usa para la detección de figuras que puedan ser expresadas matemáticamente en imágenes digitales, en nuestro caso rectas.

Este método se basa en la búsqueda de puntos que satisfagan la ecuación de una recta en su forma polar

$$\rho = x * \cos\theta + y * \sin\theta$$

Para distintos valores de ρ y θ. Para ello, transforma el plano de coordenadas (x, y) en el plano de parámetros (ρ, θ).

El conjunto de rectas que pasan por (x, y) viene dado por la ecuación de la recta en forma polar, que corresponde a una sinusoide en el plano (ρ, θ). Es decir, en este plano se representa, para cada punto de la imagen, una curva sinusoidal que representa todas las rectas que pasan por ese punto.

Por tanto, el problema de detectar líneas se reduce a buscar curvas concurrentes, ya que un conjunto de puntos que forman una recta produce sinusoides con intersección en los parámetros de esa recta [4].

Por otro lado, al procesar documentos en formato de imagen, muchas veces es necesario obtener el texto presente en ellos. Esta información puede obtenerse mediante el proceso conocido como Reconocimiento Óptico de Caracteres (Optical Character Recognition, OCR), que se basa en el reconocimiento de texto en imágenes con técnicas de procesamiento de imagen y comparación con patrones [5].

Además, mediante la orientación del texto detectado se puede obtener el grado de orientación de la imagen. Esta información puede obtenerse mediante funciones de librerías de OCR de código abierto, como Tesseract, que proporcionan el grado de orientación de la imagen con un cierto grado de confianza. La detección de la orientación de la imagen de esta librería se basa en el entrenamiento de un clasificador de formas con una serie de scripts de caracteres, uno por cada lenguaje diferente posible. Para la imagen a detectar, el clasificador se ejecuta sobre cada componente conexa de la imagen independientemente y se repite el proceso rotando cada componente conexa un número determinado de grados. El proceso guarda el número estimado de caracteres detectados de cada script para una orientación determinada y la confianza acumulada del clasificador para cada orientación candidata. La orientación de la imagen se

determina como la orientación candidata con mayor confianza acumulada del clasificador [6].

Otro método utilizado para este fin consiste en encontrar el rectángulo de área mínima que comprende todo el texto de la imagen y hallar su grado de rotación. Este proceso se realiza guardando todos los píxeles del color del texto, preferiblemente después de binarizar la imagen, y hallando el rectángulo que los contiene a todos y su orientación. Con la información obtenida, se puede rotar la imagen de forma que el texto esté correctamente orientado para facilitar las tareas de detección que se llevan a cabo sobre la imagen.

Machine Learning

El Machine Learning o Aprendizaje Automático es un campo de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a un agente aprender de la experiencia. En concreto, Tom Mitchell [7] en 1997 lo definió como "Un programa se dice que aprende de una experiencia E respecto a una tarea T y alguna medida de rendimiento P, si su rendimiento en T, medido mediante P, mejora con la experiencia E".

El aprendizaje automático puede ser de dos tipos [1, pp. 740–742]:

- **Aprendizaje supervisado.** Se conoce la salida esperada para los datos de entrada. Es decir, los datos están "etiquetados". Dentro de este tipo se encuentran la regresión y la clasificación.
    - **Regresión.** El valor de salida para un dato es un valor real.
    - **Clasificación.** El valor de salida para los datos es un valor nominal o discreto.
- **Aprendizaje no supervisado.** No se conoce la salida deseada para los datos de entrada. El clustering es de este tipo, se basa en la agrupación de datos según su estructura o características comunes.

Algunos de los principales modelos de aprendizaje utilizados para resolver problemas de clasificación son:

- Regresión logística. Establece una única frontera de decisión entre clases en base a una ecuación que da diferentes pesos a los atributos y predice la clase de un dato en base a su probabilidad de ocurrencia.

- Árboles de decisión. Clasificadores que dividen el espacio de datos en varias regiones pequeñas, una o varias para cada clase, en base a los valores de los atributos, por lo que establece varias fronteras de decisión.
- SVM. También conocidas como Máquinas de Vectores Soporte, permiten clasificar datos de entrada basándose en la construcción de un hiperplano que separa los ejemplos de cada clase. La construcción de este hiperplano se realiza con principios geométricos, es decir, se busca el plano equidistante de ambas clases y que maximice el margen entre él y las clases. Estos modelos no aceptan clasificación multiclase inicialmente, aunque puede realizarse con otros métodos (One-vs-One, One-vs-All) que descomponen el problema en problemas binarios, es decir, de dos clases.
- Redes neuronales. Son modelos que tratan de imitar el aprendizaje humano.

Nos centraremos en las redes neuronales dado que las vamos a utilizar en este trabajo.

Redes neuronales

Las redes neuronales son algoritmos que tratan de simular el comportamiento del cerebro humano. Su unidad logística, por analogía con las neuronas, es el perceptrón, que puede observarse en la Figura 4. En esta figura, x_i es el atributo i del ejemplo X de entrada, θ_i el peso para el atributo i y $h_\theta(x)$ la salida del perceptrón.

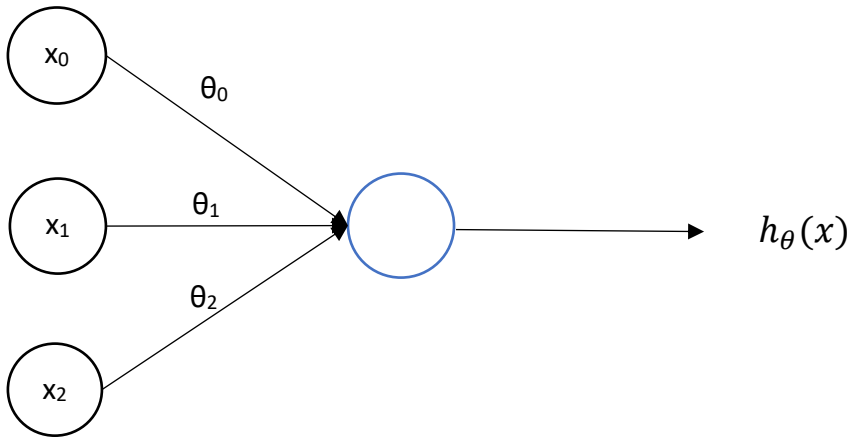


Figura 4. Representación de un perceptrón

Un perceptrón recibe como entrada los atributos del ejemplo a clasificar y produce como salida la clasificación del ejemplo utilizando para ello una función de activación que suele ser la función sigmoide, cuya fórmula viene dada por:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Una extensión de esta función para el caso de problemas multiclase, donde la salida es un vector cuya dimensión coincide con el número de clases, es la función conocida como softmax. Esta función asigna probabilidades decimales a cada clase, cuya suma debe ser 1. Su fórmula para K clases se muestra a continuación:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Para $i = 1, \ldots, K$ y $z = (z_1, \ldots, z_K) \in \mathbb{R}^K$.

Sin embargo, se pueden utilizar muchas otras funciones. Entre ellas la conocida como ReLU (Rectified Linear Unit), cuya fórmula viene dada por:

$$R(z) = \max(0, z)$$

Esta última es una de las funciones más utilizadas actualmente en Deep Learning, dado que permite un aprendizaje más rápido por su rápida convergencia y una mayor capacidad de generalización si se utiliza en las capas ocultas. Esta función soluciona un posible problema de la función sigmoide, que puede hacer que el gradiente tienda a 0 en las capas cercanas a la de entrada y ralentiza así su entrenamiento.

La salida del perceptrón con la función sigmoide como función de activación se calcula con la siguiente fórmula:

$$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)$$

donde x_0 es el conocido como bias y es igual a 1, dado que no es parte de los atributos de entrada, sino que se incorpora para poder operar θ y X en su representación matricial.

A partir de esta unidad logística se construyen las redes neuronales multicapa (MLP), que se componen de capa de entrada, que es la capa de los datos, capas ocultas o intermedias con una o más neuronas en cada una y capa de salida, que corresponde a la salida proporcionada por la red neuronal. En la Figura 5 se puede observar una arquitectura con dos capas ocultas.

En el caso de problemas multiclase, esta capa de salida tendrá tantas neuronas como clases, siendo la salida un vector de dimensión el número de clases con valores entre 0 y 1, donde la posición en que se encuentra el mayor valor corresponde a la clase predicha.

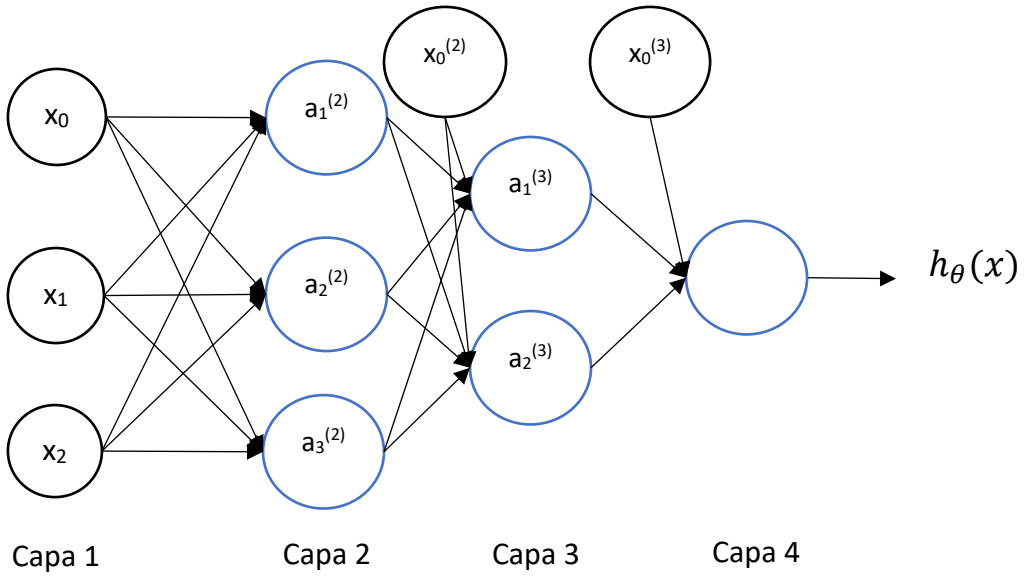


Figura 5. Red neuronal multicapa con dos capas ocultas

Además, la red neuronal tiene asignada una matriz de pesos θ^j para cada capa j con los pesos de las conexiones entre la capa j y la capa j + 1. Cada neurona tiene un valor de activación dado por la aplicación de la función de activación a los valores de entrada de dicha neurona junto con los pesos correspondientes. Es decir, los valores de activación de las neuronas de una capa se calculan en base a los valores de la capa anterior.

En la Figura 5, $a_i^{(j)}$ representa la activación de la neurona i en la capa j. Por ejemplo, la activación para la neurona 1 de la capa 3 se calcularía de la siguiente forma:

$$a_1{}^{(3)} = g(\theta_{10}{}^{(2)} x_0{}^{(2)} + \theta_{11}{}^{(2)} a_1{}^{(2)} + \theta_{12}{}^{(2)} a_2{}^{(2)} + \theta_{13}{}^{(2)} a_3{}^{(2)})$$

El cálculo de estos valores de activación desde la entrada de los datos hasta la salida de la red neuronal se conoce como propagación hacia adelante. Las capas ocultas permiten que la red neuronal aprenda sus propias características.

La potencia principal de las redes neuronales, además de en la propagación hacia adelante, reside en la propagación hacia atrás o backpropagation, dado que permite actualizar los pesos de la red en función del error obtenido. Esta se realiza para entrenar la red y consiste en, tras realizar la propagación hacia adelante, calcular el gradiente del error total respecto de cada peso de la red para actualizar los valores de los pesos con descenso por gradiente u otros métodos con el objetivo de minimizar la función de coste [1, pp. 838–851].

Esta función de coste, $J(\theta)$, determina la diferencia entre el valor estimado y el valor real. Se pueden utilizar muchos tipos de funciones de coste. Para la clasificación de variables categóricas destaca la entropía cruzada categórica, cuya fórmula viene dada por:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

donde n es el número de ejemplos de entrenamiento, m es el número de clases, p_{ij} es el valor predicho para el ejemplo i y la clase j y y_{ij} es el valor real para el ejemplo i y la clase j [8, pp. 205–210].

Además, para prevenir el sobreajuste de la red neuronal se puede utilizar regularización, que es un método que penaliza la complejidad del modelo. Cuanto más complejo es un modelo, más tiende a sobreajustarse a los datos de entrenamiento.

La regularización se realiza añadiendo un término a la función de coste del modelo. Una de las técnicas más utilizadas en redes neuronales es la regularización Lasso o L1, que penaliza los pesos en proporción a la suma de los valores absolutos de los pesos [8, pp. 144–147].

Por último, como en la mayoría de modelos de aprendizaje automático, es recomendable normalizar los datos de entrada antes de pasárselos al modelo para hacer que sigan una distribución normal y evitar que unos atributos tengan más relevancia que otros. Generalmente, esta normalización se realiza, para cada dato x, de la siguiente forma:

$$z = \frac{x - \mu}{\sigma}$$

donde μ es la media de los ejemplos de entrenamiento y σ su desviación típica.

Bag of Words

Para la clasificación de texto es necesario realizar un procesamiento del lenguaje natural que pueda transformarlo en un vector de características numéricas, proceso que recibe el nombre de vectorización.

Un modelo comúnmente utilizado para representar documentos es el conocido como Bag of Words [9, pp. 67–77]. Este se basa en la representación de un documento por las palabras que contiene sin importar el orden.

Este modelo necesita un conjunto de documentos de entrenamiento, conocido como corpus, del que guarda todas las palabras en un diccionario. A continuación construye una matriz X de conteos en la que el elemento X[i, j] representa el número de ocurrencias de la palabra de la posición j del diccionario en el documento i. Por tanto, el número de características generado es igual al número de palabras diferentes presentes en el corpus. En la implementación se puede establecer un conjunto de palabras comunes del idioma para que sean ignoradas puesto que no aportan significado semántico.

Sin embargo, esta definición implica que documentos más largos tendrán mayores conteos que documentos cortos, incluso si ambos tratan del mismo tema. Se introduce, por tanto, el concepto de frecuencia (Term Frequency, TF), que consiste en dividir los conteos de cada palabra en un documento entre el número total de palabras del documento. Además, estas frecuencias se modifican para que las palabras que aparecen en una gran cantidad de documentos tengan una importancia reducida. Esto se debe a que las palabras que aparecen en muchos documentos del corpus (como preposiciones) proporcionan menos información que las que aparecen en una pequeña parte, puesto que estas últimas serán más específicas del tema que traten los documentos y tendrán, por tanto, mayor poder de diferenciación. Este proceso recibe el nombre de TF-IDF (Term Frequency times Inverse Document Frequency). Es decir, la frecuencia obtenida anteriormente se multiplica por la inversa de la frecuencia en todos los documentos. Esta inversa se calcula de la siguiente forma:

$$idf(t) = \log \frac{1 + n_D}{1 + df(D,t)}$$

donde n_D es el número total de documentos del corpus y *df(D, t)* es el número de documentos del corpus que contienen la palabra t. La suma de una unidad en numerador y denominador se realiza para no obtener valores inválidos para el logaritmo.

Por último, los valores obtenidos tras el cálculo de TF-IDF se normalizan por la norma euclídea. En la práctica, el modelo de Bag of Words se entrena con un conjunto de entrenamiento del que aprende las palabras y los documentos de test se transforman a esta representación en base al modelo creado.

Con un conjunto de entrenamiento grande, el número de características devuelto por este método puede ser muy alto, dado que el vocabulario es muy extenso. Para evitar el sobreajuste del modelo de clasificación a los ejemplos de entrenamiento, puede realizarse un proceso de selección de variables que reduzca el conjunto de atributos obtenidos por Bag of Words a uno menor.

En este caso, un método de selección de variables recomendado es la Descomposición en Valores Singulares (SVD) truncada, puesto que la matriz resultante del proceso tiene muchos ceros. Esta técnica reduce la dimensionalidad de la matriz realizando su descomposición en valores singulares. Para una matriz M de dimensión m x n, su descomposición viene dada por:

$$M = U\Sigma V^T$$

donde U es una matriz unitaria m x m, Σ es una matriz diagonal rectangular m x n con números no negativos en la diagonal que reciben el nombre de valores singulares de M y V es una matriz unitaria n x n.

El truncado consiste en la selección de únicamente las k columnas de U y las k filas de V^T que corresponden a los k mayores valores singulares Σ_k [10]. Es decir, para la matriz de entrenamiento M, su aproximación de bajo rango viene dada por:

$$M_k = U_k \Sigma_k {V_k}^T$$

Tras aplicar esta operación a la matriz de entrenamiento, el conjunto de entrenamiento transformado con k atributos es ${U_k \Sigma_k}^T$. Para transformar una matriz ejemplos de test X, se realiza el producto XV_k.

Evaluación de modelos

Para conocer la capacidad de generalización que tiene un modelo, es decir, si realizará un buen trabajo en la clasificación de nuevos ejemplos, es importante evaluar dicho modelo.

Para poder realizar esta evaluación, es necesario dividir el conjunto de datos etiquetados disponible en dos conjuntos distintos: train y test. El conjunto de train se utiliza para entrenar el modelo aprendiendo los parámetros de forma que se minimice el error. A continuación, se predicen los datos del conjunto de test con el modelo aprendido y se calcula el error de dichos datos. Esta división se realiza para conocer de una manera fiable la capacidad de generalización, puesto que, si se utilizan para evaluar ejemplos utilizados para entrenar el modelo, los resultados son optimistas [11, pp. 228–254].

Por otro lado, existen muchas opciones a la hora de establecer el valor de los macroparámetros del modelo tales como el parámetro de regularización, el número de neuronas en una capa, etc. La búsqueda de los valores óptimos para estos parámetros se realiza mediante el proceso conocido como selección de modelos.

Esta selección exige la división del conjunto de datos etiquetados en tres conjuntos. Además de los conjuntos de train y test necesarios para la evaluación, se requiere un tercero llamado conjunto de validación. Esto se debe a que, como antes, el conjunto de test no se puede utilizar como un estimador del error de generalización si ha sido utilizado para seleccionar el valor de un parámetro del modelo, puesto que sería un estimador optimista. Estos parámetros se eligen, por tanto, de forma que minimicen el error del conjunto de validación.

Sin embargo, el método de validación cruzada de k particiones permite evitar la división del conjunto de datos en tres conjuntos. Con esta técnica solo es necesario tener un conjunto de entrenamiento y otro de test, puesto que se puede realizar la validación sobre el conjunto de entrenamiento sin que esto suponga un sobreajuste del modelo.

La validación cruzada de k particiones divide el conjunto de entrenamiento en k particiones manteniendo la distribución de las clases. Posteriormente, se fusionan k-1 de estas particiones para formar el conjunto de entrenamiento y la partición restante se utiliza como conjunto de test. El proceso se repite k veces utilizando una partición diferente como conjunto de test cada vez.

Además, se utilizan métricas para analizar el modelo. Una métrica comúnmente utilizada para modelos de clasificación y que usaremos en este trabajo es la conocida como exactitud o ratio de clasificación (accuracy), que indica la cantidad de predicciones del modelo que son correctas (cociente entre predicciones correctas y número total de predicciones). Sin embargo, esta métrica no tiene en cuenta el número de ejemplos de cada clase, por lo que no refleja exactamente la calidad del clasificador.

Por ello, son necesarias otras medidas de evaluación que se representan en la matriz de confusión de la Tabla 1 para dos clases, pero puede generalizarse para varias clases.

	Predicción positiva	**Predicción negativa**
Clase positiva	True Positive (TP)	False Negative (FN)
Clase negativa	False Positive (FP)	True Negative (TN)

Tabla 1. Matriz de confusión

A partir de esta matriz de confusión se pueden establecer las siguientes métricas:

- Precisión. Mide la cantidad de ejemplos predichos como una clase que son realmente de esa clase.

$$Precisión = \frac{TP}{TP + FP}$$

- Recall. Mide la cantidad de ejemplos de una clase que se han predicho correctamente.

$$Recall = \frac{TP}{TP + FN}$$

Para medir la calidad de un modelo, sería necesario calcular el balance entre estas dos métricas. Este balance, denominado valor-F o F_1 score, se calcula con la media armónica para evitar obtener valores altos cuando una de las dos métricas tiene muy buen resultado pero la otra no.

$$F_1\ score = 2 * \frac{Precisión * Recall}{Precisión + Recall}$$

Cuanto mayor sea el resultado de este balance, mejor será el modelo [12].

Herramientas utilizadas

Para el desarrollo de este trabajo hemos utilizado como lenguajes de programación Python y Java. El uso del primero de ellos se debe, principalmente, a las facilidades que ofrece para el desarrollo de técnicas de procesamiento de imagen y Machine Learning. Java, por su parte, es el lenguaje utilizado por la aplicación de la empresa en la que hemos integrado el algoritmo, por lo que su uso es necesario para la integración.

El algoritmo completo está desarrollado, por tanto, con la ayuda de librerías disponibles para el lenguaje Python. Las principales librerías que hemos utilizado son:

- PyTesseract. Es una librería de reconocimiento óptico de caracteres que adapta el software de código abierto de Google llamado Tesseract OCR. Entre las funciones que ofrece están la de obtener la orientación de una imagen de un documento y la de obtener el texto presente en ella con la ayuda de un diccionario del idioma que se le indique. [13]
- Keras. Es la API de alto nivel de TensorFlow, librería de código abierto para aprendizaje automático desarrollada por Google. Se especializa en la construcción de modelos de Deep Learning. [14]
- Scikit-learn. Librería de código abierto especializada en aprendizaje automático. Incluye múltiples algoritmos de Machine Learning y herramientas necesarias durante el proceso de aprendizaje. Entre todas ellas, en este trabajo hemos utilizado sobre todo técnicas de preprocesamiento de datos (Codificación One-Hot, normalización), de extracción de características de texto (vectorización y proceso TF-IDF), de selección de variables (SVD truncada) y de normalización (Standar Scaler). [15]
- Stop-words. Librería de código abierto disponible para varios idiomas que ofrece conjuntos de palabras comunes del idioma que no aportan significado semántico y por tanto podrían ignorarse en el proceso de Bag of Words. [16]
- OpenCV. Es una librería de código abierto indicada para procesos de Visión Artificial. Ofrece múltiples funcionalidades de tratamiento de imagen que hemos utilizado para procesar las imágenes y detectar tablas en ellas. [17]
- Scikit-image. Es una librería de código abierto para el procesamiento de imágenes. También la hemos utilizado para procesar las imágenes, principalmente para realizar el proceso de binarización con la binarización de Sauvola. [18]

Para la integración en la aplicación web de la empresa, por otro lado, hemos utilizado las siguientes tecnologías:

- Maven. Es una herramienta software de Apache indicada para la creación y gestión de proyectos en lenguaje Java. Facilita el desarrollo de proyectos basándose en un POM (Proyect Object Model) para gestionar las dependencias y la construcción. Con este fin, proporciona herramientas de compilación y empaquetado de código. [19]
- Tomcat. Este software de código abierto de Apache implementa las tecnologías Java Servlet, JavaServer Pages (JSP), Java Expression Language y Java WebSocket, es decir, es un contenedor web con soporte para Servlets y JSP. Permite, por tanto, desplegar aplicaciones web. [20]

PROBLEMÁTICA

El problema, como hemos comentado antes, surge en la empresa en la que se han realizado las prácticas curriculares de los dos últimos semestres. Esta empresa, IECISA, se especializa en la provisión de soluciones digitales enfocadas a la transformación digital de compañías y administraciones públicas. En esta empresa he formado parte del equipo de desarrollo informático.

La idea del proyecto surgió, pues, en este equipo, en el que se propuso una solución de innovación tecnológica para automatizar la clasificación de documentos en los puestos de digitalización. En la investigación y desarrollo de esta solución surgieron ideas que dieron lugar al planteamiento del algoritmo completo a desarrollar en este trabajo.

El sistema a desarrollar planteado debe cumplir los siguientes requisitos:

- Automatizar la clasificación de documentos en cinco tipos diferentes de documentos administrativos mediante técnicas de Deep Learning.
- Las clases disponibles para clasificar serán:

    - Acuse de recibo. Confirmaciones de recepción de notificaciones.
    - Contestación a reclamación, solicitud o queja. Respuestas a solicitudes de información o inspección y reclamaciones.
    - Notificación. Notificaciones de resultados de resoluciones.
    - Propuesta de acuerdos. Propuestas para acordar medidas entre las partes.
    - Resolución. Documentos formales de resolución de trámites, expedientes o acuerdos.

- El sistema debe recibir como entrada un documento en formato PDF o imagen.
- El sistema completo deberá integrarse en la aplicación de registro de documentos de la empresa de forma que cada documento digitalizado sea procesado y clasificado por el algoritmo.
- La integración debe permitir la corrección manual de la clase predicha por el modelo. Además, este modelo debe ser ampliable con nuevos ejemplos para poder incorporar estos documentos de los que se corrige la clase.

Para poder cumplir estos requisitos, realizaremos los siguientes procedimientos:

- Entrenaremos un modelo de clasificación con redes neuronales multicapa a partir de un conjunto de entrenamiento formado por documentos administrativos de las cinco clases disponibles.
- Del documento recibido como entrada se debe obtener su información relevante, como texto y tablas, para poder realizar la clasificación. Para este fin utilizaremos procedimientos de OCR y un algoritmo de detección de tablas en imágenes.
- Las características obtenidas del documento se deben proporcionar como entrada al modelo de clasificación, que predecirá su clase.
- Realizaremos la integración del sistema completo desarrollado en el gestor documental de la empresa. Esta integración se hará en la parte de digitalización de la aplicación web, y podrá verse el estado de procesamiento de cada documento.
- En la integración se permitirá cambiar la clase de un documento manualmente. Cuando se corrija la clase del documento, este pasará a formar parte del conjunto de entrenamiento y el modelo se reentrenará, consiguiendo así un modelo ampliable.

DESARROLLO E IMPLEMENTACIÓN

Como hemos comentado anteriormente, el trabajo desarrollado consta de dos partes principales: Procesamiento de imágenes para la detección de tablas junto con reconocimiento de texto y clasificación documental.

A continuación, se detallan las funciones y la metodología empleada para desarrollar cada una de las diferentes partes.

Visión por Computador

Preprocesamiento de imágenes

El algoritmo comienza con una fase de preprocesamiento de imágenes para mejorar su calidad y optimizar las tareas posteriores.

En primer lugar, si el formato del documento entrada es de tipo PDF, este se convierte a imagen (una imagen por cada página del documento). En consiguiente se trabaja sobre cada imagen correspondiente al documento.

Sobre la imagen se aplican varios procedimientos de tratamiento de imágenes que permiten mejorar su calidad. Por los fines para los que se utiliza la imagen, se utilizan los siguientes métodos:

- **Conversión a escala de grises.** La imagen se transforma a escala de grises en el momento de su lectura para facilitar las tareas dado que el formato de color en tres dimensiones no aporta gran información al algoritmo.
- **Detección de orientación y rotación.** Se detecta la orientación de la imagen a partir de la librería Pytesseract, que proporciona dicha orientación con cierta confianza. Si la confianza es menor que un determinado umbral fijado, se aplica otro método con la ayuda de la librería OpenCV que se basa en encontrar el rectángulo de área mínima que comprende todo el texto y hallar su grado de rotación. Con la información obtenida, se rota la imagen de forma que el texto esté correctamente orientado, como puede visualizarse en la Figura 6 y Figura 7.

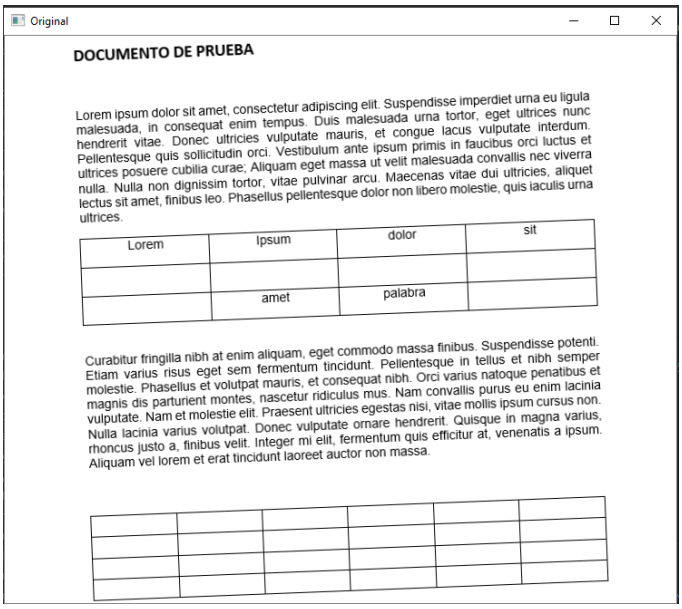


Figura 6. Imagen de entrada

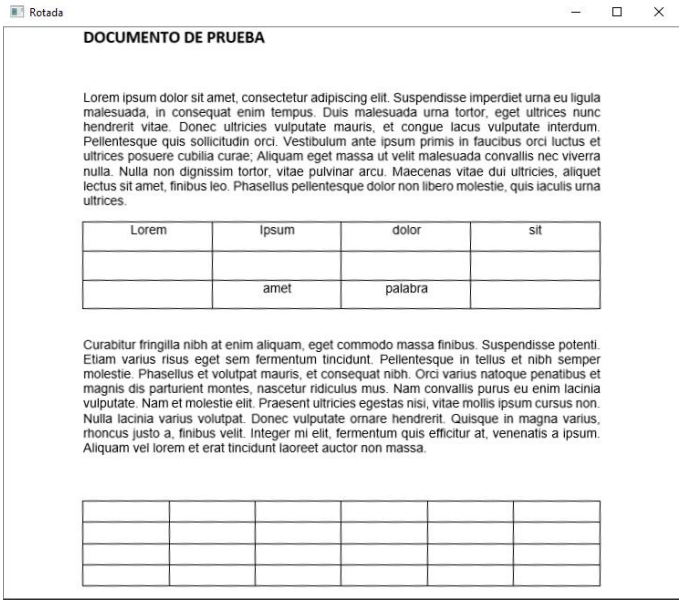


Figura 7. Imagen obtenida al rotar la imagen original el ángulo detectado (-2.231 grados)

- **Binarización de Sauvola.** La binarización es un proceso muy utilizado en el procesamiento de imágenes que tiene como objetivo reducir el ruido y distinguir objetos del fondo de objetos del primer plano (segmentar) estableciendo un umbral. Existen diferentes formas de binarizar, y una de las más apropiadas para el reconocimiento de texto es la conocida como binarización de Sauvola. Está indicada para el tratamiento de documentos donde el fondo no es uniforme y, por tanto, dado que el fondo de los documentos contiene muchas veces ruido o

manchas, ofrece mejores resultados en imágenes de documentos que otras binarizaciones. Este método calcula varios umbrales para cada píxel en lugar de un único umbral global teniendo en cuenta la media y la desviación estándar de sus píxeles vecinos (binarización adaptativa). La mejora que se obtiene con esta binarización respecto a la imagen original y a la binarización clásica (un umbral global) se puede observar en la Figura 8, la Figura 9 y la Figura 10.

Figura 8. Imagen de entrada [21]

Figura 9. Procesamiento de la imagen en la Figura 8 con binarización clásica

Figura 10. Procesamiento de la imagen en la Figura 8 con binarización de Sauvola

- **Filtrado de puntos pequeños.** Esta opción es configurable a partir de un argumento que recibe el algoritmo y que indica si se realiza o no, dado que es un proceso ligeramente más costoso que el resto. Consiste en obtener las componentes conexas de la imagen y filtrarlas por tamaño para eliminar así puntos de ruido. De esta forma puede eliminarse gran parte del ruido de la imagen, aunque los puntos grandes o manchas no se eliminarán, dado que no hay forma de distinguirlos de las letras por tamaño. El resultado obtenido por esta técnica para la imagen de la Figura 11 puede observarse en la Figura 12.

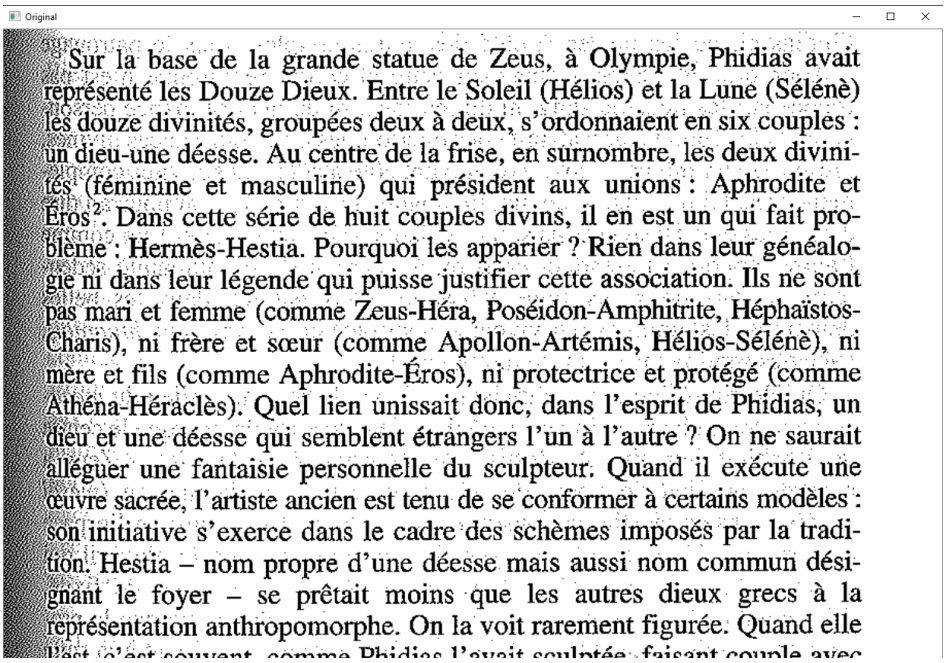


Figura 11. Imagen de entrada de un libro escaneado [22]

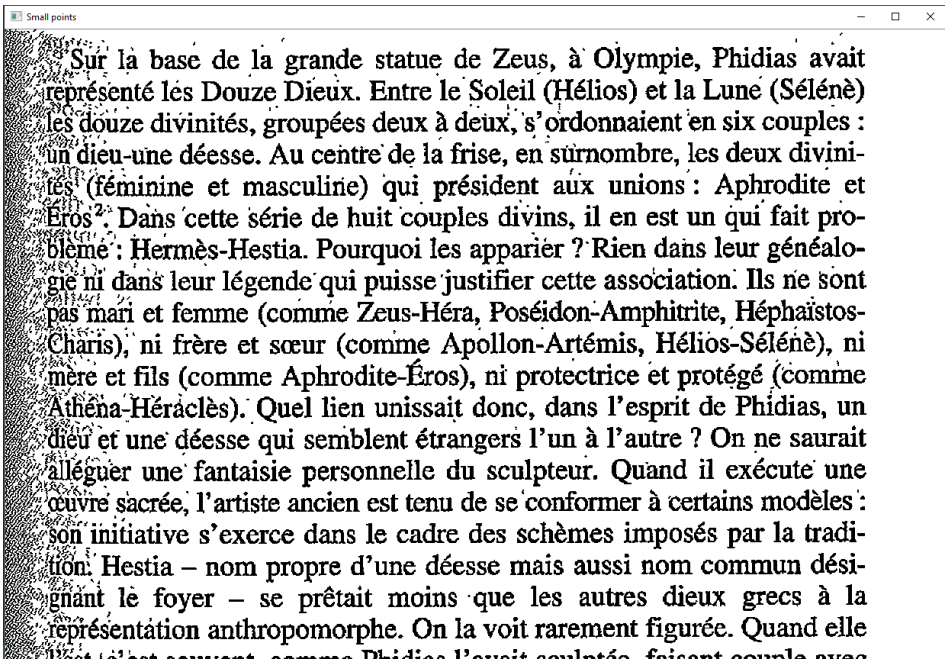


Figura 12. Transformación de la imagen de la Figura 11 tras aplicar el filtrado de puntos pequeños

Todos estos métodos consiguen una imagen considerablemente mejorada que facilita y optimiza los procesos de detección que se llevan a cabo posteriormente.

Detección de tablas

Sobre la imagen preprocesada se aplica un algoritmo de detección de tablas. Este algoritmo consta de varios pasos que se describen a continuación.

- **Detección de líneas.** Se utiliza una función de la librería OpenCV (Hough Lines) sobre la imagen invertida que detecta líneas en una imagen a través de principios matemáticos como es la transformada de Hough. Estos principios se basan en detectar intersecciones entre las sinusoides que representan las familias de líneas que pasan por un punto determinado del plano. La intersección entre dos sinusoides implica que los dos puntos a los que representa cada una de ellas pertenecen a la misma línea. Las líneas obtenidas de la imagen de la Figura 13 con este método se pueden observar en color rojo en la Figura 14.

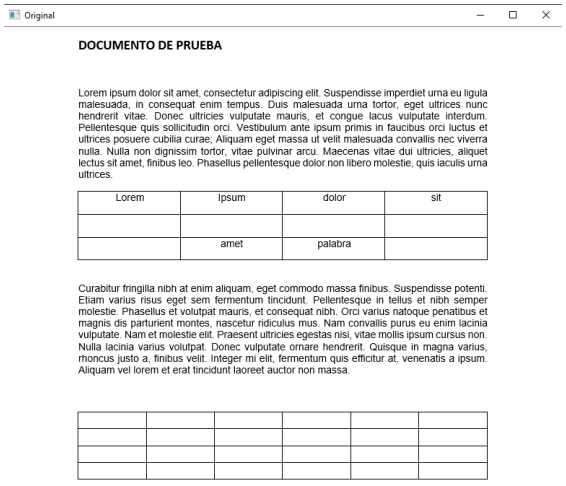


Figura 13. Imagen de entrada

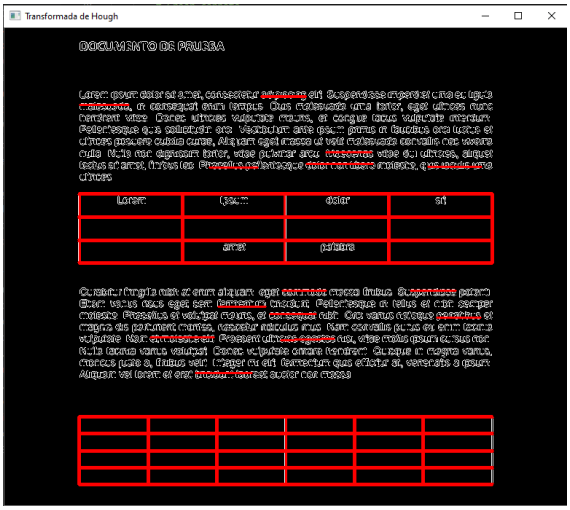


Figura 14. Líneas detectadas en la imagen de la Figura 13 con la Transformada de Hough

Además, se realiza una segunda detección de líneas horizontales y verticales aplicando erosión y dilatación a la imagen invertida con elementos estructurales en forma de rectángulos horizontales o verticales. La dilatación expande las zonas claras, mientras que la erosión las disminuye, lo que permite obtener las líneas. En la Figura 15 se puede observar la suma de las líneas horizontales y verticales obtenidas a partir de la imagen de la Figura 13.

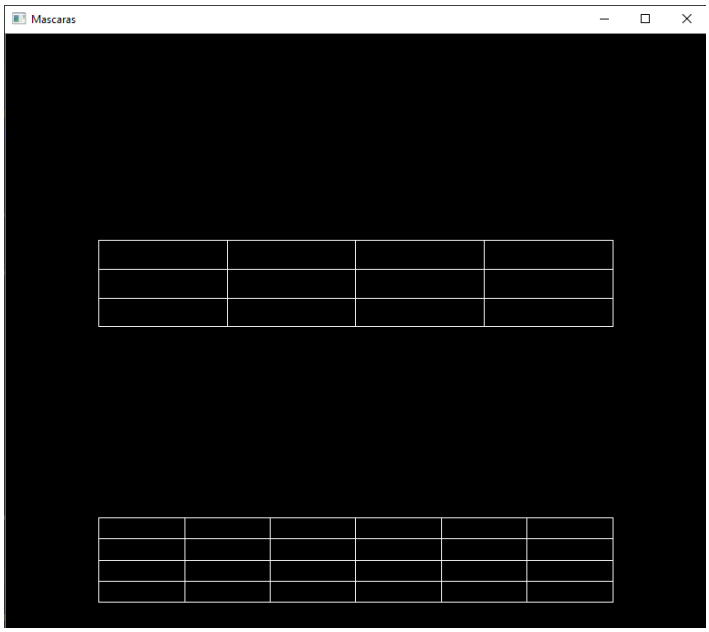


Figura 15. Líneas obtenidas en la imagen de la Figura 13 con máscaras horizontal y vertical

- **Intersección de líneas.** A partir de las líneas obtenidas a través de los dos procesos anteriores, se detectan los puntos de intersección entre líneas horizontales y verticales. Dependiendo de qué parte de los segmentos formen la intersección, se clasifican como puntos que definen las esquinas de la tabla o puntos de intersección internos. En la Figura 16 se muestran los puntos de intersección obtenidos representados por círculos centrados en ellos. Los círculos verdes representan los puntos de la esquina izquierda superior de una tabla, los rojos los puntos de la esquina derecha superior y los azules claros los puntos de la esquina izquierda inferior de una tabla. Con estos tres tipos de puntos se tiene información suficiente para delimitar una tabla.

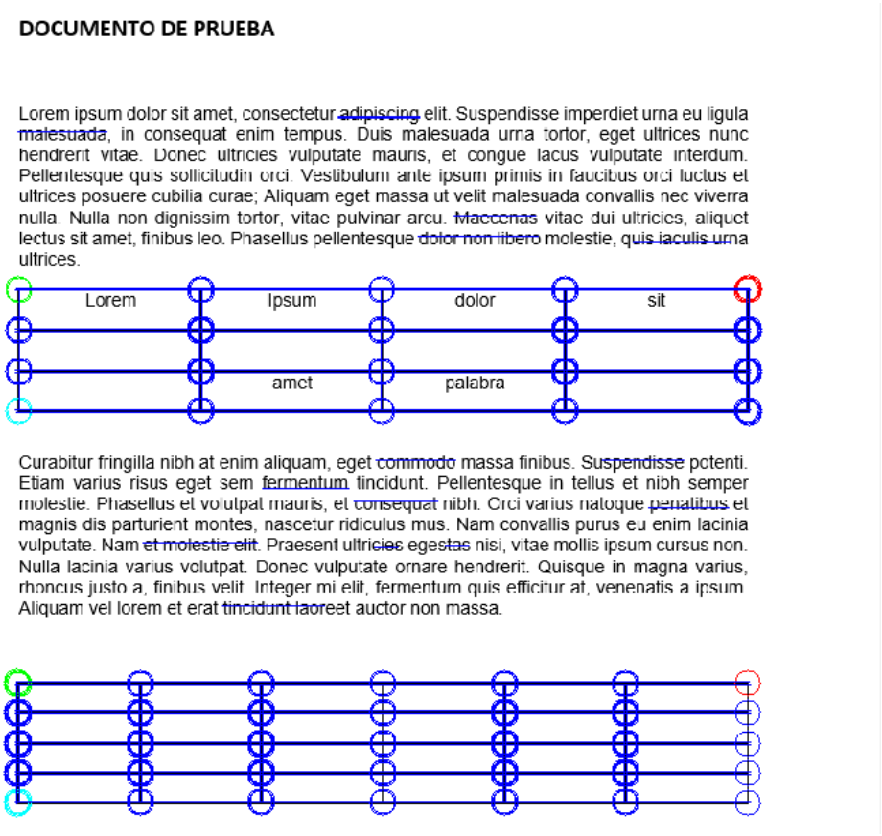


Figura 16. Puntos de intersección obtenidos en la imagen de la Figura 13

- **Filtrado de puntos.** Se estudian los puntos de intersección que pueden dar lugar a errores en la detección de tablas.
    - Se eliminan los puntos de intersección que están muy cerca y se deja uno como representante. La doble detección de líneas puede dar lugar a varios puntos de intersección muy cercanos que representan el mismo punto de la tabla, lo que posteriormente puede interpretarse como la existencia de varias tablas en el mismo punto en lugar de una.
    - Se eliminan puntos de intersección de esquinas de la tabla que parecen erróneos. Por ejemplo, puntos que han sido detectados como una esquina superior izquierda entre un punto del mismo tipo y otro de esquina superior derecha. Esta situación, de no ser corregida, podría interpretarse como la existencia de una segunda tabla dentro de la tabla real. En general, se busca obtener estructuras en las que para cada punto de inicio de tabla se encuentre un punto de fin de tabla (fin de filas y de columnas) con únicamente puntos de intersección interiores entre ambos.

También se estudian casos en los que puntos detectados como interiores pueden ser esquinas realmente (si una tabla no tiene esquina superior derecha pero sí tiene esquina superior izquierda y puntos de intersección en la misma línea, etc). Los puntos obtenidos finalmente se pueden observar en la Figura 17.

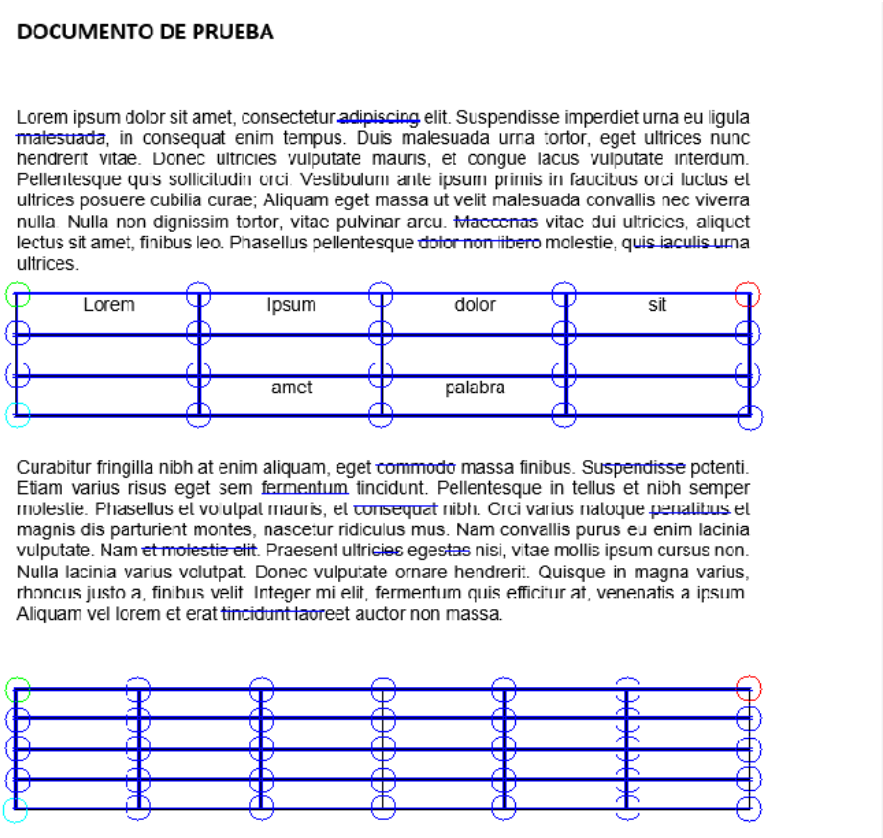


Figura 17. Puntos obtenidos tras filtrar puntos de intersección

- **Creación de tablas.** A partir de los puntos de intersección obtenidos, se crea la estructura de la tabla si es posible contando el número de intersecciones en la primera columna (entre los puntos que delimitan la tabla) para determinar el número de filas de la tabla y contando después el número de intersecciones en cada fila (contenidos entre los puntos límite de la tabla) para obtener el número de columnas.

  La tabla se representa por una lista L en la que cada elemento L[i] indica el número de columnas que tiene la fila i de la tabla. El algoritmo devuelve una lista de todas las tablas encontradas en la imagen. Para el ejemplo estudiado, la salida del algoritmo es la siguiente:

  [[4, 4, 4], [6, 6, 6, 6]].

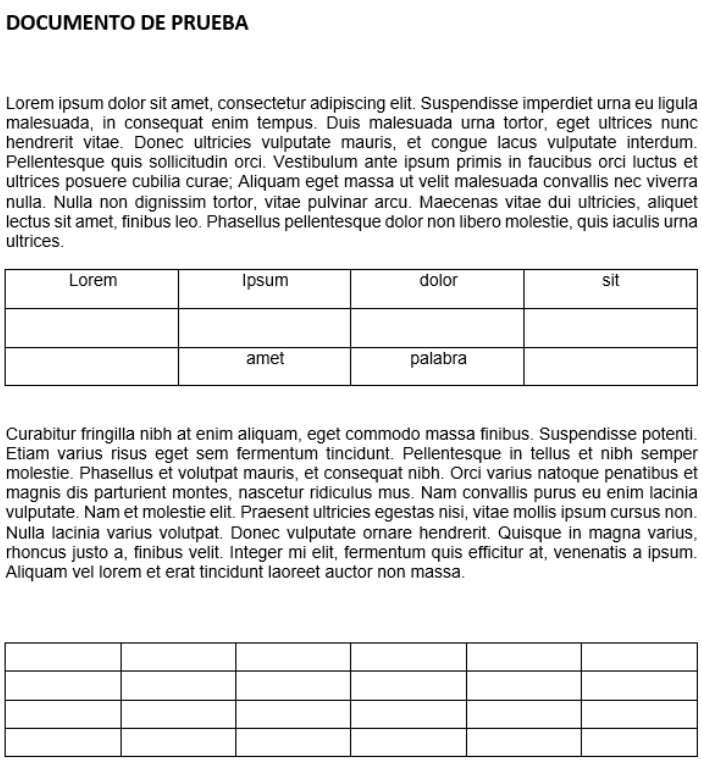
Reconocimiento de texto

El reconocimiento de texto de la imagen se realiza con la librería de código abierto de OCR Pytesseract. Dicha librería obtiene todas las palabras que es capaz de detectar en la imagen con la ayuda de un diccionario del idioma que se le indique.

Esta librería detecta el texto a través del reconocimiento de patrones de caracteres, aunque recientemente ha incorporado redes neuronales para reconocer líneas. Una de las principales razones por las que realizamos el preprocesamiento antes explicado es para mejorar la calidad de la imagen que se les pasa a los métodos de esta librería. De esta forma se facilita su tarea y es más probable obtener un reconocimiento preciso.

Finalmente, transformamos cada imagen o conjunto de imágenes de entrada en un fichero de texto con una estructura establecida. Este fichero, además del texto reconocido del documento, contiene características del documento para facilitar su recuperación posterior. Estas características son: número de palabras, número de páginas, densidad de palabras (cociente entre palabras y número de páginas) y densidad de tablas (cociente entre número de tablas y número de páginas).

En la Figura 19 se muestra la transformación obtenida a partir de la imagen del documento de la Figura 18.



DOCUMENTO DE PRUEBA

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse imperdiet urna eu ligula malesuada, in consequat enim tempus. Duis malesuada urna tortor, eget ultrices nunc hendrerit vitae. Donec ultricies vulputate mauris, et congue lacus vulputate interdum. Pellentesque quis sollicitudin orci. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Aliquam eget massa ut velit malesuada convallis nec viverra nulla. Nulla non dignissim tortor, vitae pulvinar arcu. Maecenas vitae dui ultricies, aliquet lectus sit amet, finibus leo. Phasellus pellentesque dolor non libero molestie, quis iaculis urna ultrices.

Lorem	Ipsum	dolor	sit
	amet	palabra	

Curabitur fringilla nibh at enim aliquam, eget commodo massa finibus. Suspendisse potenti. Etiam varius risus eget sem fermentum tincidunt. Pellentesque in tellus et nibh semper molestie. Phasellus et volutpat mauris, et consequat nibh. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam convallis purus eu enim lacinia vulputate. Nam et molestie elit. Praesent ultricies egestas nisi, vitae mollis ipsum cursus non. Nulla lacinia varius volutpat. Donec vulputate ornare hendrerit. Quisque in magna varius, rhoncus justo a, finibus velit. Integer mi elit, fermentum quis efficitur at, venenatis a ipsum. Aliquam vel lorem et erat tincidunt laoreet auctor non massa.

Figura 18. Imagen de entrada

```
num_palabras:201
num_paginas:1
densidad:201.0
densidad_tablas:2.0
texto: DOCUMENTO DE PRUEBA

Lorem ipsum dolor sit amet, consectetur adipiscing elit Suspenkisse imperdiet uma eu ligula
malesuada, in consequat enim tempus Duts malesuada uma tortor, eget ultrices nunc
hendrent vitae Donec ulincies vulputate mauns, et congue lacus vulputate interdum
Pellentesque quis sollictudin ora Vestibulum ante ipsum primis in faucibus orci luctus et
ulinces posuere cubilia curae, Alkjuam eget massa ut velit malesuada convallis nec viverra
nulla Nulla non dignissim tortor, vitae pulvinar arcu Maecenas vitae dul uttncies, aliquet
lectus sit amet, finibus leo Phasellus pellentesque dolor non libero molestie, quis laculis urna
ulinces


Lorem Ipsum ‘dolar ait



amet palabra




‘Curabitur fringilla nibh at enim aliquam eget commode massa finibus Suspendisse potent
Etiam varus risus eget sem fermentum tincidunt Pellentesque in tellus et mbh semper
Malestie Phasellus et volutpat mauns, et consequat nibh Orci vanus natoque penatibus et
magnis dis parturent montes, nascetur ndiculus mus Nam convallis purus eu enim lacitia
vuiputate Nam et molestie elt Praesent ultncies egestas nisi, vitae mollis ipsum cursus non
Nulla lacinia varus volutpat Donec vulputate ornare hendrent Quisque in magna vanus,
thoncus justo a, finibus velit Integer mi elit fermentum quis efficitur at, venenatis a ipsum.
Aliquam vel lorem et erat tincidunt faoreet auctor non massa
```

Figura 19. Fichero de texto resultado de la transformación de la imagen de la Figura 18

Modelo de clasificación

La segunda parte es la correspondiente a la clasificación de los documentos de entrada. Para ello, utilizamos una red neuronal para clasificación.

Características

Los documentos que el modelo reciba como entrada deben estar representados en formato vector, de forma que cada documento sea un vector con una fila y tantas columnas como atributos tenga el modelo.

Como ya hemos comentado antes, para representar los documentos utilizamos la técnica de Bag of Words. De esta forma, cada documento tendrá tantas características como palabras aprenda el modelo de Bag of Words, y los valores de estas características para cada documento se obtendrán con la técnica TF-IDF.

Además de estas características, añadimos tres más, dado que uno de los principales objetivos de este proyecto era ampliar la información que el modelo recibe de cada documento con datos relevantes y decisivos para la clasificación. Los nuevos atributos son:

- **Número de palabras.** La cantidad de palabras que tiene un documento puede ser decisivo a la hora de clasificarlo. Los documentos del mismo tipo probablemente tengan longitudes similares.

- **Densidad de palabras.** Este atributo hace uso de la información de número de páginas, razón por la que no incorporamos este dato como característica del modelo. Creemos que es relevante la cantidad de palabras por página del documento, puesto que los documentos pueden ser muy extensos y tener pocas palabras. Esta característica puede ser compartida por documentos de la misma categoría.

- **Densidad de tablas.** Relación entre número total de tablas detectadas en el documento y número de páginas del mismo. Guardamos la densidad en lugar del número de tablas porque, al igual que en las frecuencias del modelo de Bag of Words, documentos largos tendrán mayor conteo de tablas. La cantidad de tablas presentes en un documento puede aportar gran capacidad de diferenciación a la clasificación.

Arquitectura

El modelo de clasificación está formado por una red neuronal multicapa. Como se especifica en los requisitos del proyecto, hay 5 clases diferentes disponibles para clasificar los documentos. Por tanto, esta red neuronal realizará una clasificación multiclase, por lo que la salida será un vector de dimensión 5.

La red neuronal que hemos construido consta de dos capas ocultas completamente interconectadas (capas 2 y 3). Para determinar el número de neuronas de cada capa, así como el número de ejemplos que el modelo procesa antes de actualizar los valores de

los pesos de la red (batch size), las iteraciones que se realizan sobre el conjunto de entrenamiento (epochs) y la función de activación a utilizar, realizamos un proceso de búsqueda de parámetros óptimos mediante validación cruzada de 10 particiones con la función GridSearchCV de Scikit-learn.

Para cada parámetro, probamos el siguiente conjunto de valores:

- Neuronas capa 2: {5, 8, 12}
- Neuronas capa 3: {5, 8, 12}
- Batch size: {8, 10, 50, 80 100}
- Iteraciones: {10, 50, 100}
- Función en cada capa: {ReLU, softmax}

La configuración que mejor ratio de clasificación ofrece (98.67%) es la siguiente:

- 8 neuronas en la capa 2
- 8 neuronas en la capa 3
- batch size de 100 ejemplos
- 50 iteraciones
- Función ReLU en capas ocultas
- Función softmax en capa de salida

La función ReLU es una de las funciones más utilizadas actualmente en Deep Learning, dado que permite un aprendizaje más rápido y una mayor capacidad de generalización si se utiliza en las capas ocultas, como hemos podido comprobar en el proceso de validación cruzada. En la capa de salida, sin embargo, utilizamos la función softmax, extensión de la función sigmoide clásica para problemas multiclase, para obtener probabilidades entre 0 y 1.

Fijamos el número de neuronas de las capas ocultas al número óptimo obtenido por el proceso de validación cruzada.

Como función de coste utilizamos la entropía cruzada categórica, que está indicada para problemas de clasificación multiclase.

Por último, aplicamos también la conocida como regularización L1 para evitar el riesgo de sobreajuste de la red a los ejemplos de entrenamiento.

Entrenamiento

Para poder predecir la clase de documentos, la red neuronal debe ser previamente entrenada. Realizamos este entrenamiento a partir de un conjunto de documentos de entrenamiento.

De los documentos etiquetados de los que disponemos, más de 6000, realizamos una partición en dos conjuntos: entrenamiento y test.

Nuestros documentos están en formato PDF, por lo que tenemos que aplicar el algoritmo de transformación a archivo de texto a cada uno de ellos. Una vez obtenemos el fichero de cada documento, obtenemos de él las características y lo añadimos a la matriz de entrenamiento, formada por tantas filas como documentos de entrenamiento y tantas columnas como características tiene el modelo.

Para obtener estas características realizamos el proceso de Bag of Words con modelos de vectorización y de TF-IDF, que se entrenan con los documentos de entrada. Además, realizamos selección de variables sobre las características obtenidas con estos modelos con un modelo de SVD truncado para evitar el sobreentrenamiento, dado que el número de características es mucho mayor que el número de ejemplos del conjunto de entrenamiento.

Estos modelos se guardan persistentemente para obtener las características de los documentos de test a partir de ellos, ya que estos últimos tendrán que tener las mismas características.

Los datos de la matriz de entrada se normalizan antes de pasárselos al modelo de clasificación. El modelo de normalización entrenado también se guarda para normalizar los datos de test.

Por otro lado, transformamos la salida de los documentos con la codificación One-Hot, necesaria para redes neuronales multiclase, de modo que la clase i se representa con un vector unidimensional V de tamaño 5 (número de clases) en el que V[i] tiene valor 1 y el resto de posiciones del vector tienen valor 0. Las salidas de los documentos de entrenamiento se representan, por tanto, con una matriz con tantas filas como documentos y 5 columnas.

Antes de realizar el entrenamiento, realizamos la validación cruzada con el conjunto de entrenamiento como hemos explicado antes. Posteriormente realizamos el entrenamiento con dicho conjunto completo y los parámetros obtenidos.

Realizamos el entrenamiento fijando a 100 el número de ejemplos que el modelo procesa antes de actualizar los valores de los pesos de la red. Por otra parte, se realizan 50 iteraciones sobre el conjunto de entrenamiento.

Finalmente, se predice el conjunto de test con el modelo aprendido y se obtiene el error con respecto a la salida esperada.

En la Tabla 2 se muestran los ratios de clasificación obtenidos para cada conjunto.

	Exactitud (accuracy)
Conjunto de entrenamiento	98.88 %
Conjunto de test	97.67 %

Tabla 2. Ratio de clasificación del modelo en train y test

Como ya hemos comentado, esta métrica no es del todo fiable para medir la calidad de un modelo. Por tanto, utilizamos también una matriz de confusión que se muestra en la Tabla 3 para medir la precisión, el recall y F1-score del modelo. Los resultados de estas métricas para cada clase se muestran en la Tabla 4.

		CLASE PREDICHA				
		Acuse de recibo	**Contestación**	**Notificación**	**Propuesta de acuerdos**	**Resolución**
CLASE REAL	**Acuse de recibo**	90	0	0	0	0
	Contestación	1	83	0	1	0
	Notificación	1	0	86	1	0
	Propuesta de acuerdos	1	2	0	67	0
	Resolución	0	0	1	2	93

Tabla 3. Matriz de confusión para 429 ejemplos de test

	Precisión	Recall	F_1-score
Acuse de recibo	0.968	1	0.984
Contestación	0.976	0.976	0.976
Notificación	0.988	0.977	0.983
Propuesta de acuerdos	0.944	0.957	0.95
Resolución	1	0.969	0.984

Tabla 4. Métricas de precisión, recall y F_1-score para cada clase en ejemplos de test

Como podemos observar, el modelo obtenido tiene buena capacidad de generalización en base a los datos obtenidos para el balance entre precisión y recall.

Guardamos el modelo entrenado persistentemente para poder acceder a él desde cualquier instancia del programa y predecir la categoría a la que pertenecen nuevos documentos.

Predicción

Para clasificar un nuevo documento, realizamos el mismo proceso. El algoritmo lo transforma a un archivo de texto que convertimos a vector con los modelos de vectorización, TF-IDF y selección de características entrenados anteriormente. Además, normalizamos los datos con el modelo de normalización.

Pasamos el vector que representa al documento como entrada al modelo de clasificación, que produce como salida la clase a la que pertenece el documento.

Integración en gestor documental

Uno de los requisitos de este trabajo es la integración del sistema desarrollado en un gestor documental. El gestor en el que vamos a integrar el algoritmo es la aplicación de registro telemático de documentos de la empresa. En la Figura 20 se muestra la pantalla

principal de la aplicación web. Esta aplicación está desarrollada en lenguaje Java junto con la herramienta Maven y se despliega con el contenedor de servlets Tomcat.

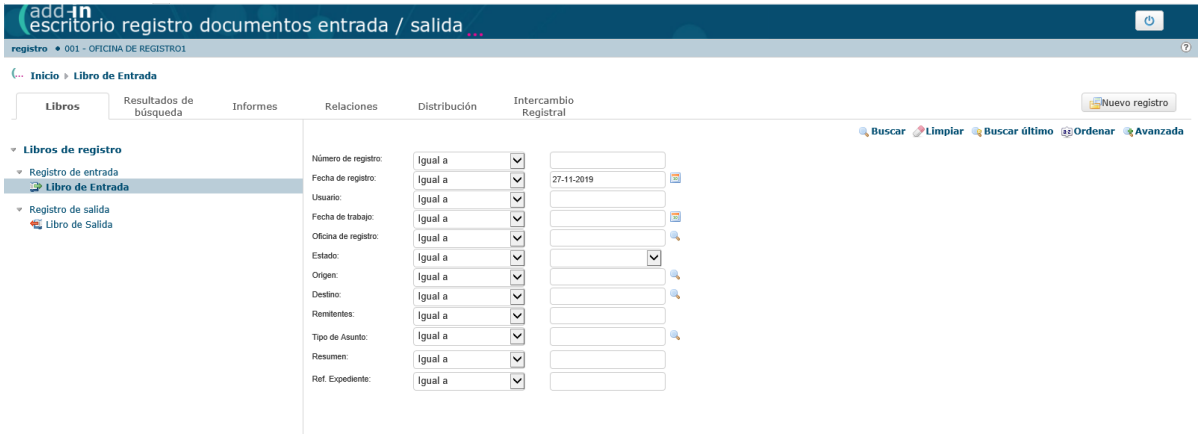


Figura 20. Aplicación web de registro de IECISA

La aplicación web permite crear y administrar registros a los que se pueden añadir documentos digitalizados. Consta de libros de registros de entrada y de registros de salida. La digitalización de documentos se realiza en los registros de entrada.

Queremos incorporar el sistema desarrollado en este trabajo para automatizar el proceso de clasificación documental, que normalmente realizan manualmente los trabajadores de los puestos de digitalización.

El objetivo es que cada documento nuevo que se digitalice sea procesado por el algoritmo de clasificación. Además, necesitamos guardar la información relativa a la clasificación. Para ello añadimos una nueva tabla a la Base de Datos de la empresa que guarda, por cada documento, los siguientes campos: identificador único, estado en el que se encuentra el proceso de clasificación, fechas de inicio y fin de este proceso, clase predicha por el modelo y tipo de error en caso de que se haya producido un error durante el proceso.

Para incorporar esta información a la interfaz de la aplicación, añadimos una pestaña llamada “Clasificación” que solo es seleccionable cuando se ha seleccionado un libro de entrada. En la Figura 21 se puede observar la vista de la aplicación al iniciar sesión y sin haber seleccionado ningún libro.

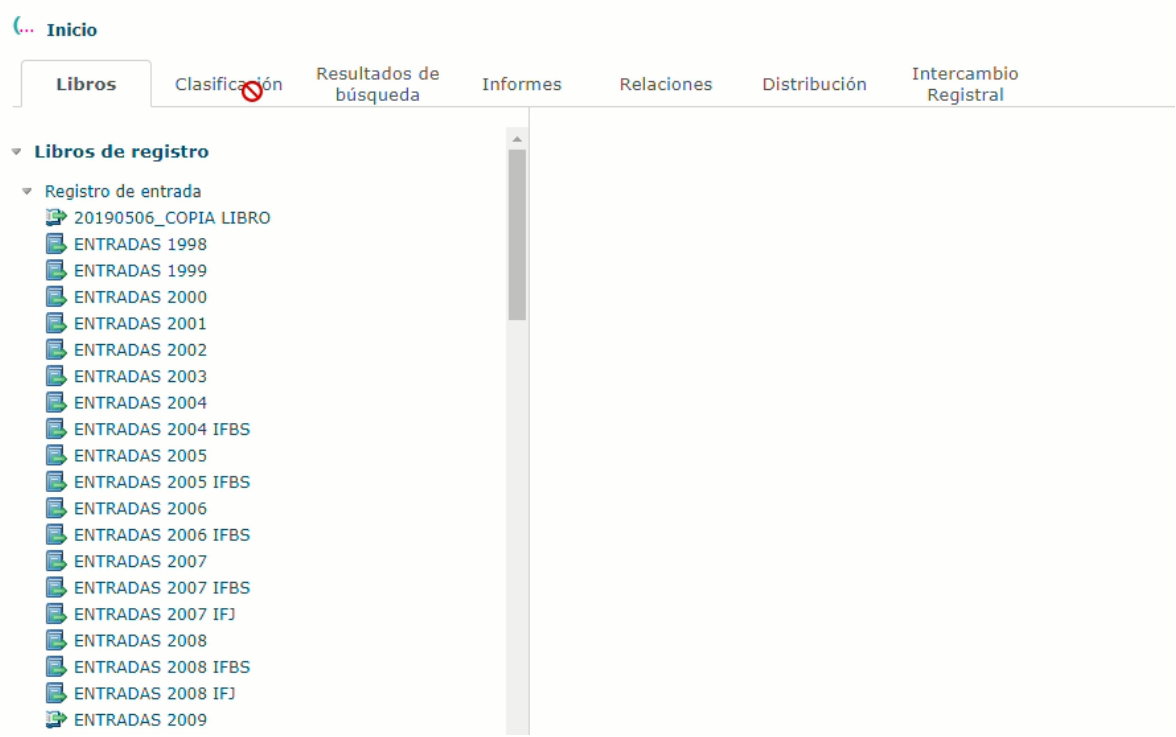


Figura 21. Vista pantalla inicial aplicación web sin seleccionar libro

Al pulsar sobre dicha pestaña, se muestra una tabla con la información del proceso de clasificación de los documentos del libro seleccionado. Esta tabla se puede observar en la Figura 22.

Figura 22. Vista de la tabla que contiene la información del proceso de clasificación

Los documentos tienen cuatro estados de clasificación posible: “pendiente”, “procesando”, “clasificado” y “error”.

Cuando en un libro se crea un nuevo registro, se pueden anexar uno o varios documentos PDF mediante escaneo con la herramienta mostrada en la Figura 23, lo que se conoce como digitalización.

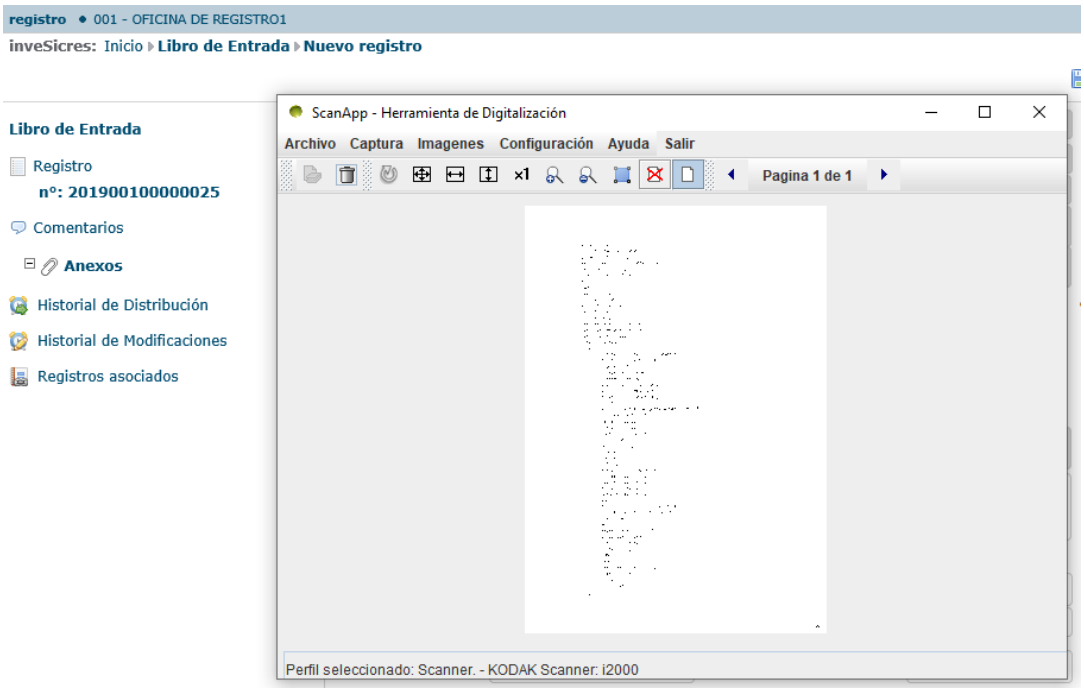


Figura 23. Herramienta de digitalización para anexar documentos a un registro

Una vez se guarda el registro con los documentos digitalizados, dichos documentos se insertan en la tabla del proceso de clasificación con estado “pendiente”, como se puede observar en la Figura 24.

Figura 24. Vista de la tabla con un documento añadido en espera de ser procesado

Los documentos en estado “pendiente” son recuperados por un planificador de tareas (scheduler) que se ejecuta cada un determinado número de minutos fijado (en nuestro caso fijado a 3 minutos).

Una vez recuperados, para cada documento llamamos desde la clase Java que los recupera a una instancia del programa Python que realiza el proceso de transformación de PDF y clasificación del documento. Los documentos que se comienzan a procesar por el algoritmo pasan a estado “procesando”, como se puede observar en la Figura 25. Cada instancia se ejecuta de forma asíncrona, de forma que se puedan realizar de forma simultánea y sin bloquear el resto de la aplicación.

Figura 25. Vista de la tabla con documento en estado “procesando”

Por último, se actualiza la información de los documentos en la tabla estableciendo la salida del algoritmo como “Clase” del documento y las fechas de inicio y fin del

procesamiento en caso de que el proceso de clasificación haya finalizado correctamente. Además, se actualiza el estado a “clasificado”.

En caso de que se produzca un error en la clasificación del documento, el estado del documento se actualiza a “error” y se almacena en el campo “Errores” el tipo de error que se ha producido. En la Figura 26 se muestran un documento nuevo clasificado y otro cuyo proceso de clasificación ha sufrido un error.

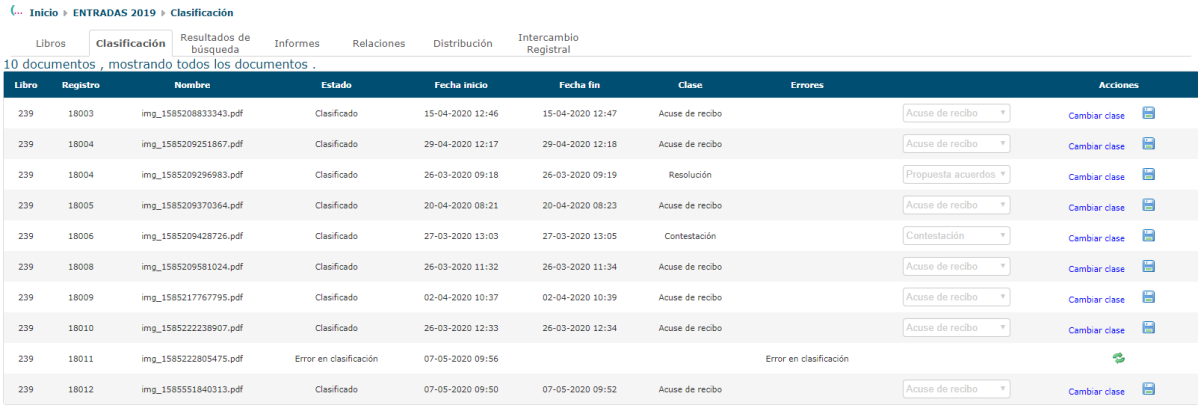


Figura 26. Vista de la tabla con un nuevo documento clasificado y otro cuya clasificación ha producido un error

La clasificación de los documentos que han sufrido un error durante el proceso se puede reintentar pulsando el botón indicado. El documento vuelve a pasar a estado “pendiente” y se vuelve a intentar su clasificación, como se puede observar en la Figura 27.

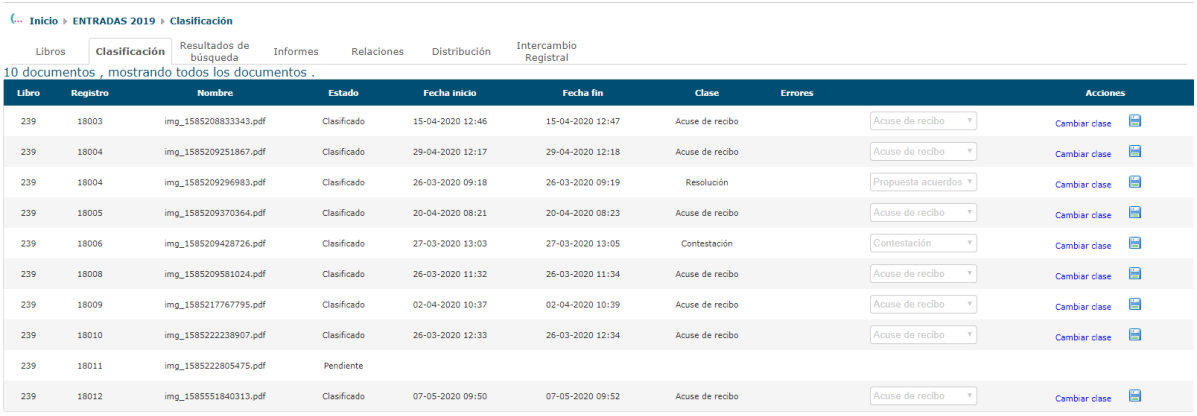


Figura 27. Vista de la tabla en la que se ha reintentado la clasificación de un documento

Por otro lado, la clase predicha por el modelo puede ser corregida manualmente a través de una lista desplegable que se activa al pulsar “Cambiar clase”, como se muestra en la Figura 28.

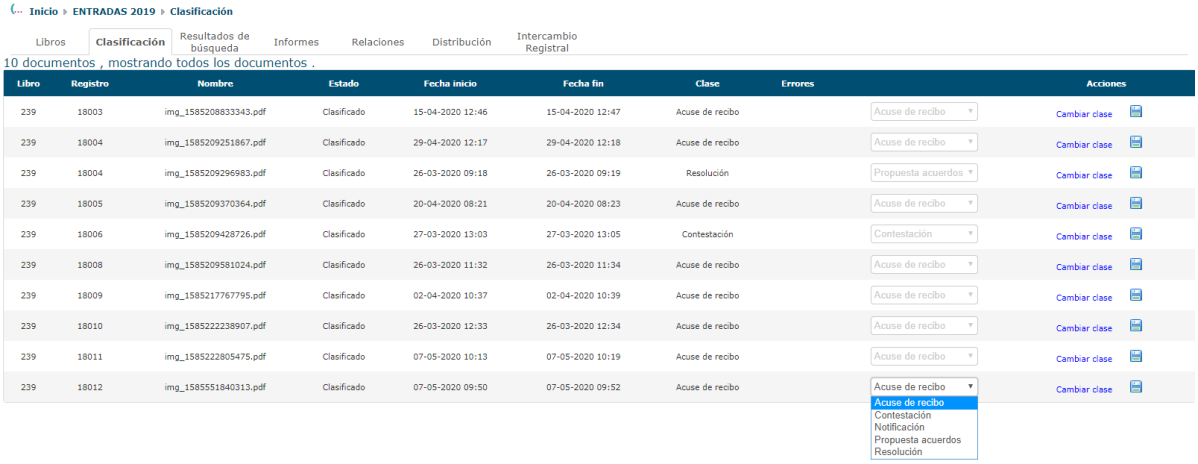


Figura 28. Vista del desplegable para elegir la clase del documento tras pulsar “Cambiar clase”

Tras seleccionar la nueva clase y pulsar el icono de “Guardar”, la clase se actualiza en Base de Datos, como se observa en la Figura 29, y el documento se pasa al modelo junto con su clase para que este se vuelva a entrenar con un ejemplo más de entrenamiento.

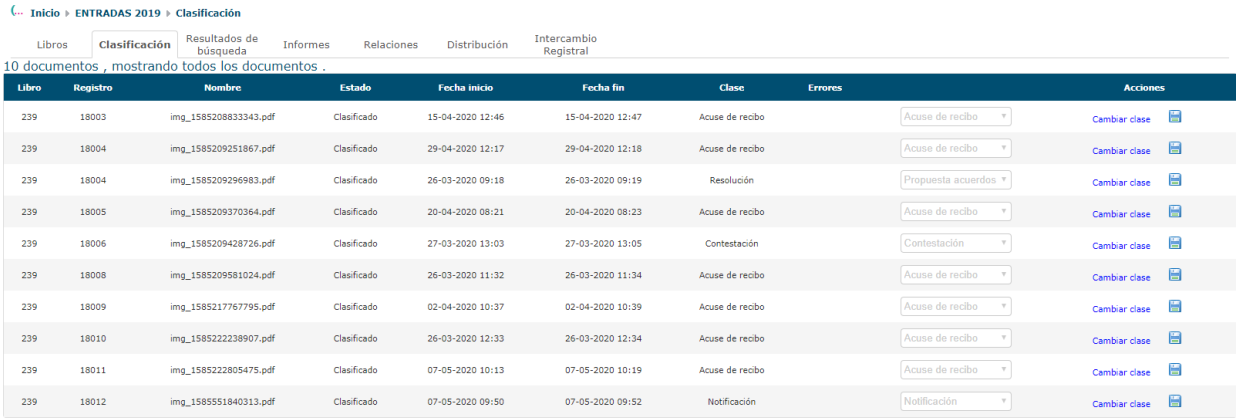


Figura 29. Vista de la tabla tras corregir la clase del último documento

Esta funcionalidad permite que el modelo sea ampliable con nuevos ejemplos de entrenamiento. Esto posibilita la mejora del modelo.

CONCLUSIONES

El trabajo realizado permite la clasificación de documentos en formato imagen o PDF en una de las cinco clases proporcionadas con un grado de confianza elevado. El entrenamiento de la red neuronal con los parámetros óptimos y un gran número de documentos previamente procesados permite la obtención de un modelo con una gran capacidad de generalización. Los documentos clasificados incorrectamente son, en su mayoría, documentos que no poseen texto, como contestaciones que son simplemente gráficos o mapas, o presentan una gran cantidad de anotaciones manuscritas que se superponen al texto y dificultan su legibilidad. La clasificación de este tipo de documentos es complicada incluso manualmente.

El algoritmo desarrollado se ha integrado en una aplicación de gestión concreta en la que hemos encontrado un campo de aplicación del mismo, pero puede extenderse a muchos otros casos. El modelo es ampliamente adaptable a diferentes escenarios, dado que puede modificarse fácilmente para incorporar características y clases distintas.

Este es un ejemplo de aplicación conjunta de varias áreas de la Inteligencia Artificial, como son la Visión Artificial y el Aprendizaje Automático, a un caso real y usual. Demuestra el potencial de este tipo de sistemas para facilitar la realización de tareas y el tratamiento de grandes cantidades de datos mediante mecanismos que tratan de simular la inteligencia humana. Además, se hace evidente el amplio abanico de posibilidades que ofrecen.

Finalmente, se puede concluir la necesidad de utilizar varias técnicas del área de la Inteligencia Artificial en conjunto para garantizar la buena actuación de cada una de ellas. Los ejemplos que encontramos en aplicaciones reales, como los documentos escaneados del caso estudiado, difieren considerablemente del caso teórico óptimo, y por tanto es necesario aplicar otras técnicas de procesamiento previamente para que algoritmos como el de clasificación automática de textos puedan ser aplicados sin ver atenuada su capacidad.

LÍNEAS FUTURAS

Como ya hemos concluido, el sistema es altamente ampliable y por tanto se puede modificar para incluir nuevas funcionalidades o tratar otros problemas.

Entre las líneas futuras del proyecto que podrían tratarse y que no se han implementado por la necesidad de desarrollo en un tiempo limitado, se encuentran:

- Mejoras en la integración realizada en este trabajo. Se pueden incorporar nuevas funcionalidades a la integración, entre otras:

    - Posibilidad de visualizar el documento desde la tabla de estados con el fin de facilitar la corrección de clasificación de forma manual.
    - Modificación de configuración para que el modelo no vuelva a entrenarse cada vez que se corrige la clase de un documento, sino que un planificador de tareas recoja cada cierto tiempo los documentos corregidos nuevos y reentrene el modelo.

- Adaptación de clases disponibles para el modelo para distintas aplicaciones
- Adaptación de características para diferentes problemas:

    - Por ejemplo, para un caso de uso de trámites con diferentes fases, puede añadirse como característica del modelo la fase en que se ha anexado el documento. El tipo de documento puede estar altamente relacionado con la fase en que se anexa, por lo que esta información puede ser decisiva para la clasificación.
    - En este mismo caso, podría ampliarse el objetivo del algoritmo para predecir, además del tipo de documento, el trámite que se va a llevar a cabo a continuación. Esto requeriría añadir otro modelo con características de las fases o trámites superados hasta el momento. Las fases anteriores, junto con el tipo de documento que se anexa (predicho por el modelo de este trabajo), pueden ayudar a predecir el trámite siguiente (aplicable en asistentes virtuales para la realización de trámites).

BIBLIOGRAFÍA

[1] S. J. Russell and P. Norvig, *Inteligencia artificial : un enfoque moderno / Stuart J. Russell y Peter Norvig ; traducción [de la 2ª ed. en inglés], Juan Manuel Corchado Rodríguez... [et al.] ; revisión técnica, Juan Manuel Corchado Rodríguez... [et al.] ; coordinación general de la traducción y revisión técnica, Luis Joyanes Aguilar.* 2004.

[2] R. C. Gonzalez and R. E. Woods, *Digital image processing / Rafael C. Gonzalez, Richard E. Woods.* 2018.

[3] J. Sauvola and M. Pietikak, "Adaptive document image binarization," 2000. Accessed: May 04, 2020. [Online].

[4] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972, doi: 10.1145/361237.361242.

[5] N. Sahu and M. Sonkusare, "A Study on Optical Character Recognition Techniques," *International Journal of Computational Science, Information Technology and Control Engineering*, vol. 4, no. 1, pp. 01–15, Jan. 2017, doi: 10.5121/ijcsitce.2017.4101.

[6] R. Unnikrishnan and R. Smith, *Combined Script and Page Orientation Estimation using the Tesseract OCR engine*. 2009.

[7] T. M. Mitchell, *Machine Learning Machine Learning*. 1997.

[8] M. Jordan, J. Kleinberg, and B. Schölkopf, "Pattern Recognition and Machine Learning." Accessed: May 06, 2020. [Online].

[9] Y. Goldberg, "Neural Network Methods for Natural Language Processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–311, 2017, doi: 10.2200/S00762ED1V01Y201703HLT037.

[10] "8 Matrix decompositions and latent semantic indexing." Accessed: May 05, 2020. [Online].

[11] T. Hastie, R. Tibshirani, and J. Friedman, "Springer Series in Statistics The Elements of Statistical Learning Data Mining, Inference, and Prediction." Accessed: May 02, 2020. [Online].

[12] A. Tharwat, “Classification assessment methods,” *Applied Computing and Informatics*, Aug. 2018, doi: 10.1016/j.aci.2018.08.003.

[13] “pytesseract · PyPI.” https://pypi.org/project/pytesseract/ (accessed May 04, 2020).

[14] “Home - Keras Documentation.” https://keras.io/ (accessed May 04, 2020).

[15] “scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation.” https://scikit-learn.org/stable/ (accessed May 04, 2020).

[16] “stop-words · PyPI.” https://pypi.org/project/stop-words/ (accessed May 04, 2020).

[17] “OpenCV.” https://opencv.org/ (accessed May 04, 2020).

[18] S. van der Walt *et al.*, “Scikit-image: Image processing in python,” *PeerJ*, vol. 2014, no. 1, 2014, doi: 10.7717/peerj.453.

[19] “Maven – Welcome to Apache Maven.” https://maven.apache.org/ (accessed May 04, 2020).

[20] “Apache Tomcat® - Welcome!” http://tomcat.apache.org/ (accessed May 04, 2020).

[21] “Old Newspaper | The front page of an issue of the "Eugene Re… | Flickr.” https://www.flickr.com/photos/orodreth_99/1271809522 (accessed May 01, 2020).

[22] J.-P. Vernant, “Hestia-Hermès. Sur l’expression religieuse de l’espace et du mouvement chez les Grecs,” *L’Homme*, vol. 3, no. 3, pp. 12–50, 1963, doi: 10.3406/hom.1963.366578.