Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo de un Sistema de Clasificación Documental mediante OCR y Aprendizaje Profundo

Autor: Daniel García Rodríguez

Tutor(a): Antonio Jesús Diaz Honrubia

Madrid, enero 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Desarrollo de un Sistema de Clasificación Documental mediante OCR y Aprendizaje Profundo
Enero 2024

Autor: Daniel García Rodríguez

Tutor:
Antonio Jesús Diaz Honrubia Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

En el contexto actual de creciente volumen de información, la necesidad de clasificar documentos de manera eficiente y precisa se ha vuelto imperativa. Este Trabajo de Fin de Grado presenta un enfoque innovador basado en inteligencia artificial para abordar esta problemática, utilizando Python como lenguaje de desarrollo clave en este sector. La elección de Python no solo responde a su popularidad y versatilidad, sino también a su papel fundamental en la implementación de soluciones de inteligencia artificial.

El desarrollo del clasificador de documentos se ha llevado a cabo utilizando tecnologías avanzadas, destacando TensorFlow como la biblioteca principal para la construcción y entrenamiento del modelo de clasificación. La robustez y eficacia de TensorFlow proporcionan una base sólida para la creación de un sistema capaz de aprender y adaptarse a patrones complejos en los documentos.

Además, para la extracción de información crucial de los documentos, se ha integrado Optical Character Recognition (OCR). Esta tecnología ha demostrado ser una herramienta esencial para la digitalización de documentos, permitiendo la conversión de datos impresos en información digital accesible y procesable. La combinación de TensorFlow para la clasificación y OCR para la extracción de datos amplía significativamente la capacidad del sistema, ofreciendo una solución integral para la gestión eficiente de documentos en entornos diversos.

Los resultados obtenidos de este clasificador de documentos son prometedores, no precisamente por su precisión y eficiencia, sino principalmente por la escalabilidad del software para mejoras futuras. La capacidad del modelo para adaptarse y aprender de conjuntos de datos variados sugiere un potencial significativo para futuras versiones.

Abstract

In the current context of increasing information volume, the need to classify documents efficiently and accurately has become imperative. This Bachelor's Thesis presents an innovative approach based on artificial intelligence to address this issue, using Python as a key development language in this sector. The choice of Python not only responds to its popularity and versatility but also to its fundamental role in implementing artificial intelligence solutions.

The development of the document classifier has been carried out using advanced technologies, with TensorFlow standing out as the main library for the construction and training of the classification model. The robustness and effectiveness of TensorFlow provide a solid foundation for creating a system capable of learning and adapting to complex patterns in documents.

Furthermore, for the extraction of crucial information from documents, Optical Character Recognition (OCR) has been integrated. This technology has proven to be an essential tool for document digitization, allowing the conversion of printed data into accessible and processable digital information. The combination of TensorFlow for classification and OCR for data extraction significantly enhances the system's capability, providing a comprehensive solution for efficient document management in diverse environments.

The results obtained from this document classifier are promising, not primarily due to its precision and efficiency, but mainly because of the scalability of the software for future improvements. The model's ability to adapt and learn from various datasets suggests significant potential for future versions.

Contenido

1 Introducción 1
1.1 Motivación y necesidad del proyecto 1
1.2 Objetivos 1
1.3 Planificación 2
1.4 Estado del arte 2
1.5 Estructura de la memoria 3
2 Soluciones existentes y tecnologías empleadas 5
2.1 Clasificador de documentos registrados en el ENI 5
2.2 Tecnologías y herramientas utilizadas 6
3 Desarrollo del clasificador de documentos 9
3.1 Análisis de requisitos 9
3.1.1 Requisitos funcionales 9
3.1.2 Requisitos no funcionales 9
3.2 Diseño del clasificador propuesto 10
3.3 Descripción de tecnologías clave 12
3.4 Implementación del modelo 15
3.4.1 Módulo Principal 16
3.4.2 ClasificarPDF 17
3.4.3 Entrenamiento 20
3.4.4 TableOCR 22
4 Demostración y evaluación 25
4.1 Demostración 25
4.2 Evaluación y discusión del modelo 32
4.2.1 Evaluación del modelo propuesto 32
5 Resultados y conclusiones 34
5.1 Conclusiones 34
5.2 Logros personales 34
5.3 Futuras ampliaciones del clasificador 35
6 Análisis de Impacto 37
7 Bibliografía 39
8 Anexos 40

Figuras

Figura 1.1 Diagrama de Gantt 2
Figura 3.1 Diagrama de flujo de la aplicación a nivel global 10
Figura 3.2 Composición de los distintos módulos y clases del software 15
Figura 4.1 Primera página del documento de entrada al programa 25
Figura 4.2 Segunda página del documento de entrada al programa 26
Figura 4.3 Traza de la ejecución nº1 28
Figura 4.4 Traza de la ejecución nº2 28
Figura 4.5 Traza de la ejecución nº3 29
Figura 4.6 Primera página del documento cambiada de pdf a png 29
Figura 4.7 Segunda página del documento cambiada de pdf a png 30
Figura 5.1 Progreso de la integración de la aplicación 36
Figura 8.1 Informe de originalidad creado por la herramienta Turnitin 40

1 Introducción

En la era digital actual, la gestión eficiente de documentos se ha convertido en un desafío ineludible, dada la abrumadora cantidad de información que fluye constantemente en entornos virtuales. La clasificación constituye una parte esencial de los procedimientos técnicos implementados en las instituciones de información. Proporciona los medios necesarios, tanto para la organización interna como para la búsqueda y recuperación eficientes de documentos almacenados [1]. Este Trabajo de Fin de Grado se sumerge en la esencia misma de esta problemática, abordando el diseño e implementación de un Clasificador de Documentos basado en Tesseract-OCR y desarrollado en el entorno de programación Python.

1.1 Motivación y necesidad del proyecto

Si bien es cierto que existen otros clasificadores de documentos, el que se pretende desarrollar en este trabajo consiste en la elaboración de uno de carácter privado para la empresa para la que trabajo, Inetum, y se basará principalmente en cubrir aquellos documentos registrados en el Esquema Nacional de Interoperabilidad (ENI) el cual consiste en un marco normativo y técnico establecido por el país con el objetivo de garantizar la interoperabilidad entre las administraciones públicas [2].

Se puede resaltar la necesidad del clasificador en los siguientes puntos:

Personalización y adaptabilidad: al tener un clasificador propio basado en el modelo del ENI, se puede adaptar y personalizar la clasificación de documentos según las necesidades y peculiaridades específicas de los clientes. Esto permite ofrecer soluciones más ajustadas y precisas a las demandas individuales.

Mejora de la eficiencia: automatizar la clasificación de documentos mediante un modelo propio permite optimizar los procesos internos de la empresa y los de sus clientes. Esto conlleva a una mayor eficiencia operativa al reducir el tiempo dedicado a la clasificación manual de documentos, liberando recursos y permitiendo un enfoque más estratégico en otras áreas.

Incremento en la exactitud: al desarrollar un clasificador propio basado en el modelo del ENI, se puede lograr una mayor precisión en la clasificación de documentos, minimizando errores y mejorando la calidad del servicio ofrecido a los clientes.

Confidencialidad y seguridad: al tener un clasificador interno, se puede asegurar la confidencialidad de los datos de los clientes al manejar la información de manera interna sin depender de servicios externos para la clasificación de documentos sensibles.

Capacidad de escalabilidad: desarrollar un clasificador propio nos brinda la flexibilidad para adaptarse a un crecimiento futuro. Esto significa que el modelo puede ajustarse y escalar según las necesidades cambiantes de la empresa y de sus clientes sin depender de soluciones externas.

1.2 Objetivos

Se va a establecer como objetivo principal de este trabajo el desarrollo de un nuevo modelo para un clasificador de documentos y para ello se tiene que cumplimentar una serie de objetivos específicos donde la realización de todos ellos culmine en el principal.

Los objetivos específicos son los siguientes:

1. Diseñar la aplicación a nivel global especificar los diferentes módulos y scripts que constituyen el clasificador.
2. Implementar la estructura propuesta en Python.
3. Realizar las correspondientes pruebas y test.

1.3 Planificación

A continuación, se va a establecer una lista de tareas y una duración estimada.

Este proyecto de fin de carrera va a ser realizado, como mencionaba anteriormente, con la empresa Inetum y la duración será de unas 300 horas aproximadamente divididas en las distintas tareas, por lo que se dedicarán 5 horas diarias durante 12 semanas, de octubre a diciembre.

Las tareas por realizar son las siguientes:

1. Familiarización con los entornos de desarrollo y programación.
2. Realizar un diseño a nivel global de la aplicación.
3. Integrar algoritmos de deep learning para entrenar y crear un modelo capaz de identificar la tipología de documentos de manera precisa.
4. Desarrollar un software que extraiga los textos y las características importantes de ellos dado un pdf.
5. Implementar un predictor para poder llevar a cabo la demostración del funcionamiento de los programas anteriores.
6. Realizar las correspondientes pruebas o tests.
7. Escritura de la memoria final.
8. Preparación de la presentación de TFG.

En la Figura 1.1 podemos ver como se reparte el tiempo en las tareas recién mencionadas en el orden tal cual están estipuladas en el listado anterior a lo largo de las 12 semanas de la duración del trabajo.

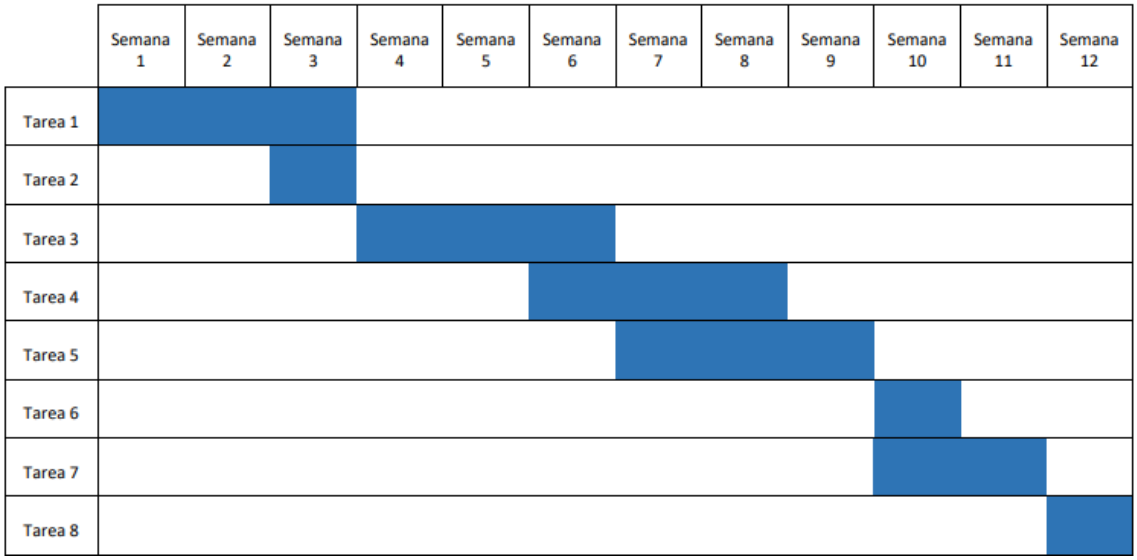


Figura 1.1 Diagrama de Gantt

1.4 Estado del arte

El estado del arte en el área de trabajo de este proyecto revela avances significativos en el reconocimiento óptico de caracteres (OCR) y la clasificación

documental. Investigaciones recientes han demostrado un interés creciente en la aplicación de técnicas de aprendizaje automático y procesamiento de imágenes para mejorar la precisión y eficiencia de los sistemas de gestión documental.

En cuanto al reconocimiento óptico de caracteres, tecnologías como Tesseract-OCR han emergido como referentes en la conversión de imágenes y documentos escaneados en texto editable. La versatilidad de estas herramientas ha catalizado su implementación en diversas aplicaciones, desde la extracción de texto en documentos históricos hasta la digitalización de archivos en entornos empresariales.

En el ámbito de la clasificación documental, investigaciones previas han explorado enfoques tradicionales y modernos basados en aprendizaje automático. La incorporación de modelos de machine learning ha permitido la creación de clasificadores más sofisticados capaces de adaptarse a una variedad de tipos de documentos y estructuras. Al nutrirse de manera constante con datos sectoriales y regulatorios, los modelos se retroalimentan y mejoran a partir de los juicios de los usuarios en su proceso de clasificación. Este enfoque permite corregir posibles inferencias de precisión en distintas iteraciones dentro de una organización específica. Gracias a la inteligencia artificial, el sistema puede sugerir al usuario tipos de clasificación, eliminando la necesidad de que este esté capacitado para clasificar documentos a diferencia de los sistemas más básicos. Se puede lograr una precisión del 97% al identificar la propiedad intelectual [3].

Además, se observa una tendencia hacia la integración de tecnologías Python en proyectos de gestión documental, aprovechando su flexibilidad y robustez. La comunidad académica y profesional ha destacado la eficacia de Python en la implementación de sistemas escalables y personalizables.

Sin embargo, se identifican desafíos persistentes, como la mejora continua de la precisión del reconocimiento, la adaptabilidad a documentos con formatos variados y la gestión eficiente de grandes volúmenes de datos. Este panorama del estado del arte proporciona un contexto sólido para la investigación y desarrollo propuestos en este proyecto, destacando áreas clave donde se puede innovar y mejorar la eficacia de los sistemas existentes.

1.5 Estructura de la memoria

Se proporciona de manera muy esquematizada los diferentes capítulos que se abordan en el documento:

Capítulo 2: el capítulo presenta la base del proyecto con los tipos de clasificaciones registradas y resalta las tecnologías clave empleadas.

Capítulo 3: en este capítulo se desarrolla e implementa el diseño del modelo en cuestión revisando las diferentes clases y módulos de los que se compone y supervisando sus requisitos funcionales y no funcionales.

Capítulo 4: compuesto de una demostración visual del funcionamiento del software, acompañado de la evaluación de este.

Capítulo 5: en este capítulo se abordan los resultados obtenidos tanto a nivel personal como profesional, así como las futuras aplicaciones sobre el proyecto.

Capítulo 6: este capítulo contiene alguno de los impactos que el trabajo podría producir en el mundo, así como los impactos producidos como solución a algunas propuestas de los ODS.

2 Soluciones existentes y tecnologías empleadas

En este capítulo se van a estudiar los tipos de documentos que cubrirá la versión final del modelo y qué vamos a utilizar para ello.

2.1 Clasificador de documentos registrados en el ENI

Para esta primera versión del modelo se requiere una demostración de su potencial antes de invertir más recursos en la obtención de otros muchos más tipos documentales para ponerlo en marcha.

Dichos documentos son reconocidos por el Esquema Nacional de Interoperabilidad (ENI) y se diferencia en 20 tipos diferentes de documentos.

Tipos de Documentos:

Documentos administrativos:

1. Resolución: documento oficial que establece decisiones o disposiciones administrativas.
2. Acuerdo: documento que refleja el consentimiento o decisión tomada por una entidad o entidades.
3. Contrato: acuerdo legal entre dos o más partes que establece obligaciones y derechos.
4. Convenio: acuerdo formal entre partes que establece compromisos o condiciones.
5. Declaración: documento que expresa una posición oficial, una manifestación de voluntad o una información específica.
6. Comunicación: escrito formal que transmite información relevante dentro de una entidad o entre entidades.

Documentos de transmisión:

1. Notificación: documento oficial que informa a una persona o entidad sobre una acción o situación.
2. Publicación: documento que comunica información de interés general al público.
3. Acuse de recibo: confirmación escrita de haber recibido un documento o información.

Documentos de constancia:

1. Acta: registro escrito de lo discutido o acordado en una reunión, sesión o acto.
2. Certificado: documento que acredita la autenticidad o veracidad de algo.
3. Diligencia: anotación oficial que se hace en un documento para constatar un hecho.

Documentos de juicio:

1. Informe: documento que presenta información detallada y análisis sobre un tema específico.

Documentos de ciudadano:

1. Solicitud: escrito mediante el cual una persona solicita algo a una entidad.
2. Denuncia: comunicación escrita en la que se informa sobre una irregularidad o falta.
3. Alegación: escrito en el que se argumenta o defiende una posición frente a algo.
4. Recursos: documento que impulsa una revisión o reconsideración de una decisión.
5. Comunicación ciudadano: escrito que transmite inquietudes o sugerencias de un ciudadano.
6. Factura: documento que detalla los bienes o servicios adquiridos y su coste.
7. Otros incautados: documentos incautados o relacionados con incautaciones que no se clasifican fácilmente en las categorías anteriores.

El Esquema Nacional de Interoperabilidad (ENI) es un marco normativo y técnico establecido en España que regula la interoperabilidad entre las administraciones públicas. Su objetivo principal es facilitar la cooperación, el intercambio de información y la prestación de servicios entre los distintos organismos y entidades gubernamentales a nivel nacional, autonómico y local.

El ENI establece principios, políticas y estándares que permiten a las diferentes entidades gubernamentales compartir datos, documentos y servicios de manera eficiente y segura. Define pautas para la creación y gestión de documentos electrónicos, asegurando la integridad, autenticidad, accesibilidad y conservación de la información.

Además, el ENI promueve la homogeneización de los procedimientos administrativos, facilitando la interoperabilidad de los sistemas de información y la comunicación entre las distintas administraciones. Esto garantiza que la información sea accesible y utilizable por todas las entidades públicas, mejorando la eficiencia y la calidad de los servicios prestados a los ciudadanos y empresas.

En resumen, el ENI constituye un marco regulatorio esencial en España para asegurar la armonización y la compatibilidad de los sistemas de información y comunicación entre las administraciones públicas, con el fin de promover la colaboración eficiente y efectiva entre ellas.

Esta demo será planteada con 5 de estos 20 tipos de documentos y la muestra de datos ha sido obtenida de kaggle y de la base de datos de la empresa, donde he recopilado más de 6000 documentos de tipo pdf etiquetados de los 5 primeros tipos de la lista: resolución, acuerdo, contrato, convenio y declaración.

Estos datos proporcionan una base sólida para el entrenamiento y la validación del modelo, permitiendo así una evaluación exhaustiva de su capacidad para clasificar diferentes tipos de documentos y valorar su precisión y eficacia.

2.2 Tecnologías y herramientas utilizadas

Se detallan las diversas bibliotecas y herramientas que forman parte del entorno de desarrollo del proyecto. La lista incluye bibliotecas de Python con sus respectivas versiones y otras herramientas específicas para el procesamiento de

datos, la manipulación de documentos y la implementación de modelos de aprendizaje automático.

```
absl-py==2.0.0
astor==0.8.1
astunparse==1.6.3
cachetools==4.2.4
certifi==2023.7.22
charset-normalizer==3.3.1
cycler==0.11.0
deskew==0.9.0
flatbuffers==23.5.26
gast==0.3.3
google-auth==1.35.0
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.59.0
h5py==2.10.0
idna==3.4
imageio==2.31.2
importlib-metadata==6.7.0
imutils==0.5.3
joblib==1.3.2
Keras==2.4.0
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.2
kiwisolver==1.4.5
libclang==16.0.6
lxml==4.9.3
Markdown==3.4.4
MarkupSafe==2.1.3
matplotlib==3.1.1
networkx==2.6.3
numpy==1.17.0
oauthlib==3.2.2
opencv-python==3.4.7.28
opt-einsum==3.3.0
packaging==23.2
```

```
pdf2image==1.7.0
Pillow==9.5.0
protobuf==3.19.6
pyasn1==0.5.0
pyasn1-modules==0.3.0
pyparsing==3.1.1
pytesseract==0.3.0
python-dateutil==2.8.2
python-docx==0.8.10
PyWavelets==1.0.3
PyYAML==6.0.1
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
scikit-image==0.16.1
scikit-learn==0.21.3
scipy==1.4.1
Shapely==1.7.0
six==1.16.0
stop-words==2018.7.23
tensorboard==2.3.0
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.3.0
tensorflow-estimator==2.3.0
tensorflow-io-gcs-filesystem==0.31.0
termcolor==2.3.0
tifffile==2021.11.2
typing-extensions==4.7.1
urllib3==2.0.7
Werkzeug==2.2.3
wrapt==1.15.0
zipp==3.15.0
```

3 Desarrollo del clasificador de documentos

En este capítulo se detallan aspectos más técnicos como el diseño a nivel global de la aplicación y su posterior implementación, así como el análisis de requisitos y las tecnologías empleadas en el desarrollo.

3.1 Análisis de requisitos

En esta sección se detallan los requisitos funcionales y no funcionales del sistema, así como los casos de uso que estructuran las interacciones entre el usuario y el sistema de clasificación de documentos.

3.1.1 Requisitos funcionales

Los requisitos funcionales delinean las acciones y funcionalidades específicas que el sistema debe cumplir para satisfacer las necesidades del usuario:

1. Conversión de PDF a imagen: el sistema tiene la capacidad de transformar documentos en formato PDF a imágenes para permitir la posterior extracción de datos a través de técnicas de OCR (Reconocimiento Óptico de Caracteres).
2. Extracción de datos mediante Reconocimiento Óptico de Caracteres (OCR): el programa realiza la extracción de texto a partir de imágenes de documentos utilizando técnicas de OCR, convirtiendo así los documentos a un formato legible y procesable.
3. Clasificación de Documentos: el software clasifica los documentos en categorías predefinidas. Utilizando los modelos entrenados, asigna a cada documento una categoría específica basada en su contenido y características extraídas durante el procesamiento.

3.1.2 Requisitos no funcionales

Estos requisitos describen las características que definen la calidad y el comportamiento esperado del sistema:

Eficiencia:

El sistema debe ser diseñado desde cero para garantizar una ejecución eficiente en términos de tiempo y recursos para las operaciones de lectura, extracción, transformación y predicción de documentos. Se busca minimizar el consumo de recursos y optimizar el rendimiento en todas las fases del procesamiento.

Precisión:

Se requiere que el sistema logre niveles de precisión significativos con una confianza adecuada desde el principio. El diseño inicial debe incorporar enfoques robustos para garantizar la precisión en la clasificación de documentos, sin depender de mejoras posteriores. La meta es alcanzar un nivel aceptable de precisión sin comprometer la integridad de las operaciones.

Escalabilidad:

La arquitectura del sistema debe ser inherentemente escalable, permitiendo el procesamiento eficiente de conjuntos de datos crecientes. Desde el inicio, se busca que la aplicación sea capaz de manejar un aumento en la carga de trabajo y facilitar futuras expansiones sin perder eficiencia en el procesamiento de documentos.

3.2 Diseño del clasificador propuesto

En este apartado se analizan el flujo de la aplicación, tal y como se muestra en la Figura 3.1, y los pasos que sigue el programa

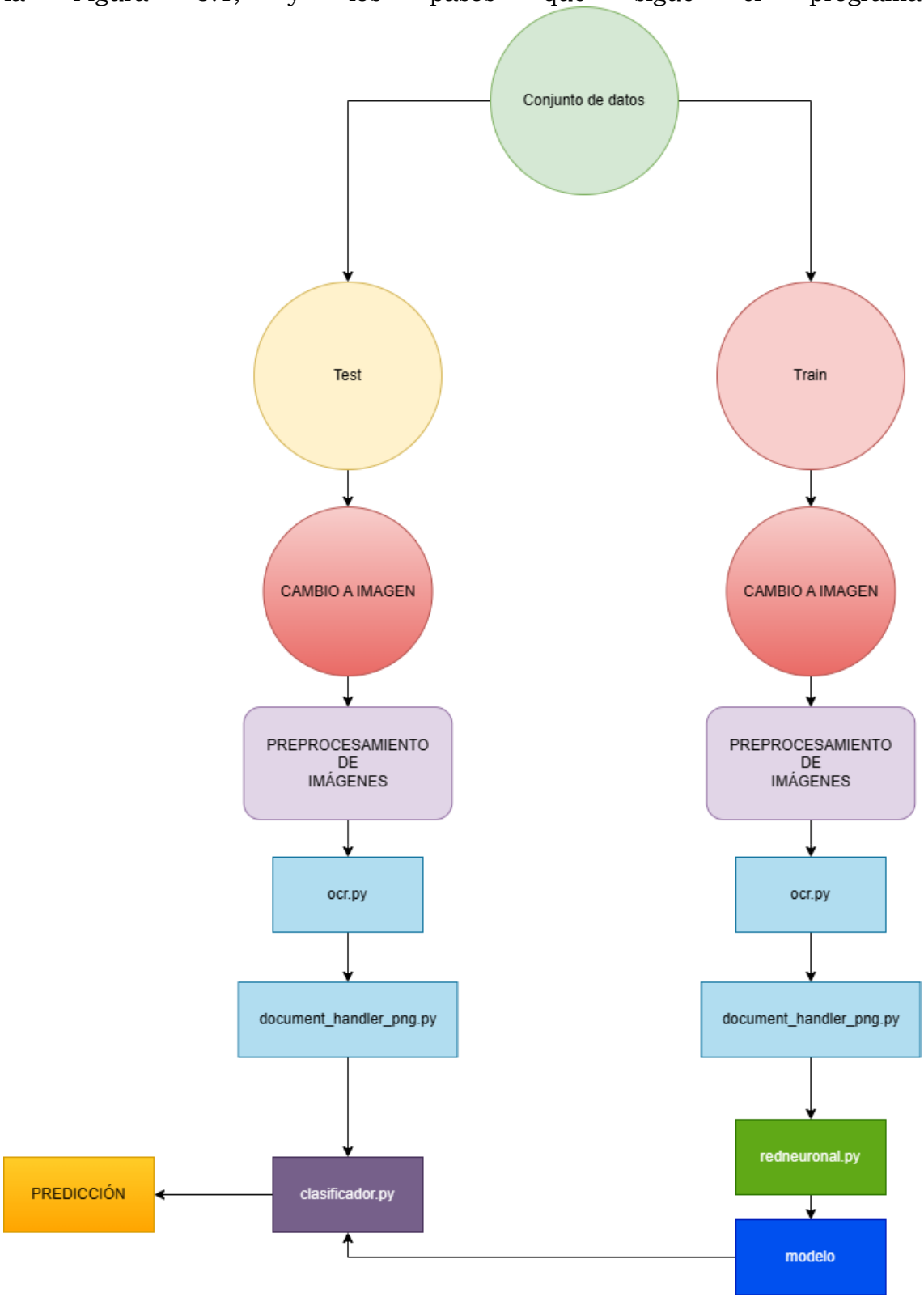


Figura 3.1 Diagrama de flujo de la aplicación a nivel global

Lectura y extracción de características de los documentos

La primera etapa implica la lectura de documentos en formato "png" que anteriormente eran archivos PDF pertenecientes a nuestro conjunto de datos. Estos archivos han pasado por una transformación utilizando la herramienta pdf2image para convertirlos en imágenes.

Tras la fase de transformación, se lleva a cabo un paso crucial de preprocesamiento de las imágenes. Este proceso tiene lugar antes de la aplicación del reconocimiento óptico de caracteres (OCR) y se centra en optimizar la calidad y la legibilidad de las imágenes. Entre las técnicas de preprocesamiento empleadas se incluyen ajustes de contraste, normalización de iluminación y eliminación de posibles ruidos visuales.

El objetivo principal del preprocesamiento es mejorar la calidad de las imágenes resultantes, facilitando así una interpretación más efectiva por parte del OCR. Este enfoque contribuye a garantizar una extracción precisa de información textual durante la etapa subsiguiente de reconocimiento de caracteres, lo que, a su vez, mejora la calidad global del proceso de clasificación.

Una vez completada la fase de preprocesamiento de imágenes, se procede a la aplicación del reconocimiento óptico de caracteres (OCR) sobre los archivos en formato "png". Este paso es esencial para convertir las representaciones visuales de los documentos en información textual comprensible. La salida de este proceso se traduce en archivos de texto (txt) que contienen la transcripción de palabras, frases y contenido textual presente en las imágenes.

Una vez transformados los archivos y aplicado el OCR, se procede a extraer características específicas de cada imagen. Estas características incluyen aspectos cuantitativos como el número de palabras, figuras, tablas y la densidad del contenido visualizado en la imagen. Estos datos proporcionan una base de datos adicional que enriquece el proceso de clasificación, permitiendo un análisis más detallado y preciso de cada documento.

Transformación a representación "Bag of Words"

La siguiente etapa es la transformación de los documentos a la representación conocida como "Bag of Words" (BoW). Esta técnica permite convertir textos en vectores numéricos representativos, donde cada elemento representa la frecuencia o presencia de una palabra en un documento.

Este proceso se inicia con la tokenización de los textos, separándolos en unidades más pequeñas, como palabras o frases. Luego, se cuenta la frecuencia de aparición de cada palabra en los documentos utilizando la librería CountVectorizer. Posteriormente, se realiza la transformación de la frecuencia de palabras a frecuencia inversa, utilizando el TfidfTransformer. Esto resulta en una matriz donde las filas representan los documentos y las columnas representan las palabras aprendidas a partir del modelo BoW. De esto se encarga el script document_handler_png.py.

Formación del Conjunto de Entrenamiento Definitivo

En esta fase, se consolida el conjunto de entrenamiento final. La matriz generada anteriormente con la representación BoW se amplía al agregar las características específicas de los documentos extraídas previamente. Esto resulta en una nueva matriz con un número de filas equivalente al número de documentos y un número de columnas compuesto por las palabras aprendidas

en el modelo BoW, junto con las características previamente extraídas (número de palabras, figuras, tablas y densidad).

Creación y entrenamiento de la red neuronal

La parte central del proceso es la creación y entrenamiento de la red neuronal, un componente fundamental en la clasificación de documentos. Para este propósito, se construye una red neuronal con múltiples capas, lo que permite modelar relaciones más complejas entre los datos. Esta red está compuesta por una capa de entrada, una o más capas intermedias y una capa de salida.

En la capa de salida, se genera un vector donde una posición específica contiene el valor 1 y el resto de posiciones tienen el valor 0, representando así la categoría predicha por la red neuronal para cada documento. Durante el entrenamiento, la red neuronal es alimentada con la matriz del conjunto de entrenamiento junto con un vector que contiene las categorías de cada documento. Este proceso permite que la red aprenda y ajuste sus pesos y parámetros internos para realizar predicciones precisas en la fase de test.

Predicción de documentos con el modelo entrenado

Una vez que la red neuronal ha sido entrenada con éxito, se procede a realizar predicciones sobre un conjunto de documentos de test. Para esto, se lleva a cabo una serie de pasos que incluyen la lectura y transformación de los documentos de test en la representación BoW utilizando los modelos previamente entrenados. La red neuronal procesa estos documentos y genera una matriz de salidas con dimensiones correspondientes al número de documentos de test y categorías disponibles.

Es importante destacar que la red neuronal utiliza una función de activación sigmoide, lo que significa que las salidas no contienen exclusivamente valores binarios (0 o 1), sino valores decimales en el rango de 0 a 1. Estos valores representan la probabilidad de pertenencia del documento a una determinada categoría, siendo la categoría predicha aquella con la probabilidad más alta para cada documento.

El resultado final es un modelo de Deep Learning capaz de clasificar documentos en las 5 diferentes categorías.

3.3 Descripción de tecnologías clave

Deep learning: el Deep Learning ha sido un componente esencial en el desarrollo de modelos avanzados para la clasificación de documentos. Esta técnica de aprendizaje automático se caracteriza por la utilización de redes neuronales profundas para aprender y extraer patrones complejos de los datos. En el contexto de este proyecto, se ha aplicado el Deep Learning para mejorar la capacidad del clasificador de documentos, permitiendo la identificación precisa de diversas categorías documentales. Para este propósito, se han utilizado herramientas como Keras y TensorFlow, que proporcionan entornos flexibles y potentes para la construcción y el entrenamiento de modelos de redes neuronales profundas.

Redes neuronales: Su denominación y estructura encuentran su inspiración en el cerebro humano, imitando la manera en que las neuronas biológicas se comunican entre sí [4]. Se compone de unidades básicas llamadas neuronas artificiales, organizadas en capas interconectadas que trabajan en conjunto para resolver problemas de aprendizaje automático.

Estas neuronas artificiales están organizadas en capas:

- **Capa de entrada:** recibe los datos y los envía a través de conexiones ponderadas a la siguiente capa.
- **Capas ocultas:** procesan la información recibida de las capas anteriores, realizando cálculos mediante operaciones matriciales y aplicando funciones de activación para generar salidas.
- **Capa de salida:** produce el resultado final o la predicción basada en el procesamiento de las capas anteriores.

Pdf2image: es una herramienta especialmente útil cuando se trabaja con documentos en formato PDF. Su funcionalidad principal radica en la capacidad de convertir páginas de un archivo PDF en imágenes, lo que permite manipular el contenido del PDF como imágenes individuales.

Esto es relevante en proyectos que involucran OCR (Reconocimiento Óptico de Caracteres) o procesamiento de imágenes, ya que trabajar con PDFs directamente puede ser más complejo debido a su naturaleza compuesta por capas, gráficos, texto y otros elementos.

PDF2Image se utiliza para extraer estas páginas del PDF y convertirlas en formatos de imagen estándar como JPEG o PNG. Esto facilita enormemente el procesamiento de cada página como una imagen independiente, lo que simplifica tareas como la detección y el análisis de texto utilizando herramientas de OCR como Tesseract.

Tesseract: Tesseract-OCR [5] es un motor de OCR (Reconocimiento Óptico de Caracteres) de código abierto y altamente preciso. Desarrollado inicialmente por HP y ahora mantenido por Google, es reconocido por su capacidad para extraer texto de imágenes, convirtiéndolas en contenido digital procesable. Utiliza algoritmos avanzados para analizar imágenes y reconocer patrones que representan caracteres en diferentes idiomas.

El funcionamiento de Tesseract se basa en modelos de redes neuronales convolucionales (CNN) y técnicas de procesamiento de imágenes para identificar y clasificar píxeles que representan caracteres. Su versatilidad y precisión lo hacen útil en diversas aplicaciones, desde la digitalización de documentos hasta la extracción de texto en tiempo real en aplicaciones móviles.

Python: Python [6] es un lenguaje de programación de alto nivel, reconocido por su sintaxis legible y versatilidad en diversos campos, incluido el desarrollo web, científico y de aprendizaje automático. Es una elección popular en proyectos de ciencia de datos y aprendizaje automático debido a su amplia gama de bibliotecas especializadas y su facilidad para prototipar y desarrollar soluciones complejas.

En el contexto de este proyecto, Python se ha utilizado debido a su ecosistema rico en herramientas de procesamiento de texto, imágenes y datos, como NumPy, y las bibliotecas específicas para procesamiento de lenguaje natural (NLP) y aprendizaje automático como TensorFlow y Keras.

Tensorflow y Keras: Keras [7] es una API de alto nivel para la construcción y el entrenamiento de modelos de aprendizaje automático, mientras que TensorFlow [8] es un marco de trabajo de código abierto para aprendizaje automático desarrollado por Google. Keras proporciona una interfaz amigable y

modular para trabajar con TensorFlow, lo que facilita la construcción y el despliegue de modelos de redes neuronales.

Ambas herramientas se han utilizado para implementar modelos avanzados de clasificación de documentos basados en aprendizaje profundo (Deep Learning). Estos modelos pueden aprender patrones complejos y representaciones de alto nivel de los datos, como imágenes de documentos, para lograr una precisión y generalización significativas en la clasificación.

Numpy: es una biblioteca fundamental en Python para cálculos numéricos, especialmente en manipulación de matrices y operaciones matemáticas. Su eficiencia en la manipulación de datos multidimensionales ha hecho que sea esencial en el preprocesamiento de datos para el aprendizaje automático, incluyendo la preparación de datos para entrenamiento y validación de modelos.

Scipy: es una biblioteca de código abierto en Python que se utiliza principalmente para matemáticas, ciencia e ingeniería. Proporciona un conjunto de herramientas y algoritmos para resolver problemas matemáticos y científicos complejos, incluyendo optimización, álgebra lineal, procesamiento de señales y estadísticas. En el contexto del procesamiento de documentos, Scipy ofrece funciones útiles para el procesamiento de imágenes, manipulación de datos y operaciones matriciales, lo que la convierte en una herramienta valiosa para la manipulación y el análisis de imágenes de documentos.

Poppler: es una biblioteca de software que se utiliza para renderizar documentos PDF. Proporciona herramientas para extraer información de documentos PDF, incluyendo texto, imágenes y metadatos. Poppler permite el acceso programático a los contenidos de los archivos PDF, lo que lo hace útil en aplicaciones que requieren la manipulación y extracción de datos de documentos en este formato. En el contexto del proyecto, Poppler ha sido utilizado para el procesamiento de documentos PDF, extrayendo información relevante para su posterior análisis y clasificación.

OpenCV (Open Source Computer Vision Library): es una biblioteca de código abierto diseñada para la visión por computadora y el procesamiento de imágenes. Fue desarrollada originalmente por Intel y ahora es mantenida por la comunidad. OpenCV proporciona una amplia gama de funciones y algoritmos que permiten a los desarrolladores trabajar con imágenes y videos de manera eficiente.

Las principales características y funcionalidades de OpenCV que aportan calidad a este proyecto son:

Manipulación de imágenes y videos: OpenCV ofrece funciones para leer, escribir, redimensionar, rotar y manipular imágenes y videos.

Procesamiento de imágenes: incluye una variedad de funciones para el procesamiento de imágenes, como filtrado, suavizado, detección de bordes, transformaciones geométricas, etc.

Detección de objetos: proporciona algoritmos y herramientas para la detección de objetos en imágenes, como Haar cascades, detección de características, etc.

Reconocimiento de patrones: ofrece herramientas para el reconocimiento de patrones y características en imágenes.

Aprendizaje automático: OpenCV incluye módulos que se integran con bibliotecas de aprendizaje automático como TensorFlow y PyTorch.

Visual studio code (VsCode): es un entorno de desarrollo integrado (IDE) de código abierto y altamente personalizable desarrollado por Microsoft. Ofrece una amplia gama de características y extensiones que lo hacen popular entre los desarrolladores. VSCode es utilizado en el proyecto debido a su versatilidad, su amplia compatibilidad con diversos lenguajes de programación y su capacidad para ofrecer herramientas específicas para el desarrollo de modelos de aprendizaje automático, como la depuración interactiva, integración con herramientas de control de versiones y extensiones especializadas para Python, TensorFlow, Keras y otros frameworks de Machine Learning. Además, su interfaz intuitiva y su comunidad activa de desarrolladores hacen que sea una elección común para la escritura de código, depuración y gestión de proyectos en el ámbito de la ciencia de datos y el desarrollo de aplicaciones.

3.4 Implementación del modelo

En la Figura 3.2 se pueden ver los principales módulos y clases que se han implementado en el desarrollo del software y se detalla a continuación cómo funcionan todas y cada una de ellas.

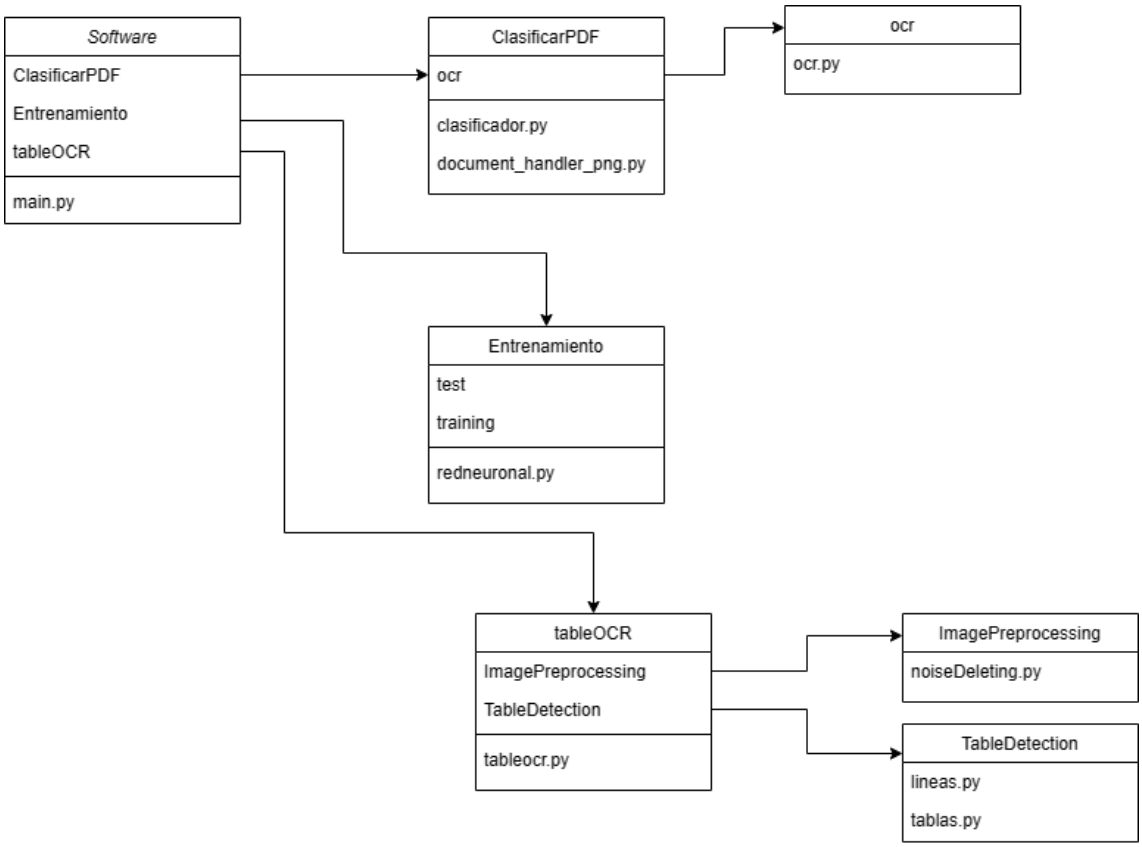


Figura 3.2 Composición de los distintos módulos y clases del software

3.4.1 Módulo Principal

Esta es la carpeta raíz del proyecto, aquí se encuentra el programa principal y los submódulos a los que este accede para desarrollar tanto la clasificación como el entrenamiento de un nuevo modelo dependiendo de los argumentos de entrada.

3.4.1.1 Main.py

Adentrándose en el funcionamiento de la clase principal, se observa el uso de los siguientes módulos:

```
from tableOCR import tableocr

from ClasificarPDF import clasificador

import constants

import sys

import logging

import cv2

import numpy as np

import os

from pdf2image import convert_from_path

import tempfile

from Entrenamiento import redneuronal

from ClasificarPDF import constants as constantsEntrenamiento
```

En cuanto su funcionamiento, main.py, procesa los argumentos de línea de comandos para obtener el nombre del fichero y la carpeta de destino.

En caso de que se proporcionan argumentos, se clasifica un documento. Si no se proporcionan argumentos, se utiliza un documento de prueba predefinido.

Se verifica si se deben aplicar técnicas de reducción de ruido (applyNoiseReduction) y se especifica la acción a realizar (funcion).

Dependiendo de la acción especificada (funcion), se llevan a cabo diferentes procesos:

Si funcion es "TRAIN", se entrena un modelo.

Si funcion es "TRAINCOMPLETE", se realiza un entrenamiento más completo, evaluación y validación del modelo.

En otros casos, o si función es “PREDICT”, se pasaría a hacer la clasificación del documento pasado por la línea de comandos.

Operaciones sobre imágenes y documentos:

Se utiliza la biblioteca pdf2image para convertir archivos PDF en imágenes PNG temporales.

Se aplican técnicas de preprocesamiento (llamada a document_handler_png).

Se guarda la imagen preprocesada en una carpeta temporal.

Se hace una llamada al clasificador.py, al cual se le pasa la imagen preprocesada.

Limpieza y salida del programa:

Se realizan acciones de limpieza, como eliminar carpetas temporales creadas durante el proceso.

Se finaliza el programa con sys.exit.

Es recomendable activar el flag de aplicar técnicas de reducción de ruido si se sabe que el pdf proporcionado para clasificar dispone de imágenes o si se desconoce, por seguridad, ya que el algoritmo de procesamiento de imágenes se beneficia significativamente de imágenes más limpias. La reducción de ruido ayuda a resaltar las características importantes y facilita la interpretación por parte de los algoritmos, además de que el exceso de ruido en los datos de entrada puede afectar negativamente al modelo y su capacidad de identificar patrones.

3.4.2 ClasificarPDF

Este módulo dispone de dos clases muy importantes, clasificador.py y document_hanlder_png.py las cuales se encargan de, como su mismo nombre indica, clasificar el documento proporcionado por los argumentos al programa principal y realizar el preprocesamiento de datos correspondiente (tokenizado y BoW) para que los datos de entrada se ajusten a los del modelo respectivamente.

3.4.2.1 Clasificador.py

Librerías y clases utilizadas en este script:

```
import os

from ClasificarPDF import document_handler_png as dh_png

import numpy as np

from keras.models import load_model

from ClasificarPDF import constants

import shutil

from ClasificarPDF.ocr import ocr

import tempfile

import sys

import time

from joblib import dump, load
```

El funcionamiento de este script es básicamente recibir y aplicar mediante la llamada al módulo ocr en esta misma carpeta el Reconocimiento Óptico de Caracteres a la imagen recibida del script principal y plasma la información recogida a un fichero de extensión “txt”.

Una vez recogida la información de entrada en un formato entendible, se hace una llamada al script encargado de preprocesar los datos para que sean

compatibles en nuestro “abecedario”, document_handler_png.py, el cuál se verá en el siguiente apartado.

Pasada la información por ello, el script está preparado para cargar y utilizar el modelo mediante la función “load” de joblib y le devuelve el resultado de la predicción a “main.py”, quien luego lo mostrará por pantalla.

3.4.2.2 Document_handler_png.py

Este de aquí se trata del módulo que extrae toda la información determinante para la clasificación de los documentos como el número de tablas, páginas, densidad, el texto y el número de palabras.

Las librerías y módulos que utiliza son:

```
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_extraction.text import TfidfTransformer

import numpy as np

import os

from stop_words import get_stop_words

from ClasificarPDF import ocr

from joblib import dump, load

from ClasificarPDF import constants

from sklearn.decomposition import TruncatedSVD
```

Una vez se ha obtenido la información más importante del texto, este proporciona una representación BoW, una técnica común en procesamiento de lenguaje natural que transforma documentos de texto en vectores numéricos, ignorando el orden y la estructura gramatical de las palabras, y enfocándose en la frecuencia de aparición de las palabras en el texto.

Se explica cómo el script realiza el tokenizado y estructura la matriz BoW:

Tokenización:

La tokenización es el proceso de dividir un texto en unidades más pequeñas llamadas tokens. En este script, la tokenización se realiza utilizando la función CountVectorizer de la biblioteca scikit-learn. Aquí está el fragmento de código relevante:

```
vectorizer = CountVectorizer(stop_words=stop_words) corpusTransformado
= vectorizer.fit_transform(corpusTrain)
```

stop_words son las palabras de parada, es decir, palabras comunes que generalmente se omiten durante el procesamiento de texto.

corpusTrain es una lista que contiene el texto de los documentos de entrenamiento.

La función fit_transform ajusta el vectorizador al corpus de entrenamiento y transforma los documentos en una matriz de conteo. Cada fila de la matriz representa un documento, y cada columna representa una palabra única en el corpus. El valor en una posición específica indica la frecuencia de esa palabra en el documento correspondiente.

Representación BoW:

Después de la tokenización, se obtiene una matriz dispersa llamada corpusTransformado. Cada fila de esta matriz representa un documento, y cada columna representa una palabra única. El valor en la posición (i, j) de la matriz indica cuántas veces aparece la palabra j en el documento i.

En este script, también se aplica la transformación TF-IDF (TfidfTransformer) para ponderar la importancia relativa de las palabras en la matriz BoW.

```
tdidf=TfidfTransformer()BoWtrain=tdidf.fit_transform(corpusTransformado.toarray())
```

Este paso ajusta el transformador TF-IDF al corpus BoW y transforma la matriz BoW en una matriz TF-IDF. La matriz resultante, BoWtrain, se utiliza para representar la información ponderada sobre la importancia de las palabras en cada documento.

Reducción de dimensionalidad:

Adicionalmente, el script utiliza la técnica de Singular Value Decomposition (SVD) para reducir la dimensionalidad de la matriz BoW. Esto se hace con la clase TruncatedSVD de scikit-learn:

```
seleccion = TruncatedSVD(n_components=1000) seleccion.fit(BoWtrain, y)
BoWtrain = seleccion.transform(BoWtrain)
```

Este paso reduce el número de dimensiones de la matriz BoW a 1000, lo cual puede ser útil para manejar matrices grandes y reducir la complejidad computacional.

Esto se aplica tanto en el proceso de entrenamiento como en el de prueba para establecer una norma en los datos, y que los datos de test, independientemente del formato, se ajusten a los de entrenamiento.

3.4.2.3 Ocr

En este módulo destaca ocr.py; Este script se encarga de extraer texto de imágenes (formato PNG) utilizando la biblioteca Tesseract OCR y luego genera un archivo de texto (.txt) con algunas estadísticas y el texto extraído.

Primeros pasos:

Se ejecuta una función que toma como entrada la ruta de una carpeta que contiene imágenes PNG, (la ruta de destino para guardar el archivo de texto, el número de tablas, y el nombre del archivo).

Procesamiento de imágenes:

Itera sobre cada imagen en la carpeta, utilizando Tesseract OCR para extraer texto y obtener datos detallados de la imagen, como cajas, confidencias y números de línea y página.

Almacenamiento de datos:

Los datos detallados se almacenan en matrices y listas para su posterior procesamiento.

Cálculo de estadísticas:

Se calculan estadísticas clave, como el número total de páginas, el número de palabras y la densidad (número de palabras por página).

Se genera una lista de líneas que contiene información sobre las estadísticas calculadas y el texto extraído de las imágenes.

Generación de archivo de texto:

Se llama a la función generate_txt del módulo txtgenerator para crear un archivo de texto con las líneas proporcionadas. La función devuelve la ruta del archivo de texto generado.

3.4.3 Entrenamiento

En el módulo entrenamiento están las carpetas correspondientes a los conjuntos de prueba y entrenamiento y el script redneuronal.py, fundamental para la creación del modelo utilizado.

Este script, implementa un sistema de clasificación de documentos que utiliza técnicas de Procesamiento de Lenguaje Natural (PLN) y Aprendizaje Profundo. El objetivo principal es clasificar documentos en formatos png en un conjunto predefinido de clases. Se emplean representaciones Bag-of-Words y Redes Neuronales Artificiales (RNA) utilizando la biblioteca Keras.

Librerias y módulos utilizados:

```
from sklearn.preprocessing import OneHotEncoder
from keras.models import Sequential
from keras.layers import Dense
from ClasificarPDF import document_handler as dh
from ClasificarPDF import document_handler_png as dh_png
import numpy as np
from ClasificarPDF import constants
from ClasificarPDF.ocr import ocr
import os
from tableOCR import tableocr
import tempfile
import cv2
from pdf2image import convert_from_path
import logging
import shutil
from keras import optimizers
from sklearn.preprocessing import StandardScaler
from joblib import dump, load
from keras.regularizers import l1
from keras.models import load_model
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix,
precision_recall_fscore_support
```

Teniendo en cuenta que se conoce como funcionan los módulos anteriores, este apartado se centra en como este script desarrolla el modelo:

Reparación de carpetas:

Se verifica si la carpeta constants.carpeta_train_txt existe. Si no existe, se crea y se le otorgan permisos.

Se itera sobre las carpetas de clases (carpetaTipo) en carpetaTrain.

Transformación de PDF a PNG:

Para cada archivo PDF (imagen) en la carpeta carpeta, se verifica si es un PDF.

Si es un PDF, se crea una carpeta en constants.carpeta_train_pdf_a_png para almacenar las imágenes PNG resultantes.

Se itera sobre las páginas del PDF y se convierten a imágenes PNG, que se guardan en la carpeta correspondiente.

Creación de archivos TXT:

Se llama a la función create_txt para cada documento PNG creado, generando archivos de texto (TXT).

Entrenamiento del modelo:

Después de convertir los PDF a PNG y crear los archivos TXT, se llama a la función trainModel para entrenar un modelo de red neuronal.

Preparación de datos:

Se llama a la función dh_png.bag_of_words_train para obtener las características (X_train) y las etiquetas (y) del conjunto de entrenamiento.

Transformación de etiquetas:

Se llama a la función transformOutput para convertir las etiquetas (**y**) en su representación one-hot encoded.

Normalización de datos:

Se utiliza StandardScaler para normalizar las características del conjunto de entrenamiento (X_train).

Definición del modelo de red neuronal:

El modelo de red neuronal utilizado consta de tres capas:

La primera capa, la capa de entrada, contiene 12 neuronas y utiliza la función de activación ReLU (Rectified Linear Unit). Esta capa recibe los datos de entrada y los procesa a través de estas neuronas.

La segunda capa, la capa oculta, tiene 8 neuronas, también empleando la función de activación ReLU. Esta capa procesa la información proveniente de la capa de entrada, realizando operaciones intermedias para comprender las relaciones más complejas entre los datos.

Finalmente, la última capa es la capa de salida, que cuenta con 2 neuronas y utiliza la función de activación sigmoide. Esta capa produce la salida final del modelo, indicando la clasificación o predicción correspondiente.

Nuestro modelo consta de una única capa intermedia debido al modelo simple que se ha elaborado en primera instancia, pero se estudiará para el modelo

definitivo una ampliación del número de capas ocultas si esto supone una mejora.

Las funciones de activación, como ReLU (Rectified Linear Unit) y sigmoide, son bloques fundamentales en una red neuronal que determinan la salida de cada neurona y, por ende, la información transmitida a las siguientes capas.

ReLU (Rectified Linear Unit): Esta función asigna cero a todos los valores negativos de entrada y deja pasar los valores positivos sin modificarlos. En términos simples, si la entrada es mayor que cero, la salida es igual a la entrada; si es menor o igual a cero, la salida es cero. ReLU es conocida por su capacidad para ayudar a resolver el problema del desvanecimiento del gradiente y suele ser eficiente en el entrenamiento, acelerando la convergencia de la red.

Sigmoide: Es una función de activación que transforma los valores de entrada en un rango entre 0 y 1. Su forma de S alarga los valores hacia los extremos, de modo que los valores muy grandes tienden a 1 y los muy pequeños tienden a 0. Se usa comúnmente en la capa de salida de una red neuronal para problemas de clasificación binaria, donde se desea obtener una probabilidad de pertenencia a una clase.

Compilación del modelo:

Se compila el modelo utilizando el optimizador Adam y la pérdida de entropía cruzada categórica.

Entrenamiento del modelo:

Se entrena el modelo en el conjunto de entrenamiento (X_train, y_train) durante 50 épocas con un tamaño de lote de 100. (el modelo pasa por todo el conjunto de entrenamiento 50 veces, actualizándose después de procesar bloques de 100 ejemplos en cada pase.)

Durante el entrenamiento, se normalizan los valores de NaN e Inf en el conjunto de entrenamiento.

Guardado del modelo:

Una vez finalizado el entrenamiento, se guarda el modelo en un archivo HDF5 utilizando la función save de Keras.

3.4.4 TableOCR

Este es el módulo que se encarga de realizar todo el preprocesamiento de las imágenes, identificar el número de líneas, número de páginas y las tablas.

Se distribuye en un script principal tableOCR.py que hace llamadas a noiseDeleting.py y líneas.py las cuales se verán a continuación en este apartado.

3.4.4.1 TableOCR.py

Este script está diseñado para procesar una imagen en busca de tablas, líneas, palabras y detectarlas e importa las siguientes librerías y módulos:

```
from tableOCR.ImagePreprocessing import noiseDeleting as ip

from tableOCR.TableDetection import lineas as td

import cv2

import imutils
```

La función principal toma un nombre de archivo de imagen y un indicador booleano, el pasado por argumentos applyNoiseReduction y realiza los siguientes pasos:

Lee la imagen original usando OpenCV.

Llama a la función get_preprocessed_image del módulo ImagePreprocessing para obtener una imagen preprocesada y el ángulo de rotación.

Rota la imagen original según el ángulo de rotación.

Llama a la función get_tables del módulo TableDetection para obtener las tablas detectadas en la imagen preprocesada y rotada.

Devuelve las tablas detectadas y la imagen preprocesada.

Este script es el motor del módulo de preprocesamiento de imágenes, pero ahora vamos a describir las clases que la hacen funcionar.

3.4.4.2 noiseDeleting.py

Se mencionan los aspectos de este script que son más importantes. Incluye las siguientes librerías:

```
import cv2

import numpy as np

from numpy.fft import fft2, ifft2

from scipy.signal import convolve2d

from skimage.filters import threshold_sauvola

from deskew import determine_skew

import pytesseract

import imutils

from pytesseract import Output

import logging
```

Su funcionamiento radica en las siguientes funciones:

ecualizacion_histograma(imagen): Realiza la ecualización del histograma en la imagen de entrada.

wiener_filter(img, kernel, K): Aplica el filtrado de Wiener a la imagen de entrada utilizando el núcleo especificado y el parámetro K.

blur(img, kernel_size=11): Aplica la convolución con un núcleo uniforme para desenfocar la imagen.

add_gaussian_noise(img, sigma): Añade ruido gaussiano a la imagen de entrada.

enderezar(img): Corrige la orientación de la imagen de entrada.

sauvola_binarization(img): Aplica el método de binarización de Sauvola a la imagen de entrada.

show_reduced_image(titulo, img, scale_percent): Muestra una versión redimensionada de la imagen de entrada.

get_preprocessed_image(image, reducepercentage, applyNoiseReduction):

Todas estas funciones conforman una limpieza total de la imagen haciendo así que los algoritmos que se utilizan posteriormente sobre ellas sean más eficaces y precisos.

3.4.4.3 Lineas.py

En este script se emplean numerosas funciones cuyo fin radica en la detección de todos los elementos que se encuentren en la imagen, ya sean palabras, líneas y tablas con ayuda de la clase auxiliar tables.py. Estas son todas las funciones que incorpora explicadas brevemente además de las librerías que incorpora sumadas a la clase auxiliar anteriormente mencionada:

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

from shapely.geometry import LineString, Point

import logging

import tableOCR.TableDetection.tablas as tables
```

Funciones:

show_reduced_image: muestra una imagen reducida en tamaño.

intersection: calcula la intersección entre dos segmentos de línea.

equal_points: compara dos puntos considerándolos iguales si están dentro de un margen de 4 píxeles.

ordenarPuntos: ordena dos puntos de un segmento, siendo el inicio el punto más a la izquierda o más arriba.

any_non_vertical_segments_intersect: detecta intersecciones entre dos grupos de segmentos.

DetectLines: utiliza la transformada de Hough para detectar líneas en una imagen.

dibujarPuntos: dibuja puntos y líneas en una imagen.

enlargeLines: aumenta la longitud de los segmentos en un píxel.

group_points_by_distance: agrupa puntos por distancia, conservando un representante por grupo.

is_vertical y is_horizontal: verifican si un segmento es vertical u horizontal.

get_more_intersection_points: obtiene puntos de intersección adicionales utilizando máscaras.

El método principal para recuperar las dimensiones de las tablas de una imagen es get_tables y sus principales funciones son las siguientes:

Detecta líneas en la imagen utilizando la transformada de Hough.

Amplía la longitud y busca intersecciones de los segmentos de línea.

Utiliza máscaras para obtener más puntos de intersección.

Agrupa y elimina puntos duplicados por distancia.

Elimina esquinas incorrectas y devuelve las coordenadas de las tablas detectadas.

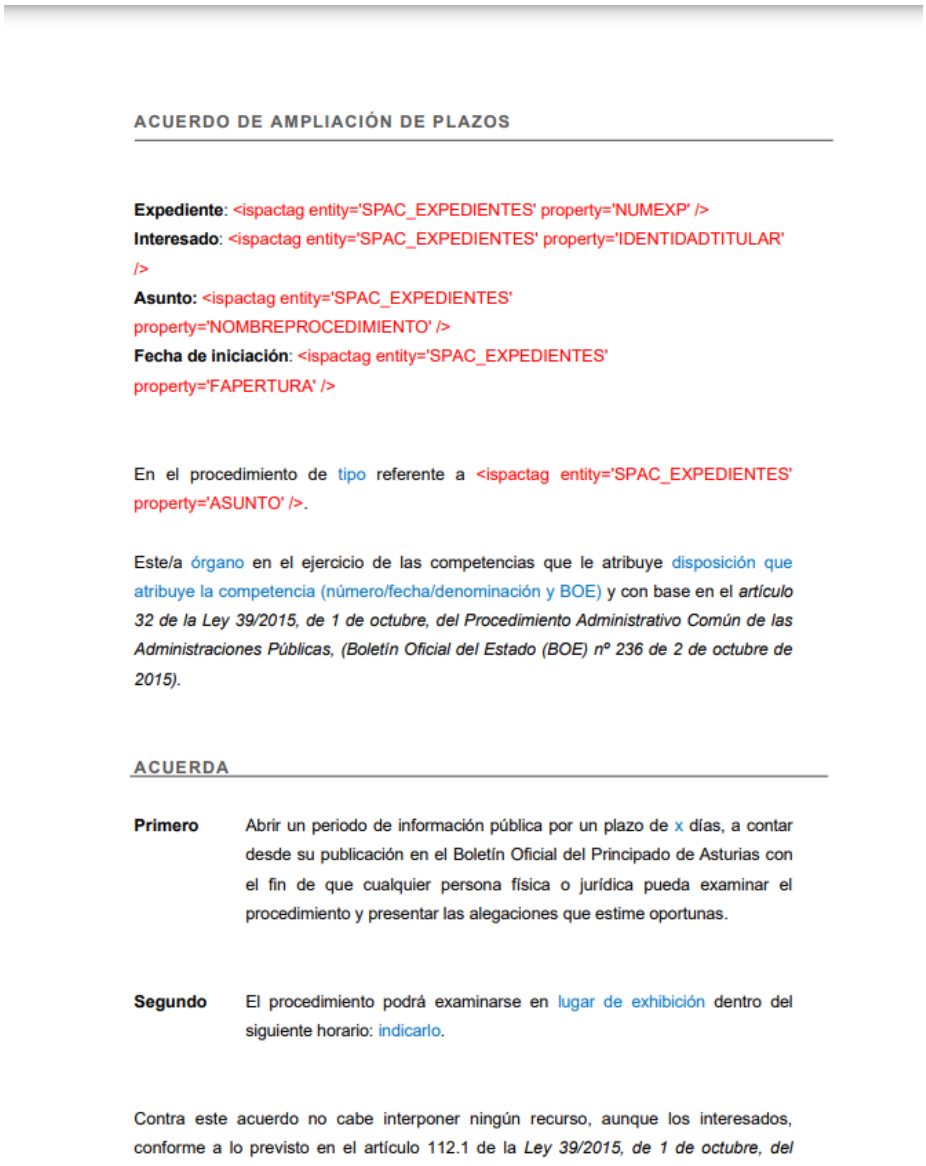
4 Demostración y evaluación

Una vez expuesta toda la teoría sobre cómo funciona el software, se muestra un ejemplo práctico para observar su funcionamiento.

4.1 Demostración

Para comprobar la capacidad del clasificador se ha elegido para este ejemplo proporcionar por línea de comandos un documento de tipo "acuerdo", perteneciente a la segunda clase de las 5 que admite.

Las Figura 4.1 y Figura 4.2 representan un ejemplo de acuerdo sin rellenar, aunque no tiene interferencia alguna en el resultado del programa ya que las palabras claves que representan que este documento se trata de tipo acuerdo están presentes en él.



ACUERDO DE AMPLIACIÓN DE PLAZOS

Expediente: <ispactag entity='SPAC_EXPEDIENTES' property='NUMEXP' />
Interesado: <ispactag entity='SPAC_EXPEDIENTES' property='IDENTIDADTITULAR' />
Asunto: <ispactag entity='SPAC_EXPEDIENTES' property='NOMBREPROCEDIMIENTO' />
Fecha de iniciación: <ispactag entity='SPAC_EXPEDIENTES' property='FAPERTURA' />

En el procedimiento de tipo referente a <ispactag entity='SPAC_EXPEDIENTES' property='ASUNTO' />.

Este/a órgano en el ejercicio de las competencias que le atribuye disposición que atribuye la competencia (número/fecha/denominación y BOE) y con base en el *artículo 32 de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas, (Boletín Oficial del Estado (BOE) nº 236 de 2 de octubre de 2015).*

ACUERDA

Primero Abrir un periodo de información pública por un plazo de x días, a contar desde su publicación en el Boletín Oficial del Principado de Asturias con el fin de que cualquier persona física o jurídica pueda examinar el procedimiento y presentar las alegaciones que estime oportunas.

Segundo El procedimiento podrá examinarse en lugar de exhibición dentro del siguiente horario: indicarlo.

Contra este acuerdo no cabe interponer ningún recurso, aunque los interesados, conforme a lo previsto en el artículo 112.1 de la *Ley 39/2015, de 1 de octubre, del*

Figura 4.1 Primera página del documento de entrada al programa

Procedimiento Administrativo Común de las Administraciones Públicas (BOE nº 236, de 2 de octubre de 2015).

Mediante este documento se
notifica a <ispactag entity='SPAC_EXPEDIENTES' property='IDENTIDADTITULAR' /> el presente acuerdo, según lo exigido en el *artículo 40.1 de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas (BOE nº 236, de 2 de octubre de 2015).*

Ciudad, <ispactag rule='FechaActual' />

El cargo

Fdo.: nombre y apellidos del firmante

Figura 4.2 Segunda página del documento de entrada al programa

Como se menciona en el capítulo 3 apartado 4 sección 1, se deben proporcionar argumentos de entrada a "main.py", los cuáles se ven de un color azulado en la Figura 4.3.

Para ello, se crea un fichero auxiliar "launch.json" que haga el arranque más cómodo:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "args": [
"Doc_ocr_classification\\ClasificarPDF\\ejemplos\\acuerdo.pdf",
                "C:\\Users\\daniel.garcia-
rodrig\\Desktop\\Salida_doc_ocr_classification",
                "1",
                "PREDICT"
            ],
            "console": "integratedTerminal",
            "justMyCode": false
        }
```

En este script se aprecia que el primer argumento corresponde al documento a clasificar, el segundo la carpeta de salida donde recibiremos los datos convertidos y procesados, el tercero corresponde al flag activado para que el programa active la eliminación de ruido y el cuarto a que ejecute la función PREDICT para que clasifique el documento. Al tratarse de un documento pdf de 2 páginas vemos como tanto en la Figura 4.3 y en la Figura 4.4 se imprime por pantalla por un lado el valor de la confianza ofrecida por las imágenes de cada una de las páginas del documento, en ambos casos > 10 por lo que no sería necesario realizar ningún preprocesamiento de imágenes y en la variable ángulo, el valor 0, lo que implica que no hay que rotarla. A continuación, observamos como se lleva a cabo el reconocimiento de elementos del texto como líneas, palabras y tablas en las trazas encabezadas por tableOCR.TableDetection.lineas.

```
(septiembre2019) PS C:\Users\daniel.garcia-rodrig\Desktop\ModeloClasificacion>  c:; cd 'c:\Users\daniel.garcia-rodrig\Deskt
:\Users\daniel.garcia-rodrig\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\deb
ugpy\launcher' '51512' '--' 'C:\Users\daniel.garcia-rodrig\Desktop\ModeloClasificacion\Doc_ocr_classification\classificateD
oc.py' 'Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pdf' 'C:\Users\daniel.garcia-rodrig\Desktop\Salida_doc_ocr_cl
assification' '1' 'PREDICT'
2024-01-09 14:05:23.373205: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library
'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2024-01-09 14:05:23.384107: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not
have a GPU set up on your machine.
Using TensorFlow backend.
2024-01-09 14:05:26,797 - __main__ - INFO - Clasificando documento Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pd
f
2024-01-09 14:05:26,799 - __main__ - INFO - ------ DETECTANDO TABLAS DE IMAGEN ------
2024-01-09 14:05:26,801 - __main__ - INFO - Transformando a png Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pdf
Guardando imagen en: C:\Users\daniel.garcia-rodrig\Desktop\Salida_doc_ocr_classification\tmpmvsulcwh\acuerdo0.png
2024-01-09 14:05:27,426 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Preprocesando imagen...
Confidence:  10.65
Angulo Tesseract:  0.0
2024-01-09 14:05:29,035 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Realizando binarización de Sauvola...
2024-01-09 14:05:29,395 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Filtrando puntos pequeños...
2024-01-09 14:05:34,260 - tableOCR.TableDetection.lineas - INFO - Detectando líneas con HoughLinesP...
2024-01-09 14:05:34,328 - tableOCR.TableDetection.lineas - INFO - Calculando intersecciones entre segmentos...
2024-01-09 14:05:34,427 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:34,427 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:34,427 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:34,427 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
Guardando imagen en: C:\Users\daniel.garcia-rodrig\Desktop\Salida_doc_ocr_classification\tmpmvsulcwh\acuerdo1.png
```

Figura 4.3 Traza de la ejecución nº1

```
Guardando imagen en: C:\Users\daniel.garcia-rodrig\Desktop\Salida_doc_ocr_classification\tmpmvsulcwh\acuerdo1.png
2024-01-09 14:05:34,593 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Preprocesando imagen...
Confidence:  10.93
Angulo Tesseract:  0.0
2024-01-09 14:05:36,043 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Realizando binarización de Sauvola...
2024-01-09 14:05:36,411 - tableOCR.ImagePreprocessing.noiseDeleting - INFO - Filtrando puntos pequeños...
2024-01-09 14:05:38,228 - tableOCR.TableDetection.lineas - INFO - Detectando líneas con HoughLinesP...
2024-01-09 14:05:38,264 - tableOCR.TableDetection.lineas - INFO - Calculando intersecciones entre segmentos...
2024-01-09 14:05:38,344 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:38,344 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:38,344 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:38,344 - tableOCR.TableDetection.lineas - INFO - Agrupando puntos por distancias...
2024-01-09 14:05:38,363 - __main__ - INFO - ------ CLASIFICANDO IMAGEN ------
2024-01-09 14:05:38.393143: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic lib
rary nvcuda.dll
2024-01-09 14:05:38.406036: E tensorflow/stream_executor/cuda/cuda_driver.cc:314] failed call to cuInit: CUDA_ERROR_UNKNOWN
: unknown error
2024-01-09 14:05:38.427012: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic informati
on for host: IBLAESMADC03797
2024-01-09 14:05:38.436612: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: IBLAESMADC03797
2024-01-09 14:05:38.442340: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with o
neAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-01-09 14:05:38.472657: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x19d59b25870 initialized for pla
tform Host (this does not guarantee that XLA will be used). Devices:
2024-01-09 14:05:38.483898: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Ve
rsion
Creando fichero txt de Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pdf
Leyendo documento C:\Users\daniel.garcia-rodrig\Desktop\tmpj3v1umwx\acuerdo.txt
********** CONJUNTO DE TEST **********
(1, 1003)
**********
Backend TkAgg is interactive backend. Turning interactive mode on.
```

Figura 4.4 Traza de la ejecución nº2

```
2024-01-10 12:37:16.227679: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: IBLAESMADC03797
2024-01-10 12:37:16.236545: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with
oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-01-10 12:37:16.281649: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x242861e2e90 initialized for pl
atform Host (this does not guarantee that XLA will be used). Devices:
2024-01-10 12:37:16.296898: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default V
ersion
Creando fichero txt de Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pdf
Leyendo documento C:\Users\daniel.garcia-rodrig\Desktop\tmp_g6zznkl\acuerdo.txt
********************* CONJUNTO DE TEST *************************
(1, 1003)
****************************************************************
El documento Doc_ocr_classification\ClasificarPDF\ejemplos\acuerdo.pdf pertenece a la clase 2
2024-01-10 12:37:40,874 - __main__ - INFO - El código de ejecución es: 2 (acuerdo)
Backend TkAgg is interactive backend. Turning interactive mode on.
```

Figura 4.5 Traza de la ejecución nº3

Tras realizar todos los algoritmos mencionados en los capítulos anteriores, se observa en la Figura 4.5 como la traza nos devuelve la predicción del clasificador evaluando en este caso correctamente que el documento se trata de un acuerdo.

ACUERDO DE AMPLIACIÓN DE PLAZOS

Expediente: <ispactag entity='SPAC_EXPEDIENTES' property='NUMEXP' />
Interesado: <ispactag entity='SPAC_EXPEDIENTES' property='IDENTIDADTITULAR' />
Asunto: <ispactag entity='SPAC_EXPEDIENTES' property='NOMBREPROCEDIMIENTO' />
Fecha de iniciación: <ispactag entity='SPAC_EXPEDIENTES' property='FAPERTURA' />

En el procedimiento de tipo referente a <ispactag entity='SPAC_EXPEDIENTES' property='ASUNTO' />.

Este/a órgano en el ejercicio de las competencias que le atribuye disposición que atribuye la competencia (número/fecha/denominación y BOE) y con base en el *artículo 32 de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas, (Boletín Oficial del Estado (BOE) nº 236 de 2 de octubre de 2015).*

ACUERDA

Primero Abrir un periodo de información pública por un plazo de x días, a contar desde su publicación en el Boletín Oficial del Principado de Asturias con el fin de que cualquier persona física o jurídica pueda examinar el procedimiento y presentar las alegaciones que estime oportunas.

Segundo El procedimiento podrá examinarse en lugar de exhibición dentro del siguiente horario: indicarlo.

Contra este acuerdo no cabe interponer ningún recurso, aunque los interesados, conforme a lo previsto en el artículo 112.1 de la *Ley 39/2015, de 1 de octubre, del*

Figura 4.6 Primera página del documento cambiada de pdf a png

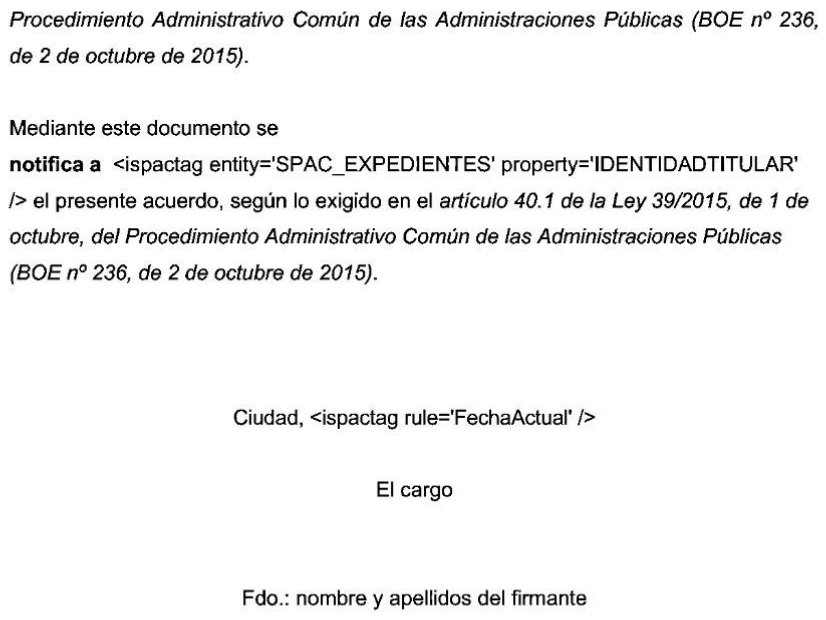


Figura 4.7 Segunda página del documento cambiada de pdf a png

En las Figura 4.6 y Figura 4.7 podemos ver los ficheros temporales de las imágenes preprocesadas antes durante la ejecución del programa que se eliminarán una vez este termine. Sobre ellas se aplicará el OCR y extracción de elementos trascendentes como se puede ver en el siguiente texto extraído del .txt generado por el programa de nombre “acuerdo.txt”.

num_palabras:256

num_paginas:2

densidad:128.0

densidad_tablas:0.0

texto: ACUERDO DE AMPLIACION DE PLAZOS

Expediente: <ispactag entity="SPAC_EXPEDIENTES' property="NUMEXP' />

Interesado: <ispactag entity="SPAC_EXPEDIENTES' property="IDENTIDADTITULAR'

[>

Asunto: <ispactag entity="SPAC_EXPEDIENTES'

property="NOMBREPROCEDIMIENTO'! />

Fecha de iniciacion: <ispactag entity="SPAC_EXPEDIENTES'

property="FAPERTURA' />

En el procedimiento de tipo referente a <ispactag entity="SPAC_EXPEDIENTES'

property="ASUNTO' />.

Este/a organo en el ejercicio de las competencias que le atribuye disposicidn que

atribuye la competencia (numero/fecha/denominaci6n y BOE) y con base en el articulo

32 de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Comun de las

Administraciones Publicas, (Boletin Oficial del Estado (BOE) n° 236 de 2 de octubre de

2015).

ACUERDA

Primero Abrir un periodo de informacién publica por un plazo de x dias, a contar desde su publicacién en el Boletin Oficial del Principado de Asturias con

el fin de que cualquier persona fisica o juridica pueda examinar el

procedimiento y presentar las alegaciones que estime oportunas.

Segundo El procedimiento podra examinarse en lugar de exhibicidn dentro del siguiente horario: indicarlo.

Contra este acuerdo no cabe interponer ningun recurso, aunque los interesados, conforme a lo previsto en el articulo 112.1 de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Comun de las Administraciones Publicas (BOE n° 236,

de 2 de octubre de 2015).

Mediante este documento se

notifica a <ispactag entity="SPAC_EXPEDIENTES' property="IDENTIDADTITULAR'

/> el presente acuerdo, segun lo exigido en el articulo 40.1 de la Ley 39/2015, de 1 de

octubre, del Procedimiento Administrativo Comun de las Administraciones Publicas

(BOE n° 236, de 2 de octubre de 2015).

Ciudad, <ispactag rule='FechaActual' />

El cargo

Fdo.: nombre y apellidos del firmante

4.2 Evaluación y discusión del modelo

El análisis y la evaluación exhaustiva de cualquier sistema son pilares fundamentales para su mejora y evolución. Esta sección, se adentra en la discusión del modelo actual, explorando sus puntos fuertes, desafíos y áreas de mejora. Se analiza su eficiencia y precisión, cada faceta contribuye a trazar el camino hacia un sistema más refinado y efectivo. Esta evaluación crítica, además, ofrece una perspectiva esencial para futuras iteraciones y para alinear el modelo con las expectativas del cliente.

4.2.1 Evaluación del modelo propuesto

A pesar del notable desarrollo del clasificador, el ejercicio práctico presentado revela que el tiempo de clasificación es de 32 segundos, lo que indica la

necesidad de una optimización significativa en términos de velocidad. Además, se llevó a cabo una prueba de precisión con 30 documentos, de los cuales solo se categorizaron correctamente 19 (63,3%). Estos resultados subrayan la urgencia de mejorar tanto la eficiencia temporal como la precisión del clasificador. Ambos aspectos serán abordados en futuras actualizaciones, donde se plantearán y explorarán algoritmos de retroalimentación y diversas técnicas de aprendizaje con el objetivo de optimizar el proceso y mejorar la exactitud de las clasificaciones.

5 Resultados y conclusiones

Este proyecto ha representado un viaje significativo en mi trayectoria profesional, marcado por la inmersión en el campo de la Inteligencia Artificial y el procesamiento de documentos. Desde la exploración de herramientas clave como Python y sus librerías hasta el desafío de realizar un proyecto funcional de IA, cada paso ha supuesto una experiencia formativa enriquecedora. En esta sección, se resumen los logros alcanzados, se identifican los desafíos enfrentados y se reflexiona sobre las contribuciones realizadas a este campo en constante evolución.

5.1 Conclusiones

En el desarrollo de este proyecto se han alcanzado notables metas relacionadas con la consecución de los siguientes objetivos clave:

Conclusión principal:

La consecución exitosa de los objetivos delineados en el capítulo 1 respalda firmemente el haber logrado el objetivo principal: el desarrollo de un nuevo modelo para clasificar documentos mediante técnicas de inteligencia artificial. El diseño integral de la aplicación, la implementación eficaz en Python utilizando las tecnologías pertinentes, la realización de pruebas exhaustivas y la identificación proactiva de áreas de mejora son evidencias tangibles de un enfoque metodológico y técnico sólido. Además de cumplir el objetivo principal, se desprenden una serie de conclusiones secundarias:

Diseño de la Aplicación a Nivel Global:

Se ha establecido una estructura eficaz para el clasificador de documentos, donde todos los módulos y scripts se integran de manera cohesionada. Este diseño permite que el clasificador funcione de manera fluida al ejecutar un solo script. Desde la conversión de archivos PDF a formato PNG hasta el preprocesamiento de imágenes y la creación de un complejo abecedario mediante tokenizado, se ha logrado consolidar un modelo integral capaz de identificar con confianza la tipología de los documentos proporcionados.

Implementación de la Estructura en Python:

Se ha logrado la implementación efectiva del diseño propuesto en Python utilizando las tecnologías explicadas en los capítulos anteriores del1 trabajo, garantizando una coherencia entre la propuesta teórica y la aplicación práctica.

Realización de Pruebas y Test:

A pesar de las posibles mejoras evidentes tanto en eficiencia como en precisión del sistema, las pruebas concluyen en que el clasificador está funcionando en su primera versión. Es eficaz y cumple con los propósitos establecidos.

5.2 Logros personales

El desarrollo de este proyecto ha sido una travesía de aprendizaje y logros, potenciando habilidades esenciales y consolidando experiencias valiosas en el campo tecnológico.

Dominio de Python y Herramientas Clave de IA:

Haber abordado el lenguaje Python, junto con sus librerías esenciales como NumPy, SciPy, TensorFlow y Keras, representa un hito importante en mi desarrollo profesional. Estas herramientas son la columna vertebral del campo

de la Inteligencia Artificial y su dominio no solo abre puertas, sino que también brinda una comprensión más profunda sobre la construcción de modelos, el procesamiento de datos y la implementación de algoritmos de aprendizaje automático. Esta inmersión en un lenguaje de programación en auge representa una inversión significativa en mi futuro en el sector.

Iniciación laboral y experiencia en proyectos a escala empresarial:

Este proyecto marca mi debut laboral contribuyendo a una empresa en la implementación de soluciones tecnológicas. Trabajar en un proyecto de esta envergadura me ha brindado una visión realista de las expectativas y desafíos que conlleva el desarrollo de soluciones tecnológicas. La responsabilidad individual asumida en este proyecto ha sido fundamental para mi crecimiento profesional.

Gestión efectiva de proyectos:

La ejecución exitosa de este proyecto se ha visto respaldada por la adopción de metodologías ágiles, tales como el uso de calendarios estructurados con sprints y que, a pesar de los inevitables contratiempos, haber logrado finalizar el proyecto en el tiempo estipulado ha propiciado el crecimiento de mi gestión personal de eficiencia de proyectos y cumplimiento de los plazos.

Desarrollo de habilidades de colaboración y trabajo en equipo:

Este proyecto ha sido una plataforma para aprender a colaborar y trabajar en equipo. La capacidad para solicitar ayuda y aprovechar los conocimientos de los compañeros con más experiencia en la materia ha sido fundamental. Estas interacciones han contribuido a fortalecer mis habilidades de colaboración y resolución de problemas en un entorno de equipo.

Agilidad en la recopilación y validación de información:

La experiencia adquirida en la búsqueda, selección y validación de información de diversas fuentes ha sido significativa. Esta habilidad mejorada en la identificación de fuentes confiables y la agilidad en la búsqueda de recursos pertinentes ha optimizado el proceso de recopilación de datos, mejorando la eficiencia y la precisión en la obtención de información relevante para el proyecto.

5.3 Futuras ampliaciones del clasificador

Actualmente, colaboro con un compañero de mi departamento en Inetum en la integración del software en la aplicación web de ISicres (producto de mi departamento). Este proyecto implica no solo la incorporación del nuevo modelo desarrollado, sino también trabajar otros aspectos de la programación relacionados con el servidor web, las bases de datos y la utilización de herramientas como Maven. Dado que mi incorporación al equipo es reciente, estoy inmerso en el proceso de familiarización con la vasta cantidad de código que compone la aplicación de ISicres.

La colaboración con mi compañero ha sido fundamental, ya que, junto a su orientación y apoyo, hemos logrado realizar avances significativos, tanto que la pestaña que albergará la funcionalidad del nuevo modelo ha sido creada recientemente a la espera de la implementación del mismo. Este logro marcará un hito en el proyecto, ya que se podrá visualizar la interfaz gráfica de la aplicación con la pestaña de clasificación abierta. La Figura 5.1 muestra el progreso de la pestaña mencionada.

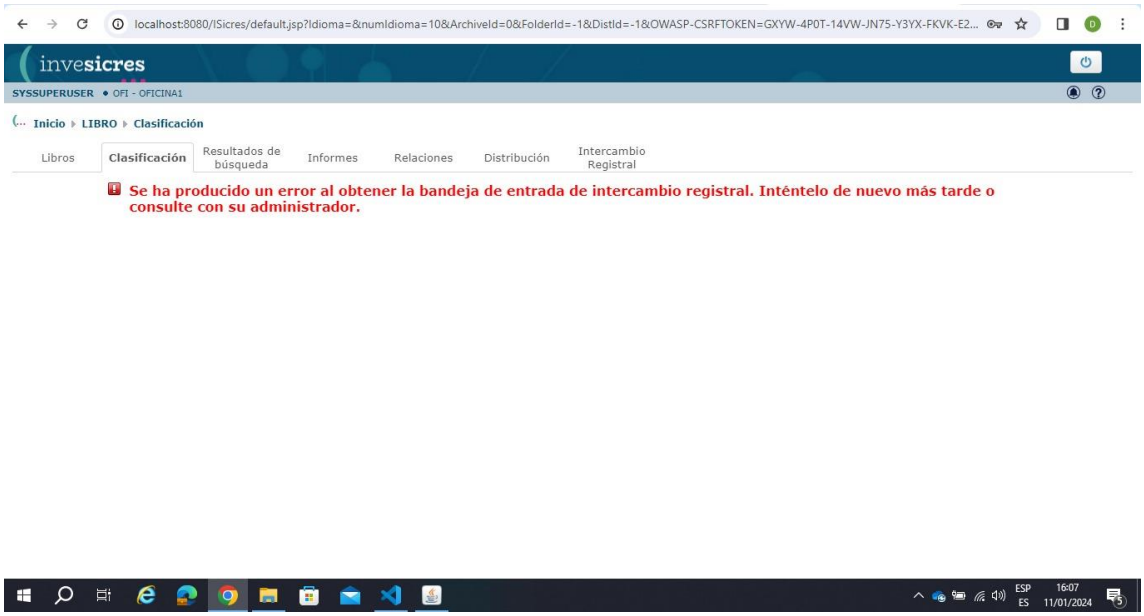


Figura 5.1 Progreso de la integración de la aplicación

Cuando se termine la integración, me gustaría proponer para sus primeras clasificaciones algún sistema de retroalimentación, donde si se efectúa mal una clasificación la persona que lo utilice pueda rectificar a mano su predicción y esto le sirva al programa para aprender de sus errores.

Otra futura implementación debe ser ir añadiendo a medida que dispongamos de los recursos más datos de entrenamiento con los otros tipos documentales que recoge el ENI para que a medida que pase el tiempo se convierta en un clasificador asociado a este marco en su totalidad.

Como último punto, comentar otro módulo extra que me gustaría añadir en el futuro al clasificador, y consiste en un módulo que, mediante técnicas de Inteligencia Artificial, sea capaz de clasificar documentos compuestos en lote, es decir, si al clasificador le pasas un pdf que no está formado por un único documento, sea capaz de identificar los diferentes documentos, separarlos y clasificarlos individualmente.

6 Análisis de Impacto

En este capítulo se analiza el impacto que puede tener un clasificador de documentos desarrollado con técnicas de aprendizaje automático en diferentes áreas de la vida.

Impacto empresarial:

Para las empresas, la implementación de un clasificador de documentos implica una optimización significativa de los procesos internos. La capacidad de categorizar automáticamente documentos como contratos o acuerdos mejora la eficiencia operativa y reduce la carga de trabajo manual.

Impacto social:

En entornos sociales, la capacidad de acceder rápidamente a información legal contenida en documentos como informes o convenios puede ser crucial. El clasificador facilita este acceso, promoviendo la transparencia y el conocimiento.

Impacto económico:

La implementación del clasificador puede traducirse en un ahorro significativo de tiempo y recursos en empresas y organizaciones al automatizar la clasificación y búsqueda de documentos. Esto podría tener un impacto positivo en la eficiencia operativa y reducir los costos asociados con la gestión documental.

La capacidad de clasificar documentos de manera efectiva puede abrir oportunidades para ofrecer nuevos servicios relacionados con la gestión documental a nivel empresarial, generando posibles ingresos adicionales.

Impacto medioambiental:

Al facilitar la gestión digital de documentos, el clasificador puede contribuir a la reducción del uso de papel en entornos empresariales y administrativos, teniendo así un impacto positivo en la conservación del medio ambiente.

Impacto cultural:

En el contexto cultural, la capacidad de clasificar y preservar documentos históricos o culturalmente relevantes puede ser esencial. El clasificador puede contribuir a la conservación de estos documentos, asegurando su disponibilidad para generaciones futuras.

Se puede relacionar el impacto generado en ciertas áreas recién mencionadas con algunos de los Objetivos de Desarrollo Sostenible (ODS), de la Agenda 2030 [8]:

Como se menciona en el impacto medioambiental, la reducción del uso del papel puede suponer una reducción de las emisiones de gases de efecto invernadero.

“Si se utiliza papel reciclado, además de reducir la presión sobre los bosques, se consigue un ahorro sustancial de energía (27%) y se reducen las emisiones de gases de efecto invernadero (74%)” [10], la razón más evidente es que los procesos de tala y refinamiento del papel generan gases de efecto invernadero como dióxido de carbono y metano.

Los ODS recogen como como meta 13.1 Fortalecer la resiliencia y la capacidad de adaptación a los riesgos relacionados con el clima y los desastres naturales en todos los países por lo que reducir las emisiones de gases de efecto invernadero supondría un impacto directo hacia esta meta.

La meta 8.2 busca “Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra” y el clasificador documental podría liberar carga de trabajo para favorecer ese uso intensivo en la mano de obra.

7 Bibliografía

[1] D. Rubier, *Evolución histórica de la clasificación documental*. 2011

[2] ENI, *PAe - Esquema Nacional de Interoperabilidad - ENI (administracionelectronica.gob.es)*

[3] Sealpath, *Ventajas de Clasificación de Datos con IA y Machine Learning (sealpath.com)*, octubre 2022.

[4] ¿Qué son las redes neuronales), *¿Qué son las redes neuronales? | IBM*

[5] R. Smith. "An overview of the Tesseract OCR engine." *Ninth international conference on document analysis and recognition (ICDAR 2007)*. Vol. 2. IEEE, 2007.

[6] P, Joshi. *Artificial intelligence with python*. Packt Publishing Ltd, 2017.

[7] A. Gulli and P. Sujit. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[8] M. A. Abu. A study on Image Classification based on Deep Learning and Tensorflow. *International Journal of Engineering Research and Technology*, 2019, vol. 12, no 4, p. 563-569. 2019

[9] ODS, *Cambio climático - Desarrollo Sostenible (un.org)*

[10] 20Minutos, *Ayuda al medio ambiente: gestos para reducir tu huella de carbono (20minutos.es)*

8 Anexos

TFG Daniel García Rodríguez

INFORME DE ORIGINALIDAD

14%	12%	3%	9%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	oa.upm.es Fuente de Internet	2%
2	github.com Fuente de Internet	1%
3	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	1%
4	mafiadoc.com Fuente de Internet	1%
5	medaenvidiatucoche.com Fuente de Internet	<1%
6	Submitted to Corporación Universitaria Minuto de Dios, UNIMINUTO Trabajo del estudiante	<1%
7	Submitted to Escuela Politecnica Nacional Trabajo del estudiante	<1%
8	Submitted to Universidad Politécnica de Madrid Trabajo del estudiante	<1%

Figura 8.1 Informe de originalidad creado por la herramienta Turnitin

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jan 15 13:51:06 CET 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)