Universidades de Burgos, León y Valladolid

Máster Universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Implementación de una herramienta basada en PLN para la detección y anonimización de datos personales en documentos

Presentado por José Manuel Simón Ramos
en Universidad de Valladolid
2 de septiembre de 2021

Tutores: Aníbal Bregón Bregón
Jorge Silvestre Vilches

Resumen

En los últimos años, el avance en el campo del Aprendizaje Automático, unido a las mejoras del hardware, y al aumento del volumen de los datos, ha motivado la utilización de técnicas de aprendizaje que empleen estos datos para automatizar procesos o extraer conocimiento a partir de los mismos. Desde el punto de vista del campo del Procesamiento del Lenguaje Natural (PLN), la utilización de los datos para generar nuevos modelos se encuentra afectada debido a la existencia de información de carácter personal en los mismos. Esto, unido a la fuerte legislación vigente sobre la Protección de Datos, hace que las administraciones y organizaciones deban tener una mayor precaución y control a la hora de utilizar o compartir documentos en los que se aparezca información personal.

El presente Trabajo Fin de Máster aborda la problemática de la detección y anonimización de entidades personales existentes en documentos administrativos (permisos, inspecciones, convenios, etc). En la línea con lo anterior, el proyecto plantea una propuesta genérica y eficiente de *pipeline* enfocada a la anonimización y generación de reemplazos para las entidades detectadas. Esta propuesta busca no solo poder ser empleada para detectar y anonimizar entidades en este tipo de documentos, sino que pretende ser una solución genérica para abordar la problemática de la detección y anonimización de entidades en cualquier tipo de documentos.

Descriptores

Aprendizaje Automático, Procesamiento del Lenguaje Natural, Anonimización, Python, Spacy, OCR.

Abstract

In recent years, progress in the area of Machine Learning, together with hardware improvements, and the increase in the volume of data, has motivated the use of learning techniques that use this data to automate processes or extract knowledge. From the point of view of Natural Language Processing (NLP), the use of data to generate new models is affected by the existence of personal information in them. This, combined with the strong legislation in force on Data Protection, means that administrations and organizations must be more cautious and have greater control when using or sharing documents which personal information appears.

This Master Thesis addresses the problem of detection and anonymization of personal entities in administrative documents (permits, inspections, agreements, etc.). In addition, the project presents a generic and efficient proposal of *pipeline* focused on the anonymization and generation of replacements for the detected entities. This proposal aims not only to be used to detect and anonymize entities in this type of documents, but also to be a generic solution to address the problem of detecting and anonymizing entities in any type of documents.

Keywords

Machine Learning, Natural Language Processing, Anonymization, Python, Spacy, OCR.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Código

Parte I

Introducción y Contexto del Proyecto

Capítulo 1

Introducción

En los últimos años, el interés y la popularidad de la Inteligencia Artificial ha sufrido un gran crecimiento, y se espera que este siga aumentando. Según el *"Artificial Intelligence Index Report 2021"* [38], elaborado por la Universidad de Stanford, las revisiones de artículos científicos, cuyo tema principal es la Inteligencia Artificial, ha pasado de ser de un 1,3 % del total de los artículos científicos en el año 2012, a un 3,8 % en el año 2019.

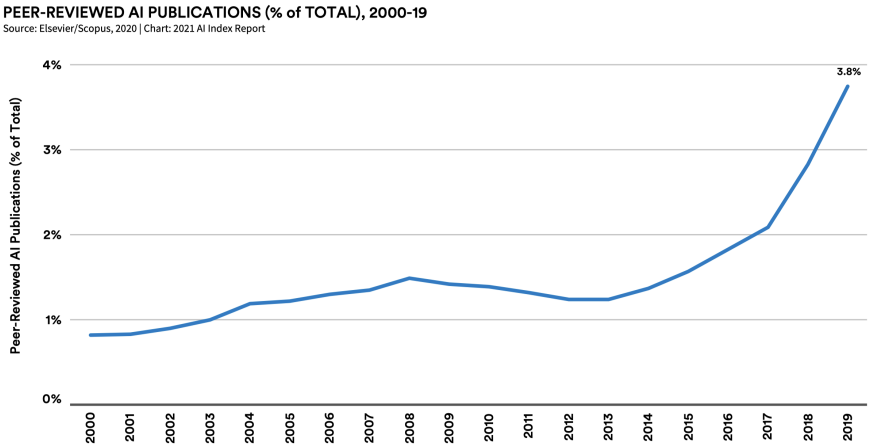


Figura 1.1: Evolución de las revisiones de artículos de Inteligencia Artificial. Fuente: [38].

La digitalización y la generación masiva de datos procedentes de diversas fuentes (IoT, sensores, *wearable*, etc), unido al avance en la potencia del hardware de los equipos, han sido claves para impulsar esta rama de la Informática, cuyas expectativas de futuro son alentadoras. Gracias a ambos factores, actualmente es posible acceder de forma libre y gratuita a grandes colecciones de datos (a partir de ahora *datasets*) de gran calidad. Esto permite tanto estudiar el desempeño de las técnicas de Aprendizaje Automático actuales

para un determinado caso de uso, como poder investigar nuevas soluciones que permitan afrontar una determinada problemática. Sin embargo, existe un problema (desde el punto de vista de la Inteligencia Artificial) asociado a los datos: **la privacidad**. Si bien es cierto que existe una gran cantidad de *datasets* disponibles de forma abierta, existen muchos datos que, debido a las políticas de privacidad que tienen las empresas o los países, no pueden hacerse públicos, limitando la aparición de nuevas propuestas o el enriquecimiento de las existentes.

Desde la perspectiva del Procesamiento del Lenguaje Natural, PLN (*Natural Language Procesing*, NLP, en inglés), esta problemática es más notoria, ya que muchas de las aplicaciones que tienen las técnicas de PLN involucran documentos o textos en los que aparece información de carácter personal (limitando así su uso y difusión). Algunos ejemplos de estas técnicas son la extracción automática de datos en textos estructurados (por ejemplo formularios), la generación de texto (creación de *chatbots* para interactuar con una persona), o la identificación de menciones en documentos (obteniendo así menciones de un tipo concreto). En términos jurídicos, la utilización o difusión de documentos en cuyo contenido se encuentre información de carácter personal se encuentra enmarcada dentro del Reglamento Europeo de Protección de Datos[1], también conocido como Reglamento General de Protección de Datos (RGPD). Este reglamento fue realizado por el Parlamento Europeo en combinación con el Consejo de la Unión Europea, siendo finalmente aprobado en abril de 2016 (aunque su aplicación no se realizó hasta mayo de 2018). El reglamento recoge como datos de carácter personal toda aquella información asociada a una persona identificada (cuya identidad es directamente conocida), o identificable (cuya entidad no se conoce directamente pero se puede obtener cruzando otras fuentes de datos). Algunos ejemplos de datos personales más importantes son: 1) Nombre y Apellido; 2) Dirección; 3) Números de Identificación (DNI, Pasaporte, etc); 4) Ingresos; 5) Direcciones IP; o 6) Datos clínicos (identificadores y datos personales de un individuo dentro del ámbito sanitario).

Asimismo, desde el punto de vista de la administración pública española (sector en el que se encuentra enmarcado el presente proyecto), la detección y anonimización de las menciones personales en los documentos no se encuentra dentro de su flujo de trabajo habitual (por lo que a priori no es algo que afecte directamente a su funcionamiento). Sin embargo, cada vez es más habitual la divulgación de datos públicos por parte de las administraciones, fomentando así el desarrollo de nuevas propuestas que permitan extraer valor a partir de esos datos, o mejorar algún área concreta con la implementación de nuevas herramientas o sistemas. Algunos ayuntamientos como el de Segovia, el de

[1] https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_es *(Visitado el 29-06-2021).*

Cáceres, el de Madrid, o el de Valladolid, disponen de portales web específicos para poder acceder a una gran variedad de datos de la ciudad[2,3] (puntos de interés turísticos, restaurantes, hoteles, instalaciones, etc).

Dentro de la publicación de los datos existen dos retos principales, los cuáles serán abordados en el presente proyecto:

- Necesidad de llevar a cabo un tratamiento sobre los documentos antes de su publicación. Este tratamiento estaría destinado a satisfacer las restricciones sobre los datos personales impuestas por el RGPD (no difusión y/o utilización de los datos personales para otros fines que aquellos para los que se ha autorizado su obtención y uso de manera explícita).

- La automatización del proceso de detección y anonimización de los datos personales en los documentos. A pesar de que los datos a publicar se pueden anonimizar de forma manual, el gran volumen de información que ha de procesarse, unido a la aparición de errores a nivel humano, hace que la inversión de tiempo y de recursos humanos a utilizar sea inviable para realizar esta labor de forma manual.

Por otro lado, atendiendo a la aplicación y contexto en el que se encuentra enmarcado el proyecto, este se sitúa dentro de las tareas de Reconocimiento de Entidades Nombradas (*Named Entity Recognition*, NER, en inglés), de textos escritos en castellano. Estas tareas se encuentran orientadas al reconocimiento y extracción de información en textos estructurados (Ver Apartado 3.2). Sin embargo, el caso de uso a resolver va más allá del propio reconocimiento de entidades personales, ya que aborda tareas de Reconocimiento Óptico de Caractéres (OCR) (Ver Apartado 3.5) para extraer el texto del que se quieren obtener las entidades personales, anonimización de las menciones utilizando mecanismos de reemplazo eficientes (Ver Apartado 8), y tareas de creación y manipulación de documentos PDF con el contenido del texto anonimizado (Ver Apartado 5).

Es por ello por lo que propuestas como las presentadas en el presente proyecto son de gran utilidad a la hora de detectar, extraer y anonimizar este tipo de menciones en documentos administrativos semi-estructurados o desestructurados, permitiendo un mejor manejo de los datos, así como un procesamiento de forma automática de los mismos, disminuyendo la cantidad de errores a nivel humano, y aumentando la velocidad de procesamiento de los documentos.

[2]http://opendata.ayto-caceres.es *(Visitado el 14-07-2021).*

[3]https://www.valladolid.es/es/temas/hacemos/open-data-datos-abiertos/catalogo-datos *(Visitado el 14-07-2021).*

En definitiva, se puede concluir con que el PLN es una de las ramas del Aprendizaje Automático más importantes en la actualidad. Sin embargo, la sensibilidad de los documentos necesarios para crear, validar y mejorar estos sistemas (debido a su contenido de carácter personal) hace que su uso o difusión se vean más restringidos que otro tipos de sistemas sin estas limitaciones.

1.1. Motivación

Uno de los principales problemas que tienen las propuestas de Aprendizaje Automático enfocadas al Procesamiento de Lenguaje Natural viene dado por la privacidad de los datos necesarios para su creación, validación y mejora. La pérdida o filtración de datos personales pueden suponer grandes problemas para la empresa, administración o responsable afectados, suponiendo graves infracciones y multas debido a estos hechos. La implantación de mecanismos de Aprendizaje Automático que permitan detectar todos estos fragmentos de información permite tener un mayor control y gestión sobre estos. Además, esto permite realizar distintas acciones interesantes sobre los documentos, como extraer esas menciones para analizarlas, e incluso censurarlas o eliminarlas del documento, evitando así que se pueda relacionar con un individuo (anonimizar el documento).

Este trabajo tiene los siguientes propósitos: 1) Detección y extracción de datos de naturaleza personal (menciones a personas), dentro de textos de carácter administrativo; y 2) Obtención de los textos de los documentos mediante el uso de OCR. Esto se debe a que un gran número de documentos se encuentran escaneados (por lo que no se puede obtener directamente el contenido del texto que aparece en él). Actualmente, el organismo público del que proceden los datos automatiza este proceso, por lo que cualquier avance o conclusión relativa a automatización será de gran utilidad para futuros proyectos. Por otra parte, unido a la automatización de la extracción de menciones personales, se va a abordar una de las aplicaciones que tienen este tipo de técnicas, la **anonimización de los datos**. En relación con esto, se desarrollará un sistema híbrido de anonimización **genérico** que permita sustituir las menciones personales detectadas por otras del mismo tipo, manteniendo la concordancia entre las mismas. Todo esto, en combinación con el manejo de documentos PDF permitirá generar un nuevo documento cuyas menciones del documento original estarán anonimizadas, y que será prácticamente indistinguible de un documento verdadero.

Para ello se llevará a cabo el análisis y optimización de modelos y técnicas orientadas a la detección de entidades en textos, así como el estudio e implementación de diversas técnicas enfocadas a la generación de reemplazos. La propuesta presentada permitirá generar nuevas entidades para anonimizar un documento de una forma rápida y eficiente. Esta implementación se realizará

en forma de *pipeline* de componentes aislados, en el que cada componente realizará una acción concreta dentro del flujo. Además de esto, el desarrollo de este *pipeline* se hará de la forma más flexible posible. De este modo podrá ser utilizado para otro tipo de documentos, otro tipo de entidades diferentes a las propuestas, u otro contexto de aplicación. Todo ello sin la necesidad de realizar modificaciones significativas en el flujo de trabajo presentado.

1.2. Objetivos

El presente Trabajo Fin de Máster afronta la problemática de detección y anonimización de entidades personales (nombres, apellidos, direcciones, matrículas de coche, etc) en textos de carácter administrativo (pagos al ayuntamiento, multas municipales, etc), así como su reemplazo por otras entidades del mismo tipo. Con todo esto, se generará un nuevo documento con valores realistas, pero sin ser identificativos de ninguna persona real. En esta línea de trabajo se identifican los siguientes objetivos y subojetivos:

- OBJ-1: Obtención del corpus documental necesario para generar el modelo.
  - OBJ-1.1: Realización de un análisis exploratorio inicial de las entidades a detectar y de los documentos.
  - OBJ-1.2: Anotación de las entidades personales que aparecen en los documentos.
- OBJ-2: Creación del *pipeline* para la detección de entidades y su anonimización.
  - OBJ-2.1: Comparativa entre los principales modelos de PLN orientados a la detección y extracción de entidades nombradas.
  - OBJ-2.2: Adaptación de técnicas de Reconocimiento Óptico de Caracteres (OCR) para la extracción del texto en documentos PDF.
  - OBJ-2.3: Implementación e incorporación de componentes adicionales al *pipeline* que complementen la detección de entidades del modelo.
  - OBJ-2.4: Desarrollo de mecanismos de reemplazo genéricos y eficientes para las entidades detectadas.
- OBJ-3: Evaluación del *pipeline* de anonimización.
- OBJ-4: Diseño e implementación de una herramienta orientada al usuario que permita anonimizar documentos a utilizando el *pipeline* desarrollado.

- OBJ-5: Realización de la documentación técnica del proyecto.

Con la finalización de los objetivos descritos anteriormente se pretende obtener una herramienta software (*OBJ-4*) que permita, a través de un documento dado, detectar las entidades de carácter personal que este contenga y generar un nuevo documento. Este nuevo documento mantendrá el contenido del documento original, a excepción de las entidades detectadas, las cuales habrán sido sustituidas por otras del mismo tipo durante su paso por el *pipeline* desarrollado (*OBJ-2*). Al mismo tiempo, se generará una memoria técnica (*OBJ-5*) en la que se plasmará todo el proceso de creación del modelo, la implementación de los mecanismos de reemplazo, evaluación de la herramienta, análisis software, etc, así como cualquier otra cuestión de interés relativa a su desarrollo.

1.3. Herramientas Utilizadas

Para llevar a cabo la realización del proyecto se han utilizado las siguientes herramientas:

- **Datasaur:** Herramienta web diseñada para el etiquetado y exportación de datos de texto para diferentes propósitos en proyectos PLN. En el caso del proyecto, esta herramienta se ha empleado para etiquetar los distintos tipos de entidad que aparecen en los documentos procesados. No obstante, dispone de funcionalidad para realizar otro tipo de tareas como el etiquetado de dependencias entre palabras, el etiquetado de co-referencias, etc.
- **GitLab:** GitLab es un servicio web gratuito de código abierto enfocado a la gestión y control de versiones mediante Git. Este servicio fue lanzado en octubre de 2011, y además de la gestión de versiones con git permite la creación de grupos tanto a nivel de equipo como a nivel de repositorios, y la generación de wikis para cada uno de los repositorios.
- **Jupyter Notebook:** Jupyter Notebook es un proyecto de código abierto que fue creado en 2014 a partir del kernel IPython. Jupyter Notebook se encuentra implementada como una aplicación cliente-servidor que permite generar documentos web siguiendo un esquema de celdas ordenadas con entradas y salidas. Las celdas de los ficheros de Jupyter Notebook soportan lenguajes de programación como Python o R, así como lenguajes de marcado de texto como Markdown o Latex.
- **Visual Studio Code:** Editor de texto desarrollado por Microsoft y lanzado de forma oficial en al año 2016. Visual Studio Code proporciona un gran número de herramientas y extensiones que le permiten dar

soporte a la mayoría de los lenguajes de programación, así como a distintos formatos de documentos, tales como Markdown (*.md*), Notebooks (*.ipynb*), o Latex (*.tex*). Además ofrece integración directa con Git, facilitando la gestión y control de versiones.

1.4. Tecnologías Utilizadas

Las tecnologías que se han utilizado para desarrollar el proyecto han sido las siguientes:

- **Python:** Python es un lenguaje de programación multiparadigma lanzado en 1991. Actualmente, Python se encuentra administrado por la *Python Software Foundation*, bajo una licencia de código abierto. Además de esto, dispone de una gran comunidad detrás que no deja de desarrollar librerías que permiten extender su funcionalidad. Entre las muchas tareas que se pueden llevar a cabo con este lenguaje cabe destacar el desarrollo de aplicaciones web, desarrollo de juegos, análisis estadístico de datos, o creación y utilización de modelos basados en aprendizaje automático.
- **Flask:** Flask es un framework desarrollado por Armin Ronacher y escrito en Python que permite crear aplicaciones web de una forma rápida y sencilla. Se encuentra basado en la especificación WSGI *Web Server Gateway Interface*, y utiliza el motor de *templates* Jinja2, el cual permite situar distintos marcadores en las plantillas correspondientes a la página web cuyo contenido será generado y renderizado en tiempo de ejecución.
- **FPDF:** FPDF es una librería escrita en Python (aunque no es original, ya que fue desarrollada como un *port*[4] de la misma librería en PHP), que proporciona herramientas para generar documentos PDF.
- **Git:** Sistema de control de versiones que registra las modificaciones realizadas en los ficheros situados en un repositorio local. Esto permite generar distintos "puntos de control" sobre el código y mantener un registro de los distintos estados de este, pudiendo navegar entre las diferentes versiones que hayan tenido estos archivos.
- **HTML y CSS:** HTML y CSS son dos lenguajes de marcado y de diseño de texto respectivamente, que permiten crear estructuras de páginas web y darles estilo. Ambos lenguajes son considerados como unos pilares dentro del desarrollo de páginas web en el lado cliente, junto con JavaScript.
- **Levenshtein:** Python-Levenshtein es una librería desarrollada en Python que proporciona funcionalidad para obtener distintas métricas relacionadas con distancia y la similitud entre palabras.

[4]Adaptación de un programa a otra plataforma o lenguaje de programación.

- **Numpy:** Numpy es una librería de código abierto escrita en Python que proporciona una gran colección de funciones matemáticas de alto nivel para realizar operaciones sobre matrices o vectores.

- **Pandas:** Pandas es una librería de código abierto para Python que permite tanto la manipulación como el análisis de datos. Para ello utiliza estructuras de datos propias (DataFrames). Esta librería es una de las más utilizadas en la actualidad, ya que permite realizar de una forma eficiente una gran variedad de acciones sobre los datos: inserciones, eliminaciones, filtrados, agrupaciones, obtención de métricas estadísticas, etc.

- **Matplotlib y Seaborn:** Matplotlib y Seaborn son dos librerías de alto nivel desarrolladas en Python enfocadas a la creación de gráficos. La principal ventaja que ofrecen ambas es que disponen de una integración directa con Pandas, lo que permite realizar visualizaciones de una forma rápida y sencilla en base a la estructura de los datos del DataFrame de Pandas.

- **Networkx:** Networkx es una librería de Python enfocada a la creación, manipulación y estudio de grafos. Además de ofrecer la funcionalidad básica para manejar estas estructuras, proporciona un gran número de funciones y algoritmos para realizar operaciones sobre los grafos: cálculo de caminos, obtención de componentes, extracción de subgrafos, etc.

- **PDF2Image:** Librería escrita y desarrollada en Python que permite transformar documentos en formato PDF a una imagen en distintos formatos (JPEG, JPG, PNG, etc).

- **Pillow:** Pillow es una librería gratuita de código abierto escrita en Python que permite abrir, modificar y guardar archivos en muchos formatos de imagen diferentes. Entre sus características se incluye la posibilidad de manipular la imagen a nivel de pixel, modificar la transparencia, nitidez, brillo, etc, y agregar texto a las imágenes, entre otras muchas funcionalidades más.

- **Pytesseract:** Pytesseract es una librería escrita en Python orientada al reconocimiento óptico de caracteres (OCR). Pytesseract utiliza el motor OCR Tesseract desarrollado en Java, actuando como *wrapper* entre Python y Java. Además dispone de soporte para un gran número de formatos de imagen comunes: JPEG, PNG, TIFF, GIF, etc.

- **Spacy:** Spacy es una librería de código abierto desarrollada en Python que ofrece una gran cantidad de opciones y funcionalidades para desarrollar proyectos enfocados al Procesamiento del Lenguaje Natural. Además de permitir crear modelos PLN desde cero, Spacy dispone de soporte nativo para más de 64 lenguajes diferentes. Por otra parte, Spacy proporciona

también distintos *pipelines* ya preentrenadas para realizar tareas básicas de PLN en 19 lenguajes.

1.5. Organización de la Memoria

El presente documento se encuentra dividido y estructurado en capítulos en los que se abordan distintos aspectos de interés del proyecto. Los capítulos que conforman el documento, así como la temática de estos son:

Capítulo 1. Introducción: En el Capítulo 1 se llevará a cabo una introducción del contexto en el que se engloba este proyecto, así como la motivación de este, y los objetivos que se pretenden lograr tras su finalización.

Capítulo 2. Estimación y Presupuestos: En el Capítulo 2 se realiza la descripción de la metodología utilizada para realizar el proyecto. Además de esto, el capítulo aborda la estimación de tiempo y costes asociados a su desarrollo.

Capítulo 3. Contexto de Desarrollo: A lo largo del Capítulo 3 se abordará el contexto en el que se encuentre desarrollado el proyecto. Por un lado, se tratará el tema del Procesamiento del Lenguaje Natural, así como sus características y aplicaciones mas habituales. Del mismo modo, se describirán las diferentes librerías y/o herramientas orientadas a la realización de proyectos utilizando esta rama del Aprendizaje Automático. Finalmente el capítulo concluye con la metodología para la extracción de textos a partir de imágenes, así como las principales características y estados de un documento desde el punto de vista de la anonimización.

Capítulo 4. Estado del Arte: En el Capítulo 4 se llevará a cabo el estudio de anteriores propuestas relacionadas con la problemática a abordar en este proyecto.

Capítulo 5. Pipeline Desarrollado: En el Capítulo 5 se realizará un breve recorrido sobre los diferentes componentes que conforman el *pipeline* desarrollado. Además de esto se describirán las principales características de estos, así como las consideraciones que se han tomado a la hora de implementarlo.

Capítulo 6. Conjunto de Datos del Corpus: En el Capítulo 6 se realizará tanto la descripción, como el análisis de los documentos utilizados para entrenar el modelo de detección de entidades. Del mismo modo, se realizará la descripción, características y consideraciones de diseño de cada uno de los tipos de menciones a detectar.

Capítulo 7. Construcción del Modelo: A lo largo del Capítulo 7 se abordará el proceso de creación y optimización del modelo NER desarrollado para la detección de las entidades. Además de esto, también se presentarán las distintas métricas obtenidas al aplicar el modelo al conjunto de datos de prueba.

Capítulo 8. Sistema de Generación de los Reemplazos: En el Capítulo 8 se llevará a cabo el análisis y de las técnicas de generación de los reemplazos implementadas.

Capítulo 9. Herramienta Software: En el Capítulo 9 se realizará un análisis software de la herramienta que se va a desarrollar (actores, casos de uso, diseño de la herramienta, etc).

Capítulo 10. Conclusiones y Trabajo Futuro: En el Capítulo 10 se presentarán las conclusiones obtenidas a lo largo del desarrollo del proyecto, así como las posibles líneas de trabajo futuro que podrían continuarse a partir de los resultados y conclusiones obtenidos.

Apéndices: En este apartado del documento se encuentra información adicional de interés asociada al desarrollo del proyecto. El Apéndice se encuentra constituido por las siguientes secciones:

- **Apéndice A. Contenido Adjunto:** En esta sección del apéndice se encuentra reflejada la estructura de directorios asociada a la entrega, así como su explicación.
- **Apéndice B. Instalación y Versiones:** En esta sección del apéndice se explica el proceso a seguir para instalar la herramienta, además de las versiones del software y librerías utilizadas para su correcto funcionamiento.
- **Apéndice C. Manual de Usuario:** En esta sección del apéndice se lleva a cabo una explicación de las distintas funcionalidades que ofrece la herramienta.
- **Apéndice D. Configuración de la Herramienta:** En esta sección del apéndice se muestra el fichero de configuración de la herramienta junto con la explicación y valores de las distintas variables de configuración.

Capítulo 2

Estimación y Presupuestos

2.1. Estimación

Dado que el presente proyecto se encuentra enmarcado dentro de la asignatura de Trabajo Fin de Máster, su alcance y duración han de ajustarse al número de horas fijado para la asignatura (9 créditos ECTS), lo que se traduce en una carga de 225 horas.

El desarrollo del proyecto se ha estructurado en bloques de tiempo de 2 semanas de duración, permitiendo así una mejor distribución de las tareas, y estipulando un periodo de control razonable para ir comprobando si se está logrando cumplir los objetivos fijados en la planificación. Teniendo esto en cuenta, el proyecto se ha dividido en 4 bloques, comenzando el proyecto el día 7 de junio de 2021, y concluyendo el último bloque el día 1 de agosto del mismo año. Por otra parte para facilitar la especificación, estructuración y organización de las tareas, se van a definir distintas Historias de Proyecto constituidas por distintas tareas más pequeñas y sencillas.

Las Historias de Proyecto que se han definido para satisfacer los objetivos del proyecto son las siguientes:

- **H-01: Estudio del Dominio del Proyecto (*OBJ-2*).**

  Esta Historia de Proyecto tiene como objetivo la obtención de los conocimientos necesarios para abordar el tema a tratar (Procesamiento del Lenguaje Natural (PLN), Reconocimiento de Entidades Nombradas (NER), Anonimización, etc), así como el estudio de otras propuestas similares, consolidando la hoja de ruta a seguir para llevar a cabo el proyecto.

- **H-02: Creación y Normalización del Conjunto de Datos (*OBJ-1*).**

Esta Historia de Proyecto aborda toda la gestión y manejo de los datos desde su inicio (documentos PDF sin etiquetar), hasta su adaptación y división en los conjuntos de Entrenamiento, Validación y Prueba. Dicho de otro modo comprende las transformaciones llevadas a cabo en los documentos originales, para obtener los conjuntos de datos necesarios para entrenar y validar el funcionamiento del modelo NER.

- **H-03: Implementación y Optimización del modelo para la detección de las menciones (*OBJ-2*).**

  Esta Historia de Proyecto tiene como objetivo implementar y optimizar el modelo encargado de detectar las menciones personales en los documentos. Para ello se realizarán comparativas entre diferentes modelos enfocados a resolver este tipo de tareas, evaluando diferentes configuraciones en los hiperparámetros de los mismos.

- **H-04: Desarrollo del mecanismo de generación de reemplazos (*OBJ-2*).**

  Esta Historia tiene como objetivo desarrollar un mecanismo eficiente para la generación de las nuevas entidades de sustitución para anonimizar los documentos. Este mecanismo se basa en un enfoque híbrido utilizando grafos y diccionarios.

- **H-05: Integración y evaluación de rendimiento de las componentes del *pipeline* (*OBJ-3*).**

  El objetivo de esta Historia de Proyecto consiste en integrar los distintos componentes que conforman el *pipeline* de anonimización y evaluar su rendimiento con distintos tipos de nuevos documentos.

- **H-06: Desarrollo de la Herramienta software de anonimización (*OBJ-4*).**

  Esta Historia de Proyecto tiene como objetivo el desarrollo de una herramienta software que permita al usuario interactuar con el *pipeline* de forma visual e interactiva. Todo ello a través de una interfaz gráfica que pueda ser desplegada de forma local en el ordenador.

- **H-07: Desarrollo de la Memoria del Proyecto (*OBJ-5*).** El objetivo de esta Historia de Proyecto consiste en la documentación de una forma clara, concisa, y utilizando la estructura de documento proporcionada en la titulación, cada una de las distintas fases que conforman el proyecto.

Para cada una de las Historias de Proyecto descritas anteriormente, se ha realizado una división en tareas mas simples. La consecución exitosa de todas las tareas se traduce en el alcance de los objetivos fijados anteriormente (ver Apartado 1.2). Por otra parte, dado que el tamaño y dificultad de las tareas no

es exactamente el mismo, se ha asignado a cada tarea un nivel de complejidad. Cada nivel de complejidad es asociado a una tarea utilizando la técnica de la **estimación de póquer**[1] (*planning poker*, en inglés), empleando como escala la sucesión de Fibonacci[2]. Gracias a la asignación de la dificultad a cada tarea se puede hacer un reparto más equitativo de las mismas entre los distintos bloques de tiempo en los que se ha dividido el proyecto.

A continuación se presenta el desglose de las tareas que conforman cada Historia de Proyecto, así como la dificultad asociada a las mismas (ver Tablas 2.1 - 2.7):

Tarea	Descripción	Dificultad
T1.1	**Estudio de los modelos de detección de entidades nombradas en documentos en la actualidad.**	*1 Punto de Dificultad*
	Estudio inicial de las distintas propuestas que existenen la actualidad enfocadas a la creación de modelos orientados a la detección de entidades en textos.	
T1.2	**Estudio de los métodos de anonimización de menciones en textos.**	*1 Punto de Dificultad*
	Consolidación de las distintas técnicas que existen actualmente para anonimizar documentos, así como el estado que ha de tener un documento para considerarse que está anonimizado.	
T1.3	**Estudio de las diferentes herramientas y librerías para llevar a cabo el desarrollo del proyecto.**	*1 Punto de Dificultad*
	Análisis de las diferentes librerías y herramientas que existen para entrenar y validar modelos NER, así como soportar las estructuras de datos desarrolladas para generar los reemplazos.	
T1.4	**Estudio de propuestas similares.**	*2 Puntos de Dificultad*
	Consolidación de las distintas propuestas similares a la presentada en el proyecto (correspondiente al estudio del estado del arte).	

Tabla 2.1: Desglose de las tareas asociadas a la Historia 1 del Proyecto.

[1] https://www.scrummanager.net/bok/index.php?title=Estimación_de_póquer *(Visitado el 22-06-2021).*

[2] https://quantdare.com/numeros-de-fibonacci/ *(Visitado el 24-06-2021).*

Tarea	Descripción	Dificultad
T2.1	**Etiquetado de las entidades del conjunto de documentos a utilizar.**	*5 Puntos de Dificultad*
	Realizar el etiquetado manual de todos los tipos de entidades que aparecen en los documentos que conforman el conjunto de datos.	
T2.2	**Transformación de los documentos al formato requerido por la herramienta para crear el modelo.**	*1 Punto de Dificultad*
	Transformar los documentos con sus anotaciones al formato propietario de la herramienta para que estos puedan ser utilizados.	
T2.3	**División de los documentos en subconjuntos.**	*1 Punto de Dificultad*
	Realización de forma equilibrada y estratificada la división de los documentos en los distintos subconjuntos de datos para entrenar, validar y probar el desempeño del modelo.	

Tabla 2.2: Desglose de las tareas asociadas a la Historia 2 del Proyecto.

Tarea	Descripción	Dificultad
T3.1	**Evaluación de los distintos tipos de modelos utilizando su configuración base.**	*1 Punto de Dificultad*
	Cálculo del desempeño de los distintos modelos NER utilizando su configuración base.	
T3.2	**Optimización y selección de los modelos.**	*3 Puntos de Dificultad*
	Mejora de los resultados obtenidos por los modelos llevando a cabo modificaciones en sus hiperparámetros y seleccionar aquel con el mejor desempeño.	
T3.3	**Creación de componentes para mejorar los resultados de detección.**	*3 Puntos de Dificultad*
	Desarrollo de un nuevo componente basado en expresiones regulares para mejorar los resultados del modelo a la hora de detectar entidades con una estructura predecible.	

Tabla 2.3: Desglose de las tareas asociadas a la Historia 3 del Proyecto.

Tarea	Descripción	Dificultad
T4.1	**Obtención de los datos de la base del conocimiento.**	*2 Puntos de Dificultad*
	Obtención de los datos utilizados para generar los reemplazos, así como la creación de las estructuras de datos necesarias para su uso.	
T4.2	**Creación del mecanismo de reemplazo basado en grafos.**	*5 Puntos de Dificultad*
	Desarrollo un mecanismo de reemplazos de entidadades relacionadas utilizando estructuras de datos basadas en grafos.	
T4.3	**Creación del mecanismo de reemplazo basado en diccionarios.**	*3 Puntos de Dificultad*
	Desarrollo un mecanismo de reemplazos de entidades no relacionadas utilizando estructuras de datos basadas en diccionarios.	
T4.4	**Integración de los mecanismos de reemplazo.**	*1 Punto de Dificultad*
	Integración de ambos mecanismos de reemplazo para que funcionen de manera conjunta y organizada.	

Tabla 2.4: Desglose de las tareas asociadas a la Historia 4 del Proyecto.

Tarea	Descripción	Dificultad
T5.1	**Integración de los componentes de detección de entidades y generación de reemplazos.**	*2 Puntos de Dificultad*
	Integración de los componentes implementados en anteriores etapas para consolidar el flujo de anonimización desarrollado.	
T5.2	**Evaluación de los tiempos de ejecución del *pipeline* para distintos tipos de documentos.**	*1 Punto de Dificultad*
	Evaluación del *pipeline* desarrollado desde el punto de vista de los tiempos de ejecución para comprobar su desempeño y mejorarlo en el caso de que fuese necesario.	

Tabla 2.5: Desglose de las tareas asociadas a la Historia 5 del Proyecto.

Tarea	Descripción	Dificultad
T6.1	**Desarrollo de la parte visual de la herramienta.**	*2 Puntos de Dificultad*
	Implementar de la parte visual de la herramienta junto con los componentes de la vista que se utilizarán para interactuar con el *pipeline* desarrollado.	
T6.2	**Desarrollo del método de extracción de textos de documentos.**	*2 Puntos de Dificultad*
	Diseño de un método que permita extraer el texto a partir de un fichero PDF.	
T6.3	**Integración del *pipeline* con la parte visual.**	*1 Punto de Dificultad*
	Integración de la parte visual de la herramienta con la parte interna de la misma, es decir, conectar la vista con el *pipeline* de anonimización.	

Tabla 2.6: Desglose de las tareas asociadas a la Historia 6 del Proyecto.

Tarea	Descripción	Dificultad
T7.1	**Documentación de la introducción y la descripción del proyecto.**	*3 Puntos de Dificultad*
	Documentación de la parte relativa a la introducción y descripción del proyecto (motivación, objetivos, metodología, contexto, etc).	
T7.2	**Documentación de la implementación del proyecto.**	*3 Puntos de Dificultad*
	Documentación de la parte relativa a la implementación del proyecto.	
T7.3	**Documentación de las conclusiones y anexos del proyecto.**	*1 Punto de Dificultad*
	Documentación de las conclusiones obtenidas tras finalizar el proyecto, así como los apéndices necesarios para complementar la memoria técnica.	

Tabla 2.7: Desglose de las tareas asociadas a la Historia 7 del Proyecto.

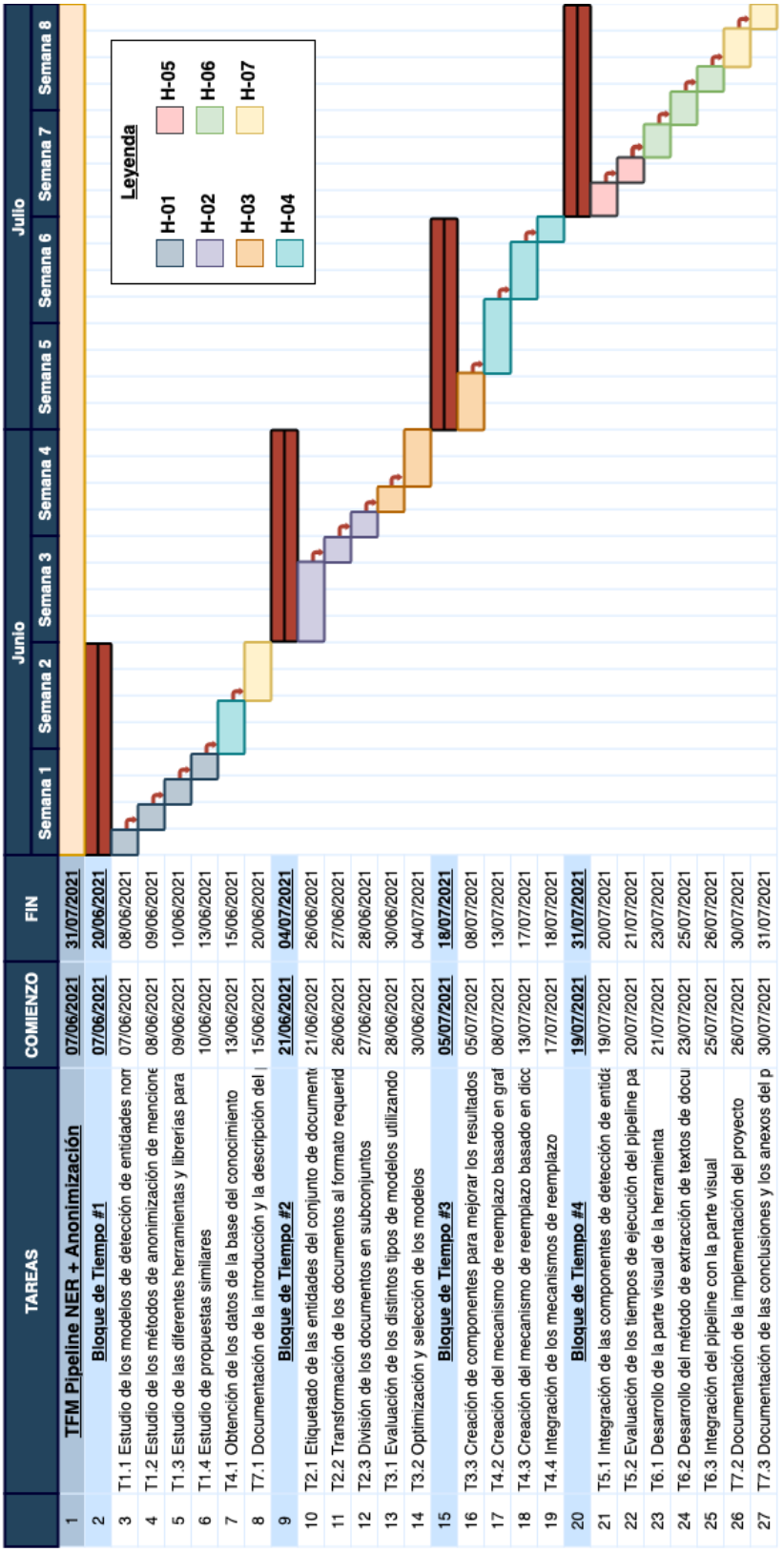
Para llevar a cabo el proyecto se han estimado un total de 45 Puntos de Dificultad a distribuir entre cada uno de los 4 bloques de tiempo en los que se divide el proyecto. Un punto importante que destacar es que a cada uno de estos bloques de tiempo se le ha asignado un total de 57 horas de trabajo efectivo, por lo que el total de horas efectivas en todos ellos equivale a las horas estipuladas para llevar a cabo el Trabajo Fin de Máster (57 horas · 4 bloques de tiempo = 228 horas).

A pesar de que no existe una relación directa y objetiva entre Puntos de Dificultad y horas efectivas, es posible realizar una distribución de las tareas entre los distintos bloques de tiempo haciendo uso de esta medida. Para comenzar, se ha asignado al primer bloque de tiempo un conjunto de tareas por valor de 11 Puntos de Dificultad, dando inicio al primer bloque de tiempo del proyecto. Al finalizar cada bloque se lleva a cabo una comparativa entre los Puntos de Dificultad completados y los estimados, obteniendo así de forma aproximada el número de Puntos de Historia que se es capaz de abordar durante un bloque de tiempo.

En la tabla 2.8 se muestra la distribución de las tareas entre los distintos bloques de tiempo. Del mismo modo, en la Figura 2.1 se presenta el diagrama de Gantt correspondiente a la planificación fijada para el proyecto. Cabe destacar que para simplificar tanto el análisis como las vistas, en esta planificación no se tomarán en cuenta ni las reuniones de seguimiento con los tutores, ni la realización del acto de defensa del proyecto.

Bloque de Tiempo	**Inicio**	**Fin**	**Tareas**	**Total Puntos de Dificultad**
Bloque de Tiempo #1	07/06/2021	20/06/2021	T1.1; T1.2; T1.3; T1.4; T4.1; T7.1	10
Bloque de Tiempo #2	21/06/2021	04/07/2021	T2.1; T2.2; T2.3; T3.1; T3.2	11
Bloque de Tiempo #3	05/07/2021	18/07/2021	T3.3; T4.2 T4.3; T4.4	12
Bloque de Tiempo #4	19/07/2021	01/08/2021	T5.1; T5.2; T6.1; T6.2; T6.3; T7.2; T7.3	12

Tabla 2.8: Distribución de tareas por bloques de tiempo.



	TAREAS	COMIENZO	FIN
1	**TFM Pipeline NER + Anonimización**	07/06/2021	31/07/2021
2	**Bloque de Tiempo #1**	07/06/2021	20/06/2021
3	T1.1 Estudio de los modelos de detección de entidades nom	07/06/2021	08/06/2021
4	T1.2 Estudio de los métodos de anonimización de mencione	08/06/2021	09/06/2021
5	T1.3 Estudio de las diferentes herramientas y librerías para	09/06/2021	10/06/2021
6	T1.4 Estudio de propuestas similares	10/06/2021	13/06/2021
7	T4.1 Obtención de los datos de la base del conocimiento	13/06/2021	15/06/2021
8	T7.1 Documentación de la introducción y la descripción del	15/06/2021	20/06/2021
9	**Bloque de Tiempo #2**	21/06/2021	04/07/2021
10	T2.1 Etiquetado de las entidades del conjunto de documenti	21/06/2021	26/06/2021
11	T2.2 Transformación de los documentos al formato requerid	26/06/2021	27/06/2021
12	T2.3 División de los documentos en subconjuntos	27/06/2021	28/06/2021
13	T3.1 Evaluación de los distintos tipos de modelos utilizando	28/06/2021	30/06/2021
14	T3.2 Optimización y selección de los modelos	30/06/2021	04/07/2021
15	**Bloque de Tiempo #3**	05/07/2021	18/07/2021
16	T3.3 Creación de componentes para mejorar los resultados	05/07/2021	08/07/2021
17	T4.2 Creación del mecanismo de reemplazo basado en graf	08/07/2021	13/07/2021
18	T4.3 Creación del mecanismo de reemplazo basado en dicc	13/07/2021	17/07/2021
19	T4.4 Integración de los mecanismos de reemplazo	17/07/2021	18/07/2021
20	**Bloque de Tiempo #4**	19/07/2021	31/07/2021
21	T5.1 Integración de las componentes de detección de entidi	19/07/2021	20/07/2021
22	T5.2 Evaluación de los tiempos de ejecución del pipeline pa	20/07/2021	21/07/2021
23	T6.1 Desarrollo de la parte visual de la herramienta	21/07/2021	23/07/2021
24	T6.2 Desarrollo del método de extracción de textos de docu	23/07/2021	25/07/2021
25	T6.3 Integración del pipeline con la parte visual	25/07/2021	26/07/2021
26	T7.2 Documentación de la implementación del proyecto	26/07/2021	30/07/2021
27	T7.3 Documentación de las conclusiones y los anexos del p	30/07/2021	31/07/2021

Figura 2.1: Diagrama de Gantt del proyecto.

2.2. Presupuestos

Una vez realizada la estimación y la planificación temporal del proyecto, se va a llevar a cabo una estimación de los costes asociados al mismo. En este presupuesto se van a incluir tanto los costes de las herramientas (hardware y software), así como los costes de los recursos humanos necesarios para su realización.

Por un lado, desde el punto de vista del hardware, toda la implementación de la herramienta se ha llevado a cabo en un ordenador personal (Procesador I7-9850H, 64 bits, 16GB de RAM y 500GB SSD), por lo que hay que tener en cuenta su precio y su vida útil para obtener el coste asociado al proyecto. En cuanto al software utilizado, todas las herramientas son de código abierto, o software de pago con licencias gratuitas debido al rol de estudiante universitario, por lo que en ambos casos su coste ha sido nulo. Sin embargo, el entrenamiento y validación del modelo NER ha sido realizado sobre una máquina virtual proporcionada por el Departamento de Informática de la Universidad de Valladolid. Si bien es cierto que su uso no supone un coste directo en el proyecto, esta se tendrá en cuenta a la hora de realizar los presupuestos para que así estos sean más acordes a una situación real. No obstante, dado que no se conoce el coste de uso por hora de esta máquina virtual, se va a realizar el cálculo basándonos en una máquina virtual de *Amazon Web Services (AWS)* con unas especificaciones similares. Las especificaciones de la máquina virtual proporcionada por el Departamento de Informática son: Procesador 8 núcleos, 64 bits, 16GB de RAM y 30 GB de almacenamiento HDD. Unido a esto, para llevar a cabo todo el desarrollo del proyecto es necesaria una conexión a internet, así como el uso de dispositivos electrónicos, por lo que los costes asociados a ambos recursos se verán también reflejados en los presupuestos. Para ello se tomará el número de horas aproximadas que estos han sido utilizados, junto con su coste por hora.

En la siguiente tabla (ver Tabla 2.9) se muestra el desglose de las herramientas hardware y software utilizadas, así como el coste asociado a cada una de ellas[3,4]:

[3]El coste energético se ha calculado como la media de los costes por hora en España en los meses de junio y julio.

[4]El coste de la máquina virtual se ha obtenido utilizando la calculadora de AWS para las instancias EC2 (alto rendimiento).

Herramienta	**Coste (mes)**	**Vida útil**	**% de uso**	**Coste Total**
Ordenador Personal	1.800€ (coste total)	60 meses (5 años)	3,33 %	59,94€
Internet	30€	2 meses	100 %	60€
Coste Energético	10,12€	2 meses	100 %	20,24€
Máquina Virtual	58,66€	0,46 meses (2 semanas)	25 %	26,98€
GitLab	0€	-	-	0€
Visual Studio Code	0€	-	-	0€
Anaconda	0€	-	-	0€
Datasaur	0€	-	-	0€
TOTAL				**167,16€**

Tabla 2.9: Costes asociados a las herramientas y recursos hardware.

Por otro lado, desde el punto de vista de los costes asociados a los recursos humanos, estos van a ser desglosados en función de los distintos roles que ha ido asumiendo el alumno a lo largo del desarrollo. Para realizar los cálculos de estos costes, se ha supuesto la dedicación habitual de un trabajador en una empresa en la actualidad (40 horas semanales o 160 horas al mes). En la Tabla 2.10 se muestran los distintos roles que el alumno ha tomado durante el desarrollo del proyecto, así como la dedicación en horas, y su coste asociado[5]:

Rol	**Salario (mes)**	**Horas**	**Coste Total**
Jefe de Proyecto	3.292€	30	617,25€
Analista de Requisitos	2.545€	48	763,50€
Desarrollador Python	2.391€	140	2.092,13€
Desarrollador Web	1.583€	10	98,94€
TOTAL			**3.571,82€**

Tabla 2.10: Costes asociados a los recursos humanos. Fuente: *Linkedin Salary (Visitado 20-07-2021).*

Finalmente, teniendo en cuenta tanto los costes hardware y software, así como el coste de los recursos humanos, el coste total previsto para el proyecto es de **3.738,98€**, de los cuales 167,16€ se corresponden a los recursos hardware, y 3.571,82€ a los recursos humanos.

[5]El coste asociado a cada rol ha sido obtenido a partir de Linkedin Salary tomando como referencia el precio medio del puesto a nivel nacional.

2.3. Balance final del Proyecto

2.3.1. Planificación Temporal

Desde el punto de vista de la planificación temporal del proyecto, este se inició en la fecha prevista, y comenzó abordando las tareas T1.1, 1.2 y 1.3, orientadas a la obtención de los conocimientos necesarios para poder llevar a cabo el proyecto de forma satisfactoria. Estas tareas fueron finalizadas en el tiempo previsto, completando así la Historia de Proyecto H-01. Tras esto, se continuó en paralelo con las tareas 7.1 y 4.1, plasmando los conocimientos obtenidos en la documentación técnica del proyecto, y obteniendo los datos necesarios para poder generar las estructuras utilizadas posteriormente para generar los reemplazos. A pesar de que el bloque de tiempo contenía tareas por un valor de 10 Puntos de Dificultad, estas fueron finalizadas antes de acabar el bloque, por lo que se comenzó con la tarea T2.1 (correspondiente al siguiente bloque) y se aumentó el número de puntos de dificultad a abordar en los bloques posteriores.

En el segundo bloque de trabajo se continuó con la tarea T2.1, y se completaron las tareas T2.2 y T2.3, finalizando así la Historia de Proyecto H-02 asociada a la obtención del conjunto de datos necesario para generar el modelo NER. Una vez se finalizaron estas tareas, se continuó con el desarrollo, y se realizaron las tareas T3.1 y T3.2, obteniendo un modelo NER optimizado para detectar las menciones en los documentos. Además de esto, al igual que ocurrió en el primer bloque de tiempo (y debido a que parte de una tarea ya había sido comenzada), se finalizaron las tareas asociadas a este bloque antes de lo planificado, por lo que se inició la tarea T3.3 correspondiente al siguiente bloque, y se volvió a aumentar el número de Puntos de Dificultad en los bloques sucesivos.

En el tercer bloque de tiempo se finalizó lo que restaba de la tarea T3.3 y se abordaron las tareas T4.2, T4.3 y T4.4, dando fin a las Historias de Proyecto H-03 y H-04 (correspondientes a la creación de los componentes de detección de entidades y generación de reemplazos respectivamente). A diferencia de anteriores bloques, en este no hubo tanta diferencia entre la finalización de la última tarea con respecto al fin del bloque, por lo que se mantuvo el ritmo y no se añadieron más tareas a los siguientes bloques.

Finalmente, en el último bloque se realizaron las tareas planificadas, concluyendo las Historias de Proyecto H-05, H-06 y H-07 (correspondientes a la integración de los componentes en la herramienta y a la redacción de la memoria del proyecto). A pesar de que este era el bloque con más tareas asociadas, estas fueron acabadas en el tiempo previsto, ya que muchas de ellas eran cortas y consistían en integrar las partes desarrolladas en tareas e historias anteriores.

2.3.2. Planificación Económica

En cuanto a la planificación económica, dado que no ha habido ningún contratiempo en el desarrollo de la práctica, y todas las tareas han sido completadas en los tiempos fijados, esta no ha sufrido ninguna modificación frente a lo estimado anteriormente. Es por esto, por lo que los costes (tanto hardware como de recursos humanos) asociados al proyecto se mantienen exactamente igual: 3.738,98€.

Capítulo 3

Contexto de Desarrollo

3.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural, PLN (*Natural Language Processing*, NLP en inglés), es la rama de la Inteligencia Artificial que comprende las técnicas por las cuales se dota a una máquina de la capacidad para procesar, analizar, interpretar y representar texto escrito o hablado, como si fuese un ser humano [27]. El primer uso de este tipo de técnicas se remonta a la década de 1940, donde Weaver y Booth [13], [28] iniciaron un proyecto de PLN orientado a descifrar códigos enemigos durante la Segunda Guerra Mundial. Han pasado 80 años desde entonces y en la actualidad, se disponen de sistemas de PLN capaces de procesar, generar e interpretar tanto texto escrito como hablado.

El método más representativo para simbolizar lo que sucede dentro de un sistema PLN es utilizar el **modelo síncrono del lenguaje** [21] (también conocido como "enfoque mediante niveles"). Este enfoque plantea la hipótesis de que el procesamiento del lenguaje humano se lleva a cabo de una forma estrictamente secuencial. Sin embargo, la investigación psicolingüística[1] sugiere que el procesamiento del lenguaje es un proceso mucho más dinámico, ya que un nivel puede interactuar con otro a pesar de que dichos niveles no se encuentren situados de manera contigua. Unido a esto, la introspección revela que el ser humano utiliza con frecuencia la información obtenida en niveles superiores de procesamiento para así ayudarse a entender la información situada en niveles inferiores [8]. Los distintos niveles del lenguaje ordenados de menor a mayor junto con su significado son los siguientes:

- **Fonología:** Es el nivel más bajo del lenguaje. Se encarga de la interpretación de los sonidos del habla de las palabras. Dentro de este nivel existen distintos tipos de reglas que se utilizan en el análisis fonológico:

[1] http://revistamito.com/que-es-la-psicolinguistica/ *(Visitado el 30-06-2021).*

- **Reglas Fonéticas**: Son las reglas que describen los sonidos dentro de las palabras. Por ejemplo, en el caso del castellano, todas las palabras terminadas en consonante tienen el golpe de voz en la última sílaba (balón, ventilador, tambor, etc).
- **Reglas Fonémicas**: Son las reglas que rigen las variaciones en la pronunciación cuando las palabras se pronuncian juntas.
- **Reglas Prosódicas**: Son las reglas que determinan la fluctuación en el acento y en la entonación a lo largo de una oración.

El conjunto de estas reglas es muy importante en sistemas de PLN capaces de interpretar texto hablado, ya que las ondas sonoras asociadas a la vocalización del texto son analizadas digitalmente y pueden ser comparadas mediante estas reglas, mejorando la capacidad de interpretación de los sistemas de PLN.

- **Morfología:** Este nivel se encarga de analizar la naturaleza de las componentes que conforman las palabras, siendo el morfema la unidad más pequeña e indivisible que posee significado gramatical. Por ejemplo, la palabra *aterrizaje* se puede dividir en dos morfemas y un lexema: siendo *a-* un morfema prefijo; *-terr-* un lexema que representa al sustantivo tierra (además de ser el núcleo de la palabra); y el morfema *-izaje* que es un sufijo que hace referencia a una *"acción de algo"*. Debido a que el significado de los morfemas es siempre el mismo en todas las palabras, el ser humano puede descomponer una palabra desconocida en sus morfemas para así comprender su significado. Volviendo al ejemplo anterior, si se descompone la palabra *alunizaje*, se puede ver como esta se encuentra constituida por un prefijo *a-*, un lexema *-lun-* (del sustantivo luna), y un morfema *-izaje* que es una acción de algo. A partir de esta descomposición, y sin saber el significado de la palabra, se puede deducir que esta hace referencia a una acción que se lleva a cabo en la luna. De manera análoga, un sistema de PLN es capaz de reconocer el significado de cada uno de los morfemas de una palabra, obteniendo así una mejor interpretación de las palabras.

- **Léxico:** En este nivel se interpreta el significado a través del análisis de cada una de las palabras que conforman el texto. En este nivel puede ser utilizado un léxico completo (conjunto de todas las palabras que conforman un lenguaje), o parcial (únicamente conformarlo aquellas palabras que se van a utilizar). Este léxico a su vez puede ser más complejo y, además de las palabras, puede contener también información sobre su clase semántica, los argumentos que toma, la definición de cada significado (para palabras polisémicas), etc. Dicho de otro modo, este léxico que se utiliza para analizar el significado de las palabras en este

nivel es el equivalente al diccionario de la lengua que utilizan los seres humanos.

- **Sintáctico:** Este nivel se enfoca en analizar las palabras de una oración con el objetivo de descubrir la estructura gramatical de la oración. El resultado de este nivel consiste en una representación que muestra las relaciones de dependencia que existen entre las distintas palabras que conforman la oración.

- **Semántico:** Este es el nivel en el que la mayoría de las personas piensan que se determina el significado de un texto. Sin embargo, como se ha mostrado en las anteriores definiciones, todos los niveles contribuyen en mayor o menor medida a la obtención de ese significado. El procesamiento semántico determina los posibles significados que tiene una oración, ya que se enfoca en las interacciones de los significados entre las distintas palabras que conforman la oración. Además de esto, este nivel de procesamiento también incluye la desambiguación semántica de palabras polisémicas (palabras con más de un significado). Esta desambiguación permite seleccionar un único sentido-significado de la palabra a la hora de llevar a cabo la representación semántica de la oración. No obstante, para establecer este significado de entre todos los posibles, es necesario obtener información del resto de palabras que conforman la oración. Para ello, existen diferentes métodos como el análisis de frecuencias de cada uno de los significados dentro del corpus utilizado, análisis del contexto local de la palabra dentro de la oración, análisis del significado dentro del dominio en el que se encuentra el documento, etc.

- **Discurso:** Mientras que el léxico, la sintaxis y la semántica funcionan con unidades de palabras u oraciones, el nivel de discurso en un sistema PLN funciona a nivel de unidades de texto. Esto quiere decir que no interpreta textos como múltiples oraciones concatenadas entre sí y cuyo significado se analiza por separado para cada una de ellas. Más bien, el nivel de discurso se centra en obtener las propiedades del texto en su conjunto, obteniendo el significado de todo el texto al llevar a cabo las conexiones entre las distintas oraciones que lo componen. En este nivel se pueden producir varios tipos de procesamiento del discurso, sin embargo los dos más comunes son los siguientes:

  - **Resolución de Anáforas:** Consiste en la sustitución de palabras semánticamente vacías (como los pronombres), por la entidad apropiada a la que se está refiriendo.
  - **Reconocimiento de la Estructura del Texto:** Determina las funciones de las oraciones o fragmentos que conforman el texto, lo que permite generar una representación significativa del mismo. Por ejemplo un artículo científico se puede deconstruir en componentes

del discurso como: resumen, introducción, estado del arte, planteamiento propuesto, evaluación-discusión y conclusiones.

En la actualidad, los sistemas de PLN tienden a implementar módulos que permiten procesar los niveles más bajos del procesamiento (morfología, léxico y sintáctico). Esto puede deberse a varias razones: 1) La aplicación que se implemente no requiera de una interpretación de los niveles superiores del texto; 2) Los niveles más bajos del procesamiento son los más investigados, por lo que al no haber propuestas que procesen los niveles superiores, no hay sistemas PLN que procesen estos niveles; o 3) En los niveles inferiores se trata con unidades de análisis más pequeñas (morfemas, palabras, oraciones, etc), las cuáles **se rigen por reglas ya definidas y concretas** (en lugar de los niveles superiores que se rigen por la regularidad y por el conocimiento del dominio o el contexto).

3.1.1. Enfoques para el Procesamiento del Lenguaje Natural

Existen diferentes enfoques que utilizan los sistemas de PLN para el procesamiento del lenguaje [21]. Estos enfoques se dividen en cuatro categorías en función del método empleado:

- **Enfoques Simbólicos:** Los enfoques simbólicos llevan a cabo un análisis profundo de los fenómenos lingüísticos. Este tipo de enfoques se basan en la representación explícita de hechos sobre el lenguaje mediante esquemas de representación del conocimiento, y algoritmos asociados a ellos [41]. De hecho, la descripción de los niveles de análisis del lenguaje llevada a cabo anteriormente se encuentra definida desde una perspectiva simbólica. Los enfoques simbólicos son los enfoques más básicos que se pueden definir, y se ajustan a la forma del lenguaje humano ya que, al igual que este, se genera mediante reglas y léxicos definidos. Un ejemplo de este tipo de enfoques se ve en los sistemas lógicos o los sistemas basados en reglas. En los sistemas lógicos, la estructura simbólica adopta la forma de proposiciones lógicas que se definen mediante procedimientos de inferencia que preservan la verdad. Por otra parte, los sistemas basados en reglas constan de un conjunto de reglas (que conforman la base de conocimiento), y un motor de inferencia que se encarga de aplicarlas.

- **Enfoques Estadísticos:** Los enfoques estadísticos utilizan distintas técnicas matemáticas y grandes corpus de texto, para desarrollar modelos generalizados y aproximados de ejemplos reales, a partir de la información proporcionada en el corpus utilizado. A diferencia de los enfoques simbólicos, los enfoques estadísticos utilizan los datos del corpus como principal fuente de información para generar el modelo. Este tipo de enfoques se utiliza normalmente en tareas en las que no existen un conjunto de reglas

definidas que se satisfagan en la totalidad de los casos, por ejemplo tareas de reconocimiento de voz, análisis sintáctico, etiquetado de partes de la voz, etc.

- **Enfoques Conexionistas:** De forma similar a los enfoques estadísticos, los enfoques conexionistas desarrollan modelos generalizados a partir de ejemplos lingüísticos (a partir de un corpus de ejemplo). La diferencia que existe entre este tipo de enfoques y los enfoques estadísticos se encuentra en que los modelos conexionistas combinan el aprendizaje estadístico con varias teorías de representación. Esto hace que en los enfoques conexionistas se puedan realizar transformaciones, inferencias y manipulación de fórmulas lógicas (al igual que ocurría en los enfoques simbólicos). En términos generales, un modelo conexionista no es más que una red de unidades de procesamiento simple interconectadas con conocimiento almacenado en los pesos de las conexiones entre unidades de procesamiento. Este modelo puede verse como la arquitectura común de una red neuronal, en el que las unidades de procesamiento son las neuronas, y el conocimiento es el equivalente a los pesos asociados a cada una de las entradas de las neuronas. Dentro de los enfoques conexionistas existen dos tipos:

  - **Modelos Localistas:** En este tipo de modelos cada una de las unidades de procesamiento representa un concepto concreto, y los pesos asociados a las conexiones entre los distintos conceptos, representan la relación que existe entre ellos. Por ejemplo, si una unidad de procesamiento representa el concepto "libro" y otra unidad el concepto "biblioteca", la relación entre ambos conceptos se encuentra codificada por los pesos que hay entre ambas unidades de procesamiento. Cuanto mayor sea la relación que existe entre ambos conceptos, mayor será el peso asociado a su conexión. Este tipo de modelos tienen un buen desempeño en tareas de desambiguación, generación del lenguaje o inferencia.
  - **Modelos Distribuidos:** En este tipo de modelos, a diferencia de los modelos localistas, un concepto se representa en función de la activación simultánea de varias unidades (en lugar de tener una unidad por cada concepto). Este tipo de modelos es adecuado para tareas de análisis sintáctico, traducción o recuperación asociativa (recuperación de la información a partir del conocimiento parcial de su contenido).

3.1.2. Desafíos del Procesamiento del Lenguaje Natural

A pesar de que el Procesamiento del Lenguaje Natural es una herramienta muy potente que ofrece un gran número de beneficios, y puede ser utilizada para

una gran variedad de aplicaciones (ver Apartado 3.1.3), el lenguaje humano es muy complejo. Es por esto por lo que existen desafíos e inconvenientes que el Procesamiento del Lenguaje Natural aún no ha sido capaz de solventar por completo [33]. Algunos de estos desafíos son:

- **Polisemia y Homonimia:** En un texto, varias palabras o frases pueden tener distintos significados en función del contexto en el que se encuentren. Además, existen palabras que se escriben exactamente igual, pero su significado semántico es diferente (palabras homónimas). Este tipo de situaciones son fáciles de identificar e interpretar para un ser humano, ya que leemos el contexto de la oración y conocemos los distintos significados de las palabras. Sin embargo, a pesar de que los modelos PLN pueden haber aprendido también todas estas definiciones a partir del corpus dado, diferenciar entre palabras a partir del contexto, sigue siendo un desafío que aún no ha sido resuelto en su totalidad.

- **Sinónimos:** Al igual que ocurre con la homonimia, la aparición de palabras sinónimas en el texto dificulta su comprensión contextual en los sistemas PLN. Además de esto, la utilización de unos sinónimos u otros depende en gran medida del vocabulario personal de cada individuo, y distintas personas pueden utilizar diferentes sinónimos para hacer referencia a un mismo término. Del mismo modo, también influye el nivel de complejidad de los sinónimos y, mientras hay palabras que apenas tienen 1 o 2 sinónimos, existen otras que tienen muchos más (dificultado la caracterización del contexto del texto). Es por esto por lo que es muy importante incluir en un sistema PLN todos los significados de una palabra y todos los sinónimos posibles, disminuyendo así el error.

- **Ironía y Sarcasmo:** En los sistemas PLN, la ironía y el sarcasmo presentan grandes problemas, ya que su significado e intencionalidad son diferentes a los aparentes o más inmediatos (al igual que ocurre con las metáforas).

- **Ambigüedad:** El término ambigüedad en sistemas PLN hace referencia a oraciones que pueden tener potencialmente más de un significado. Dentro de esta ambigüedad se distinguen tres tipos posibles:

  - **Ambigüedad Léxica:** Palabras que en función de su uso tienen diferentes tipos morfológicos: sustantivo, adjetivo, verbo, etc.
  - **Ambigüedad Semántica:** Se refiere a la interpretación de una oración en el contexto. Por ejemplo, la frase *"El otro día vi a mi hermano jugando al fútbol con mis gafas"*. Esta frase podría significar que vi a mi hermano a través de mis gafas, o que era mi hermano el que estaba utilizando mis gafas mientras jugaba al fútbol.

  - **Ambigüedad Sintáctica:** Cuando en la sintaxis de una frase una misma palabra puede pertenecer a dos grupos sintácticos diferentes. Por ejemplo en la frase *"Me vuelvo a cambiar"* puede ser que la persona vaya a ir a un determinado lugar a cambiarse, o que se cambie dos veces.

- **Idioma:** El idioma es el desafío más importante que tienen los sistemas PLN. A lo largo de este apartado, se han descrito problemas concretos que se presentan en el castellano (aunque son extensibles a otros idiomas como el inglés). Sin embargo, cada idioma tiene sus particularidades concretas, las cuáles presentan nuevos retos a resolver en los sistemas PLN. Si nos centramos en un único idioma (en este caso el castellano), pueden surgir dos problemáticas asociadas a él:

  - **Dominios Concretos:** A pesar de utilizar un determinado idioma, existen diferentes dominios en los que tanto la forma de expresarse como el lenguaje utilizado es muy diferente a otro. Por ejemplo, textos del ámbito médico y del ámbito jurídico pueden estar escritos en un mismo idioma, pero la forma de expresarse y el lenguaje utilizado es totalmente diferente.
  - **Bajos Recursos:** Los modelos actuales de Aprendizaje Automático orientados al PLN han sido construidos para lenguajes comunes y ampliamente utilizados (inglés, castellano, chino, etc). Es por esto, por lo que en el caso de querer desarrollar un sistema PLN en otro idioma u otra lengua, o no va a ser posible, o va a ser más costoso, ya que es probable que esta no se encuentre soportada. Sin embargo, existen técnicas como los Transformadores Multilingües como *BERT-Multilingüe*[2], que permiten identificar las similitudes universales que existen entre distintos idiomas para poder ser utilizados (a pesar de que la base de desarrollo de la lengua de destino sea reducida).

- **Errores en el Texto o la Voz:** A menudo, los textos que procesan los sistemas PLN no son correctos al 100 %. Es bastante probable que estos sistemas tengan que lidiar con textos en los que haya faltas de ortografía o palabras mal utilizadas en el contexto. Si bien es cierto que estos sistemas tienen procesos de corrección gramatical automática, no siempre pueden solventar todos los errores, o no pueden entender la intención del escritor.

  En el caso del lenguaje hablado ocurre algo similar, errores o variaciones en el habla como pronunciaciones erróneas, tartamudeos o acentos, dificultan el procesamiento de la voz para los sistemas PLN. Sin embargo, con el

[2]https://github.com/google-research/bert/blob/master/multilingual.md *(Visitado el 29-06-2021).*

aumento del uso de asistentes de voz, y por consiguiente el aumento en las bases de datos de idiomas, la frecuencia de estos problemas es menor.

- **Coloquialismos y Argot:** Cuando se trata del lenguaje informal o en un contexto muy concreto y reducido, es común el uso de unas expresiones o una jerga específica. Esto es un reto para los sistemas PLN, especialmente para sistemas orientados a cubrir un caso de uso genérico. En el caso de los coloquialismos, puede que estos no tengan una definición en el diccionario como tal, o que si la tienen, su significado sea distinto en función del área geográfica donde se encuentre. Para el caso del argot ocurre algo similar, ya que el lenguaje es cambiante, se expande y transforma continuamente, por lo que cada día aparecen nuevas palabras. Por otro lado, al igual que ocurre con las variaciones en el habla, disponer de una mayor cantidad de datos hace que estos problemas disminuyan.

3.1.3. Aplicaciones del Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural dispone tanto de teoría como de implementaciones para una gran variedad de aplicaciones. De hecho, cualquier aplicación que utilice algún tipo de procesamiento con textos es candidata a ser una aplicación PLN. Dentro de la variedad de aplicaciones que utilizan PLN, las más habituales son las siguientes [21], [8]:

- **Recuperación de Información:** La Recuperación de Información (*Information Retrieval*, IR, en inglés), consiste en la obtención de información de interés, a partir de una consulta de entrada dada. Por ejemplo un buscador web, en el que, a partir de una consulta, recupera aquellas páginas que se encuentran relacionadas con el criterio de búsqueda proporcionado.

- **Extracción de Información:** La Extracción de Información (*Information Extraction*, IE, en inglés), consiste en el reconocimiento y obtención automática de información en un conjunto de textos estructurado (formularios), semi-estructurados (informes, facturas, etc), o no estructurados (capítulos de un libro). Dentro de esta aplicación destaca el Reconocimiento de Entidades Nombradas (*Named Entity Recognition*, NER, en inglés) del cuál se tratará más en profundidad en el Apartado 3.2.

- **Análisis de Sentimientos:** El Análisis de Sentimiento (*Sentiment Analysis*, en inglés), consiste en la obtención, cuantificación y estudio de la información subjetiva (comentarios, opiniones, valoraciones, etc) tanto en textos como en el habla. Habitualmente esta cuantificación se lleva a cabo en base a tres niveles: positivo, negativo y neutral, aunque en modelos más complejos, puede ser un valor numérico y generar tantos nuevos niveles de emoción como sean necesarios. Los casos de uso más

comunes de este tipo de aplicaciones son: análisis de satisfacción de clientes, análisis de reseñas de productos, análisis de redes sociales, etc.

- **Generación de Textos:** La Generación de Textos (*Text Generation*, en inglés), agrupa el conjunto de técnicas cuyo objetivo consiste en generar un texto de salida en base a la entrada recibida. Dentro de esta aplicación existen distintas ramas como el Resumen de Textos (*Text Summarization*, en inglés), cuyo objetivo se centra en crear una síntesis reducida de grandes volúmenes de textos; o los Sistemas de Diálogo (*Dialogue Systems*, en inglés), cuyo objetivo consiste en mantener una conversación de forma realista y adaptativa en función de la entrada recibida, por ejemplo un chatbot.

- **Modelado de Temas:** El Modelado de Temas (*Topic Modeling*, en inglés), es uno de los métodos utilizados para identificar temas en el texto. El objetivo principal de este tipo de técnicas consiste en, a partir de un conjunto de textos, obtener los temas principales que se tratan en ellos, y clasificar los textos en función de esos temas. A diferencia de las aplicaciones anteriores, las técnicas de Modelado de Temas no requieren de un conjunto de datos de entrenamiento para poder ser utilizadas, ya que estas se encuentran enmarcadas dentro de la rama del Aprendizaje Automático No Supervisado. Alguno de los algoritmos utilizados en este tipo de técnicas son el LSA[3] (*Latent Semantic Analysis*), el LDA[4] (*Latent Dirichlet Allocation*), o el CTM[5] (*Correlated Topic Model*).

3.2. Reconocimiento de Entidades Nombradas

El Reconocimiento de Entidades Nombradas (*Named Entity Recognition*, NER, en inglés), es una subtarea englobada dentro de las técnicas de Extracción de Información (*Information Extraction*, IE, en inglés)(ver Apartado 3.1.3). Este tipo de técnicas se centran en la identificación automática de información en un conjunto de textos. Una de las principales características del NER es que permite obtener entidades concretas (personas, organizaciones, direcciones postales, etc) en todo tipo de textos, ya sean estructurados, semi-estructurados o desectructurados, lo que hace que pueda ser utilizado en la gran mayoría de casos de uso.

Un punto importante a destacar es que para entidades comunes (países, organizaciones, etc), existen modelos ya entrenados capaces de detectarlas

[3]https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python *(Visitado el 28-06-2021).*

[4]https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2 *(Visitado el 28-06-2021).*

[5]https://towardsdatascience.com/intuitive-guide-to-correlated-topic-models *(Visitado el 28-06-2021).*

directamente sin la necesidad de llevar a cabo todo el proceso de creación y entrenamiento. Además de esto, las tecnologías actuales permiten reentrenar este tipo de modelos para que sean capaces de identificar cualquier tipo de entidad, lo cuál es algo útil si se quieren identificar entidades no soportadas, o procesar textos en otro idioma. En la Figura 3.1 se puede ver como el modelo base de Spacy[6] en castellano es capaz de detectar personas, localizaciones y organizaciones, sin la necesidad de haber realizado ningún tipo de entrenamiento previo.

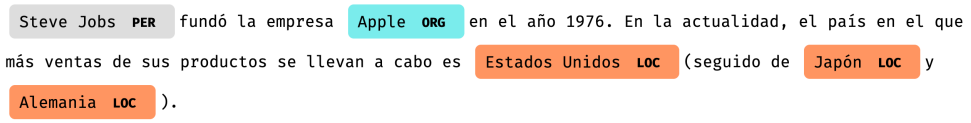


Figura 3.1: Ejemplo de Reconocimiento de Entidades Nombradas utilizando el modelo base de Spacy en castellano.

Dentro del Reconocimiento de Entidades Nombradas, existen diferentes tipos de modelos, en función del tipo de aprendizaje utilizado para llevar a cabo la tarea. Estos tipos de modelos [37] son:

- **Modelos basados en Reglas:** Este tipo de modelos es la aproximación más simple para el Reconocimiento de Entidades Nombradas. Su funcionamiento se basa en la aplicación de reglas y heurísticas definidas manualmente para detectar las entidades, por ejemplo *"después de una palabra **Calle** o **Avenida**, las siguientes palabras que comiencen por mayúscula pertenecen al tipo de entidad **LOCALIZACIÓN**"*. En el caso de tener el texto *"¿Sabes como puedo llegar a la calle Amanecer?"*, la regla descrita anteriormente asignaría la etiqueta **LOCALIZACIÓN** a **Amanecer**. Como se puede ver, la simplicidad de este tipo de modelos hace que no puedan ser utilizados en situaciones en las que la variedad sea muy elevada, o existan un gran número de entidades a detectar. El motivo de esto se debe a que la creación de reglas para un gran variedad de formas en las que se pueden encontrar las entidades en los textos es inviable.

- **Modelos basados en Aprendizaje Supervisado:** En este tipo de modelos, el aprendizaje se lleva a cabo comparando los resultados obtenidos por el modelo durante el entrenamiento, con el resultado esperado que se debería obtener. Es por esto, por lo que al igual que ocurre con los modelos de Aprendizaje Supervisado convencionales, es necesario disponer de un conjunto de datos etiquetado para poder entrenar el modelo. Sin embargo, existen diferencias entre los conjuntos de datos etiquetados para

[6] https://spacy.io *(Visitado el 01-07-2021).*

entrenar modelos de NER, y los conjuntos etiquetados para otras tareas de clasificación. Mientras que para tareas de clasificación convencionales el conjunto de datos se encuentra estructurado en instancias con los datos de entrada y su salida esperada, en el caso del NER se encuentra estructurado por palabras y la etiqueta correspondiente a la entidad a la que pertenece. Existen diferentes formatos de etiquetado para el NER (ver Apartado 3.2.1), sin embargo, el más habitual (y el que se utilizará en este proyecto) es el sistema de etiquetado IOB (*Inside-Outside-Beginning*, en inglés).

- **Modelos basados en Aprendizaje Semi-Supervisado:** Este tipo de modelos se encuentra en un punto intermedio entre el Aprendizaje Supervisado y el No Supervisado. En este tipo de modelos, el conjunto de datos de entrenamiento se encuentra formado por datos etiquetados (en menor proporción), y datos sin etiquetar (en mayor proporción). El objetivo que se persigue es que, a partir de los datos etiquetados, se puedan ir etiquetando correctamente textos sin etiquetar, aumentando así el conjunto de textos etiquetados y mejorando los resultados del modelo. La ventaja que tiene este tipo de modelos es que no es necesario llevar a cabo el etiquetado de la totalidad de los textos, eliminando así la tediosa labor del etiquetado (sobre todo cuando el conjunto de datos con el que se quiere trabajar es muy grande).

- **Modelos basados en Aprendizaje No Supervisado:** En este tipo de modelos, el conjunto de datos utilizado para generar el modelo final no se encuentra anotado. Esto hace que el modelo tenga que llevar a cabo un proceso de inferencia que le permita agrupar o clasificar las distintas palabras del texto, en el tipo de entidad que se corresponda.

En el presente Trabajo Fin de Master se emplearán modelos basados en Aprendizaje Supervisado, por lo que será necesario llevar a cabo un paso previo de etiquetado de los datos que se utilizarán para crear el modelo.

3.2.1. Proceso de Creación de un Sistema de Reconocimiento de Entidades Nombradas

Como se ha mencionado anteriormente, el modelo que se empleará para detectar las entidades de los textos es un modelo basado en Aprendizaje Automático Supervisado. Es por ello por lo que antes de comenzar con la división de los datos en subconjuntos, el entrenamiento, y la evaluación del modelo, es necesario llevar a cabo un paso previo. Este paso consiste en realizar un etiquetado de las entidades que se quieren detectar en los textos que se utilizarán para entrenar y evaluar el modelo.

El proceso de etiquetado de datos consiste en asignar una etiqueta a cada una de las palabras del texto en función de la entidad a la que esta pertenezca. Existen diferentes formatos[7] para llevar a cabo este proceso de etiquetado. Los más habituales son los siguientes:

- **IO:** IO (*Inside-Outside*, en inglés) es el método más simple para el etiquetado de entidades nombradas. Este formato consiste en asignar a cada palabra del texto el tipo de entidad a la que pertenece. Además, de esto, se añade un prefijo[8] en función de si esta forma parte de una entidad (I), o si por el contrario se encuentra fuera de la entidad (O).
- **IOB:** El formato IOB (*Inside-Outside-Beginning*, en inglés) es el formato de etiquetado estándar en la actualidad. Fue inventado por Ramshaw y Marcus en 1995 y mejora la información que proporciona el formato IO. El formato IOB añade un nuevo prefijo (B) para hacer referencia a la primera palabra de cada una de las entidades existentes en el texto. Esto es una diferencia respecto al formato IO ya que, mientras en este todas las palabras de una entidad se denotaban con I, el formato IOB añade información sobre la primera palabra de cada entidad. Esto permite diferenciar entre entidades simples (una palabra) y compuestas (más de una palabra).
- **BILOU:** El formato BILOU (*Beginning-Inside-Last-Outside-Unit_Lenght*), o BMEWO (*Beginning-Middle-End-Whole-Outside*), es una evolución del formato IOB. En este formato se añade el prefijo (L) a la última palabra de entidades con múltiples palabras (a diferencia del formato IOB, en el que se denotaba la primera palabra con el prefijo B y el resto con el prefijo I). Además de esto, se añade un nuevo prefijo (U) a las entidades con una única palabra. De este modo se consigue diferenciar al instante si una determinada entidad se encuentra conformada por una o más palabras, añadiendo un extra de información al documento etiquetado.

En la siguiente tabla (ver Tabla 3.1) se muestra el etiquetado correspondiente a la frase *"Luis Pérez Rodríguez compró un billete para viajar a España"* en los distintos formatos presentados anteriormente.

[7] https://lingpipe-blog.com/2009/10/14/coding-chunkers-as-taggers-io-bio-bmewo-and-bmewo/ *(Visitado el 02-07-2021).*

[8] En el caso de que la palabra no forme parte de ninguna entidad a etiquetar, únicamente se le asigna el valor "O" (en lugar de la asignación "prefijo-tipo_entidad").

Palabra	**IO**	**IOB**	**BILOU**
Luis	I-PERSONA	B-PERSONA	B-PERSONA
Pérez	I-PERSONA	I-PERSONA	I-PERSONA
Rodríguez	I-PERSONA	I-PERSONA	L-PERSONA
compró	O	O	O
un	O	O	O
billete	O	O	O
para	O	O	O
viajar	O	O	O
a	O	O	O
España	I-PAIS	B-PAIS	U-PAIS

Tabla 3.1: Ejemplo de etiquetado de una frase en los formatos IO, IOB y BILOU.

Además de los formatos de etiquetado, existen también formatos de texto estructurados en los que se pueden representar las etiquetas de las palabras que conforman un texto. Estos formatos de texto se utilizan en combinación con los formatos de etiquetado, para entrenar los modelos de NER, o para llevar a cabo la visualización de las entidades en un texto. El motivo de esto se debe a que al ser formatos de texto estructurados, permite representar de una forma estándar las etiquetas de las palabras (posibilitando la automatización y la representatividad de estas características de una forma sencilla). Además de esto, los formatos de textos estructurados mejoran la legibilidad de las anotaciones, facilitando la labor de revisión o de modificación en caso de error. Los formatos de texto más habituales para representar las anotaciones son:

- **TSV:** El formato TSV (*Tab Separated Values*) es uno de los formatos más simples para representar las anotaciones. Este formato es muy similar al CSV (incluso puede ser el mismo dependiendo del separador utilizado), ya que cada palabra del texto se encuentra en una línea del fichero, y se utiliza el tabulador para separar la palabra con su etiqueta asociada. A diferencia del formato CSV, en el formato TSV siempre se utiliza el tabulador para separar la palabra y su etiqueta asociada. Este tipo de formato es el más simple pero a su vez es el menos legible de todos, dado que el tener una palabra en cada línea dificulta la lectura continua de las frases.

- **JSON:** El formato de texto JSON (*JavaScript Object Notation*) es uno de los formatos que permiten una mayor representatividad de la información de todos los componentes que conforman el texto, así como de su estructura. Esto se debe a que JSON es un formato que permite dividir los objetos de forma jerárquica y representar sus características de una forma anidada. Gracias a esto, se puede emular una estructura

similar a la que tiene el texto original, dividiendo el texto en párrafos, los párrafos en oraciones, y las oraciones en palabras. Toda esta división se representa de forma anidada y aportando información de valor en cada uno de los niveles (identificador de la palabra, palabra, etiqueta asociada, etc). A continuación se muestra un ejemplo simple de la representación de las anotaciones en formato IOB y JSON (ver Código 3.1).

```
{
    "id":0,
    "paragraphs":[
    {
        "sentences":[
        {
            "tokens":[
            {
                "id":0,
                "orth":"Luis",
                "ner":"B-PERSONA"
            },
            {
                "id":1,
                "orth":"Pérez",
                "ner":"I-PERSONA"
            },
            {
                "id":2,
                "orth":"Rodríguez",
                "ner":"I-PERSONA"
            },
            {
                "id":3,
                "orth":"compró",
                "ner":"O"
            },
        ...
```

Código 3.1: Representación del etiquetado de un texto en formato IOB y JSON.

- **XML:** El formato XML (*eXtensible Markup Language*) es un formato de etiquetado utilizado habitualmente para representar de forma visual las palabras del texto y su etiqueta asociada. Este tipo de formato (al igual que el HTML), es un formato de marcado mediante etiquetas, por

lo que es fácilmente legible y entendible. Este tipo de formatos suele ser utilizado a la hora de representar de forma gráfica las etiquetas que el modelo NER a asignado a un texto cualquiera. Es decir, este formato se utiliza para visualizar la inferencia del modelo en lugar de para entrenarlo. Ambos formatos (XML y HTML) son compatibles con los navegadores web y con otras librerías como CSS o JavaScript, lo que permite añadir características especiales en función de cada entidad. En el ejemplo de la Figura 3.1, el código fuente sin renderizar del texto mostrado (y de sus entidades), es un código HTML generado a partir de las entidades que el modelo ha asignado a cada una de las palabras del texto de entrada. A continuación se muestra un ejemplo en texto plano del formato de etiquetado XML (ver Código 3.2).

```
<PERSONA> Luis Pérez Rodríguez </PERSONA> compró un
↪  billete para viajar a <PAIS> España </PAIS>.
```

Código 3.2: Representación del etiquetado de un texto en formato XML.

3.2.1.1. Evaluación de Sistemas de Reconocimiento de Entidades Nombradas

Una vez se ha llevado a cabo el proceso de etiquetado sobre todos los textos que conformaran el corpus empleado para generar el modelo, es necesario realizar una división de los documentos en subconjuntos. Los subconjuntos generados se utilizarán para entrenar, validar y probar el desempeño del modelo con datos que este nunca "ha visto". Este paso es idéntico al que se realiza en cualquier proyecto de esta índole, y el objetivo principal consiste en lograr que el modelo final implementado sea capaz de generalizar correctamente la problemática a tratar. No obstante, un punto muy importante a destacar es que los datos que se encuentren en cada uno de los subconjuntos han de ser lo más representativos posible del conjunto de datos inicial. De no ser así, el modelo construido estaría sesgado y únicamente funcionaría correctamente para unos casos muy concretos (mala generalización), en lugar de funcionar bien para todos o para la mayoría. Los subconjuntos en los que se divide el corpus, así como su utilidad son los siguientes:

- **Entrenamiento:** El conjunto de entrenamiento (*Training Set*) se encuentra formado por todos aquellos documentos que se emplearán para llevar a cabo el aprendizaje el modelo. Dicho de otro modo, serán los documentos que el modelo emplee para ajustar sus parámetros internos.
- **Validación:** El conjunto de validación (*Validation Set*) es un subconjunto que, al igual que el subconjunto de entrenamiento, se emplea durante la

fase de aprendizaje del modelo para ajustar sus hiperparámetros. Otro uso que se le da a este conjunto es para comprobar si el funcionamiento del modelo es correcto, o si por el contrario existe un sobreajuste que puede empeorar su capacidad de generalización. En definitiva, el modelo es entrenado con los datos de entrenamiento, y se utiliza este conjunto para comprobar que su funcionamiento es correcto.

- **Prueba:** El conjunto de prueba (*Test Set*) es un conjunto totalmente independiente de los dos anteriores. El objetivo principal de este conjunto consiste en determinar qué tan bien es capaz el modelo de generalizar. Es por esto por lo que los documentos que se encuentren en este conjunto de datos no pueden ser utilizados en ninguna de las fases de aprendizaje del modelo, entendiendo como fase de aprendizaje aquellas fases en las que se producen modificaciones en los parámetros internos del modelo.

Finalmente, una vez se ha creado y entrenado el modelo del NER, el paso final consiste en evaluar su desempeño. Es en este punto donde entra en juego el conjunto de prueba anteriormente definido. A pesar de trabajar con textos, el modelo generado no es más que un modelo de clasificación, cuya función consiste en asignar a cada una de las palabras del texto la etiqueta asociada a la entidad a la que pertenece. Es por esto por lo que las métricas de este modelo son las mismas que las utilizadas para evaluar modelos de clasificación. Todo modelo de clasificación realiza una clasificación sobre un conjunto de datos de acuerdo con la Tabla 3.2. Esta tabla, también conocida como *matriz de confusión*, se utiliza para obtener las métricas asociadas al modelo, permitiendo determinar cuantitativamente la eficacia del modelo.

	Clasificado Positivo	**Clasificado Negativo**
Ejemplo Positivo	Verdadero Positivo (VP)	Falso Negativo (FN)
Ejemplo Negativo	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 3.2: Matriz de Confusión.

Donde:

- **Verdadero Positivo (VP):** La palabra real lleva asociada una entidad y el tipo de dicha entidad es del mismo tipo que el predicho por el modelo. Ejemplo: Real → **B-NOMBRE**; Predicción → **B-NOMBRE**.
- **Falso Positivo (FP):** La palabra real no pertenece a ningún tipo de entidad, y el modelo le asigna una etiqueta de entidad. Ejemplo: Real → **O**; Predicción → **B-NOMBRE**.
- **Falso Negativo (FN):** La palabra real pertenece a una entidad, y el modelo la clasifica como si no perteneciente a ninguna. Ejemplo: Real → **B-NOMBRE**; Predicción → **O**.

- **Verdadero Negativo (VN):** La palabra real no se encuentra asociada con ningún tipo de entidad, y el modelo predice que no pertenece a ninguna entidad. Ejemplo: Real → **O**; Predicción → **O**.

Las métricas más habituales para evaluar el desempeño de un modelo, así como la forma de obtenerlas a partir de la matriz de confusión son:

- **Precision:** Representa la proporción de muestras que se han clasificado correctamente como positivas del total de ejemplos que se han clasificado como positivos. La fórmula para obtener la *precision* de un modelo a partir de la matriz de confusión es la siguiente:

$$precision = \frac{VP}{VP + FP} = \frac{clasificados\ correctamente\ como\ positivos}{todos\ los\ clasificados\ como\ positivos}$$

  Donde lo ideal es que *precision = 1*

- **Recall:** Representa la proporción de muestras que se han clasificado correctamente como positivas de la totalidad de los ejemplos positivos. La fórmula para obtener el *recall* de un modelo a partir de la matriz de confusión es la siguiente:

$$recall = \frac{VP}{VP + FN} = \frac{clasificados\ correctamente\ como\ positivos}{todos\ los\ positivos}$$

  Donde lo ideal es que *recall = 1*

- **F1:** El F1 o *F1-Score* es la métrica que representa el desempeño del modelo. Su cálculo viene dado a partir de las métricas *precision* y *recall* y se obtiene a partir de la siguiente fórmula:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

  Donde lo ideal es que *F1 = 1*

Un punto importante que destacar es que las métricas descritas están diseñadas para clasificadores binarios, es decir, clasificadores en los que únicamente hay dos clases posibles a predecir. Esto es algo prácticamente imposible en un modelo de NER, ya que por norma general, este deberá ser capaz de clasificar una palabra entre todas las entidades posibles para las que ha sido entrenado (casi siempre serán más de dos tipos de entidades). La solución a este problema pasa por transformar la evaluación de un modelo multiclase,

a un modelo binario. Para ello, lo que se hace es obtener por separado las métricas de cada una de las entidades, de tal forma que la clasificación de esa clase pasa de ser multiclase a binaria. Por lo tanto, la evaluación se resume en calcular si una determinada instancia pertenece a la clase que se está evaluando o no. Teniendo esto en cuenta, el cálculo de la *precision* y del *recall* pasan de calcularse de la siguiente forma:

$$precision = \frac{1}{N}\sum_{N=1}^{N} precision\ para\ la\ entidad\ N$$

$$recall = \frac{1}{N}\sum_{N=1}^{N} recall\ para\ la\ entidad\ N$$

$$F1 = \frac{1}{N}\sum_{N=1}^{N} F1\ para\ la\ entidad\ N$$

Donde N = *Tipos de entidades distintas que el modelo clasifica*, y lo ideal continúa siendo que *precision = 1*; *recall = 1*; y *F1 = 1*.

3.2.2. Librerías orientadas al Reconocimiento de Entidades Nombradas

A lo largo del apartado anterior se ha descrito el proceso necesario para llevar a cabo la creación y evaluación de un sistema NER. Sin embargo, el desarrollar soluciones *ad-hoc* desde cero, es un proceso complejo y costoso, por lo que utilizar herramientas y librerías que automaticen o faciliten el trabajo es un punto fundamental a la hora de elaborar un sistema NER. En este apartado se describirán a grandes rasgos algunas de las herramientas más populares tanto para el etiquetado de documentos, como para el entrenamiento y evaluación de los modelos, así como sus principales características.

3.2.2.1. Herramientas para etiquetado de documentos

Dentro de las herramientas enfocadas a etiquetar los documentos que conformarán el Corpus de datos, así como exportar las anotaciones realizadas utilizando alguno de los formatos mencionados en el Apartado 3.2.1, las principales herramientas son:

- **Prodigy:** Prodigy[9] es una herramienta de anotación con interfaz web basada en Python, orientada a crear conjuntos de datos de entrenamiento y evaluación para modelos de Aprendizaje Automático. Estos conjuntos

[9]https://prodi.gy *(Visitado el 02-07-2021).*

de datos no se limitan únicamente a tareas de procesamiento de textos, si no que también se pueden generar datos para tareas de visión por computador, anotación de videos, audios, etc. Así mismo, Prodigy proporciona distintos flujos de trabajo en función de las necesidades del usuario:

- **Manual 100 %:** En este flujo de trabajo se presentan los documentos en texto plano y se va realizando el etiquetado de estos de una forma totalmente manual (seleccionando un fragmento del texto e indicando su clase).
- **Manual con sugerencias por patrones:** Este flujo de trabajo es similar al modo manual con la diferencia de que durante el etiquetado, Prodigy realiza sugerencias de etiquetado de entidades en función de los patrones que se han establecido previamente en su fichero de configuración. Esto permite que entidades con una estructura fija como números de teléfono, DNI, cuentas bancarias, etc, se etiqueten automáticamente, lo que facilita el proceso de etiquetado.
- **Manual con sugerencias del modelo:** Este flujo de trabajo es idéntico al anterior con la diferencia de que en lugar de recibir las sugerencia a partir de patrones y reglas definidas previamente, las sugerencias se generan mediante un modelo NER previamente entrenado. Esto es útil cuando se dispone de un buen modelo y se quiere añadir soporte para un nuevo tipo de entidad. En este caso, el modelo se encargaría de detectar las entidades antiguas, y únicamente habría que etiquetar manualmente el nuevo tipo de entidad a soportar, o corregir las anotaciones del modelo (semi-supervisado).
- **Utilizando Aprendizaje Activo:** Este flujo de trabajo se utiliza para refinar un modelo ya existente. El modelo creado va realizando las sugerencias de las entidades, y estas se corrigen manualmente (en el caso de que estén mal). La diferencia entre este flujo de trabajo y el anterior, es que el modelo es entrenado teniendo en cuenta las correcciones realizadas, lo cuál hace que vaya aprendiendo a medida que se corrigen las sugerencias que este haya efectuado de forma errónea. De este modo se va realizando un aprendizaje continuo y activo a medida que se va corrigiendo el etiquetado que el modelo realiza sobre los textos.

Una de las principales ventajas que tiene Prodigy frente al resto de herramientas es su integración con las funcionalidades de Spacy (ver Apartado 3.2.1). Esto permite por un lado que los datos etiquetados generados con esta herramienta sean totalmente compatibles con cualquier

modelo de Spacy, y que los modelos generados por Spacy puedan ser utilizados y refinados empleando flujos de trabajo de aprendizaje activo. Por otra parte, su principal desventaja es que es un software de pago.

- **Datasaur:** Datasaur[10] es una herramienta web de etiquetado orientada al procesamiento de textos (etiquetado de entidades, dependencias entre palabras, co-referencias, y otras tareas de etiquetado de textos). A diferencia de Prodigy, no cuenta con integración de modelos personalizados de aprendizaje, o sugerencias en base a reglas. Sin embargo, cuenta con funcionalidades que permiten realizar el etiquetado de una forma más rápida (selección automática de palabras, asignación de etiquetas a teclas, etc). Por otra parte, todo el proceso de anotación se lleva a cabo desde una aplicación web alojada en sus servidores, lo que permite almacenar las anotaciones y continuar en cualquier momento desde diferentes equipos. No obstante, el principal inconveniente es que al alojar los datos en un lugar ajeno sobre el que no se dispone de un control total, hace que se pierda la privacidad de los mismos.

  Alguna de las características más relevantes de Datasaur son:

  - **Atajos de teclado para entidades:** Una de las características de Datasaur es su integración con el teclado mediante un manejo de la herramienta basado en atajos. Un ejemplo de esto es la posibilidad de asociar a cada número del teclado una entidad, de tal forma que permite ir etiquetando sin la necesidad de utilizar el ratón. Además de esto, una vez asignada la entidad a una palabra mediante el teclado, el propio software pasa automáticamente a la siguiente palabra para continuar con la anotación. Del mismo modo, en el caso de entidades con múltiples palabras, es posible realizar una selección múltiple con los atajos convencionales (*shift + flechas de control*) y etiquetar la entidad.
  - **Exportación en múltiples formatos:** Una vez se ha realizado la anotación de los documentos, Datasaur permite realizar la exportación de estos en distintos formatos. Esto es una característica interesante ya que hace que pueda ser utilizado en un mayor número de modelos, al ser compatible con distintos formatos. Entre los formatos de exportación disponibles se encuentra TSV y JSON con formato de etiquetado IOB.
  - **Modificación del texto:** A menudo, dependiendo de la fuente de origen de la que se ha obtenido el texto a etiquetar, puede ser habitual que este contenga errores gramaticales o de formato (tildes, enes en lugar de eñes, palabras sin espacios, etc). Para solucionar esto, Datasaur permite editar el texto desde la propia

[10] https://datasaur.ai *(Visitado el 02-07-2021).*

herramienta, todo esto manteniendo las anotaciones que pudieran tener las palabras modificadas (siempre y cuando el número nuevo de palabras de una anotación sea el mismo que el antiguo).

- **Extensiones:** La herramienta web de Datasaur proporciona distintas extensiones que facilitan el flujo de anotación de documentos. Alguna de estas extensiones son la vista de documentos (listado con los documentos que contiene el proyecto), diccionario, creador dinámico de entidades (para crear, eliminar o modificar las entidades que se están utilizando), información sobre el documento actual (número de filas, número de entidades anotadas, tipos de entidades, etc) o buscador de palabras.
- **Soporte con los modelos base de otras plataformas:** Datasaur dispone de integración con los modelos base de otras herramientas como Spacy. Esto le permite poder etiquetar de forma automática entidades y tipos de palabras que soporte el **modelo base de Spacy**. Esto puede ser útil cuando no es necesario etiquetar entidades personalizadas.

3.2.2.2. Librerías para la creación de modelos de NER

Desde el punto de vista de las librerías destinadas a la creación y evaluación de modelos orientados al Reconocimiento de Entidades Nombradas, existen diferentes alternativas en función del lenguaje de programación sobre el que se basan, las funcionalidades que ofrecen, su tipo (si son *open-source* o privativas), los idiomas de texto que estas soportan, etc. Alguna de las principales librerías para la creación de modelos orientados al NER son las siguientes:

- **Spacy:** Spacy[11] es una librería de código abierto escrita en Python, que fue desarrollada por Matt Honnibal en 2015. Esta librería permite realizar distintas tareas orientadas al Procesamiento del Lenguaje Natural (Reconocimiento de Entidades Nombradas, análisis de dependencias entre palabras, asignación de las categorías sintácticas del texto, etc). Además de esto, proporciona otras funcionalidades que complementan las tareas, como la tokenización de texto (división del texto entre las palabras y elementos que lo conforman), soporte para más de 64 idiomas, modelos preentrenados para algunos lenguajes, o visualizadores para poder mostrar de una forma sencilla y gráfica las entidades, propiedades y dependencias entre los distintos elementos del texto. En este punto no se va a profundizar más sobre Spacy ya que, al ser la librería utilizada para la implementación del modelo en este proyecto, tendrá un apartado íntegro asociado posteriormente (ver Apartado 3.3).

[11] https://spacy.io *(Visitado el 01-07-2021).*

- **NLTK:** NLTK[12] (*Natural Language Toolkit*, en inglés), es una librería de código abierto escrita en Python que ofrece herramientas para el procesamiento simbólico y estadístico orientado al Procesamiento del Lenguaje Natural. Además de funcionalidades enfocadas al PLN como la tokenizacion o la visualización de dependencias, NLTK se caracteriza por ser una librería orientada a la investigación y al aprendizaje del PLN. Es por esto, por lo que proporciona una gran cantidad de demostraciones gráficas de tareas, una colección de datos de ejemplo para comenzar a desarrollar, e incluso un libro[13] en el que se van tratando los aspectos más relevantes del PLN. No obstante, para llevar a cabo un desarrollo de Reconocimiento de Entidades Nombradas como el que se está realizando en este trabajo, la librería NLTK se queda lejos de ofrecer toda la funcionalidad requerida. Debido a esto, la mayoría de los proyectos NER que utilizan esta librería, están desarrollados en base a un sistema mixto. Este sistema mixto combina funcionalidades de NLTK para la extracción y el procesamiento de los textos, con funcionalidades de otras librerías como Spacy para la creación y evaluación de los modelos creados.

- **Google Cloud NLP:** El módulo de NLP que ofrece la plataforma Google Cloud[14] proporciona herramientas para llevar a cabo cualquier tipo de tarea PLN (análisis de sentimientos, extracción de textos de audios o imágenes, NER, acceso a los modelos de Deep Learning que utiliza Google, etc). A diferencia de Spacy y NLTK, Google Cloud no proporciona soporte local, por lo que todo tanto los datos como el procesamiento han de estar alojados en los servidores de Google. Este módulo de PLN se encuentra formado por los siguientes componentes:

  - **AutoML Natural Language:** Proporciona las herramientas y la funcionalidad de *AutoML* para entrenar modelos personalizados de PLN que sean capaces de extraer y clasificar opiniones. Estos modelos se crean y entrenan sin la necesidad de escribir ninguna línea de código y bajo una interfaz gráfica que facilita el proceso. Existen diferentes tipos de *AutoML*[15] en función de la tarea a resolver: *AutoML Natural Language* para el análisis de textos, *AutoML Translator* para traducir textos de forma dinámica, *AutoML Vision* para vídeos, etc.
  - **API de Natural Language:** Este API permite a los desarrolladores aplicar funciones de Comprensión del Lenguaje Natural (*Natural Language Comprension*, CLN, en inglés), a las aplicaciones desarro-

[12] https://www.nltk.org *(Visitado el 03-07-2021).*
[13] https://www.nltk.org/book/ *(Visitado el 03-07-2021).*
[14] https://cloud.google.com/natural-language *(Visitado el 04-07-2021).*
[15] https://cloud.google.com/automl *(Visitado el 04-07-2021).*

lladas. Alguna de estas funciones CLN son el análisis de entidades nombradas, análisis sintáctico del texto, análisis de opiniones, etc.

- **API Natural Language de Healthcare:** Este módulo de Google Cloud permite realizar un análisis en tiempo real de datos que se encuentren en textos médicos sin estructurar. Además permite combinar funciones de AutoML para crear modelos personalizados y que los datos y entidades extraídas de estos textos médicos sean personalizadas en función de cada caso de uso.

3.3. Spacy

Spacy es una librería de código abierto desarrollada en Python que ofrece una gran cantidad de funcionalidades para la realización de proyectos de PLN (Reconocimiento de Entidades Nombradas, análisis de dependencias entre palabras, clasificación del texto en función de su sintaxis, etc). Algunas de las características principales de Spacy son las siguientes:

- Dispone de soporte para más de 64 lenguajes y 55 flujos de trabajo preentrenados para 17 lenguajes. Entre estos flujos de trabajo preentrenados se encuentran flujos destinados a la transformación de palabras a vectores (*word-vectors*), o modelos basados en Transformers [40], como por ejemplo BERT [11].

- El procesamiento interno de los componentes se encuentra implementado en base al estado del arte, lo que proporciona una mayor velocidad y eficiencia.

- La arquitectura de procesamiento de documentos está implementada de forma modular como una tubería (*Pipeline*), en la que cada funcionalidad actúa como un componente aislado. Estos componentes reciben el resultado del componente anterior (independientemente de cuál sea), lo procesa, y lo envía al siguiente componente (sin importar cuál sea). Esta implementación aislada e independiente, permite realizar modificaciones tanto en el orden de ejecución como en las propiedades de los componentes, de una forma sencilla. Entre los componentes disponibles por defecto destacan el tokenizador, el detector de entidades o el detector morfológico.

- Permite añadir y eliminar componentes en el *Pipeline* de una forma simple y personalizable. Por un lado, en el caso de necesitar eliminar componentes en el flujo (porque no se van a utilizar, y se quiere reducir el tiempo de cómputo), simplemente se especifica en la configuración. Por otro lado, en el caso de querer añadir nuevos componentes personalizados, únicamente hay que tener en cuenta que la salida de este nuevo componente sea la

que espera Spacy, y añadir en la configuración el nuevo componente del flujo.

- Ofrece soporte para modelos personalizados en otros frameworks de Aprendizaje Automático como PyTorch o TensorFlow.

3.3.1. Arquitectura

La arquitectura de Spacy se divide en componentes aislados que interactúan entre sí (ver Figura 3.3.1). Gracias a esto, se consigue mejorar el mantenimiento y personalización del sistema, ya que, en el caso de necesitar modificar algún parámetro de algún componente, se puede modificar directamente en el componente necesario (sin afectar al resto). Los componentes que conforman la arquitectura de Spacy son:

- **Language:** El *Language* es el componente central, y el punto de entrada de cualquier modelo de Spacy. Este componente es el que se carga cada vez que se importa un modelo para su uso. En ella se almacenan los metadatos asociados al idioma que se está utilizando, su vocabulario (componente *Vocab*), y los componentes que conforman el conjunto de transformaciones y operaciones a aplicar sobre el documento (componente *Pipeline*).

- **Vocab:** Contiene la información sobre el vocabulario del idioma utilizado. Este vocabulario puede estar definido por defecto (a partir de los modelos de idioma que Spacy proporciona), o puede ser personalizado. En cualquiera de los dos casos, el componente *Vocab* almacena la representación de cada palabra en texto plano y su hash (en el *String Store*), y en forma vectorial (en el *Vectors*). Estas representaciones se utilizarán para llevar a cabo el proceso de *Word Embedding* (proceso por el cual se transforma cada palabra en texto plano a su representación vectorial), para que pueda ser interpretada correctamente por el modelo.

- **Pipeline:** El *Pipeline* o tubería, es el componente que más se va a modificar desde el punto de vista del desarrollo. Esto se debe a que en ella se encuentran definidos tanto el modelo de Aprendizaje Automático, como las distintas operaciones y/o transformaciones que se aplicarán sobre el texto para resolver la tarea objetivo (reconocimiento de entidades, relación entre palabras, etc).

- **Doc:** Es la unidad de representación de textos en Spacy. Un *Doc* representa un contenedor constituido por una secuencia de *tokens* junto con sus etiquetas (entidad a la que pertenece ese *token*, morfología, sintaxis, etc). Todo modelo de Spacy se aplica sobre un objeto *Doc* a cuyos *tokens* se les añade la etiqueta de la categoría a la que estos

pertenezcan. Este tipo de objeto será también el que se utiliza para el entrenamiento, validación y evaluación de los modelos, así como para la visualización de los resultados. A su vez, el objeto *Doc* se encuentra constituido por los objetos *Token* (cada uno de los *tokens* del texto y sus etiquetas asociadas), y el objeto *Span* (agrupación de objetos del tipo *Token*).

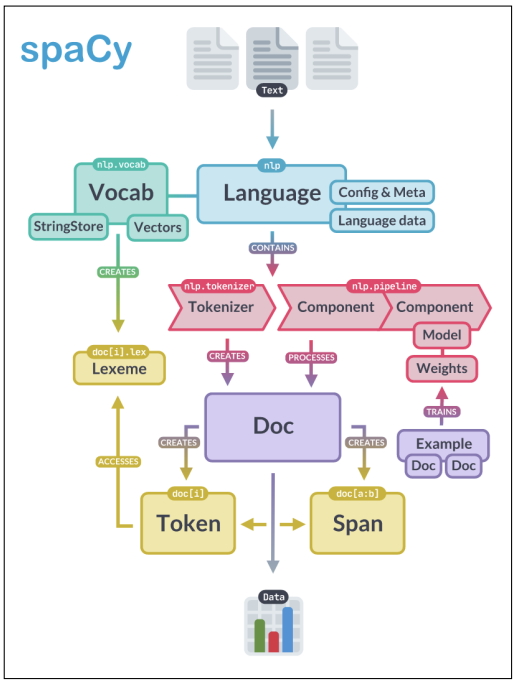


Figura 3.2: Arquitectura de Spacy. Fuente: *https://spacy.io/api (Visitado el 29-06-2021).*

3.3.2. Pipeline

El *Pipeline* define el conjunto de operaciones y transformaciones (así como su orden) que se aplicarán sobre el texto. Existen diferentes *Pipelines* que ofrece Spacy de forma nativa para tareas de análisis morfológico, sintáctico, o para desambiguación (ver Figura 3.3). No obstante, este componente es totalmente personalizable y se pueden eliminar y añadir componentes personalizadas para un caso de uso más específico. Los componentes que conforman el *Pipeline* base de Spacy así como su funcionalidad son los siguientes:

- **tok2vec:** El componente *tok2vec* es la primera transformación que se le aplica a un texto. En este punto se divide el texto de entrada en los distintos *tokens* que lo conforman para poder ser procesados posteriormente. Dicho de otro modo, hace la labor de un tokenizador.

- **tagger:** El *tagger* es el encargado de "escuchar" al *tok2vec*, e ir asignando a cada *token* que este genere la etiqueta correspondiente a su categoría gramatical (nombre, pronombre, verbo, adjetivo, etc). La asignación de una etiqueta u otra vendrá dada en función de las reglas definidas en *attribute_ruler*.

- **parser:** Al igual que ocurre con el *tagger*, el *parser* se encuentra "escuchando" al *tok2vec* a la espera de *tokens* para asignar a cada uno de ellos la etiqueta UPOS[16] (*Universal POS Tags*) correspondiente. Nuevamente, la asignación de una etiqueta u otra vendrá dada en función de las reglas definidas en *attribute_ruler*.

- **attribute_ruler:** El componente *attribute_ruler* define el conjunto de reglas que se utilizarán para asignar a un determinado *token* su etiqueta UPOS correspondiente. Este componente dispone de reglas definidas por defecto para una gran variedad de idiomas. No obstante, es posible añadir nuevas reglas en el caso de que fuese necesario.

- **lemmatizer:** El *lemmatizer* define el conjunto de reglas que se utilizarán para asignar a cada uno de los *tokens* la etiqueta correspondiente a su lema (forma base de una palabra). Al igual que ocurría con el *attribute_ruler*, este componente ya dispone de reglas predefinidas para una gran variedad de idiomas. No obstante, es posible añadir nuevas reglas en el caso de que fuera necesario.

- **ner:** El ner es el componente encargado de procesar los *tokens* y asignar a cada uno de ellos la categoría a la que este pertenece.

Además de estos componentes, Spacy ofrece la posibilidad de añadir componentes personalizados a este *Pipeline* para mejorar o extender su funcionalidad. Alguno de los componentes extra que se pueden añadir son:

- **EntityRuler:** Este componente permite añadir etiquetas del tipo de entidad a los *tokens* del texto. Para ello utiliza reglas basadas en coincidencias de palabras o por coincidencias exactas de frases. Por ejemplo: *"a todos los tokens **"patata"** en el texto, se les asignará la etiqueta de entidad **"Hortaliza"**"*.

- **Matchers:** Los *Matchers* son componentes que permiten asignar a un *token* (o conjunto de *tokens*) una etiqueta utilizando reglas basadas en sus atributos. Este componente va un paso más allá que el *EntityRuler* ya que permite asignar más de un tipo de etiqueta (no solo la etiqueta de categoría). Otra característica es que permite crear reglas más complejas

[16] https://universaldependencies.org/docs/u/pos/ *(Visitado el 28-06-2021).*

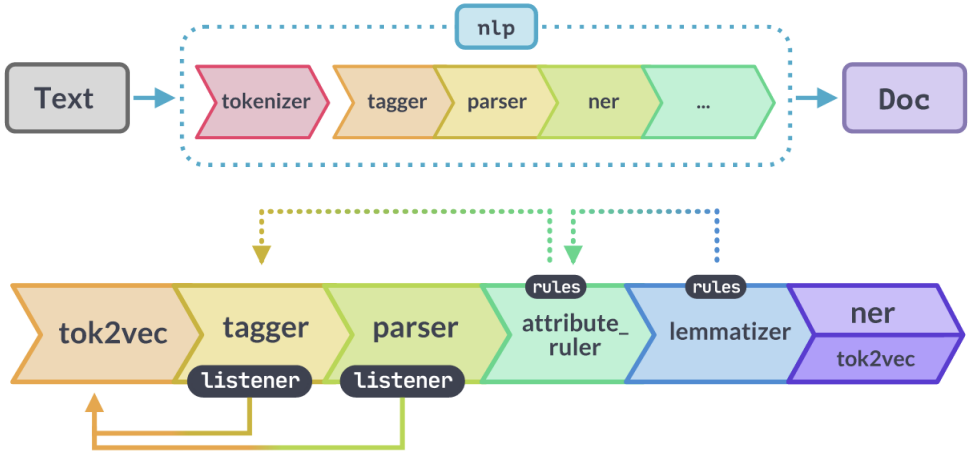


Figura 3.3: *Pipeline* de Spacy. Fuente: *https://spacy.io/api (Visitado el 29-06-2021).*

que la coincidencia exacta de *tokens*. Algunos ejemplos de reglas que soporta este componente son las asignaciones de etiquetas UPOS en base al lema del *token*, asignación de etiquetas de categoría utilizando expresiones regulares, o aplicar una combinación de condicionales: asignar una determinada etiqueta en base al texto, lema, longitud del *token* o cualquier otro atributo que este tenga.

3.3.3. Displacy

Una de las funcionalidades que dispone Spacy es la posibilidad de crear visualizaciones de los textos (objetos *Doc*), y sus atributos o entidades. Estas visualizaciones son generadas utilizando el paquete **Displacy**[17] (el cual se encuentra incluido en la librería de Spacy). Displacy permite generar visualizaciones de dos tipos:

- **Dependencia:** En este tipo de visualizaciones se muestran los atributos asociados a los *tokens* del documento, así como las dependencias que existen entre ellos (ver Figura 3.4).
- **Entidad:** Este tipo de visualización muestra las distintas entidades que existen en el documento, así como su tipo y los *tokens* que pertenecen a ella (ver Figura 3.5).

[17] https://spacy.io/usage/visualizers *(Visitado el 30-06-2021).*

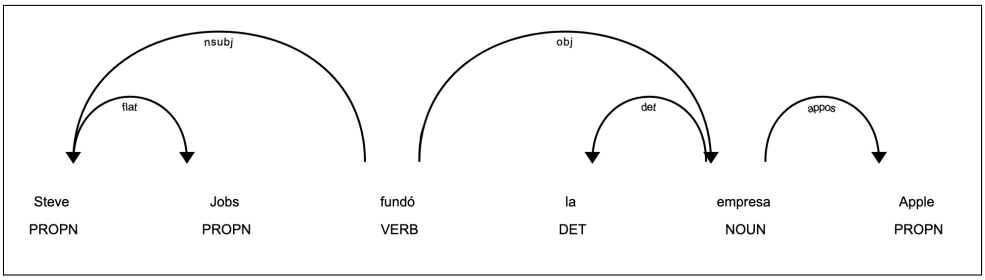


Figura 3.4: Ejemplo de Visualización en Displacy en modo "Dependencia".

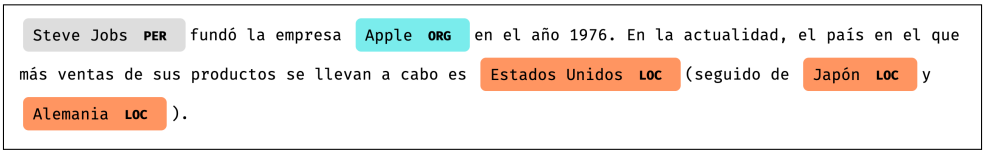


Figura 3.5: Ejemplo de Visualización en Displacy en modo "Entidad".

La gran ventaja de estas visualizaciones es que pueden ser generadas automáticamente a partir de los atributos y etiquetas de un documento, o bien pueden ser generadas en base a los atributos de un fichero JSON personalizado (Displacy interpretará ese JSON como un *Doc* con los atributos y etiquetas que se especifiquen en el propio JSON). Esta característica es interesante ya que permite generar visualizaciones sin la necesidad de tener que crear un objeto *Doc* y aplicar un *Pipeline* para predecir sus atributos y entidades. Otra característica que ofrece Displacy es la posibilidad de exportar las visualizaciones como imágenes o como código HTML. Esto permite que puedan ser utilizadas en la gran mayoría de aplicaciones (páginas webs, aplicaciones de escritorio, etc), además de disponer de soporte para cuadernos de Jupyter.

3.4. Anonimización de Documentos

La anonimización o de-identificación[18] es un proceso en el cual se llevan a cabo un conjunto de técnicas y operaciones para evitar que se obtenga la identidad personal de alguien. En otras palabras, el objetivo principal de estas técnicas consiste en eliminar la asociación entre una persona y un texto [3]. Teniendo esto en cuenta, se pueden establecer tres tipos de identificación en función del estado en el que se encuentren los datos (ver Figura 3.6):

- **Identificado:** En esta categoría se encuentran aquellos documentos en cuyos datos se encuentra información directa (nombres, direcciones, identificaciones fiscales, etc) para poder identificar a una persona. No

[18] https://www.truevault.com/resources/developer/what-is-data-de-identification *(Visitado el 28-06-2021).*

obstante, en este punto hay que hacer una distinción entre **identificado** e **identificable**. Una persona o individuo es identificable en un conjunto de datos siempre que sea razonable esperar que, con los datos disponibles de forma inmediata (nombres, direcciones, etc) o con otra información externa (metadatos del documento, procedencia, etc) sea posible identificarle de forma unívoca. Por otro lado, un individuo es identificado cuando se conocen (y se pueden asociar directamente) los datos del documento a dicho individuo. Es importante hacer esta distinción ya que no siempre el disponer de datos de carácter personal hace que sea posible identificar a un individuo.

- **Pseudonimización:** En esta categoría se encuentran aquellos documentos en los que los identificadores directos han sido sustituidos por pseudónimos o por identificadores falsos que impiden identificar al individuo de forma directa. No obstante, los identificadores indirectos no se ven afectados, por lo que una persona puede continuar siendo identificable tras realizar este procesamiento. La correspondencia entre las menciones originales y los pseudónimos utilizada puede ser almacenada para asegurar la reversibilidad del proceso.

- **Anonimizado:** En esta categoría se encuentran los documentos en los que se han eliminado tanto los identificadores directos como los indirectos. Esto añade un nivel más de seguridad que en el caso de la pseudonimización ya que al no disponer de ningún tipo de identificador, no se puede llegar a la identidad del individuo original.

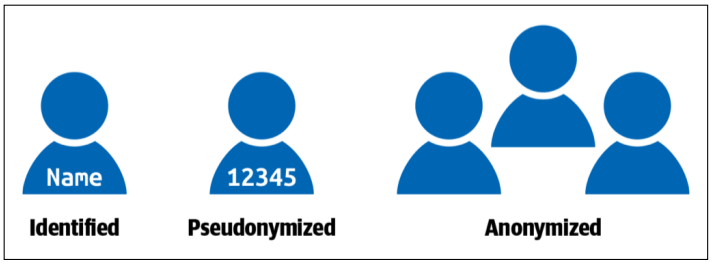


Figura 3.6: Tipos de anonimización en función del estado del documento. Fuente: [3].

Por norma general, el objetivo marcado consiste en transformar los documentos con un estado **identificado**, a un documento **pseudo-anonimizado** o **anonimizado**. La diferencia entre uno u otro dependerá en gran medida del uso que se le vaya a dar a los documentos resultantes. Por ejemplo, en un entorno jurídico puede ser más interesante sustituir las menciones personales por otras realistas para facilitar la comprensión del documento y distinguir entre los distintos tipos de menciones. Por lo tanto en este caso se utilizaría

la pseudonimización. Del mismo modo, en el caso de documentos en los que lo importante es el qué en lugar del quién (por ejemplo en una resolución administrativa que afecta a un particular), se utilizaría la anonimización del documento.

Dentro de este proceso de anonimización existen diferentes técnicas en función del resultado final que se desee. A continuación se indican las técnicas más habituales para pasar de un documento identificado a uno pseudo-anonimizado o anonimizado [22]:

- **Eliminación de identificadores:** Este tipo de técnicas es la más básica e intuitiva en materia de anonimización. Consiste en eliminar el identificador del documento anonimizado de tal forma que este no sea visible y no aparezca en el nuevo documento. La principal desventaja que tienen este tipo de técnicas es que al eliminar partes del documento sin sustituirlas por otras, dificulta la lectura y la interpretación del texto (ya que en lugares donde deberían aparecer identificadores no aparece nada). Por otro lado, una de las grandes ventajas es que en términos de ejecución es muy rápido, y en términos de computación es eficiente y fácil de implementar.
- **Generación de mecanismos de reemplazo:** Este tipo de técnicas tienen como objetivo el suplir la pérdida de legibilidad del documento anonimizado que sufren las técnicas de anonimización basadas en la eliminación de las menciones. Para ello lo que se hace es sustituir cada uno de los identificadores por caracteres, u otras entidades del mismo tipo que la original. Con esto se consigue tanto mantener la legibilidad del documento, como hacer que el mismo parezca más natural. Dentro de este tipo de reemplazos existen diferentes técnicas en función de la forma en la que se realice tanto la generación, como la aplicación de los reemplazos. Entre estas técnicas, las más habituales son:
  - **Supresión:** Consiste en sustituir el identificador a anonimizar por una secuencia de caracteres fija (*****, ######, ——, etc).
  - **Reemplazo por el tipo de la entidad-identificador:** Consiste en sustituir el identificador a anonimizar por el texto plano correspondiente al tipo de identificador al que pertenece (nombre, dirección, teléfono, etc).
  - **Reemplazo por entidades-identificadores sintéticos:** Este tipo de técnicas consiste en generar de forma artificial un identificador del mismo tipo que el que se quiere anonimizar y reemplazarlo por el original. La generación del nuevo identificador puede realizarse de diversas formas en función del formato que este tenga, por ejemplo en el caso de los números de teléfono, bastaría con generar un número

aleatorio entre 600000000 y 699999999. Otra forma de generar estos reemplazos sintéticos es mediante diccionarios (estructuras de datos que contienen reemplazos ya definidos para unos determinados identificadores) u otra estructura similar. De este modo, si ahora se quiere generar un nombre sintético únicamente habría que ir a la estructura de datos correspondiente y tomar uno de los valores de nombres que se tienen almacenados. Dentro de la generación de reemplazos sintéticos, existen diferentes categorías en función del tipo de estructura de datos utilizada para generar los reemplazos:

- **Estáticos:** Los reemplazos almacenados en la estructura de datos empleada es siempre la misma.
- **Dinámicos:** Los reemplazos almacenados en la estructura de datos empleada varía a medida que se anonimizan documentos. En estos casos, se parte de un conjunto de reemplazos base y a medida que se van sustituyendo identificadores en el documento, se actualizan también en la propia estructura de datos. De este modo, a medida que se anonimizan los documentos, se va enriqueciendo el catálogo de reemplazos.

En la siguiente tabla (ver Tabla 3.3) se muestra un ejemplo de anonimización de un texto utilizando las técnicas anteriormente descritas.

Original	**Eliminación**	**Supresión**
Hola mi nombre es **Juan Sánchez García** y mi número de teléfono es **623904582**	Hola mi nombre es y mi número de teléfono es	Hola mi nombre es ****** ******* ******** y mi número de teléfono es *************

Original	**Tipo de Identificador**	**Identificadores Sintéticos**
Hola mi nombre es **Juan Sánchez García** y mi número de teléfono es **623904582**	Hola mi nombre es **NOMBRE APELLIDO APELLIDO** y mi número de teléfono es **TELÉFONO**	Hola mi nombre es **Paco Gómez Casas** y mi número de teléfono es **674560147**

Tabla 3.3: Ejemplos de anonimización de textos utilizando distintas técnicas.

3.5. Reconocimiento Óptico de Caracteres (OCR)

El Reconocimiento Óptico de Caracteres (*Optical Character Recognition*, OCR, en inglés), es un proceso por el cual un software es capaz de detectar los

caracteres que se encuentran en una imagen (fotografía o documento escaneado), de tal forma que estos sean reconocidos por un ordenador, pudiendo interactuar con estos posteriormente mediante software de visualización o editores de texto. El proceso que sigue un OCR para llevar a cabo la detección del texto en una imagen es el siguiente:

1) **Preprocesamiento de la imagen:** La primera etapa que se lleva a cabo es la de realizar un preprocesamiento de la imagen a tratar. Este proceso consiste en aplicar sobre la imagen distintos filtros que faciliten la extracción del texto en las etapas posteriores. Alguno de los filtros a aplicar son el reescalado de la imagen, la binarización (normalización de los píxeles), transformación a blanco y negro, etc. A pesar de que esta etapa no es obligatoria dentro del flujo del OCR, es altamente recomendable, ya que ayuda a mejorar el desempeño de las etapas posteriores, lo que se traduce en unos mejores resultados en el reconocimiento de los caracteres de la imagen.

2) **Segmentación:** Una vez se ha llevado a cabo todo el preprocesamiento de la imagen, el siguiente paso consiste en dividir la imagen en distintas zonas donde se encuentra situado el texto (ver Figura 3.8). Esta división se realiza en dos fases:

   - **Extracción de regiones de interés:** En esta primera fase se lleva a cabo una división de la imagen en distintas áreas de donde se encuentra el texto a procesar. Estas áreas reciben el nombre de Regiones de Interés (*Region Of Interest*, ROI, en inglés). Por ejemplo, en el caso de un documento, la división de la imagen en regiones de interés pueden ser los distintos párrafos que lo conforman. Del mismo modo, en el caso de una imagen en la que aparezca la fachada de un restaurante, la región de interés puede ser la zona en la que aparece el cartel con el nombre del restaurante. El objetivo de esta fase consiste en reducir el espacio de búsqueda, limitándose únicamente a aquellas zonas de la imagen que contienen texto.
   - **División de caracteres:** Una vez se han definido las distintas regiones de interés dentro de la imagen, el siguiente paso consiste en realizar una división de los caracteres dentro de esa región. Esta es la etapa más delicada y a la vez complicada, ya que una mala segmentación de los caracteres hace que estos no puedan ser reconocidos. En términos generales, la obtención de estos caracteres se realiza mediante el análisis de los píxeles, o métodos de proyección de histogramas (ver Figura 3.7), por lo que el hecho de haber preprocesado y normalizado la imagen toma un papel importante en esta etapa ya que ayuda a la detección de las zonas correspondientes a caracteres.

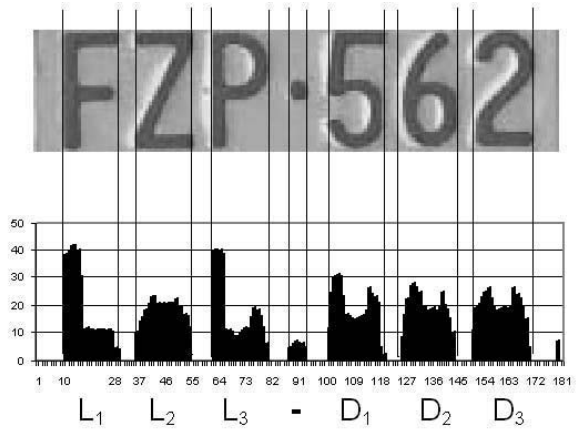


Figura 3.7: Ejemplo segmentación mediante histogramas. Fuente: [31].

Existen casos como los formularios, en los que esta delimitación de los caracteres es sencilla (ya que cada uno se encuentra en una celda distinta, como ocurre en el ejemplo de la Figura 3.8). No obstante, en el caso de textos manuscritos, esta tarea puede ser muy compleja ya que no se dispone de esta división tan inmediata.

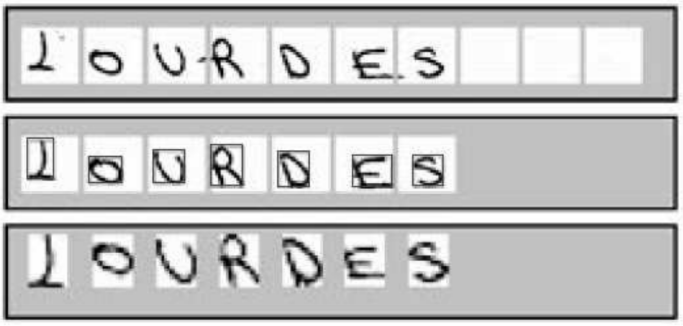


Figura 3.8: Ejemplo de normalización y segmentación de la imagen en formularios. Fuente: [12].

3) **Reconocimiento del carácter:** Finalmente, una vez se han obtenido los distintos caracteres del texto, el último paso consiste en identificar que carácter es. Esta identificación se lleva a cabo utilizando algoritmos de Aprendizaje Automático entrenados con grandes conjuntos de datos de caracteres. Alguno de los modelos utilizado para este tipo de tareas es el KNN (*K-Nearest-Neighbors*), los Árboles de Decisión, o las Redes Neuronales. No obstante, el avance en el campo del Deep Learning en materia de procesamiento de imágenes, ha hecho que estas propuestas sean las más utilizadas para este tipo de tareas.

En la actualidad existen muchos y muy diversos usos de este tipo de tecnología. Alguno de estos usos son los siguientes:

- Automatización de la extracción de datos en formularios.
- Reconocimiento de señales de tráfico en vehículos autónomos.
- Traducción del texto que aparece en una imagen.
- Digitalización de documentos.

Capítulo 4

Estado del Arte

En los últimos años, el creciente interés en la privacidad ha propiciado el desarrollo de sistemas de protección de datos. Estos sistemas requieren de mecanismos que permitan tanto detectar las entidades personales en los textos, como clasificarlas según su tipo. Asimismo, la entrada en vigor de leyes enfocadas a la protección de datos, en especial el Reglamento General de Protección de Datos (RGPD), ha ampliado significativamente el alcance de estos sistemas de detección y clasificación de menciones personales en textos.

La primera propuesta orientada a detectar entidades en textos se remonta al año 1991 en la *7º Conferencia IEEE en Aplicaciones de la Inteligencia Artificial*. Esta propuesta [32] estaba enfocada a reconocer únicamente nombres de compañías en textos escritos basándose en heurísticas y en reglas escritas (ver Figura 4.1). No obstante, a pesar de su sencillez, era capaz de extraer correctamente el 97,5 % de las compañías en los textos de evaluación utilizados.

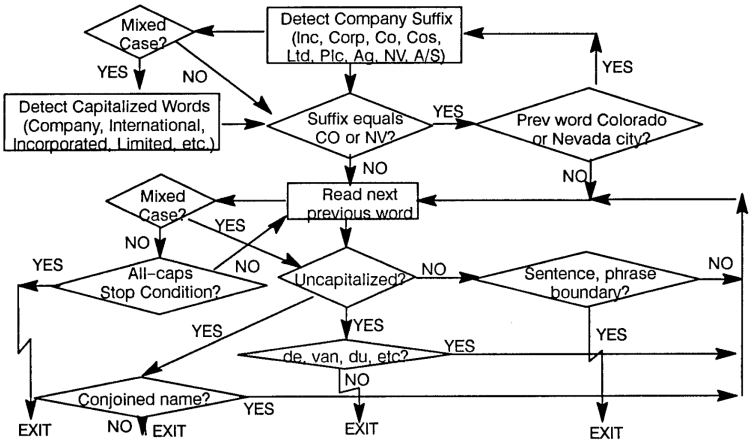


Figura 4.1: Flujo de detección y extracción de entidades del primer NER basado en heurísticas y en reglas escritas. Fuente: [32].

Estos resultados generaron un mayor interés en este campo, lo cual hizo que, en los siguientes años, aparecieran nuevas propuestas enfocadas a detectar un mayor número de tipos de entidades [37]. Estas propuestas se encontraban divididas según la metodología empleada para extraer las entidades de los textos. Por un lado, surge una nueva rama orientada a aplicar técnicas y modelos de Aprendizaje Automático sobre los textos (siendo el Aprendizaje Supervisado el tipo de Aprendizaje Automático más utilizado). En esta línea caben destacar las propuestas [34] y [4], en las que se plantean alternativas basadas en Árboles Binarios y Máquinas de Vectores de Soporte (*Support Vector Machines*, SVM en inglés) respectivamente. Ambos modelos permitían detectar entidades nombradas sin necesidad de tener que establecer previamente de forma manual tanto las reglas escritas, como las heurísticas necesarias para su detección. Gracias a este tipo de técnicas se consiguió incrementar el número de tipos de entidades detectadas de 1 (únicamente compañías), a 7 en el caso de la propuesta basada en Árboles Binarios, y a 8 en el caso de la propuesta basada en SVM. Por otra parte, y en cuanto a las métricas, se pasó de un 83,51 % de F1 en el modelo de Árbol Binario a un 87,21 % en las SVM.

Por otra parte, se continuó con la propuesta inicial (basada en heurísticas y en reglas escritas), mejorando tanto el número de entidades de distinto tipo a detectar, como los resultados en las métricas asociadas a la extracción de dichas entidades. En esta línea destaca la propuesta [36], en la que se desarrolló un sistema de detección de entidades jerárquico capaz de extraer hasta 150 tipos de entidades distintas (lo cual supuso un salto cuantitativo en cuanto al número de tipos de entidades detectadas). En relación con esta, se encuentra la propuesta [35] presentada en CONLL-2003, la cual permitía detectar 200 tipos de entidades en textos utilizando una combinación de reglas escritas a mano y diccionarios.

Sin embargo, en los últimos años el avance en el campo del Deep Learning ha hecho que las propuestas basadas en este tipo de técnicas se conviertan en predominantes (ver Figura 4.2). Esto hizo que las distintas metodologías que había hasta entonces enfocadas al reconocimiento de entidades nombradas perdiesen interés, y se tomase el Deep Learning como punto común sobre el que continuar avanzando [20]. Dentro de este campo existen diferentes propuestas en función de la arquitectura de la red neuronal utilizada:

- **Redes Neuronales Convolucionales:** Este tipo de arquitecturas fue propuesta en [9] y consiste en generar un vector de características globales del texto, para posteriormente asignar a cada palabra la probabilidad de que esta pertenezca a un determinado tipo de entidad. Alguna de las aplicaciones de este tipo de arquitecturas son [43], en el que se utiliza para detectar entidades biomédicas (proteínas, tipos de células, etc) utilizando

el corpus biomédico *GENIA*[1]. Otra propuesta similar es [42], en la que se detectan entidades médicas (enfermedades, pruebas, medicación, etc) en textos médicos escritos en chino.

- **Redes Neuronales Recurrentes:** Las arquitecturas basadas en redes recurrentes y sus variantes: *Gated Recurrent Unit* (GRU), *Long Short Term Memory* (LSTM) o *Bidirectional-RNNs*, se caracterizan por la existencia de conexiones en ambas direcciones entre nodos de la misma capa, y/o nodos de capas anteriores o posteriores, permitiendo procesar la información teniendo en cuenta el contexto (predicciones pasadas). Algunas propuestas en la que se utilizan este tipo de arquitecturas son [17] y [16], en las que se aprovechan las características de este tipo de modelos para reconocer y detectar entidades anidadas.

- **Transformers:** Esta arquitectura surgió a finales de 2017 desarrollada en Google. Su característica principal consistió en la sustitución de las capas recurrentes por las llamadas **capas de atención**. Estas capas permitían generar una representación matemática del contexto del texto, mejorando así la capacidad de extracción del conocimiento [40]. Algunos ejemplos de este tipo de arquitecturas son las *Generative Pre-trained Transformers* (GPT) [30], o *Bidirectional Encoder Representations from Transformers* (BERT) [11].

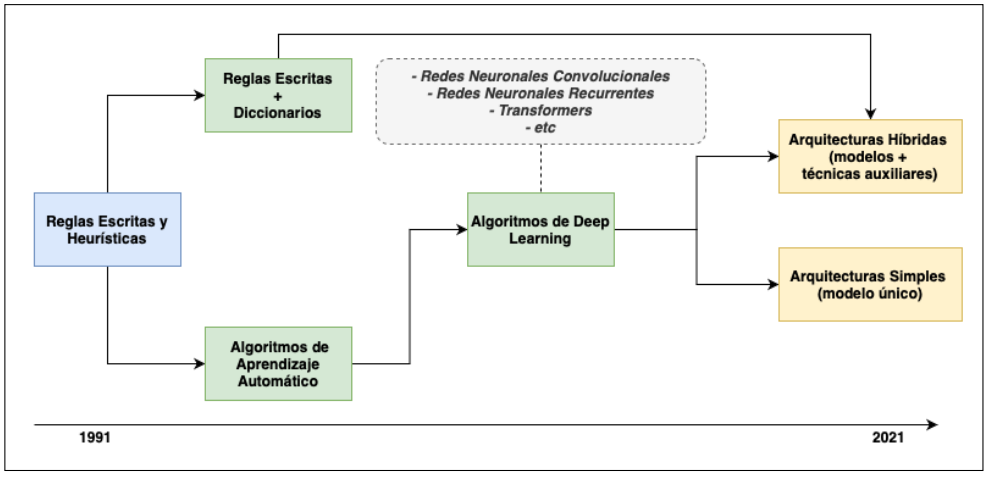


Figura 4.2: Tendencias de técnicas para la extracción de entidades nombradas en textos.

En cuanto al contexto en el que se encuentra enmarcado este proyecto (caso de uso y lenguaje empleado), el número de propuestas disminuye considerablemente. Desde el punto de vista del lenguaje del texto utilizado

[1]http://www.geniaproject.org/genia-corpus *(Visitado el 02-07-2021).*

destacan las propuestas [2] y [26], en las que se utilizan arquitecturas basadas en Redes Neuronales Convolucionales y Recurrentes para la detección de sustancias y/o componentes de medicamentos en textos en castellano. Otra propuesta englobada en esta línea es [24] en la que se aplican y evalúan todas las arquitecturas presentadas anteriormente para la detección de patologías asociadas al cáncer en diagnósticos médicos.

Desde el punto de vista de la detección de entidades de carácter personal en textos, existen diferentes propuestas en función del enfoque empleado. Los enfoques simples se basan en la utilización de una única técnica para la detección de entidades. Un ejemplo es [10], en el que se utiliza una arquitectura basada en Redes Neuronales Convolucionales con varios *encoders*, que permiten mejorar la contextualización del texto. Gracias a esta arquitectura y a esa mejora, se puede realizar una clasificación de "grano fino" de 134 entidades de carácter personal (nombre, correo electrónico, número de teléfono, prefijo telefónico, extensión, tarjeta de crédito, etc). Otro tipo son los enfoques híbridos, en los que se utilizan modelos de Aprendizaje Automático en combinación con técnicas y/o modelos auxiliares que permiten complementar las detecciones realizadas por estos modelos. En esta línea se encuentra la propuesta [23], en la que se utiliza el algoritmo de Aprendizaje Supervisado basado en Campos Aleatorios de Márkov (*Márkov Random Fields*, MRF en inglés), en combinación con diccionarios de inferencia para detectar nombres de personas en correos electrónicos. Otra propuesta similar a la anterior es [7], en la que se combinan modelos de redes neuronales con una detección basada en reglas escritas. En esta propuesta se obtienen entidades de carácter personal (nombre, dirección postal, etc) en informes policiales.

Una característica importante que destacar es que, en todas las propuestas presentadas a lo largo de la presente sección, el texto empleado es un texto plano escrito o transcrito a ordenador (lo cual hace que su interpretabilidad por medios automáticos sea mayor). Sin embargo, existen otras propuestas de extracción de entidades de carácter personal que realizan un paso previo a este proceso, extrayendo el texto que se va a utilizar a partir de una imagen. En esta línea cabe destacar la propuesta [1], en la que a partir de imágenes de direcciones postales, se utiliza un OCR para efectuar un reconocimiento del texto que aparece en la imagen y posteriormente detectar las entidades personales (nombre, dirección, organización, etc), utilizando redes BLSTM (*Bidirectional Long Short Term Memory*). En relación con lo anterior se encuentra la propuesta [19], en la que se combina un flujo de procesamiento mediante el módulo OCR de Spark (Spark OCR) para la extracción de textos en documentos PDF, con un flujo PLN enfocado a la detección y censura de las entidades personales existentes en los PDF procesados. Entre las entidades que detecta se encuentran nombres, números de teléfono, datos médicos o fechas, entre otros.

En relación con la anonimización de datos, existen diversas técnicas en función del propósito deseado. [22] aporta una visión global de los tipos de técnicas que hay, implementando, evaluando y definiendo los dos grandes grupos de métodos de anonimización existentes (ver Figura 4.3). Por un lado se encuentra el grupo de los métodos basados en la supresión de entidades o reemplazo por etiqueta (consistentes en eliminar las entidades detectadas en el texto o sustituirlas por la etiqueta de la clase a la que pertenezcan). Estos métodos son muy eficientes pero tienen la desventaja de la pérdida de información del texto. Un ejemplo de esto es [25] en el que se combina la detección de entidades por OCR con una sustitución mediante la etiqueta de la entidad, anonimizando datos personales en radiografías.

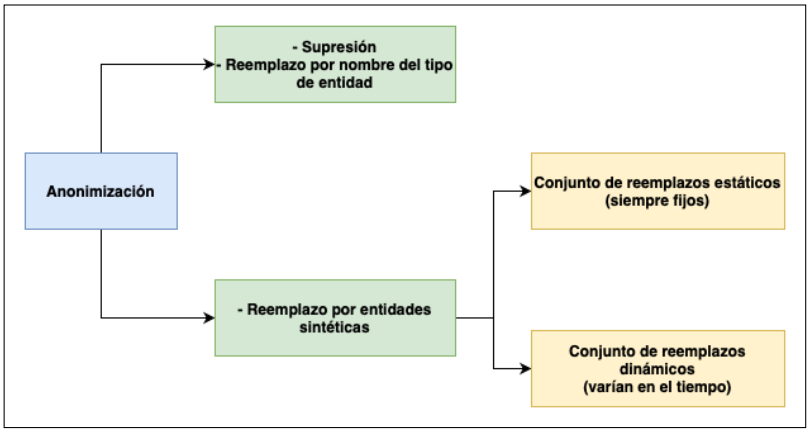


Figura 4.3: Tipos de técnicas para la anonimización de entidades nombradas en textos.

Por último, se encuentran las técnicas de reemplazo basadas en la sustitución por entidades sintéticas-artificiales. Estas técnicas consisten en sustituir las entidades a anonimizar por otras del mismo tipo, permitiendo así preservar el contexto del documento pero sin hacer referencia a ninguna entidad personal. En esta línea se encuentran dos tipos de propuestas en función del tipo de conjunto de entidades sintéticas: por un lado se encuentran los conjuntos de entidades sintéticas estáticas (en las que el conjunto de entidades para reemplazar son fijas). En esta línea se centra la propuesta [29] en el se dispone de un conjunto de entidades artificiales estáticas para la sustitución de entidades personales en textos médicos en castellano. Por otro lado, se encuentran los conjuntos de entidades sintéticas dinámicas, las cuáles parten de un conjunto de entidades reducidas y estas se van actualizando e incrementando a medida que se procesan documentos. En esta línea se encuentran las propuestas [6] y [15] en las que se utilizan grupos de entidades artificiales dinámicas para mejorar la anonimización de las entidades personales en informes médicos del *SARS-CoV-2* (*COVID-19*).

En el presente Trabajo Fin de Máster se abordan distintas técnicas en el campo del PLN tratadas a lo largo de esta sección. Por un lado, para la detección de entidades nombradas se retomará la línea de [7], utilizando modelos híbridos basados en Redes Neuronales y detección mediante expresiones regulares. Por otra parte, para la generación de los reemplazos se continuará en la línea de [22], utilizando mecanismos de reemplazos estáticos. Como novedad frente a las anteriores propuestas, se añadirá un paso previo a la detección de las entidades extrayendo el texto de los documentos mediante OCR. Del mismo modo, se emplearán arquitecturas de Redes Neuronales más complejas (concretamente modelos basados en Redes Neuronales Recurrentes), siguiendo la línea presentada en [43]. Finalmente, se mejorará el sistema de generación de reemplazo de entidades, desarrollando un mecanismo de sustitución basado en grafos que permita que las nuevas entidades que se generen guarden relación entre sí, por ejemplo que las nuevas direcciones generadas pertenezcan a la nueva ciudad generada, etc.

Parte II

Implementación del Proyecto

Capítulo 5

Pipeline Desarrollado

Dado que el *pipeline* desarrollado para anonimizar los documentos se encuentra conformado por múltiples componentes con distintos propósitos, se va a realizar un paso previo de explicación de dichos componentes antes de profundizar en cada uno de ellos.

La implementación del *pipeline* se ha desarrollado bajo una estructura basada en componentes. En esta estructura, cada uno de los componentes actúa como un sistema aislado e independiente del resto, cuyo objetivo se centra en recoger la entrada proporcionada por el componente anterior, realizar una tarea determinada, y enviar el resultado al siguiente componente. Todo este proceso se realiza de forma aislada, es decir, dentro de un componente únicamente puede fallar algo relacionado con la tarea que ha de llevar a cabo el componente. Gracias a esto se consigue mejorar tanto la mantenibilidad, como el aislamiento de los posibles fallos que puedan ocurrir en la herramienta, ya que si fallase un proceso en uno de los componentes, el error se encontraría dentro de él, y no habría que buscar en los componentes ejecutados previamente. Otras ventajas que ofrece la estructura de *pipeline* propuesta es la modularidad, y la actuación como modelo de caja negra: recibe una entrada, la procesa, y la envía al siguiente componente con el que está conectado. Además de esto, la disposición del flujo mediante este tipo de esquemas permite ampliar y reducir la funcionalidad de una forma más sencilla, dado que la única consideración que se ha de tener en cuenta es el formato de la entrada que esta va a recibir, así como mantener el formato de salida esperado por los otros componentes. Esta característica amplía las posibilidades que ofrece actualmente este *pipeline*, y lo convierte en una solución genérica y altamente adaptable a otros flujos, o a otros casos de uso en los que necesiten realizar distintas acciones y/o modificaciones sobre los documentos, o sobre las menciones detectadas.

Los componentes que conforman el *pipeline* de anonimización desarrollado (ver Figura 5.1), junto con su objetivo son las siguientes:

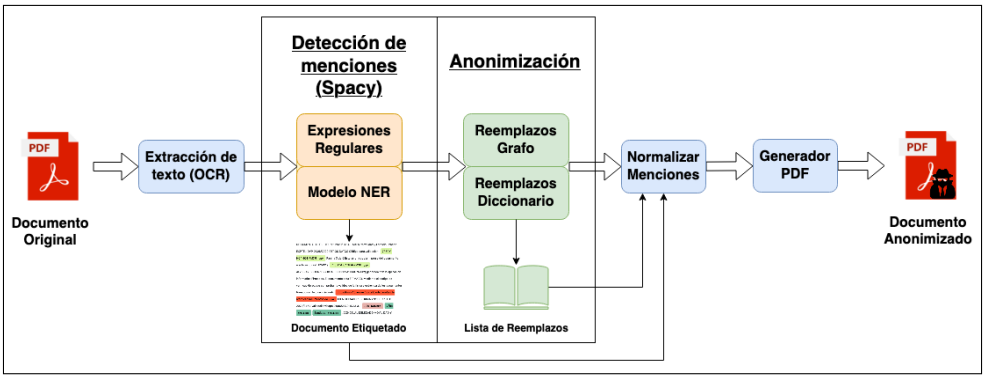


Figura 5.1: Estructura de componentes del *pipeline* desarrollado.

- **Extracción de texto:** El componente de extracción de texto es la primera etapa que se lleva a cabo del *pipeline*. Este componente recibe como entrada un documento en formato PDF y se encarga de leerlo, transformar cada una de las páginas del PDF a imagen y, aplicar un OCR a cada una de esas imágenes para obtener el texto. El texto obtenido a partir del PDF es utilizado para generar un documento de texto plano (*.txt*) que será la combinación de los textos obtenidos al aplicar el OCR a cada una de las páginas del PDF original. Todo este proceso se lleva a cabo utilizando las librerías de manejo de PDF *Pillow*, *pdf2image* y *pytesseract* (ver Apartado 1.3).

  Una vez generado el documento de texto plano, se envía el resultado al siguiente componente, el cual se encargará de detectar las menciones personales que existan en dicho documento de texto. En la siguiente figura (ver Figura 5.2) se muestra en detalle el flujo que sigue este componente:

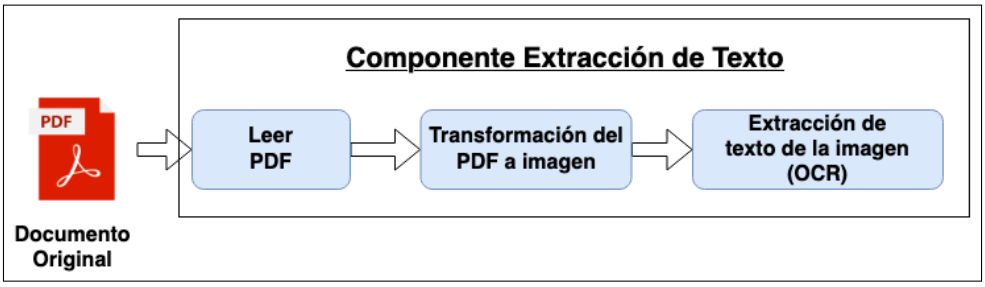


Figura 5.2: Desglose del flujo del "Componente Extracción de Texto".

- **Detección de menciones:** El componente de detección de menciones se encarga de, a partir de un fichero de texto plano, generar un objeto de tipo *Doc* de Spacy y detectar las menciones que este pudiera contener. Esta detección de menciones se realiza en una primera instancia mediante el uso de expresiones regulares para detectar las menciones altamente

predecibles (ver Apartado 7.3), y en una segunda instancia mediante la aplicación del modelo NER entrenado con los documentos del subconjunto de entrenamiento (ver Apartado 7). Finalmente este componente genera un documento anotado que por un lado se utilizará para crear el nuevo documento ("Componente Normalizar Documento" + "Componente Generador PDF"), y por otro lado para obtener las menciones que se han de anonimizar ("Componente Anonimización").

- **Anonimización:** Este componente se encarga de anonimizar las menciones que han sido detectadas por el componente "Detección de menciones". Para ello, utiliza un enfoque híbrido: por un lado, para las menciones que guardan relación entre sí como las ciudades, provincias y códigos postales, se generará un grafo en el que se representen dichas relaciones entre las entidades. La estructura de este grafo se utilizará para obtener un subgrafo similar a partir de la base de conocimiento, logrando así que las nuevas menciones mantengan la misma jerarquía y relación que las originales (ver Apartado 8.1). Por otro lado, para el resto de las menciones se utilizará una anonimización basada en diccionarios, en el que la mención será sustituida por otra con las mismas características (tipo de mención y número de palabras) (ver Apartado 8.2). El resultado obtenido de este componente es una lista en la que se encuentran las menciones originales (detectadas por el componente de detección), y la mención por la cual se va a sustituir (obtenida en este componente).

- **Normalización de Entidades:** Este componente se encarga de comparar el formato de las menciones del documento original, con el de las nuevas menciones, haciendo que estas últimas tengan el mismo formato que las originales. El motivo de la introducción de este componente se debe a que, en muchos casos, hay menciones que se encuentran escritas con todas las letras en mayúsculas, todas en minúsculas, sólo la primera en mayúscula, etc. Por lo tanto, si se hace un reemplazo por las nuevas menciones sin tener esto en cuenta, pueden aparecer fragmentos en el documento que den pie a sospechar que el documento ha sido modificado (por ejemplo que haya un fragmento de texto con todas las letras en mayúsculas, en cuyo interior se encuentre un nombre de persona escrito en minúsculas). Para solucionar este problema, este componente se encarga de obtener el formato de la mención original (*uppercase*, *lowercase* o *capitalize*), y aplicar la misma transformación a la mención por la cual se va a sustituir, modificando así el listado de reemplazos obtenido en el "Componente de Anonimización". El resultado de este componente se enviará al "Componente de Generación de PDF" para crear el documento final anonimizado.

- **Generación del nuevo PDF:** Este componente se encarga de tomar el texto del documento original y la información de las menciones

normalizadas, para generar un nuevo texto a partir de ellas. En este nuevo texto las menciones originales ya han sido sustituidas por sus reemplazos normalizados correspondientes, dando lugar a un texto anonimizado. Una vez se dispone del texto anonimizado, se crea un archivo PDF vacío, se introducen los estilos de diseño de este (cabecera, imágenes, etc), y se escribe el texto anonimizado en él. Una vez se ha terminado de escribir el texto, se persiste el nuevo PDF en disco y se finaliza la ejecución del *pipeline*. En la siguiente figura (ver Figura 5.3) se muestra un fragmento de un documento anonimizado a través de este *pipeline*.

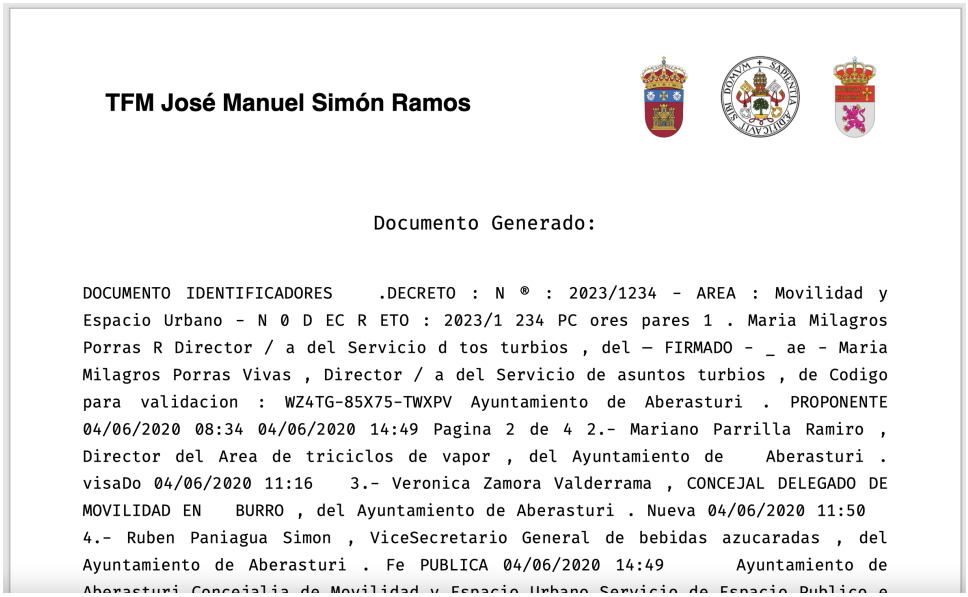


Figura 5.3: Ejemplo de PDF anonimizado una vez finalizada la ejecución del *pipeline*.

Capítulo 6

Conjunto de Datos del Corpus

Los datos son una parte fundamental a la hora de entrenar y evaluar cualquier modelo de Aprendizaje Automático. Es por esto, por lo que realizar una buena elección y división de estos tiene una gran influencia en los resultados finales del modelo. Una buena elección del conjunto de datos tiene como objetivo tomar una muestra (cuanto más grande mejor) de ese conjunto de datos, de tal forma que los datos contenidos en dicha muestra sean variados y representativos del problema que se quiere resolver.

El conjunto de datos utilizado para el presente trabajo se encuentra constituido por documentos en formato PDF pertenecientes a distintas áreas de una Administración Pública de carácter local (movilidad, turismo, servicios sociales, urbanismo, etc). En total se dispone de un conjunto de datos formado por 309 documentos de distinto tipo, clase y objeto (permisos de ocupación de vía pública, concesión de ayudas y subvenciones, permisos de obras, cédulas urbanísticas, inspecciones técnicas a inmuebles, etc).

Todos los documentos que conforman el conjunto de datos tienen una estructura común, por lo que los elementos principales del mismo (firmas, nombres, área de la administración, etc) se encuentran situadas en la misma zona. Dentro de esta estructura se encuentran diferenciadas las siguientes zonas de interés (Ver Figura 6.1):

- **Cabecera:** Se corresponde con la parte superior del PDF. En ella se encuentran los metadatos del documento: firmas, códigos de verificación (CSV), área de la administración, cargo que ocupa, etc. El contenido de esta zona se encuentra delimitado por líneas discontinuas, lo que permite separar esta área del resto.
- **Verificación del documento:** Esta zona se encuentra situada en el margen izquierdo del documento. En ella se muestra, de forma vertical,

un mensaje indicando que el documento ha sido firmado, así como la forma de verificar la integridad del mismo. Además de esto, esta zona dispone de un código de barras para poder escanear la referencia del documento.

- **Contenido:** Esta zona se encuentra situada en la parte central del documento, y se corresponde con su contenido. Al igual que la cabecera, esta zona también se encuentra delimitada (en este caso por un borde sólido).

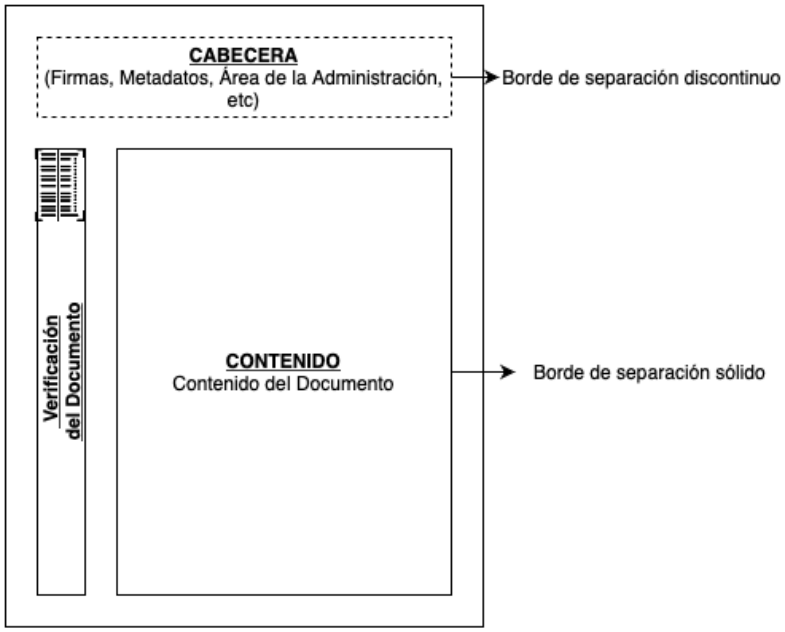


Figura 6.1: Estructura de los documentos utilizados.

Estos documentos han sido etiquetados de forma manual y serán utilizados para entrenar el modelo de Deep Learning encargado de detectar automáticamente estas entidades en los nuevos documentos. Las entidades que se van a detectar y anonimizar, así como sus características, particularidades y consideraciones que se han llevado a cabo para cada una son las siguientes:

- **Nombre:** En esta categoría se incluyen todos los nombres personales referentes a personas físicas (no se han tomado en cuenta nombres de empresa, organizaciones ni administraciones). En el caso de los nombres compuestos (formados por más de una palabra) como "José Manuel" o "Luis Alberto", se unirán ambos en una única entidad en lugar de interpretarlas como entidades independientes y compuestas: <Nombre>**José Manuel**

</Nombre> en lugar de <Nombre>**José**</Nombre> <Nombre> **Manuel**</Nombre>.

- **Apellido:** En esta categoría se incluyen todos los apellidos (por separado) referentes a personas físicas. A diferencia de los nombres personales, en este caso de los apellidos se ha llevado a cabo una división en esta entidad, de tal forma que el primer apellido va separado del segundo: <Apellido>**Simón**</Apellido> <Apellido>**Ramos**</Apellido>, en lugar de <Apellido>**Simón Ramos**</Apellido>. Sin embargo, si uno de los dos apellidos es compuesto, se mantiene la forma de representarlo de los nombres (ambas partes del apellido compuesto en una misma entidad).

  El motivo de realizarlo de este modo se debe a que existen documentos en los que los apellidos no se encuentran posicionados de forma contigua en el texto una vez este ha sido procesado. Un ejemplo de esto se da en las tablas en las que existen diferentes columnas para cada entidad del nombre personal (columna nombre, columna primer apellido, columna segundo apellido). En estos casos los apellidos no estarían situados de forma contigua, sino que se encuentran separados por el carácter de separación de las celdas de la tabla. Para mantener la representatividad se ha optado por identificar de este modo los apellidos personales

- **DNI:** En esta categoría se encuentran comprendidas todas las menciones de números de identificación fiscal y personal de España en los documentos. Esto incluye los DNI (Documento Nacional de Identidad), los NIF (Número de Identificación Fiscal) y los CIF (Código de Identificación Fiscal). A pesar de que en el caso de los residentes en España el DNI y el NIF es el mismo (y el caso de uso que ocupa el proyecto se encuentra enmarcado dentro del contexto español), se ha decidido introducir ambos tipos ya que en el caso de que estos no coincidan, su posición y estructura en el documento es la misma. Por lo tanto, el tener en cuenta esta entidad proporciona información de gran interés desde el punto de vista de la estructura del documento. Además de esto, el hecho de que sean o no iguales no implica ninguna consideración especial dado que se trata de información de carácter personal y por tanto debe ser anonimizada.

- **Dirección:** En esta categoría se encuentran todas las referencias a direcciones postales en los documentos. Estas referencias únicamente comprenden el nombre de la dirección postal a la que se refieren, por lo que las partes “Calle”, “Avenida”, “Polígono”, etc, de la dirección y sus derivadas “C\”, “Av.”, etc no se incluyen en las menciones de esta categoría. Además de esto, tampoco se tienen en cuenta otros identificativos de la localización, como el número del piso, el portal, o la escalera. El motivo de esto se debe a que, a pesar de que aportan

información referente a la dirección, esa información no es significativa por sí misma, si no que es información complementaria a la dirección principal. Del mismo modo, la multitud de formas en las que se puede encontrar representada esta información en un documento, junto con la variedad de documentos utilizados dificulta su identificación de una forma correcta.

- **Ciudad:** En esta categoría se encuentran comprendidas todas las referencias a las ciudades y municipios de los documentos.

- **Provincia:** En esta categoría se engloban las menciones a las provincias que se encuentren en los documentos.

- **CP:** En esta categoría se engloban las menciones a los códigos postales de las ciudades y/o municipios que se encuentren en los documentos.

- **Teléfono:** En esta categoría se encuentran las referencias de números de teléfono (tanto móviles como fijos) en los documentos. Un punto importante que destacar es que se incluirán todos los números independientemente de su formato. Por ejemplo, algunos de los formatos de teléfono más habituales en documentos de carácter administrativo y que se contemplan en esta categoría son: 983-874-985 o 983 874 985.

- **Referencias Catastrales:** En esta categoría se incluyen todos aquellos identificadores de bienes inmuebles que se referencien en los documentos. Dentro de esta categoría se incluyen tanto las referencias catastrales urbanas (en las que se incluye información sobre la parcela, la hoja del plano y el inmueble, por ejemplo 872023 VH5797S 0001 WX), y las referencias catastrales rurales (en las que se incluye información sobre la provincia, el municipio y el sector de la parcela, por ejemplo 13 077 A 018 00039 0000 FP).

- **Seguridad Social:** En esta categoría se encuentran todas las menciones referentes a los números de la seguridad social. Este identificador está constituido por 12 dígitos, y contiene información asociada al centro donde se dio de alta el trabajador, el año de alta, el año de nacimiento del trabajador, etc.

- **Cuenta Bancaria:** En esta categoría se encuentran todas las referencias a números de cuenta bancaria que se referencien en los documentos. A diferencia de las direcciones, en el caso de las cuentas bancarias se va a incluir toda la referencia de esta, es decir se va a incluir tanto los dígitos correspondientes a la cuenta bancaria, como los dígitos correspondientes a la entidad, sucursal, así como los dígitos de control y el código de país (en este caso es: *ES*).

- **Email:** En esta categoría se engloban todas las referencias a correos electrónicos que se encuentren en los documentos. Estos correos pueden ser de cualquier tipo y de cualquier dominio. Desde dominios habituales (Gmail, Outlook, Yahoo, etc), hasta dominios particulares de entidades, empresas o instituciones (alumnos.uva.es, inf.uva.es, etc).

- **Matrícula:** En esta categoría se incluyen las referencias a las matrículas de los vehículos realizadas en los documentos.

- **CSV:** El Código Seguro de Verificación (CSV), es un código que se utiliza para identificar de forma unívoca un documento electrónico. Además de esto, este identificador se encarga de garantizar la integridad del documento en el que se encuentra (evitando así que este pueda ser modificado). A pesar de que el código como tal no hace referencia a ninguna entidad personal, es necesario que este sea procesado por el modelo ya que, en el caso de no anonimizarlo, es posible llegar a otros documentos cuyas entidades personales no han sido anonimizadas. Es por esto, por lo que anonimizar esta entidad es de vital importancia ya que permite eliminar todas las relaciones que existen entre distintos documentos a través de este código.

- **URL:** En esta categoría se engloban todas las direcciones y enlaces web que se encuentren en los documentos. Para el caso de las url's, se van a tener en cuenta todo tipo de referencias web siempre y cuando esta sea válida: protocolo (*http*, *https*) + host (*google.com*, *facebook.com*) + ruta (*/login*, */verificacion*) + variables opcionales en peticiones GET (*?variable1=valor1&variable2=valor2*).

Una característica importante que destacar sobre la identificación de estas entidades es que, en el caso de tener entidades compuestas (entidades formadas por más de una palabra), estas se considerarán como menciones únicas en lugar de como una combinación de varias menciones simples. Un ejemplo de esto es el ya descrito en el caso de las menciones de nombres personales (el nombre "José Manuel" se considera un único nombre en lugar de considerarse como la combinación del nombre "José" y el nombre "Manuel"). El motivo de representar las menciones compuestas como una única entidad, se debe a la premisa de que la información que aporta una entidad es la misma (independientemente del número de palabras que esta tenga). Este mismo supuesto se mantiene en otro tipos de documentos, como los formularios, en los que entidades compuestas (nombre, apellido, dirección, etc) se procesan como una única mención.

Por otra parte, desde el punto de vista de la representatividad y la estructuración de la información en documentos administrativos (y en una gran mayoría de otros tipos de documentos), es mucho más habitual e intuitivo

asociar menciones de identificación personal a estructuras del tipo NOMBRE-APELLIDO-APELLIDO, o APELLIDO-APELLIDO-NOMBRE, en lugar de NOMBRE-NOMBRE-APELLIDO-APELLIDO[1]. De este modo, además de generalizar mejor las distintas estructuras y características de las menciones que aparecen en ellas, se consigue simplificar el problema a resolver, mejorando (en una primera instancia) la eficacia del modelo.

En la siguiente tabla (ver Tabla 6.1) se muestran las menciones totales de cada tipo de entidad en todos los documentos, el total de documentos distintos en el que aparece cada tipo de mención, así como el porcentaje de cada tipo de mención en relación con el resto.

ENTIDAD	DOCUMENTOS	DOCUMENTOS (%)	APARICIONES
NOMBRE	309	100.00	2.263
APELLIDO	309	100.00	4.526
DNI	79	25,57	710
DIRECCIÓN	160	51,68	323
CIUDAD	309	100.00	1.994
PROVINCIA	5	1,62	7
CP	119	38,51	141
TELÉFONO	117	37,86	118
REFERENCIA CATASTRAL	10	3,24	36
SEGURIDAD SOCIAL	-	-	-
CUENTA BANCARIA	17	5,50	56
EMAIL	3	0,97	4
MATRÍCULA	6	1,94	7
CSV	309	100.00	309
URL	282	91,26	282

Tabla 6.1: Total de menciones individuales y por documento de las entidades del conjunto de datos utilizado.

Como se puede ver en la tabla anterior (ver Tabla 6.1), no existe ninguna mención del tipo *Seguridad Social* en los documentos utilizados para entrenar el modelo, lo que supone que el modelo final no es capaz de identificar ninguna mención de este tipo de entidad. No obstante, como se verá en el Apartado 7.3, además de la detección mediante el modelo, se hará uso de expresiones

[1]Esta estructura considera que la única mención personal compuesta es el nombre. Sin embargo, puede darse el caso en el que alguno de los apellidos sea también compuesto, generando una estructura más compleja.

regulares que permitirán mejorar la detección de algunos tipos de entidades como los DNI o las cuentas bancarias. Esto se traduce en que, a pesar de que el modelo no haya sido entrenado con este tipo de entidades (y por ende no sea capaz de identificarlas), estas puedan ser identificadas haciendo uso de otras técnicas (en este caso mediante expresiones regulares).

6.1. Preparación del Conjunto de Datos

Una vez finalizado el proceso de etiquetado y exportación de los documentos, el siguiente punto a realizar consiste en dividir los datos en subconjuntos y entrenar el modelo. Sin embargo, Spacy requiere que los ficheros de los documentos que se utilicen para entrenar y evaluar el modelo se encuentren en un formato propietario (formato *.spacy*). Si bien es cierto que en anteriores versiones esto no era necesario, la utilización de este nuevo formato de los documentos aporta una mayor eficiencia y velocidad a la hora de realizar estas tareas (ya que se reduce la variabilidad de formatos y se puede optimizar mejor el rendimiento).

Por lo tanto, antes de comenzar con el entrenamiento, hay que transformar los documentos etiquetados (en formato *.tsv*), al formato de Spacy (formato *.spacy*). Para ello, Spacy CLI *Command Line Interface* proporciona una funcionalidad que permite llevar a cabo esta transformación de un modo totalmente transparente para el usuario. Para llevar a cabo este cambio de formato, únicamente hay que ejecutar la siguiente instrucción por consola:

```
find annotated -name "*.tsv" | xargs -I _ python -m spacy
↪  convert _ annotatedBin -c iob -n 10
```

Código 6.3: Conversión de los documentos de *.tsv* a *.spacy*.

Donde:

- *annotated:* Directorio donde se encuentran todos los documentos anotados (formato *.tsv*).
- *annotated_bin:* Directorio donde se van a almacenar los documentos exportados (en formato *.spacy*).
- *-c iob:* Indica el tipo de conversión que Spacy ha de aplicar a los documentos. En este caso es *IOB* (Ver Apartado 3.2.1) ya que los documentos que se van a transformar se encuentran etiquetados en ese formato.

- *-n 10:* Indica el número de oraciones por documento. En este caso se ha asignado el valor de 10 ya que ha sido el valor que Spacy ha recomendado utilizar en ejecuciones donde este parámetro no ha sido establecido.

En cuanto al comando propuesto en el Código 6.3, lo que se hace es obtener del directorio donde se encuentran los documentos anotados (directorio *annotated*), el listado de todos los documentos en formato .tsv (primera parte del pipeline). Una vez se obtiene dicho listado, para cada fichero, se ejecuta la segunda parte del pipeline (correspondiente al *spacy convert ...*). Para ello se hace uso del comando *xargs* que, entre otras cosas, permite ejecutar una orden para cada uno de los elementos que recibe. El motivo de realizar el proceso de este modo se debe a que el comando *convert* de Spacy acepta únicamente un documento, por lo que habría que repetirlo para cada documento anotado. De este modo se consigue aplicar en una única orden la conversión de los documentos uno a uno sin necesidad de repetir el comando para cada uno de ellos.

Una vez se disponen de todos los datos en el formato adecuado, el siguiente paso consiste en realizar la división de estos en distintos subconjuntos, que se emplearán para entrenar y validar el desempeño del modelo. En este caso se van a dividir los datos en tres subconjuntos: 1) Conjunto de Entrenamiento (70 % de los documentos); 2) Conjunto de Validación (10 %); y 3) Conjunto de Test (20 %). Un punto importante que destacar es la forma en la que se ha llevado a cabo esta división en los datos. Dado que existe un sesgo en torno a diversas entidades que aparecen en los documentos (hay entidades con un gran número de apariciones y otras con pocas o muy pocas), la asignación aleatoria de documentos entre los distintos subconjuntos podría no ser correcta, y hacer que el modelo final esté sesgado. Esto se debe a que es posible que en documentos de alguno de los subconjuntos no se encuentren todas las entidades (ya que las que tienen un menor número de apariciones pueden haberse asociado a otro subconjunto distinto).

Existen diversas formas de solucionar esta problemática en función del método empleado para su división. Una de las formas evaluadas para aplicar en este caso de uso, consistió en la agrupación de los documentos por temáticas, realizando una división en base a los subgrupos temáticos que se generen con esta agrupación, y posteriormente generar los subconjuntos de manera estratificada en base a cada una de las agrupaciones realizadas. A pesar de que esta propuesta es la más razonable (e intuitiva), fue descartada ya que existe un gran número de tipos de documentos distintos. Esta particularidad implicaba que, en ocasiones, no se dispusiese de documentos suficientes de un mismo tipo para ser repartidos entre los distintos subconjuntos de datos. La alternativa a esta propuesta (y la que ha sido utilizada finalmente para realizar la división de los documentos), ha sido el reparto mediante tipos de entidades.

Para ello, se generan N distintos subgrupos[2] de documentos para cada uno de los tipos de entidades que aparecen en él (el subgrupo correspondiente al tipo de entidad A está constituido por los documentos X, Y, Z; el subgrupo de la entidad B por los documentos M, N, etc). Una vez se tienen estos subgrupos, se van realizando divisiones estratificadas de los documentos en base a las entidades (de entidades con un menor número de apariciones en documentos, a entidades con un mayor número de apariciones), tal y como se muestra en la Figura 6.2. De este modo se consigue que, a pesar de que el reparto en porcentajes no sea exacto para todas las entidades y/o documentos (ver Tabla 6.2), se consigan dividir correctamente aquellas con un menor número de apariciones. El hecho de que al realizar la división de este modo implique que el reparto no sea completamente exacto se debe a que, al contabilizar el número de documentos en los que aparece cada entidad, no se están teniendo en cuenta los duplicados, por lo que un documento en el que aparezca una entidad NOMBRE y una entidad DNI serán contemplados como distintos a pesar de que es el mismo. Por lo tanto, a la hora de asignar un documento a un subconjunto en función de una entidad, no se tienen en cuenta el resto de las entidades que también se encuentran en ese documento, lo que ocasiona que el reparto no sea 100 % exacto. No obstante, como se ha mencionado anteriormente, esto permite que entidades con pocas, o muy pocas apariciones aparezcan en todos los subconjuntos, lo cuál hace que se estén teniendo en cuenta tanto para el entrenamiento, como para la validación del modelo.

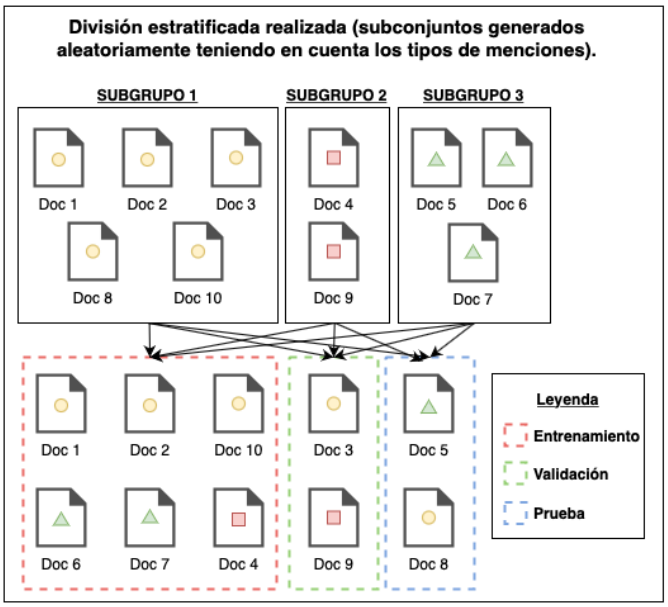


Figura 6.2: División de los datos en los subconjuntos de Entrenamiento, Validación y Prueba.

[2]siendo N el número de tipos de entidades distintas que se van a detectar.

En la siguiente tabla (ver Tabla 6.2) se muestra el número de documentos únicos en los que aparece cada entidad que debería tener cada subconjunto de datos (columna *Esp*), así como el número real de documentos en cada subconjunto en los que aparece cada entidad (columna *Real*):

ENTIDAD Total (309)	**Entrenamiento (70 %)**		**Validación (10 %)**		**Prueba (20 %)**	
	Esp	**Real**	**Esp**	**Real**	**Esp**	**Real**
NOMBRE	216	216	31	31	62	62
APELLIDO	216	216	31	31	62	62
DNI	55	51	8	10	16	18
DIRECCIÓN	112	107	16	20	32	33
CIUDAD	216	216	31	31	62	62
PROVINCIA	4	3	1	1	1	2
CP	83	78	12	16	24	25
TELÉFONO	82	77	12	16	23	24
REFERENCIA CATASTRAL	7	7	1	1	2	2
SEGURIDAD SOCIAL	-	-	-	-	-	-
CUENTA BANCARIA	12	12	2	2	3	3
EMAIL	2	2	1	1	1	1
MATRÍCULA	4	4	1	1	1	1
CSV	216	216	31	31	62	62
URL	197	196	28	28	56	57

Tabla 6.2: División de los documentos en los subconjuntos de Entrenamiento, Validación y Prueba.

Capítulo 7

Construcción del Modelo

Una vez se han convertido los datos al formato esperado por Spacy, y estos se han divido de forma estratificada en los subconjuntos de Entrenamiento, Validación y Prueba, el siguiente paso a realizar consiste en entrenar el modelo NER encargado de detectar las menciones personales de los documentos definidas en el Apartado 6.

Desde el punto de vista de la arquitectura de la red neuronal empleada, se hará uso de un *pipeline* preentrenado que ofrece Spacy para trabajar con textos en español. El motivo de utilizar un *pipeline* en lugar de únicamente un modelo se debe a que, desde que se leen los datos del texto hasta que estos llegan al modelo, hay que realizar previamente varios pasos de procesamiento. Entre estos pasos se incluye la *tokenización del texto* (dividir el texto en los *tokens* que lo conforman), y el *word-embedding* (representar los *tokens* del texto en forma de vector). Por lo tanto, si únicamente se crea el modelo, habría que desarrollar también estos otros componentes (dificultando la creación de la solución propuesta). Para evitar esto, se hará uso de un *pipeline* preentrenado de Spacy compuesto por el *tokenizador* y el modelo NER vacío[1]. Sobre este modelo se llevará a cabo el proceso de entrenamiento que permitirá detectar las menciones personales sobre el conjunto de datos.

Para el caso del español, Spacy proporciona dos tipos de *pipelines* con diferentes modelos base optimizados, pudiendo elegir entre uno u otro en función de las necesidades finales. Los dos tipos de *pipelines* que proporciona Spacy en función de su optimización son los siguientes:

- **Eficiencia:** Este tipo de *pipelines* se encuentra optimizado para ser lo más ligero posible. La principal diferencia frente al *pipeline* enfocado

[1]Este *pipeline* contiene más componentes como el *tagger*o el *morphologizer*. Sin embargo, dado que la funcionalidad que estos ofrecen no es relevante dentro del caso de uso abordado, no se añadirán al *pipeline* para evitar una carga innecesaria de procesamiento extra.

en la precisión reside en el tamaño del vocabulario (lo que se traduce en un menor número de vectores de representación de palabras). No obstante, esto hace que su velocidad en cuanto al entrenamiento sea muy alta, y el tamaño del *pipeline* final entrenado en disco sea muy bajo (en torno a los 13-15MB). Esta característica hace que pueda ser utilizada en una gran mayoría de sistemas al no necesitar de demasiados recursos para su funcionamiento. Dentro de todos los *pipelines* disponibles para el castellano, el orientado a la eficiencia es el *es_core_news_sm*[2].

- **Precisión:** Este tipo de *pipeline* cuenta con los mismos componentes que el anterior. Sin embargo, este *pipeline* cuenta con un gran número de representaciones de vectores en palabras (concretamente cuenta con 500.000 palabras y vectores, además de un total de 300 dimensiones). Si bien es cierto que esto no es algo fundamental para su funcionamiento, le permite tomar en cuenta un mayor número de características para realizar la predicción, lo cual se traduce en un mejor desempeño final. No obstante, desde el punto de vista de la eficiencia, esta empeora respecto a la tipología anterior. El entrenamiento de los *pipelines* de este tipo es mucho más lento y costoso que los enfocados a la eficiencia, y el espacio en disco que ocupa cada uno de estos *pipelines* es muy superior a los anteriores (aproximadamente 550-650MB frente a los 5MB del los orientados para la eficiencia), lo cual es algo a tener en cuenta según qué tipo de caso de uso. Dentro de los *pipelines* que ofrece Spacy en castellano, el enfocado a la precisión es el *es_core_news_lg*[3].

Dado que en la problemática abordada en este trabajo no se cuenta con restricciones en cuanto a tamaño o tiempo de ejecución, se va a realizar el análisis de los hiperparámetros del modelo para ambas optimizaciones de *pipelines*.

7.1. Proceso de Entrenamiento

El proceso que utiliza Spacy para entrenar el modelo es el mismo que el que se utiliza para entrenar cualquier modelo basado en redes neuronales[4]. Para llevar a cabo el entrenamiento, se realizan distintas iteraciones sobre los documentos que conforman el conjunto de datos de entrenamiento (cada una de las iteraciones completas a todo el conjunto de datos se le conoce como **época**). En cada época, para cada uno de los documentos se toma por un lado el texto, y por el otro lado las anotaciones asociadas al mismo, y se

[2]https://spacy.io/models/es#es_core_news_sm *(Visitado el 14-07-2021).*

[3]https://spacy.io/models/es#es_core_news_lg *(Visitado el 14-07-2021).*

[4]https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ *(Visitado el 16-07-2021).*

aplica el modelo sobre el texto del documento, obteniendo así las predicciones del modelo para ese documento. Estas predicciones son comparadas con las anotaciones reales que se disponen y, en base a un algoritmo de optimización del gradiente (en este caso se utiliza el optimizador Adam [18]), se lleva a cabo una modificación en los parámetros internos de las neuronas que conforman la red (modificaciones en los pesos de las entradas de las neuronas). El objetivo es reducir la diferencia entre las anotaciones del modelo y las reales (lo cual se traduce en una mejor calidad de las predicciones). Este proceso es repetido hasta conseguir que exista una coincidencia exacta en todas las anotaciones de todos los documentos (conseguir un 100 % de precisión), o hasta que se realice un número n de iteraciones (**épocas**) sobre todos los documentos del conjunto de datos. Finalmente, el modelo (junto con su configuración de hiperparámetros y neuronas) es exportado para su posterior uso. En el caso de Spacy, se exportan siempre dos modelos: por un lado se tiene el modelo que mejores resultados ha obtenido junto con su configuración (llamado *model-best*), y por otro lado se exporta el modelo correspondiente a la última época llevada a cabo (llamado *model-last*). La elección del mejor modelo la realiza Spacy de forma transparente en función de los pesos que se asigne a cada una de las métricas (precisión, recall, F1, etc)[5].

En la Figura 7.1 se muestra el flujo de ejecución que utiliza Spacy para entrenar los modelos:

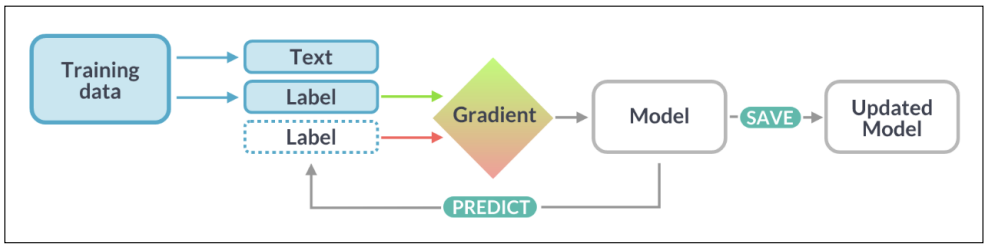


Figura 7.1: Flujo de entrenamiento de un modelo en Spacy. Fuente: *spacy.io (Visitado el 30-06-2021).*

7.2. Evaluación de los hiperparámetros

En esta sección se llevará a cabo la descripción de los distintos hiperparámetros que se han modificado para optimizar la eficacia del modelo, así como

[5]En este caso se ha asignado el 100 % del peso al F1, por lo que Spacy tomará como mejor modelo aquel con el mayor F1.

los valores utilizados y los resultados obtenidos[6]. Los hiperparámetros que se han modificado son los siguientes:

- **Hiperparámetros relativos al optimizador:** Estos hiperparámetros son los que se utilizan a la hora de llevar a cabo la optimización del modelo. La buena elección de estos parámetros permite realizar de forma más efectiva las actualizaciones de los parámetros internos de las neuronas (pesos), lo que se traduce en una mayor y más rápida disminución del error del modelo (diferencia entre el resultado esperado y el resultado real). Entre los distintos hiperparámetros de este tipo los que se han modificado son los siguientes:

  - **Learning Rate:** El Learning Rate, también conocido como factor de aprendizaje, es un parámetro que representa en qué proporción se modificarán los pesos de las neuronas. A grandes rasgos, indica qué tan rápido aprende la red neuronal.
  - **Beta 1:** El valor Beta 1 (β_1) se corresponde con el factor de decaimiento inicial del optimizador.
  - **Beta 2:** El valor Beta 2 (β_2) se corresponde con el factor de decaimiento para el segundo instante del optimizador. Este parámetro se utiliza en combinación con β_1 para estimar los primeros instantes del gradiente, dicho de otro modo, sirven para inicializar la componente relativa al gradiente en el algoritmo.
  - **Épsilon:** El valor de épsilon (ϵ) es un valor muy pequeño que se utiliza para prevenir que se produzcan divisiones por cero dentro del algoritmo, evitando así posibles errores.
  - **Decay Factor:** El factor de decaimiento determina el peso de los valores obtenidos en cada una de las iteraciones del algoritmo frente a los valores existentes.

- **Hiperparámetros relativos al proceso de entrenamiento:** Estos hiperparámetros determinan las características del entrenamiento, por ejemplo, el número de iteraciones que se van a realizar sobre el conjunto de datos, si los datos van a ser procesados en pequeños lotes (*batches*) en lugar de todos juntos, etc. Los parámetros de este tipo que se han modificado durante el entrenamiento son los siguientes:

  - **Batch Size:** Esta variable determina el número de documentos que se van a procesar antes de llevar a cabo el cálculo del error y la

[6]Se ha fijado un valor de semilla ($semilla = 23$) para controlar la generación de los valores aleatorios que se producen en las distintas fases del entrenamiento. De este modo, todos los experimentos realizados pueden ser reproducidos, obteniendo así los mismos resultados que los presentados en este documento.

optimización del modelo (en lugar de realizar esta optimización tras iterar sobre todos los documentos del conjunto de entrenamiento). En el caso de no especificar nada, se entiende que únicamente existe un lote (compuesto por todos los datos del conjunto de entrenamiento), y que la optimización no se llevará a cabo hasta haber iterado sobre todos ellos.

- **Dropout:** El *dropout* es una técnica que se aplica durante el proceso de entrenamiento de una red neuronal para prevenir el sobreajuste. Esta técnica consiste en "omitir" neuronas de forma aleatoria en cada época del entrenamiento, y realizar el ajuste de los pesos sin tener en cuenta a dichas neuronas. La variable *dropout* determina la proporción de neuronas que se omitirán de forma aleatoria en cada una de las épocas del entrenamiento del modelo.

En este punto cabe destacar que, los hiperparámetros correspondientes al optimizador tienen una fuerte dependencia entre sí, por lo que su modificación ha de realizarse en pequeños incrementos en lugar de grandes variaciones o la utilización de valores aleatorios. Todo esto, unido a la complejidad que tiene seleccionar unos buenos parámetros, hace que este proceso de *"prueba y error"* pueda ser demasiado largo. Para solventar esto, se ha tomado como referencia distintas combinaciones de hiperparámetros del optimizador que utilizan las principales librerías actuales orientadas al Deep Learning: TensorFlow[7] y Keras (ver Tabla 7.1).

Librería	**Learning Rate**	**Beta 1** (β_1)	**Beta 2** (β_2)	**Epsilon** (ϵ)	**Decay Factor**
TensorFlow/Torch	0.001	0.9	0.999	10^{-8}	-
Keras	0.001	0.9	0.999	10^{-8}	0

Tabla 7.1: Valores de los hiperparámetros del optimizador en otras librerías de Deep Learning. Fuente: *machinelearningmastery.com (Visitado el 02-07-2021).*

7.2.1. Evaluación de Resultados Globales

Una vez se han definido los distintos parámetros que se van a modificar para maximizar la eficacia del modelo, en esta sección se llevará a cabo la comparativa y análisis de los resultados obtenidos para las distintas pruebas realizadas. Cabe destacar que, al haber probado con tantas variaciones en los hiperparámetros, se va a llevar a cabo un filtro inicial para evitar profundizar en el análisis con configuraciones que no son adecuadas. Para ello, se llevará a cabo

[7]En lo relativo al establecimiento de estos valores por defecto, también se ha tenido en cuenta la librería Torch. Sin embargo, dado que los valores que emplea son los mimos que los de TensorFlow se ha decidido mencionar únicamente esta.

un primer estudio de todas las configuraciones en base a sus métricas generales (Precisión, Recall y F1). De este estudio, se tomarán las mejores configuraciones y sobre ellas se profundizará en el análisis (tratando las mismas métricas pero bajando al nivel de entidad). En las Tablas 7.2 y 7.3 se muestran las métricas globales de las distintas configuraciones de hiperparámetros utilizadas para los modelos optimizados para mejorar la eficiencia, y los modelos optimizados para mejorar la precisión. Del mismo modo en las Figuras 7.2 y 7.3 se muestra la comparativa entre la métrica F1 de las distintas configuraciones utilizadas.

Para obtener estas métricas se han utilizado los documentos del subconjunto de datos de entrenamiento (procesados en *batches*). Al finalizar cada una de las épocas, el modelo resultante se aplica sobre los documentos del subconjunto de validación y prueba (generando así sus métricas asociadas). Del mismo modo, se calcula la tasa de error del modelo, la cuál será utilizada por el algoritmo de optimización (Adam) para modificar los parámetros internos del modelo, y así tratar de disminuir la tasa de error en épocas posteriores.

ID Modelo	Configuración Optimizador	Batch	Dropout	Learning Rate	Decay Factor	Precision	Recall	F1
1	TensorFlow	50	0,1	0,001	-	0,948	0,935	0,941
2	TensorFlow	100	0,1	0,001	-	0,948	0,935	0,941
3	Keras	50	0,1	0,001	0	0,955	0,928	0,941
4	Keras	100	0,1	0,001	0	0,955	0,928	0,941
5	TensorFlow	50	0,2	0,001	-	0,943	0,936	0,939
6	TensorFlow	100	0,2	0,001	-	0,943	0,936	0,939
7	Keras	50	0,2	0,001	0	0,953	0,930	0,942
8	Keras	100	0,2	0,001	0	0,953	0,930	0,942
9	TensorFlow	50	0,3	0,001	-	0,957	0,912	0,934
10	TensorFlow	100	0,3	0,001	-	0,957	0,912	0,934
11	Keras	50	0,3	0,001	0	0,961	0,918	0,939
12	Keras	100	0,3	0,001	0	0,961	0,918	0,939
13	TensorFlow	50	0,1	0,01	-	0,871	0,809	0,839
14	TensorFlow	100	0,1	0,01	-	0,871	0,809	0,839
15	Keras	50	0,1	0,01	0	0,870	0,864	0,867
16	Keras	100	0,1	0,01	0	0,870	0,864	0,867
17	TensorFlow	50	0,2	0,01	-	0,858	0,819	0,838
18	TensorFlow	100	0,2	0,01	-	0,858	0,819	0,838
19	Keras	50	0,2	0,01	0	0,926	0,728	0,815
20	Keras	100	0,2	0,01	0	0,926	0,728	0,815
21	TensorFlow	50	0,3	0,01	-	0,908	0,809	0,855
22	TensorFlow	100	0,3	0,01	-	0,908	0,809	0,855
23	Keras	50	0,3	0,01	0	0,892	0,764	0,823
24	Keras	100	0,3	0,01	0	0,892	0,764	0,823

Tabla 7.2: Resultados obtenidos para distintas configuraciones del modelo basado en la **Eficiencia** Verde (Mejor); Amarillo (Segundo Mejor); Rojo (Peor).

ID Modelo	Configuración Optimizador	Batch	Dropout	Learning Rate	Decay Factor	Precisión	Recall	F1
1	TensorFlow	50	0,1	0,001	-	0,940	0,948	0,944
2	TensorFlow	100	0,1	0,001	-	0,940	0,948	0,944
3	Keras	50	0,1	0,001	0	0,933	0,880	0,910
4	Keras	100	0,1	0,001	0	0,933	0,880	0,910
5	TensorFlow	50	0,2	0,001	-	0,933	0,899	0,916
6	TensorFlow	100	0,2	0,001	-	0,933	0,899	0,916
7	Keras	50	0,2	0,001	0	0,943	0,891	0,917
8	Keras	100	0,2	0,001	0	0,943	0,891	0,917
9	TensorFlow	50	0,3	0,001	-	0,931	0,888	0,909
10	TensorFlow	100	0,3	0,001	-	0,931	0,888	0,909
11	Keras	50	0,3	0,001	0	0,947	0,918	0,933
12	Keras	100	0,3	0,001	0	0,947	0,918	0,933
13	TensorFlow	50	0,1	0,01	-	0,809	0,737	0,771
14	TensorFlow	100	0,1	0,01	-	0,809	0,737	0,771
15	Keras	50	0,1	0,01	0	0,864	0,808	0,835
16	Keras	100	0,1	0,01	0	0,864	0,808	0,835
17	TensorFlow	50	0,2	0,01	-	0,886	0,751	0,813
18	TensorFlow	100	0,2	0,01	-	0,886	0,751	0,813
19	Keras	50	0,2	0,01	0	0,841	0,781	0,810
20	Keras	100	0,2	0,01	0	0,841	0,781	0,810
21	TensorFlow	50	0,3	0,01	-	0,849	0,720	0,779
22	TensorFlow	100	0,3	0,01	-	0,849	0,720	0,779
23	Keras	50	0,3	0,01	0	0,884	0,742	0,807
24	Keras	100	0,3	0,01	0	0,884	0,742	0,807

Tabla 7.3: Resultados obtenidos para distintas configuraciones del modelo basado en la **Precisión** Verde (Mejor); Amarillo (Segundo Mejor); Rojo (Peor).

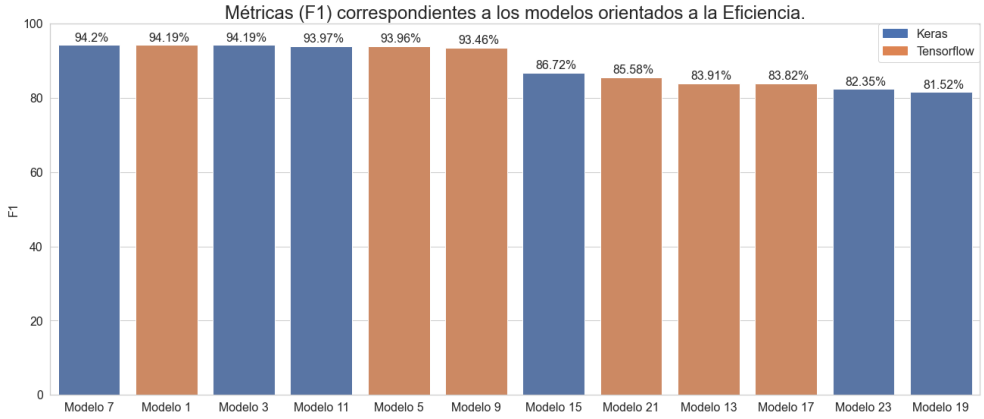


Figura 7.2: Comparativa de la métrica F1 de los modelos enfocados a la Eficiencia.

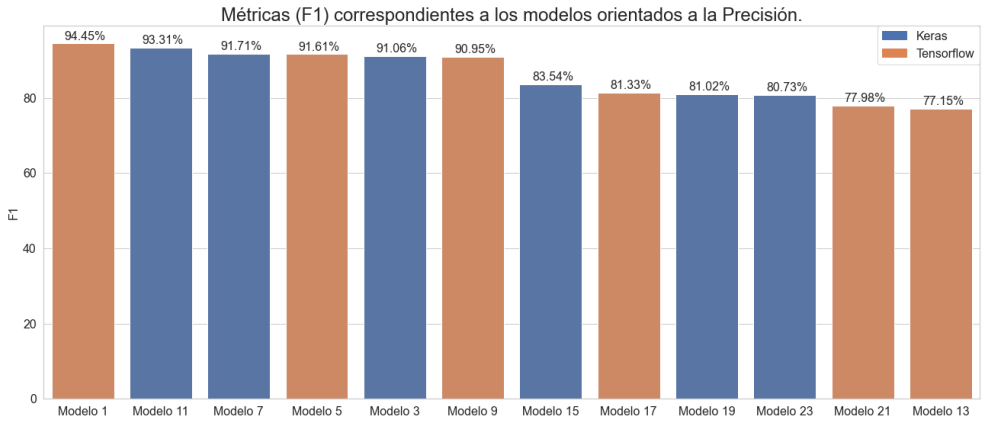


Figura 7.3: Comparativa de la métrica F1 de los modelos enfocados a la Precisión.

Conclusiones

En base a los resultados obtenidos para las distintas configuraciones utilizadas, así como los diferentes modelos enfocados a la optimización de diferentes características, las conclusiones que se pueden sacar son las siguientes:

- **Supresión de parámetros:** Como se puede ver en los resultados anteriores, tanto para el caso de modelos enfocados en la eficiencia, como modelos enfocados en la precisión, el valor del parámetro *batch size* no afecta al desempeño del modelo (los resultados son siempre los mismos independientemente del valor que este parámetro tome). Teniendo esto en cuenta, este parámetro dejará de ser analizado en la memoria y se pasará a utilizar el valor por defecto ($batch\ size = 50$).

- **Disminución de los resultados a medida que se aumenta el dropout:** En todos los casos probados, los resultados globales obtenidos o bien se mantienen, o bien empeoran de forma sustancial a medida que se incrementa el valor de *dropout*. Teniendo esto en cuenta, a la hora de comparar entre dos modelos con resultados parejos, se priorizará aquel cuyo valor de *dropout* sea mayor. Esto se debe a que, en términos generales, un modelo al que se le ha aplicado un *droput* más fuerte durante su entrenamiento, tiende a generalizar mejor (ya que no sufrirá de tanto sobreajuste).

- **Disminución de los resultados a medida que aumenta el factor de aprendizaje:** Al igual que ocurre en el caso del *dropout*, los resultados obtenidos al aumentar el valor del factor de aprendizaje disminuyen de forma notoria en todos los casos probados. Una hipótesis de esta particularidad es que, al utilizar un valor tan elevado, la optimización no es tan buena como para valores más pequeños, ya que es posible que las modificaciones que hayan de realizarse sobre los parámetros de la red sean tan pequeñas y sutiles, que son imposibles de obtener al aumentar el factor de aprendizaje. Otro motivo de esto puede venir asociado a una disminución del sobreajuste por parte del modelo.

- **Los modelos orientados a la eficiencia obtienen mejores resultados que los orientados a la precisión:** Este resultado es muy interesante ya que no es algo directamente intuitivo a nivel teórico. El motivo de obtener mejores resultados en modelos enfocados a mejorar la eficiencia se debe a que la información extra que tienen los modelos enfocados a la precisión (vectores, relaciones entre palabras, etc), no es útil, o no se puede aprovechar del todo para resolver la situación que se está desarrollando. Posiblemente, en otros casos de uso, o casos de uso más complejos en los que haya que obtener las relaciones semánticas entre palabras, su tipología, u otras características, la información extra que ofrecen estos modelos orientados a la precisión pueda ser utilizada de una forma más eficaz, obteniendo unos mejores resultados finales. Sin embargo, para el caso de detección de entidades nombradas en este tipo de documentos, ese exceso de información que aporta un modelo frente a otro no es aprovechado al 100 %. Por lo tanto, se puede concluir que la utilización de un modelo más complejo para un caso de uso simple empeora los resultados en la mayoría de las pruebas llevadas a cabo. Asímismo, podemos afirmar que para este caso de uso concreto, el modelo simple es capaz de generalizar mejor el problema que el modelo complejo.

- **Las configuraciones del optimizador son muy similares:** Finalmente, desde el punto de vista de las configuraciones del optimizador (TensorFlow vs Keras), los resultados obtenidos no permiten sacar unas conclusiones firmes sobre su eficacia. Existen situaciones en las que una

configuración obtiene mejores resultados que la otra (Modelo 6 vs Modelo 7 o Modelo 22 vs Modelo 23). Por lo tanto, dado que el uso de unos parámetros u otros es prácticamente despreciable en términos de tiempos de ejecución, la elección de una configuración u otra se limitará a aquella que proporcione unos mejores resultados finales.

Teniendo en cuenta estas conclusiones, se van a seleccionar los modelos: Modelo 7 (Eficiencia); y Modelo 1 (Precisión) (ver Figura 7.4) para continuar con el análisis y profundizar en los resultados de ambos a nivel de entidad.

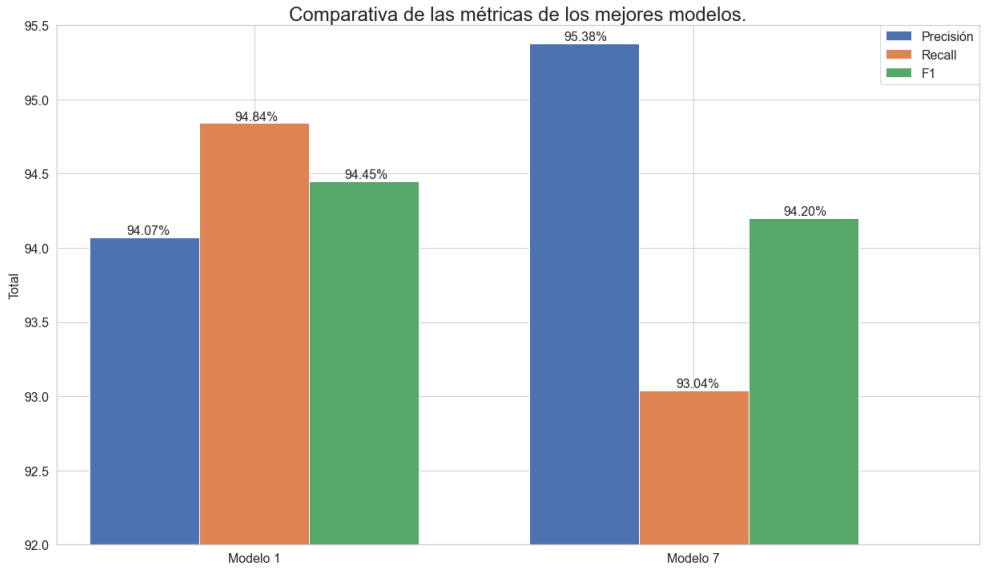


Figura 7.4: Comparativa de las métricas de los modelos seleccionados.

7.2.2. Evaluación de Resultados a nivel de Entidad

Tras haber realizado un primer análisis probando distintas configuraciones de hiperparámetros y modelos, se han seleccionado los modelos: Modelo 7 (Eficiencia); y Modelo 1 (Precisión) para analizar a nivel de entidad. El objetivo de esta sección consiste en, teniendo en cuenta los resultados obtenidos anteriormente, y los resultados que se presentan a continuación, analizar y seleccionar el modelo que tenga un mejor desempeño. Este modelo será el que se utilice en la herramienta desarrollada, y será el núcleo principal de la detección de las menciones personales.

En las Tablas 7.4 y 7.5, se muestran los resultados de ambos modelos a nivel global, así como para cada uno de los tipos de entidad a detectar por la herramienta:

Modelo	Precision	Recall	F1
Modelo 7 (Eficiencia)	0,953	0,930	0,942
Modelo 1 (Precision)	0,940 ▼0,013	0,948 ▲0,018	0,944 ▲0,002

Tabla 7.4: Comparativa métricas globales de los mejores modelos.

Entidad	Modelo 7 (Eficiencia)			Modelo 1 (Precisión)		
	Precision	Recall	F1	Precision	Recall	F1
NOMBRE	0,953	0,961	0,957	0,954 ▲0,001	0,979 ▲0,018	0,966 ▲0,009
APELLIDO	0,962	0,968	0,965	0,961 ▼0,001	0,977 ▲0,009	0,969 ▲0,004
DNI	0,871	0,90	0,885	0,816 ▼0,055	0,827 ▼0,084	0,821 ▼0,064
DIRECCION	0,887	0,588	0,707	0,703 ▼0,184	0,650 ▲0,062	0,675 ▼0,032
CIUDAD	0,995	0,987	0,991	0,982 ▼0,013	0,992 ▲0,005	0,987 ▼0,004
PROVINCIA	1	1	1	1 =	1 =	1 =
CP	0,929	0,788	0,852	1 ▲0,071	0,818 ▲0,030	0,900 ▲0,048
TELEFONO	1	1	1	1 =	1 =	1 =
REFERENCIA CATASTRAL	0	0	0	0,125 ▲0,125	0,083 ▲0,083	0,100 ▲0,100
CUENTA BANCARIA	0,056	0,067	0,061	0 ▼0,056	0 ▼0,067	0 ▼0,061
EMAIL	0	0	0	0 =	0 =	0 =
MATRICULA	0	0	0	0 =	0 =	0 =
CSV	0,968	0,968	0,968	0,968 =	0,968 =	0,968 =
URL	1	0,966	0,983	1 =	0,966 =	0,983 =
SEGURIDAD SOCIAL	-	-	-	-	-	-

Tabla 7.5: Comparativa de las métricas de los dos modelos a nivel de entidad.

Conclusiones

Como se puede ver en la Tabla 7.4, a nivel global, los resultados generales obtenidos por el modelo orientado a la precisión son mejores que los del modelo optimizado para la eficiencia (en torno a un 0,74 % mejor). Esta diferencia se ve también reflejada en los resultados presentados en la Tabla 7.5, en la

que en términos generales, hay un mayor número de mejoras de métricas en el caso del modelo enfocado a la precisión que el orientado a la eficiencia. No obstante, si se profundiza en los aumentos-decrementos de los resultados, la diferencia que existe entre un modelo y otro no es tan elevada. Por ejemplo, en el caso de la entidad "CP", el F1 del modelo optimizado para la precisión mejora un 4,80 %, mientras que en el caso de la entidad "DNI" hay un descenso del F1 del 6,40 %. Por otro lado, en el caso de la entidad "NOMBRE" hay un incremento del 0,9 % frente al decremento del 0,4 % de la entidad "CIUDAD". En términos generales, para aquellas entidades que se ha conseguido una determinada mejora, existe otra entidad cuyos resultados han empeorado en una proporción similar (esto tiene sentido ya que los resultados globales de ambos modelos presentados en la Tabla 7.4 son muy parecidos, por lo que apenas hay diferencia entre ambos). Del mismo modo, si se toma como métrica de referencia el F1, las diferencias entre ambos modelos son aún más pequeñas si cabe. Teniendo esto en cuenta, y viendo que las entidades en las que el modelo enfocado a la precisión pierde eficacia son de las más sensibles ("DNI", "CUENTA BANCARIA" y "DIRECCION"), junto con el coste extra tanto de cómputo como de espacio en disco que supone utilizar un modelo u otro, la decisión que se ha tomado al respecto es prescindir de las mejoras (en términos globales) que ofrece el modelo optimizado para la precisión, y utilizar como modelo final el modelo orientado a la eficiencia. El motivo de esto se debe a que, en base a los resultados obtenidos tanto en este apartado como en el anterior, además del análisis realizado, el coste extra que supone utilizar el modelo basado en precisión no merece la pena comparando los resultados finales de ambos modelos. Por lo tanto, el modelo final a utilizar es el: **Modelo 7 (Orientado a la Eficiencia)**.

7.3. Mejora de los resultados utilizando Expresiones Regulares

Debido a que en los resultados obtenidos a nivel de entidad por el modelo seleccionado (ver Tabla 7.5) existen entidades con unas métricas muy bajas (o incluso nulas), y la estructura de estas entidades es predecible, se va a añadir al sistema de detección de entidades un componente basado en expresiones regulares (RegEx). El objetivo de este nuevo componente se centra en complementar al modelo a la hora de detectar las menciones personales, y mejorar las métricas de las entidades cuya estructura se pueda representar de forma paramétrica.

El componente desarrollado cubre 7 de las 15 tipos de menciones a detectar, y se ha diseñado para capturar las formas más frecuentes de aparición de este tipo de menciones. No obstante, cabe destacar que además del formato "ideal" de cada una de estas entidades, se han incluido variaciones menos frecuentes para mejorar la capacidad de detección de este método. En la Tabla

7.6 se incluyen algunos ejemplos de cada tipo de entidad detectada por este componente, la forma ideal esperada para cada entidad, así como los formatos compatibles (variaciones) que se han contemplado:

Entidad	**Ideal**	**Variaciones Compatibles**
DNI	DNI: 12345678A NIE: X9999999A NIF: A99999999	12345678-A, 12.345.678-A X-9999999-A A-99999999, X-99.999.999
EMAIL	usuario_1234@dominio.com	usuario_1234@organizacion .dominio.com
MATRICULA	Sistema Antiguo: AA 1234 BB Sistema Actual: 1234BBB	AA-1234-AA, A-1234-A, AA-1234-A 1234 BBB, 1234-BBB
REFERENCIA CATASTRAL	Urbana: 999999 XX9999X 9999 XX Rustica: 99 999 X 999 99999 9999 XX	999999 XX9999X 9999 XX, 999999XX9999X9999XX 99 999 X 999 99999 9999 XX, 99999X999999999999XX
CUENTA BANCARIA	ES11 2222 3333 44 555555 ↪ 5555 ES11 2222 3333 4444 5555 ↪ 6666	ES-11-2222-3333-4444-55555-66 ↪ 6666 ES-11-2222-3333-4444-55555-66 ↪ 6666
CSV	XXXXX-XXXXX-XXXX	XXXXX-XXXXX-XXXXX XX XXX-XX XXX-XX XXX XXX XX-XXX XX-XXX XX
URL	https://www.ejemplo.com	https://ejemplo.com www.ejemplo.com/ruta1/ruta2

Tabla 7.6: Estructura de las menciones ideales y compatibles detectadas por las expresiones regulares.

Para minimizar la probabilidad de aparición de un gran número de falsos positivos, o errores a la hora de asignar una etiqueta a un *token*, se han añadido restricciones generales a las expresiones regulares. La primera restricción consiste en que cualquier coincidencia ha de estar precedida por un espacio o un punto. La segunda restricción definida consiste en que toda mención ha de ir seguida de un punto, un espacio, o un final de línea. Estas restricciones garantizan que cualquier mención detectada se encuentre formada por *tokens* completos, evitando así que se produzcan coincidencias con solamente una parte de un *token*.

En la práctica, este componente se aplica de forma secuencial junto con el modelo, por lo que se puede decidir si aplicar primero el componente basado en expresiones regulares para detectar las menciones que sean altamente predecibles (y que luego el modelo detecte las menciones restantes), o aplicar primero el modelo, y posteriormente utilizar las expresiones regulares para obtener aquellas menciones que el modelo haya pasado por alto. En cualquiera de ambos casos, los métodos no se pueden combinar entre sí, es decir, las entidades no pueden superponerse. Esto se traduce en que, en el caso de que se

apliquen primero las expresiones regulares, el modelo no modificará ni eliminará ninguna mención que coincida con una coincidencia de una expresión regular. Del mismo modo, en el caso de aplicar primero el modelo, el componente basado en expresiones regulares no modificará ninguna etiqueta que el modelo haya asignado a un *token*, a pesar de que esta coincida con una expresión regular. En la siguiente tabla (ver Tabla 7.7) se muestran los resultados obtenidos al aplicar el componente basado en expresiones regulares antes y después de aplicar el modelo:

Entidad	**NER + RegEx**			**RegEx + NER**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NOMBRE	0,953	0,961	0,957	0,953 =	0,961 =	0,957 =
APELLIDO	0,962	0,968	0,965	0,962 =	0,968 =	0,965 =
DNI	0,766	0,96	0,852	0,915 ▲0,149	0,973 ▲0,013	0,942 ▲0,090
DIRECCION	0,887	0,588	0,707	0,887 =	0,588 =	0,707 =
CIUDAD	0,995	0,987	0,991	0,995 =	0,987 =	0,991 =
PROVINCIA	1	1	1	1 =	1 =	1 =
CP	0,929	0,788	0,852	0,929 =	0,788 =	0,852 =
TELEFONO	1	1	1	1 =	1 =	1 =
REFERENCIA CATASTRAL	1	0,333	0,500	1 =	0,812 ▲0,479	0,891 ▲0,391
CUENTA BANCARIA	0,150	0,200	0,171	1 ▲0,850	0,800 ▲0,600	0,889 ▲0,718
EMAIL	1	1	1	1 =	1 =	1 =
MATRICULA	1	1	1	1 =	1 =	1 =
CSV	0,968	0,968	0,968	0,968 =	0,968 =	0,968 =
URL	1	0,966	0,983	1 =	0,966 =	0,983 =
SEGURIDAD SOCIAL	-	-	-	-	-	-

Tabla 7.7: Comparativa de las métricas aplicando expresiones regulares antes y después del modelo NER.

Como se puede ver en la Tabla 7.7, los resultados obtenidos son mejores al aplicar el componente basado en expresiones regulares antes de aplicar el modelo. El motivo es que el número de apariciones de las entidades que se han tratado

mediante expresiones regulares es inferior al resto (ver Tabla 6.1). Por lo tanto, a pesar de que el modelo es capaz de identificarlas correctamente, la detección mediante una expresión regular funciona mejor (al tratarse de entidades con una estructura definida). Esto, unido al hecho de que las anotaciones realizadas por un componente no pueden ser modificadas por el otro, hace que los resultados mejoren, ya que en una primera instancia se están detectando menciones altamente predecibles, y se está evitando que el modelo las clasifique de manera incorrecta. Dicho de otro modo, el aumento de las métricas se debe tanto a la detección directa de las entidades por parte de las expresiones regulares, como a la imposibilidad de modificación de estas etiquetas por parte del modelo, evitando que este se equivoque y disminuyan los valores de las métricas.

Finalmente, en las Tablas 7.8 y 7.9 se muestra la comparativa de los resultados obtenidos únicamente por el modelo (ver Apartado 7), y los nuevos resultados obtenidos al combinar el modelo con el componente basado en expresiones regulares aplicada antes del modelo.

Tipo de detección	**Precision**	**Recall**	**F1**
RegEx	0,028	0,059	0,038
NER	0,953 ▲0,925	0,930 ▲0,871	0,942 ▲0,904
RegEx + NER	0,936 ▼0,017	0,949 ▲0,019	0,945 ▲0,003

Tabla 7.8: Comparativa de las métricas globales sin aplicar expresiones regulares y aplicándolas.

Entidad	NER			RegEx + NER		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NOMBRE	0,953	0,961	0,957	0,953 =	0,961 =	0,957 =
APELLIDO	0,962	0,968	0,965	0,962 =	0,968 =	0,965 =
DNI	0,871	0,90	0,885	0,915 ▲0,044	0,973 ▲0,073	0,942 ▲0,057
DIRECCION	0,887	0,588	0,707	0,887 =	0,588 =	0,707 =
CIUDAD	0,995	0,987	0,991	0,995 =	0,987 =	0,991 =
PROVINCIA	1	1	1	1 =	1 =	1 =
CP	0,929	0,788	0,852	0,929 =	0,788 =	0,852 =
TELEFONO	1	1	1	1 =	1 =	1 =
REFERENCIA CATASTRAL	0	0	0	1 ▲1	0,812 ▲0,812	0,891 ▲0,891
CUENTA BANCARIA	0,056	0,067	0,061	1 ▲0,944	0,800 ▲0,733	0,889 ▲0,828
EMAIL	0	0	0	1 ▲1	1 ▲1	1 ▲1
MATRICULA	0	0	0	1 ▲1	1 ▲1	1 ▲1
CSV	0,968	0,968	0,968	0,968 =	0,968 =	0,968 =
URL	1	0,966	0,983	1 =	0,966 =	0,983 =
SEGURIDAD SOCIAL	-	-	-	-	-	-

Tabla 7.9: Comparativa de las métricas sin aplicar expresiones regulares y aplicándolas.

Conclusiones

Como se puede observar en los resultados presentados a lo largo del presente apartado, la introducción de expresiones regulares como soporte adicional para la detección de las entidades ha supuesto una mejora en la eficacia del modelo de un 0,3 %. Si bien es cierto que esta mejora no es muy elevada en términos generales (ver Tabla 7.8), si que es notoria si se atiende a cada una de las entidades de forma independiente (ver Tabla 7.9). En este punto, algunos tipos de entidad como las referencias catastrales, los números de cuenta bancaria, las matrículas o los correos electrónicos, han pasado de tener unas métricas de detección prácticamente nulas, a lograr detectar la totalidad de las mismas.

Sin embargo esta mejora no es visible desde el punto de vista de las métricas globales, ya que existe un sesgo en torno a la cantidad de entidades de cada tipo (ver Tabla 6.1). Dado que los tipos de menciones cuyas métricas han mejorado más con la aplicación de expresiones regulares son también las que tienen un menor número de apariciones en los documentos, todas las mejoras que se hagan sobre las mismas apenas son visibles en los resultados a nivel global. Esto también se puede comprobar en los resultados generales obtenidos al aplicar únicamente expresiones regulares (ver Tabla 7.8), cuyo F1 global es apenas del 4 %.

Por lo tanto, teniendo todo esto en cuenta, se puede concluir que la introducción de expresiones regulares ha mejorado considerablemente los resultados del modelo, logrando detectar (con prácticamente un 100 % de acierto) aquellas entidades con una estructura fija y predecible (referencias catastrales, emails, matrículas, etc), a pesar de que el modelo por sí solo no era capaz de detectarlas.

Capítulo 8

Sistema de Generación de los Reemplazos

Uno de los puntos más importantes del proyecto consiste en llevar a cabo la implementación del sistema de anonimizacion de los documentos. Este sistema de anonimización permite desligar cada uno de los documentos de las personas, entidades, instituciones, etc, que se mencionan en él. A pesar de que esta anonimizacion puede muy ser sencilla e inmediata (simplemente eliminando las entidades detectadas del documento), en este proyecto se va a implementar un sistema más complejo y realista, atendiendo a las características del documento (idioma) y al entorno en el que se encuentra enmarcado. Las características que tiene el sistema de generación de reemplazos implementado son:

- **Robustez:** El sistema es capaz de generar reemplazos de una forma genérica para cualquier combinación de entidades que se le presente. Además, en el caso de no poder encontrar reemplazos válidos (reemplazos que no se ajusten a las características del generador), el sistema genera una alternativa que continúa manteniendo la integridad del nuevo documento.
- **Integridad:** Los reemplazos generados mantienen la integridad del documento. Esto se traduce en que las nuevas entidades generadas mantienen la tipología (tipo de entidad) y estructura (número de palabras) de la entidad a la que reemplazan.
- **Concordancia:** En el caso de entidades que guarden una relación estrecha o fuerte (provincias, ciudades, códigos postales, etc), el sistema es capaz de generar los reemplazos de forma que se mantenga una concordancia entre ellas. De este modo, si en un documento hay que reemplazar una ciudad y su provincia, los reemplazos generados mantendrán esta relación, y la nueva provincia generada contendrá la nueva

ciudad. Además de esto, el sistema es capaz de manejar relaciones débiles como las calles y los números de teléfono, de tal forma que estos se ajustan a las nuevas provincias y ciudades generadas. Del mismo modo, en el caso de no poder garantizar la concordancia (debido a la vulneración de otros factores que se han de tener en cuenta como la robustez o la integridad), el sistema es capaz de mantener concordancias parciales entre las entidades afectadas.

- **Acorde a la realidad:** El nuevo documento generado que utiliza los reemplazos es lo más parecido posible al documento real del que parte. De este modo, puede pasar desapercibido y no parecer que haya sido alterado.
- **Genericidad:** El sistema es capaz de generar reemplazos para todo tipo de documentos independientemente de su tipología.

Para que el sistema cumpla todas estas características se ha desarrollado un mecanismo de reemplazo híbrido. Por un lado, se procesan las entidades susceptibles de mantener relaciones entre sí (provincias, ciudades, etc), generando reemplazos correlacionados mediante búsqueda en grafos (ver Apartado 8.1). Una vez se han procesado las entidades relacionadas, se aplica un mecanismo de reemplazo basado en diccionarios para reemplazar las entidades que no tengan relación entre sí (nombres, apellidos, cuentas bancarias, etc) (ver Apartado 8.2).

El mecanismo de generación de los reemplazos parte de una estructura de datos basada en diccionarios (o tablas hash, en función del lenguaje de programación), en el que cada documento tiene asociado un diccionario vacío. A medida que se van ejecutando los distintos mecanismos de reemplazo del sistema híbrido, este diccionario se va actualizando, creándose una entrada para cada entidad original cuyo valor será el reemplazo generado. Al finalizar el proceso, el diccionario tiene una entrada para cada una de las distintas entidades del documento junto con el valor (nueva entidad) por la que será reemplazada. Finalmente, se construye el documento final sustituyendo las entidades originales por su reemplazo correspondiente, todo ello según el valor que tenga la clave en el diccionario de reemplazos.

En la Figura 8.1 se muestra el flujo de generación de los reemplazos para las entidades de un documento.

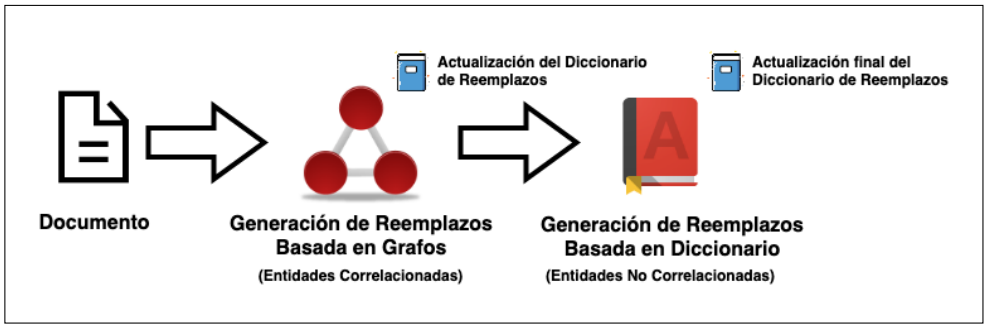


Figura 8.1: Estructura del flujo para la generación de los reemplazos.

8.1. Generación de reemplazos basados en Grafos

El primer mecanismo para la generación de reemplazos dentro del sistema híbrido implementado consiste en un sistema basado en grafos. Este sistema dispone de un grafo de conocimiento en el que se incluyen todas las provincias, ciudades y códigos postales a nivel nacional (a partir de ahora lo denominaremos **grafo general**). Cada uno de los valores únicos de estas entidades se traducirá en un nodo dentro del grafo, y sus relaciones se traducirán en aristas que conecten dichos nodos (ver Apartado 8.1.1).

A partir de la información que ofrece el grafo general, se construye el grafo correspondiente a las entidades del documento (denominado **grafo del documento**). Para ello, se realizará una creación en dos fases:

- **Fase 1: Creación de los nodos.** En esta fase se toman todas las entidades del documento susceptibles de estar relacionadas entre sí (en este caso provincias, ciudades y códigos postales), y se crea un nodo dentro del grafo del documento para cada una de ellas (ver Figura 8.2).

  Sin embargo, como se mencionará en la Fase 2, el establecimiento de las conexiones entre los nodos del grafo del documento se lleva a cabo mediante una comparativa entre pares de nodos del grafo del documento y el grafo general. De este modo, si el par de nodos del grafo del documento se encuentran conectados en el grafo general, se establecerá una conexión entre los nodos en el grafo del documento. Esto implica que la comparativa entre nodos se lleva a cabo mediante una coincidencia exacta de ambas entidades, lo cual es un gran problema. Esto hace que, en el caso de que exista alguna variación de sintaxis o de formato entre las entidades, no se establecerá una relación entre entidades que realmente si que la tienen. Para el caso de los problemas de formato la solución es sencilla ya que se puede aplicar la misma transformación a todas las entidades (eliminación de acentos, transformación a mayúsculas o minúsculas,

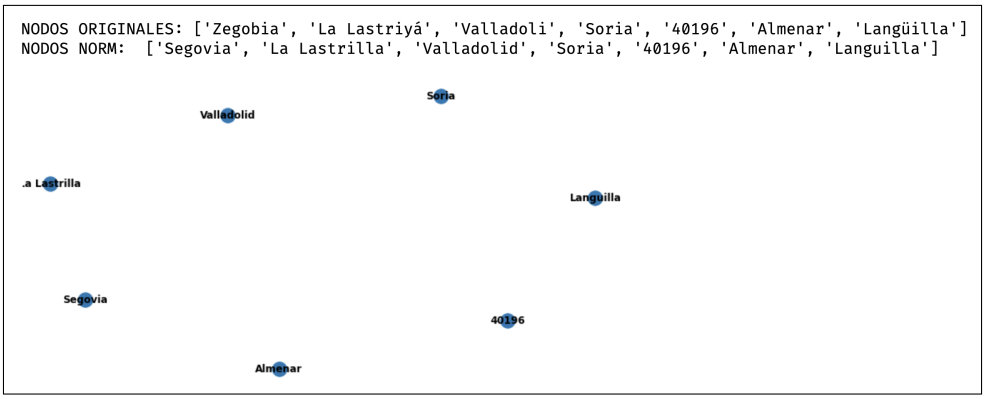


Figura 8.2: Ejemplo de entidades de un documento antes de realizar las conexiones.

etc) para que puedan ser comparadas y relacionadas. No obstante, esta solución presupone que no existen errores en cuanto a la escritura de la entidad en el documento original, algo que ni para este caso de uso (debido al OCR) ni para una solución genérica se puede presuponer (siempre pueden existir errores humanos). Sin embargo, una de las cosas que sí se puede suponer, es que las entidades que conforman el grafo general son correctas (tanto en sintaxis como en formato). Gracias a esto, se pueden emplear las entidades del grafo general como referencia de cara a la posterior comparativa con las entidades en los documentos.

Partiendo de esa premisa, se ha planteado un paso previo de normalización de entidades mediante un enfoque basado en búsquedas por similitud, también denominada **búsqueda difusa**. Para ello, en lugar de crear un grafo asociado al documento, cuyos nodos sean exactamente las mismas entidades que las del documento original, se creará un grafo cuyos nodos serán las entidades del grafo general más similares a las del documento. Por ejemplo, en el caso planteado anteriormente (ver Figura 8.2), se puede ver cómo en el documento original aparece la entidad "Valladoli" y en el grafo general esta misma entidad se encuentra definida como "Valladolid". Como con total seguridad la entidad "Valladolid" será la entidad del grafo general con una mayor similitud a la entidad del documento, el nodo que se cree en el grafo del documento será "Valladolid" (en lugar de "Valladoli"). De forma análoga ocurre lo mismo para los casos "Zegobia" → "Segovia", o "La Lastriya" → "La Lastrilla". Además de esto, para mantener la trazabilidad, todas estas transformaciones serán almacenadas para mantener la relación **Entidad del documento ⇒ Nodo del Grafo del Documento ⇒ Reemplazo generado**.

La forma que se ha seguido para determinar si dos entidades (la entidad del documento y la entidad del grafo) son la misma, viene dada por el **Ratio de similitud**. Este ratio viene dado por la siguiente fórmula:

$$Ratio = \frac{1 - (lev_{p_1,p_2}(|p_1|, |p_2|))}{max(|p_1|, |p_2|)}; \qquad Ratio \in [0, 1]$$

Donde:

- *lev:* Es el cómputo de la Distancia de Levenshtein[1](menor número de inserciones, sustituciones o eliminaciones para obtener una palabra a partir de otra) entre ambas palabras (p_1 y p_2).
- *max(p_1, p_2):* es el valor máximo de las longitudes de ambas palabras.

Por otra parte, se ha establecido un valor umbral, de tal forma que si el mayor ratio de similitud entre una entidad en el documento, y su mayor coincidencia dentro del grafo general no supera dicho umbral, la entidad del documento se mantiene tal cual en el grafo (y se procesará como entidad aislada e independiente). De este modo se consigue no añadir un exceso de ruido ni a los nodos que conforman el grafo del documento, ni a sus relaciones.

Gracias a la implementación mediante búsqueda difusa, se consigue mejorar la concordancia entre los reemplazos. Esto se debe a que, en el caso de haber continuado con métodos basados en coincidencias exactas, la concordancia entre entidades se vería afectada al no poder establecer relaciones entre entidades (o establecer un menor número de ellas). Por lo tanto, el sistema sería equivalente a realizar reemplazos de entidades por otras del mismo tipo, sin atender a las relaciones entre ellas.

- **Fase 2: Establecimiento de las conexiones:** Para cada par de nodos del grafo del documento, se comprueba si estos están conectados dentro del grafo general. En el caso de que lo estén, se creará una arista que una ambos nodos en el grafo del documento. Esta comprobación finalizará cuando se haya terminado de comparar todos los pares de nodos del grafo del documento.

  Debido al paso previo de normalización llevado a cabo en la Fase 1, se puede garantizar que todas las componentes del grafo del documento con más de un nodo se corresponden con un subgrafo dentro del grafo general. Por otra parte, en el caso de las componentes aisladas (con un único nodo), pueden corresponderse o no con subgrafos del grafo general[2].

[1]https://www.analyticslane.com/2020/06/17/la-distancia-de-levenshtein/ *(Visitado el 19-07-2021).*

[2]Esto se debe al establecimiento de un valor umbral a la hora de calcular la similitud entre palabras en la Fase 1. En el caso de no establecerlo, todas las componentes del grafo

En cambio, a pesar de solventar el problema de los errores de sintaxis y/o formato de las entidades, esta propuesta sufre una alta dependencia del estado de la base de conocimiento con la que se crea el grafo general. Si bien es cierto que el funcionamiento del mecanismo no depende de este grafo (ya que en el caso de no poder asociar los nodos, las entidades se reemplazan por otras aleatorias del mismo tipo), una base de conocimiento escasa o alejada del caso de uso a tratar implica no poder establecer relaciones entre los nodos del grafo del documento. Esto se traduce en que el grafo del documento sea un grafo constituido por N componentes simples, siendo N el número de entidades distintas en el documento. A pesar de que este problema se soluciona sustituyendo cada entidad por otra del mismo tipo, añade un tiempo de ejecución innecesario, ralentizando el proceso, y añadiendo una carga de computación extra (dado que tiene que hacer todo el proceso de búsquedas en grafos descrito anteriormente). Por este motivo, es de gran importancia disponer de una buena base de conocimiento para así evitar este tipo de problemas.

Finalmente, tras la consecución de ambas fases, se dispondrá de un grafo de entidades del documento que guardarán relación entre sí, así como las relaciones que existen entre ellas (ver Figura 8.3).

Figura 8.3: Ejemplo de entidades del documento tras haber establecido las conexiones.

8.1.1. Creación del Grafo General

El grafo general constituye el eje central del mecanismo de generación de reemplazos correlacionados. En él se encuentra toda la información tanto de

del documento se corresponderían con subgrafos del grafo general (independientemente del número de nodos que la conformasen).

las entidades que se utilizarán para reemplazar las entidades originales, así como sus características (número de palabras, tipo de entidad, etc). Una de las características que ofrece la librería de grafos utilizada (ver Apartado 1.3), es que el grafo internamente se encuentra representado e implementado como un diccionario. Esto implica que las consultas a los atributos y características de los nodos se lleven a cabo en un tiempo de ejecución constante ($\mathcal{O}(1)$). Además, asegura que no existan nodos de una misma entidad duplicados.

La creación de este grafo se lleva a cabo a partir de un fichero de texto (en formato *.csv*), en el que se encuentra la base de conocimiento de forma estructurada que se empleará para crear este grafo[3]. Esta base de conocimiento ha sido obtenida combinando los datos de las provincias y municipios (Ver fila **DIRECCIÓN** en la Tabla 8.1), con el listado de códigos postales por municipio[4]. En el Código 8.4 se muestra un ejemplo de la estructura de registros del fichero utilizado para generar el grafo general. Del mismo modo, en la Figura 8.4 se muestra la representación de estos registros en forma de grafo, una vez han sido procesados.

```
Provincia;Ciudad;CP
Valladolid;Gomeznarro;47493
Valladolid;Honcalada;47219
Segovia;Languilla;40556
Segovia;Lastras De Cuellar;40352
Soria;Almazul;42126
Soria;Almenar;42130
```

Código 8.4: Estructura de los registros del fichero para generar el grafo general.

[3]En este caso, el grafo contendrá únicamente información sobre Provincias, Ciudades y Códigos Postales, pero es extrapolable a cualquier otro tipo de relación de entidades.

[4]https://github.com/inigoflores/ds-codigos-postales-ine-es *(Visitado el 02-07-2021).*

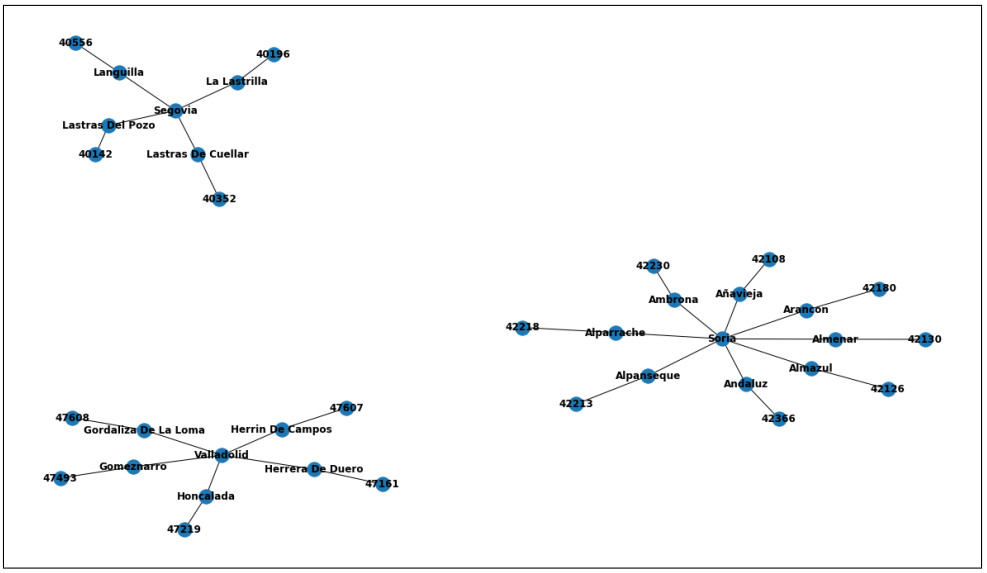


Figura 8.4: Ejemplo reducido de la estructura del grafo general.

Para cada uno de los registros del fichero de datos, se obtienen las entidades dividiendo por el separador utilizado (en este caso *;*). Seguidamente, se crea un nodo para cada una de las entidades y se les asignan atributos internos para posteriormente poder operar con estos nodos (en el caso de que el nodo ya exista no se hace nada). Los atributos internos que se les asigna a cada nodo son los siguientes:

- **n_tokens:** Este atributo almacena el valor numérico, correspondiente al número de tokens (palabras) que tiene la entidad.
- **tipo:** Atributo numérico que se utiliza para representar tanto el tipo de entidad como su nivel de profundidad o importancia dentro de la jerarquía de nodos. Este atributo toma el valor 0 en el caso de ser una provincia; 1 en el caso de tratarse de una ciudad; y 2 en el caso de ser un código postal.

Finalmente, una vez se han creado los nodos correspondientes a las entidades de ese registro, se unen entre sí siguiendo el orden en el que se encontraban en el fichero. Por ejemplo, si el orden del fichero es: provincia, ciudad, código postal; el nodo creado para esa provincia se une al nodo de esa ciudad, y así sucesivamente.

Un punto que destacar desde el punto de vista de la implementación es que, dado que para mejorar el desempeño de este mecanismo se requiere de una base de conocimiento amplia, el proceso de leer y procesar el fichero con los datos, así como generar el grafo supone un tiempo de procesamiento y cómputo

a tener en cuenta. Para evitar este coste computacional extra, además de para mejorar la eficiencia del sistema, el grafo generado se almacenará en disco en forma serializada una vez este haya sido creado. De este modo, se reduce tanto el tiempo de ejecución como la carga de procesamiento al no tener que generar siempre el grafo desde cero, haciendo que únicamente sea necesario generarlo una vez y cargarlo desde el disco en posteriores ejecuciones.

8.1.2. Obtención de reemplazos en grafos

Una vez se dispone tanto del grafo general, como del grafo asociado al documento (nodos y conexiones entre los nodos), el método que se va a utilizar para obtener los reemplazos consiste en obtener del grafo general un subgrafo con la misma estructura que el grafo del documento (ver Figura 8.5). Este método es conocido dentro de la Teoría de Grafos como **Problema de Isomorfismo de Subgrafos** [5].

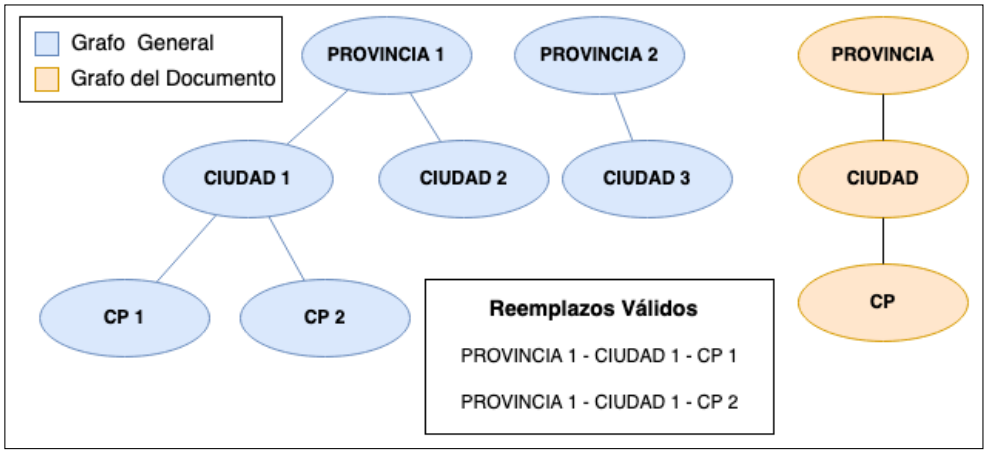


Figura 8.5: Ejemplo de búsquedas de isomorfismos de subgrafos.

El principal problema que existe en cuanto a la utilización de este tipo de métodos es que se encuentra enmarcado dentro del conjunto de problemas NP-Complejos[5]. Esto quiere decir que su resolución no puede ser realizada ni representada de forma polinomial, por lo que el tiempo y coste necesario para su resolución puede ser muy elevado (algo que se quiere evitar). Además de esto, las restricciones impuestas a las nuevas entidades (número de palabras, tipo de entidad, etc) hace que esta búsqueda sea más compleja y costosa. Si bien es cierto que los grafos que se generan a partir de los documentos utilizados en este proyecto no presentan problema (no son excesivamente complejos, por lo que esta búsqueda es relativamente rápida (del orden de milisegundos (ver Apartado 9.6))), para un caso de uso genérico en el que puede haber una mayor complejidad de estructuras sí que suponen un problema. Dado

[5]http://es.knowledger.de/0078452/NPhard *(Visitado el 18-07-2021).*

que este algoritmo viene implementado en la librería de grafos utilizada, y su optimización y creación desde cero se encuentra fuera del alcance tanto de este proyecto, como del Máster en el que se enmarca, se han desarrollado distintas estrategias a nivel de programación que permiten simplificar el problema, y reducir tanto su tiempo como su coste. Estas estrategias son las siguientes:

- **Procesamiento por componentes:** A pesar de que el algoritmo de búsqueda de isomorfismos acepta grafos completos, el tiempo de cómputo y el número de iteraciones-intentos necesarios es muy elevado. La solución desarrollada para este problema consiste en dividir el grafo en sus componentes aisladas e ir procesando estas componentes individualmente. De este modo, se consigue reducir el tiempo de búsqueda, ya que los grafos son más simples. Además de esto, las componentes que conforman el grafo son procesadas en función del número de nodos que la conformen (de mayor a menor número de nodos). El motivo es que, dado que las nuevas entidades no pueden repetirse, se consigue evitar nuevas restricciones que pueden surgir al procesar antes otras componentes. Por ejemplo, si se ha sustituido un nodo provincia de una componente por Valladolid, la siguiente componente no puede utilizar Valladolid como nueva provincia, ya que esta ya ha sido utilizada anteriormente.

- **Utilización de atributos en los nodos:** Otra de las soluciones que se han implementado para limitar el tiempo y coste de búsqueda de este algoritmo, ha sido la de utilizar atributos en los nodos como criterio para limitar la búsqueda. Dado que la tipología de la entidad que representa el nodo se encuentra definida mediante el atributo **tipo** dentro de los nodos (ver Apartado 8.1.1), se puede limitar el espacio de búsqueda filtrando nodos por esa propiedad (mejorando los tiempos de procesamiento).

- **Eliminación de aristas:** Dado que a menudo el procesar el grafo por sus componentes no es suficiente para reducir la complejidad y el tiempo de búsqueda, se ha implementado un mecanismo de eliminación de aristas. Este mecanismo tiene como objetivo el dividir una componente en dos (mediante la eliminación de una de las aristas entre sus nodos) para que así esta sea más simple y pueda ser procesada más rápidamente. La eliminación de las aristas se realiza de forma automática una vez se alcanza un determinado número de intentos fallidos sin encontrar un subgrafo válido. Además, esta eliminación se aplica entre los nodos que se encuentren en los niveles más altos de la componente (nunca por la mitad). De este modo se consigue mantener la concordancia entre un mayor número de entidades ya que estos nodos superiores son los que más conexiones tienen. Tras la eliminación de una de las aristas, se procesan las componentes originadas, y se repite el proceso hasta que se consigue encontrar un reemplazo válido para todas las entidades (en el caso de

que la componente reducida siga siendo compleja se vuelve a dividir y así sucesivamente). En Figura 8.6 se muestra un ejemplo de eliminación de aristas para obtener grafos más simples a partir de una componente.

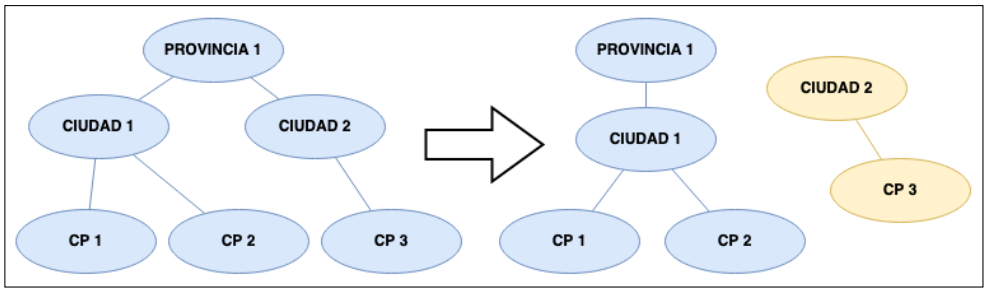


Figura 8.6: Ejemplo de división de componentes para simplificar el grafo.

Cabe destacar que todas estas estrategias se han implementado para solventar problemas de tiempo de ejecución a la hora de buscar reemplazos en todo tipo de situaciones. El caso de uso abordado en el presente proyecto no requiere de la utilización de estas técnicas debido a la sencillez de sus estructuras. No obstante, dado que uno de los objetivos fijados tenía como objeto la creación de un sistema de reemplazo de entidades genérico, ha sido necesaria la implementación de este tipo de estrategias.

8.2. Generación de reemplazos basados en Diccionarios

Tras la obtención de los reemplazos para las entidades relacionadas entre sí, el siguiente paso del flujo de anonimización consiste en generar reemplazos para el resto de las entidades en el documento. Dado que estas entidades no tienen por qué guardar una relación entre sí, sus reemplazos van a venir dados o bien por otras entidades del mismo tipo y características, o bien van a ser generados dinámicamente (ver Apartado 8.2).

Para llevar a cabo la obtención de estos reemplazos de la forma más eficiente posible, se ha implementado una estructura basada en diccionarios (a partir de ahora denominada **diccionario general**) y organizada por tipo de entidad y número de palabras (ver Figura 8.7). De este modo, cada vez que se quiera obtener un reemplazo para una entidad con un determinado número de palabras, únicamente hay que acceder al nivel correspondiente a esa entidad, acceder al nivel específico del número de palabras, y tomar uno de los valores disponibles. Un punto importante que destacar sobre este diccionario general es que para el caso de las direcciones postales (calles), se ha añadido un nivel más de profundidad. Este nivel extra se corresponde a las provincias, por lo que, además de especificar el número de palabras que ha de tener el reemplazo,

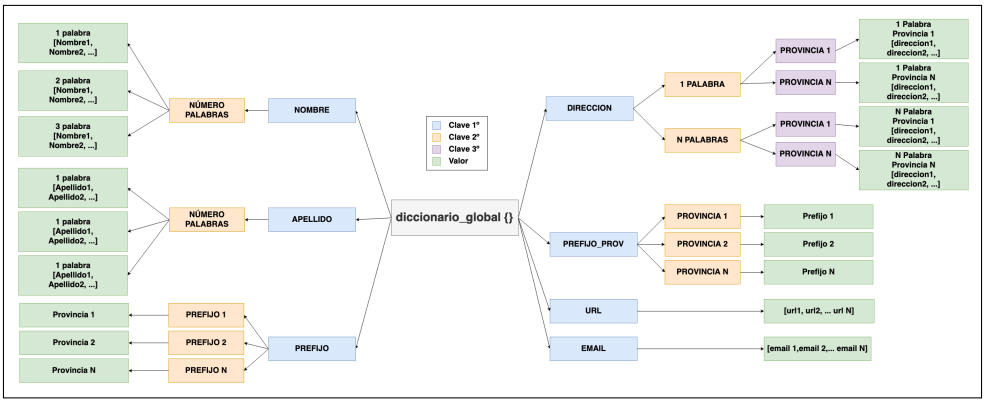


Figura 8.7: Estructura del diccionario general implementado.

hay que fijar la provincia en la que se está buscando dicho reemplazo. Esta característica es muy interesante ya que como se verá posteriormente, ayuda a solucionar algunos problemas de concordancia entre los reemplazos. Por otra parte, al igual que ocurre con el grafo general creado anteriormente (ver Apartado 8.1.1), la información empleada para generar este diccionario procede de ficheros en texto plano, por lo que la eficacia del generador reside en la variedad y cantidad de información utilizada para su creación. En la Tabla 8.1 se muestran los distintos ficheros utilizados para generar este diccionario, así como los datos que contienen y el origen de estos.

Entrada Diccionario	**Fichero (*.txt*)**	**Datos**	**Origen**
NOMBRE	DIC_NOMBRE	Antonio Francisco Javier	INE
APELLIDO	DIC_APELLIDO	Rodríguez González	GitHub
DIRECCION	DIC_DIRECCION	Abrahan Lincoln;Álava Las Acacias;Álava	INE
EMAIL	DIC_EMAIL	rnicolau@yahoo.com antonia68@gmail.com	Mockaroo
URL	DIC_URL	http://www.casal.org/ http://tormo.com/	Mockaroo
PREFIJO	DIC_PREFIJO	Álava;945 Albacete;967	Xataka

Tabla 8.1: Formato de los ficheros de datos utilizados para crear el diccionario general y fuente de donde han sido obtenidos *(Fuentes Visitadas el 05-07-2021)*.

Del mismo modo que para el grafo general, una vez este diccionario ha sido generado, es exportado de forma binaria. Esto hace que no sea necesario

generarlo continuamente desde cero con cada ejecución. Además, a pesar de que existen varios niveles de profundidad para cada entidad, el acceso a todas ellas se lleva a cabo en un tiempo constante ($\mathcal{O}(1)$), por lo que se trata de una implementación muy eficiente. En la Tabla 8.2 se muestra el número de reemplazos para cada tipo de entidad que dispone el diccionario utilizado:

ENTIDAD	**NÚMERO PALABRAS**	**TOTAL REEMPLAZOS**
NOMBRE	[1 - 3]	1.197
APELLIDO	[1 - 3]	1.029
DIRECCION	[1 - 15]	118.688
EMAIL	Indefinidos	100
URL	Indefinidos	100
PREFIJO	-	52

Tabla 8.2: Número de palabras soportadas y total de reemplazos disponibles para cada tipo de entidad.

Aclaración

Las entidades **EMAIL** y **URL** tienen un número de palabras indefinido ya que a la hora de realizar el reemplazo este será dividido en tantos *tokens* como tenga la entidad original.

Dentro de la generación de reemplazos para entidades no relacionadas se pueden distinguir dos tipos en función de la técnica utilizada para su obtención:

- **Reemplazos generados mediante Diccionario:** En esta categoría se incluyen todas las entidades cuyo reemplazo se haya generado a partir del diccionario general definido anteriormente. Los reemplazos generados de este modo se corresponden con las siguientes entidades:

  - **Nombre:** Para el caso de los nombres, el primer paso que se lleva a cabo consiste en obtener el número de palabras que conforma el nombre original. Una vez se obtiene este valor, se selecciona un valor aleatorio del diccionario global en la categoría **"Nombre"**, y en nivel correspondiente al número de palabras. Por ejemplo, si el nombre original está formado por dos palabras, el acceso a los reemplazos válidos para esa entidad es **diccionario_global[NOMBRE][2]**.
  - **Apellido:** El proceso es idéntico al llevado a cabo para los nombres. La única diferencia es que en lugar de acceder a la categoría "Nombre" en el diccionario general, se accede a la categoría **"Apellido"**.
  - **Dirección:** Para el caso de la generación de las nuevas direcciones, el procedimiento es un poco distinto. En primer lugar, al igual que

con los nombres y los apellidos, se obtiene el número de palabras que contiene la dirección. Una vez hecho esto, se genera una provincia aleatoria (la cuál se utilizará para determinar la localización de la dirección). A continuación, se comprueba si la provincia elegida aleatoriamente tiene un reemplazo asociado (debido a que esta ha sido anonimizada en la fase correspondiente a los grafos). En el caso de que la nueva provincia ya se encuentre anonimizada, la nueva dirección que se genere será una que se encuentre en la provincia por la cuál ha sido reemplazada. Por el contrario, en el caso de que la provincia no haya sido anonimizada, se genera una dirección aleatoria perteneciente a esa provincia seleccionada (que ha sido elegida aleatoriamente).

Este método es interesante ya que, a pesar de que para la mayor parte de las direcciones es difícil saber en que ciudad o provincia se encuentran (por lo que pueden pasar desapercibido), en España hay lugares muy reconocibles debido a las lenguas. Por ejemplo, si la dirección que aparece en el documento es "Rúa da Zarra", se puede deducir que el lugar donde esta se encuentra se corresponde a alguna zona de Galicia. Además de esto, si dicha dirección se encuentra asociada en el nuevo documento a la provincia Sevilla, es evidente que ha habido una manipulación en el documento, haciendo que este deje de ser realista.

- **Teléfono:** La utilización de los diccionarios para la generación de los nuevos números de teléfono se centra en la obtención de la provincia a partir del número de teléfono. Para ello, se toman los primeros tres dígitos del número antiguo (siempre que este comience por 9), y se comprueba a qué provincia se corresponde (utilizando la entrada "Prefijo" en el diccionario general). En el caso de que no pertenezca a ninguna, se vuelve a realizar la comprobación con los primeros dos dígitos (ya que en provincias como Asturias, Madrid o Valencia el prefijo se compone de dos dígitos en lugar de tres). Si tras esta segunda comprobación tampoco se corresponde con ninguna (y se tiene la certeza de que es un número fijo), se genera una provincia de manera aleatoria y se utiliza el prefijo de esta para generar el nuevo número de teléfono. Cabe destacar que, al igual que ocurre para las direcciones, en el caso de que esta nueva provincia ya haya sido anonimizada, el nuevo prefijo se corresponderá al de la provincia anonimizada. Una vez se haya finalizado este proceso, se dispondrá de un prefijo que será el que se utilice para generar el nuevo número de manera dinámica.
- **Email:** Para el caso de los reemplazos en emails, únicamente se toma un valor aleatorio dentro de la categoría **"Email"** del diccionario general. Sin embargo, debido a que a menudo los emails suelen

contener caracteres que en lenguaje natural se corresponden con caracteres de fin o de corte (. - _, etc), es muy frecuente que el *tokenizador* divida el email por estos caracteres. Esto se traduce en que un email de un único *token*, sea dividido en dos o más *tokens* a través de estos caracteres. Para solucionar este problema y mantener la integridad del documento, lo que se hace es calcular el número de *tokens* de la mención original, y dividir el nuevo email en el mismo número de *tokens*. De este modo, a pesar de que el reemplazo sea de un único *token*, este puede ser transformado para que tenga tantos *tokens* como la mención original.

- **URL:** Para el caso de las url's, el proceso que se sigue es exactamente el mismo que para los emails. La única diferencia entre ambos es que el nuevo reemplazo para esta entidad se toma de la entrada **"URL"** del diccionario general (en lugar de la de Email).

- **Reemplazos generados dinámicamente:** En esta categoría se encuentran los reemplazos que se generan aleatoriamente de forma dinámica. Los reemplazos generados de este modo se corresponden con las siguientes entidades:

  - **DNI:** Para la generación de las nuevas entidades del tipo DNI, el primer paso que se realiza consiste en la obtención del tipo de identificador al que pertenece la entidad[6]. Esta comprobación se lleva a cabo según el diagrama de flujo presentado en la Figura 8.8. Una vez determinado el tipo de identificador al que corresponde la entidad, se genera un número aleatorio comprendido entre 10^{n-1} y $(10^{n-1}) - 1$ (donde n se corresponde con el número de dígitos que tiene el identificador). Finalmente, en función del identificador al que se corresponda, se calculan y se añaden los caracteres de control asociados a dicho identificador[7,8]. Gracias a este cálculo, el identificador generado es perfectamente válido y se ajusta a las propiedades establecidas en el marco legislativo español.

[6]Esto es necesario ya que a pesar de que la entidad sea DNI, en esta categoría se incluyen DNI, CIF y NIF.

[7]https://definanzas.com/cif-o-el-codigo-de-identificacion-fiscal-calcular-y-comprobar-diferencias-con-el-nif/ *(Visitado el 16-07-2021).*

[8]http://www.interior.gob.es/web/servicios-al-ciudadano/dni/calculo-del-digito-de-control-del-nif-nie *(Visitado el 16-07-2021).*

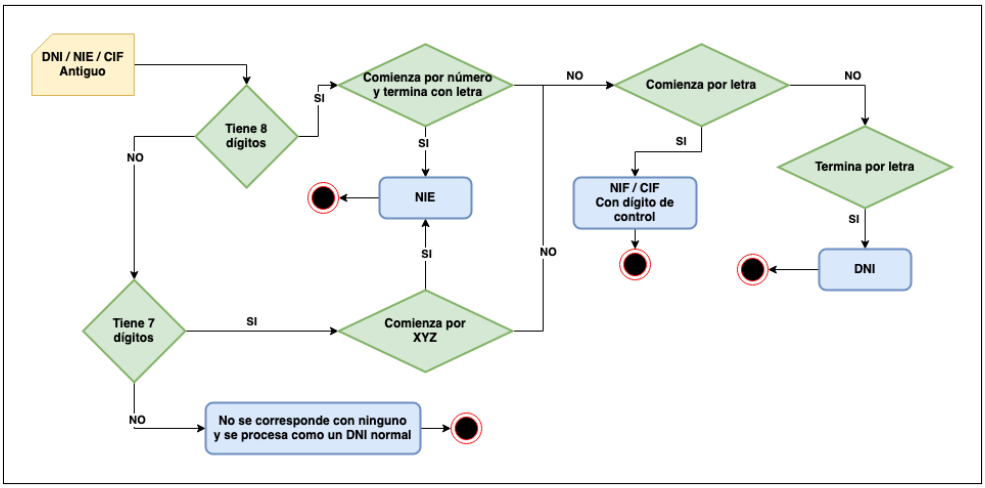


Figura 8.8: Flujo para la comprobación del tipo de entidad del DNI (DNI, NIE, CIF).

- **Teléfono:** Para el caso de los números de teléfono el proceso de generación del reemplazo es similar al ya presentado para los DNI's. Para ello, se genera un número aleatorio de 9 dígitos y, en función del nuevo valor del prefijo obtenido mediante diccionarios, se sustituyen los n primeros dígitos del nuevo teléfono por el nuevo prefijo.

- **Cuenta Bancaria:** Para la generación de las cuentas bancarias, se ha decidido anonimizar únicamente una parte del número de cuenta. El motivo de esto se debe a que, como se puede ver en la Figura 8.9, el número IBAN contiene diversos identificadores comunes en su interior (identificador de la entidad bancaria, identificador de la sucursal, etc). Dado que estos identificadores son públicos y generalmente conocidos (182 BBVA, 49 Santander, 73 Open Bank, etc), el generar un nuevo IBAN a partir de un número aleatorio (como ocurre con los teléfonos o los DNI), hace que sea más evidente que este nuevo número es falso. Por lo tanto, en el caso de las cuentas bancarias se va a mantener toda la mención original, a excepción de los últimos diez dígitos (correspondientes al número de cuenta). A pesar de que el resto del IBAN se mantenga sin modificar, la generación de un número de cuenta aleatorio es suficiente para desvincular a una persona de un determinado número de cuenta. Además de esto, el nuevo número continuará siendo realista, e incluso puede llegar a ser falso (puede ser que el número de cuenta generado no exista en la sucursal de esa entidad bancaria). Sin embargo, al tener el resto de los identificadores sin modificar, es menos evidente el hecho de que ha habido una modificación en esta entidad.

Figura 8.9: Formato número IBAN. Fuente: *kelisto.es (Visitado el 12-07-2021).*

- **CSV, Referencia Catastral, Seguridad Social y Matrícula:** Para el caso de este tipo de entidades, el procedimiento a seguir para generar el reemplazo es el mismo. Para ello, se itera sobre los caracteres de la entidad antigua y se genera un carácter aleatorio en función de su tipo. Por ejemplo, si la entidad antigua es *AB3* la nueva entidad se obtendría calculando una letra aleatoria entre la a y la z (correspondiente al carácter *A*), otra letra aleatoria (correspondiente al carácter *B*), y un número aleatorio entre el 0 y el 9 (correspondiente al carácter *3*).

Un punto importante que destacar es que, a pesar de que la explicación de la generación de los reemplazos en este tipo de entidades se ha tratado desde el punto de vista ideal (sin ningún carácter atípico o separador entre caracteres), este método mantiene el formato de la entidad antigua. Para ello, una vez generado el nuevo reemplazo, se le añaden los caracteres atípicos procedentes de la entidad original. Por ejemplo, en el caso de tener el siguiente DNI: 59**.** 863**.**210**-**Y, la generación del nuevo reemplazo se realiza sin tener en cuenta ni los espacios ni los caracteres extra (**.** y **-**). Suponemos que el nuevo reemplazo para esa entidad que se ha obtenido y validado es: **90543256F**. Una vez obtenido dicho reemplazo, se procesa para hacer que este mantenga el formato del original. Teniendo esto en cuenta, y continuando con el ejemplo descrito, la salida proporcionada por el método es: **90. 543.256-F**.

Capítulo 9

Herramienta Software

9.1. Descripción de los Actores

En la primera sección de este capítulo se llevará a cabo la descripción de los distintos actores que interactúan con la herramienta software desarrollada. Cabe destacar que, al tratarse de una herramienta básica para interactuar con el *pipeline* desarrollado (no requiere de autenticación ni verificación de ningún tipo), únicamente va a tener un actor. El actor que conforma el usuario, así como sus características viene descrito en la Tabla 9.1:

A-01	**Usuario**
Versión	1.0 (17/07/2021).
Descripción	Este tipo de actor utilizará la herramienta software desarrollada para anonimizar referencias personales en documentos en formato PDF. Además podrá exportar los resultados finales a un nuevo fichero PDF.
Comentarios	No necesita de autenticación ni de verificación de acceso.

Tabla 9.1: Descripción del Actor-01: Usuario.

9.1.1. Requisitos de Usuario

En esta sección de presentarán los distintos requisitos de usuario que la herramienta software ha de satisfacer. En la Tabla 9.2 se muestran los distintos requisitos de usuario asociados al actor *Usuario*:

ID	Nombre
RU-01	Un usuario puede cargar en la herramienta documentos en formato PDF.
RU-02	Un usuario puede visualizar las menciones personales detectadas por el modelo NER en el documento cargado.
RU-03	Un usuario puede anonimizar un documento en formato PDF cargado en la herramienta software.
RU-04	Un usuario puede visualizar el resultado de la anonimización de las referencias personales .
RU-05	Un usuario puede exportar el documento anonimizado a un nuevo documento PDF.

Tabla 9.2: Requisitos de Usuario del Sistema.

9.1.2. Casos de Uso

A lo largo de la presente sección se presentan los distintos casos de uso a satisfacer por la herramienta software, el requisito de usuario asociado y la especificación de estos. En la Tabla 9.3 se muestran los distintos casos de uso que se han obtenido a partir de los requisitos de usuario descritos en el Apartado 9.1.1:

ID	Requisito de Usuario Asociado	Nombre
CU-01	RU-01, RU-02	Cargar documento.
CU-02	RU-03, RU-04	Anonimizar documento.
CU-03	RU-05	Exportar documento anonimizado.

Tabla 9.3: Casos de Uso de la Herramienta.

Tras la definición de los casos de uso, así como los requisitos de usuario que satisfacen, se ha generado el Diagrama de Casos de Uso para el único actor que interactúa con la herramienta. De este modo se consigue plasmar de una forma visual, esquemática, simplificada y global la funcionalidad que ofrece la herramienta. En la Figura 9.1 se muestra el Diagrama de Casos de Uso de la herramienta:

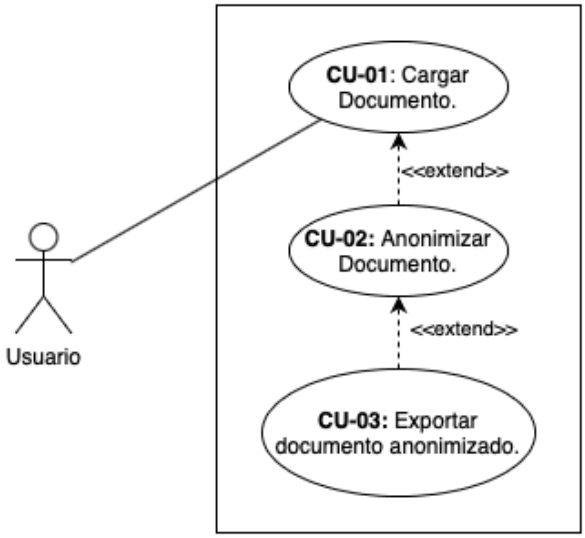


Figura 9.1: Diagrama de Casos de Uso de la herramienta.

9.1.2.1. Especificación de los Casos de Uso

En esta subsección se abordará la especificación de los casos de uso obtenidos a partir de los requisitos de usuario en el apartado anterior. A continuación se muestran las distintas especificaciones de los casos de uso asociados a la herramienta (ver Tablas 9.4 - 9.6):

CU-01	**Cargar Documento**
Versión	1.0 (17/07/2021).
Actor Principal	Usuario.
Descripción	El sistema permite al usuario cargar un fichero para anonimizar y visualizar las predicciones del modelo.
Disparador	El usuario solicita cargar el documento que quiere anonimizar
Precondiciones	El formato del fichero ha de ser PDF.
Postcondiciones	Ninguna.
Flujo Normal	1. El usuario solicita cargar un nuevo documento para anonimizar. 2. El sistema solicita la selección del documento. 3. El usuario selecciona el documento a anonimizar. 4. El sistema carga el documento y aplica el modelo para obtener las predicciones. 5. El sistema muestra el contenido del documento y las predicciones del modelo. 6. El Caso de Uso finaliza satisfactoriamente.
Flujo Alternativo	3a) El usuario cancela la acción de cargar un documento. - El Caso de Uso finaliza.
Excepciones	4. El documento seleccionado no se encuentra en el formato esperado o se encuentra corrupto. - El sistema muestra un mensaje de error. - El Caso de Uso finaliza.

Tabla 9.4: Especificación de Caso de Uso: *CU-01: Cargar Documento.*

CU-02	**Anonimizar Documento**
Versión	1.0 (17/07/2021)
Actor Principal	Usuario
Descripción	El sistema permite al usuario anonimizar un documento cargado en la herramienta.
Disparador	El usuario solicita anonimizar el documento
Precondiciones	El *CU-01: Cargar Documento* ha de ser completado satisfactoriamente.
Postcondiciones	Ninguna
Flujo Normal	1. El usuario solicita anonimizar un documento. 2. El sistema reemplaza las menciones personales detectadas en el documento por otras del mismo tipo manteniendo las propiedades del documento (ver Apartado 8). 3. El sistema muestra el contenido del nuevo documento y las menciones reemplazadas. 4. El Caso de Uso finaliza satisfactoriamente.
Flujo Alternativo	-
Excepciones	2. Ocurre un error durante el proceso de anonimización. - El sistema muestra un mensaje de error. - El Caso de Uso finaliza.

Tabla 9.5: Especificación de Caso de Uso: *CU-02: Anonimizar Documento.*

CU-03	Exportar Documento Anonimizado
Versión	1.0 (17/07/2021).
Actor Principal	Usuario.
Descripción	El sistema permite al usuario exportar un documento que haya sido anonimizado con la herramienta.
Disparador	El usuario solicita exportar un documento anonimizado.
Precondiciones	El *CU-02: Anonimizar Documento* ha de ser completado satisfactoriamente.
Postcondiciones	Ninguna.
Flujo Normal	1. El usuario solicita exportar un documento anonimizado. 2. El sistema genera un archivo PDF con el contenido del documento anonimizado. 3. El Caso de Uso finaliza satisfactoriamente.
Flujo Alternativo	-
Excepciones	2. Ocurre un error durante el proceso de exportación. - El sistema muestra un mensaje de error. - El Caso de Uso finaliza.

Tabla 9.6: Especificación de Caso de Uso: *CU-03: Exportar Documento Anonimizado.*

9.2. Requisitos Funcionales

En la siguiente sección se presentarán los distintos requisitos funcionales asociados a los casos de uso definidos anteriormente. Este tipo de requisitos son útiles ya que muestran el comportamiento que ha de tener el sistema a la hora de llevar a cabo diferentes acciones. A continuación se muestran los casos de uso del sistema, así como los sus requisitos funcionales asociados (ver Tablas 9.7 - 9.12).

ID	Nombre
RF-01	El sistema habilitará la opción de cargar un documento en la herramienta.
RF-02	El sistema solicitará al usuario la selección de un documento para cargar en la herramienta.
RF-03	El sistema mostrará un mensaje de error en el caso de que no se haya podido realizar la carga correctamente.
RF-04	El sistema mostrará un mensaje de confirmación cuando se termine de cargar satisfactoriamente el documento.

Tabla 9.7: Requisitos Funcionales asociados al Caso de Uso: *CU-01: Cargar Documento.*

ID	Nombre
RF-05	El sistema mostrará al usuario las predicciones realizadas por el modelo NER.
RF-06	El sistema permitirá al usuario moverse a través de la vista de las predicciones.

Tabla 9.8: Requisitos Funcionales asociados al Caso de Uso: *CU-02: Visualizar las predicciones del modelo NER.*

ID	Nombre
RF-07	El sistema habilitará la opción de anonimizar documento.
RF-08	El sistema comprobará que existe un documento previamente cargado en la herramienta.
RF-09	El sistema mostrará un mensaje de error en el caso de que no se haya cargado ningún documento.
RF-10	El sistema anonimizará el documento utilizando técnicas basadas en grafos y en diccionarios.
RF-11	El sistema mostrará un error en el caso de que se produzca un fallo durante el proceso de anonimización.
RF-12	El sistema mostrará un mensaje de confirmación cuando se termine de anonimizar correctamente el documento.

Tabla 9.9: Requisitos Funcionales asociados al Caso de Uso: *CU-03: Anonimizar Documento.*

ID	Nombre
RF-13	El sistema mostrará al usuario el texto del documento con las menciones detectadas anonimizadas.
RF-14	El sistema permitirá al usuario moverse a través de la vista de las anonimizaciones.

Tabla 9.10: Requisitos Funcionales asociados al Caso de Uso: *CU-04: Visualizar el Resultado de la Anonimización.*

ID	Nombre
RF-15	El sistema habilitará la opción de exportar un documento anonimizado.
RF-16	El sistema comprobará que existe un documento previamente anonimizado en la herramienta.
RF-17	El sistema mostrará un mensaje de error en el caso de que no se haya anonimizado previamente un documento.
RF-18	El sistema generará un nuevo documento PDF con el texto del documento original anonimizado.
RF-19	El sistema mostrará un mensaje de error en el caso de que falle la generación del nuevo documento.
RF-20	El sistema mostrará un mensaje de confirmación cuando se termine de exportar correctamente el documento.

Tabla 9.11: Requisitos Funcionales asociados al Caso de Uso: *CU-05: Exportar Documento Anonimizado.*

ID	Nombre
RF-21	El sistema generará un fichero tipo *.log* con las características del documento, así como el tiempo de ejecución en cada etapa.

Tabla 9.12: Requisitos Funcionales Globales.

9.3. Diseño

9.3.1. Arquitectura Lógica

En el siguiente apartado se presenta el diseño lógico que tendrá la herramienta desarrollada. La propuesta presentada se encuentra implementada en local, por lo que su despliegue se realiza dentro del propio sistema donde esta se ejecuta. Dentro del servidor se encuentran almacenado el código fuente de la aplicación (desarrollado en HTML + CSS y Python (utilizando Flask)), el modelo NER (y sus componentes), y la base de conocimiento para generar los reemplazos, así como sus representaciones serializadas. El funcionamiento de la herramienta se realiza mediante conexión segura HTTPS, en la que el usuario selecciona un documento para anonimizar y se lo proporciona a la

herramienta[1]. Tras su anonimización, y posterior solicitud de exportación, el sistema genera un nuevo fichero PDF dentro de la estructura de ficheros de la propia herramienta y finaliza el proceso. En la Figura 9.2 se muestra el diseño lógico de la herramienta:

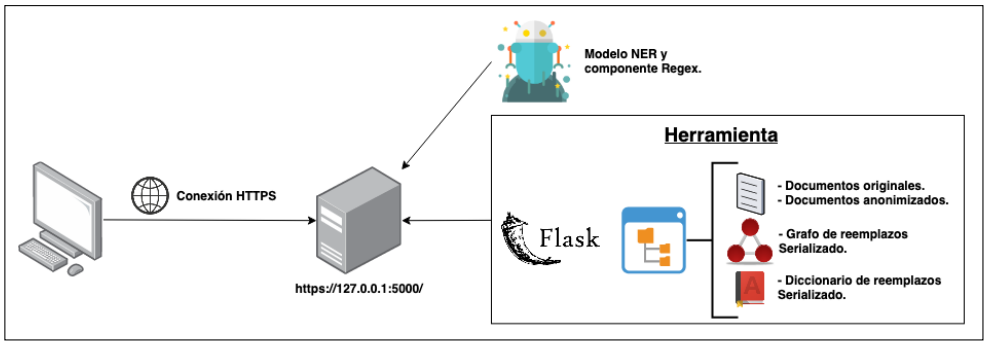


Figura 9.2: Arquitectura Lógica de la Herramienta.

9.3.2. Diseño de la Aplicación

En esta subsección se presentan las distintas vistas e interfaces que constituyen la herramienta desarrollada, así como los distintos eventos que se pueden realizar desde cada una de ellas. Un punto importante que destacar es que, a pesar de que en este apartado se muestren distintas vistas, realmente sólo hay una. La diferencia que existe entre una y otra consiste únicamente en que se produce una actualización en partes de la vista. Sin embargo, para facilitar la descripción y el entendimiento de estas, cada una de estas actualizaciones de la interfaz se ha considerado como una ventana aparte. A continuación se muestran las distintas vistas que conforman el diseño de la herramienta:

[1]Este fichero ha de encontrarse en un directorio específico dentro de la estructura de directorios de la herramienta.

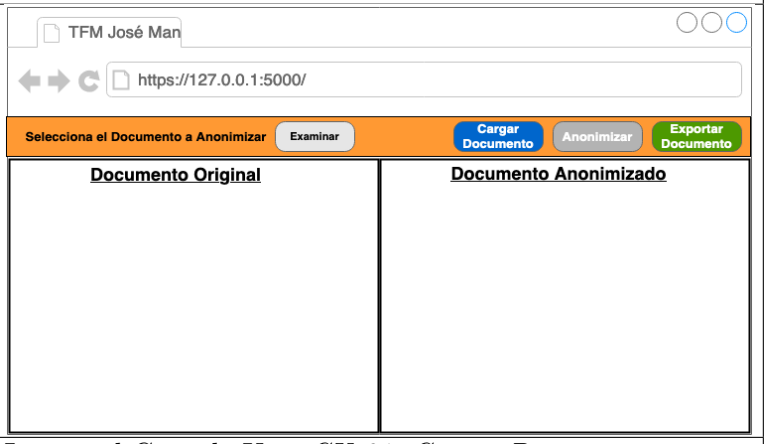
DI-01.1	Página Principal
Descripción	Página encargada de actuar como punto de entrada a la herramienta.
Activación	El usuario accede a la herramienta.
Boceto	
Eventos	Lanzar el Caso de Uso: *CU-01: Cargar Documento*.

Tabla 9.13: Diseño Interfaz *DI-01.1: Página Principal*.

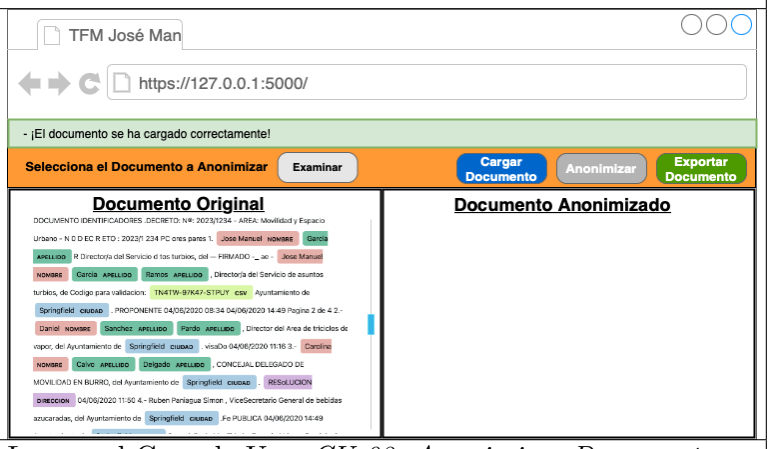
DI-01.2	Página Cargar Documento
Descripción	Página encargada mostrar al usuario las predicciones realizadas por el modelo NER sobre el documento cargado.
Activación	El usuario activa el botón "Cargar Documento" y el Caso de Uso: *CU-01: Cargar Documento* finaliza satisfactoriamente.
Boceto	
Eventos	Lanzar el Caso de Uso: *CU-02: Anonimizar Documento*.

Tabla 9.14: Diseño Interfaz *DI-01.2: Página Cargar Documento*.

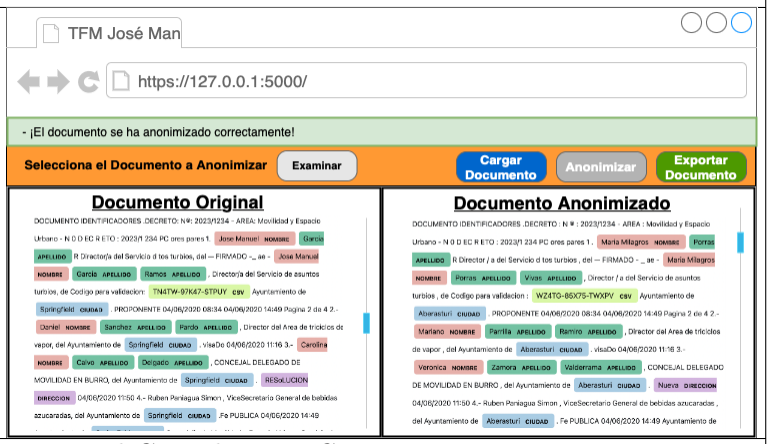
DI-01.3	Página Anonimizar Documento
Descripción	Página encargada mostrar al usuario el resultado de la anonimización de las entidades detectadas por el modelo NER en el documento.
Activación	El usuario activa el botón "Anonimizar" y el Caso de Uso: *CU-02: Anonimizar Documento* finaliza satisfactoriamente.
Boceto	
Eventos	Lanzar el Caso de Uso: *CU-03: Exportar Documento.*

Tabla 9.15: Diseño Interfaz *DI-01.3: Página Anonimizar Documento.*

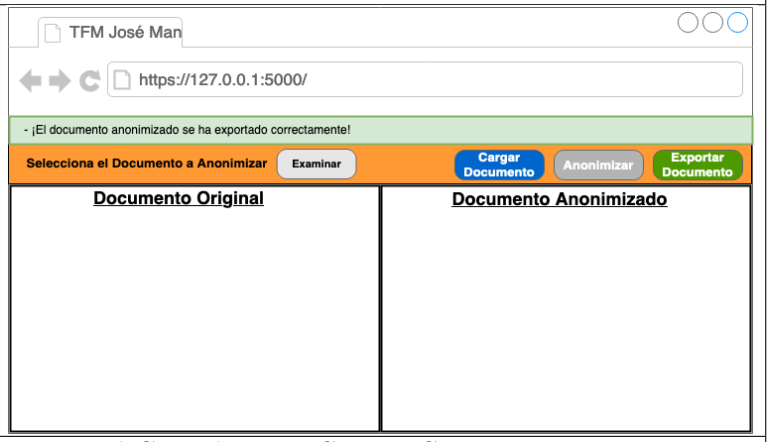
DI-01.4	Página Exportar Documento
Descripción	Página encargada mostrar al usuario el estado del proceso de exportación.
Activación	El usuario activa el botón "Exportar Documento" y el Caso de Uso: *CU-02: Anonimizar Documento* finaliza satisfactoriamente.
Boceto	
Eventos	Lanzar el Caso de Uso: *CU-01: Cargar Documento.*

Tabla 9.16: Diseño Interfaz *DI-01.4: Página Exportar Documento.*

9.4. Implementación de la herramienta

En esta sección se definen las principales características y consideraciones que se han tenido en cuenta a la hora de llevar a cabo el desarrollo de la herramienta. Además de esto, se abordará la estructura interna de la herramienta, así como la implementación de las partes más relevantes de la misma.

Desde el punto de vista de las características de la herramienta, la principal dependencia viene dada por la funcionalidad que esta proporciona. Dado que los componentes principales de la herramienta han sido desarrollados en Python, la parte gráfica (también conocida como *Front-End*), debe permitir la integración entre estos componentes. Es por esto por lo que se ha tomado la decisión de utilizar **Flask**[2] como framework principal para integrar la funcionalidad de la herramienta (detección de entidades y generación de reemplazos), con la parte visual de la misma. Flask es un framework escrito en Python que permite crear aplicaciones webs dinámicas, API's. y otros tipos de sistemas web. El principal motivo de utilizar Flask frente a otras alternativas conocidas como Django, se debe a que Flask proporciona las herramientas mínimas para poder crear y mantener una aplicación web. Esto, unido a que la herramienta a desarrollar es relativamente simple en cuanto a su complejidad, hace que no sea necesario emplear frameworks más pesados y completos, al no necesitar gran parte de la funcionalidad extra que estos ofrecen.

Por otro lado, desde el punto de vista de la implementación, se ha dividido la funcionalidad de la herramienta en 6 clases. De este modo se consigue mantener una estructura lógica dentro del código, así como aislar posibles fallos y mejorar su mantenimiento. Las clases en las que se ha dividido la herramienta son las siguientes:

1) **Clase Main:** Es el punto de entrada a la herramienta. Se encarga de crear la instancia del servidor, renderizar la vista correspondiente, e invocar una funcionalidad u otra en función de la acción del usuario.

```
from flask import Flask, render_template, request, flash
from markupsafe import Markup
...

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
```

[2]https://flask.palletsprojects.com/en/2.0.x/ *(Visitado el 14-07-2021).*

```
def main():
    if request.method == "POST":
        if request.form.get("btn-cargar") == "Cargar":
            archivo = request.form.get("input-doc")
            conversorPDF = ConversorPDF()
            conversorPDF.extraerTextoPDF(archivoPDF =
            ↪  archivo)

            ner = NER()
            ner.obtenerPrediccionesNER()
            visualizacion_prediccion =
            ↪  ner.generarVisualizacionDocumentos()

            return render_template("index.html",
            ↪  visualizacion_prediccion =
            ↪  Markup(visualizacion_prediccion) )

        elif request.form.get("btn-anonimizar") ==
        ↪  "Anonimizar":
                anonimizador = Anonimizador()
                visualizacion_nuevo_doc =
                ↪  anonimizador.AnonimizarDocumento()

            return render_template("index.html",
            ↪  visualizacion_prediccion =
            ↪  Markup(visualizacion_prediccion),
            ↪  visualizacion_nuevo_doc =
            ↪  Markup(visualizacion_nuevo_doc) )

        elif request.form.get("btn-exportar") ==
        ↪  "Exportar":
            generadorNuevoPDF = ExportarPDF()
            generadorNuevoPDF.exportarPDF()

        return render_template("index.html")

if __name__ == "__main__":
    app.run(ssl_context = ("keys/cert.pem",
    ↪  "keys/key.pem"))
```

Código 9.5: Fragmento de la Clase *Main*.

2) **Clase ConversorPDF:** Esta clase se ocupa de, a partir del archivo proporcionado por el usuario, validar que se encuentre en el formato adecuado y extraer su texto. Para ello, realiza una transformación del documento a imagen, y aplica a esta un OCR para extraer el texto. Finalmente, el texto obtenido de la imagen es exportado a un fichero de texto plano para su posterior uso.

```
from PIL import Image
import pytesseract
...

class ConversorPDF():

    def extraerTextoPDF(self, archivoPDF):
        paginas = convert_from_path(archivoPDF, 500)

        contador_paginas = 1

        for pagina in paginas:
            # Exportamos la imagen a JPEG
            pagina.save(f"doc_{contador_paginas}.jpg",
            ↪  'JPEG')
            contador_paginas += 1

        for i in range(1, contador_paginas):

            archivo_imagen = f"doc_{i}.jpg"
            fichero = open(f"doc_{i}.txt", "w")

            # Aplicamos el OCR a la imagen cargada de
            ↪  la página del PDF
            texto = pytesseract.image_to_string(
            ↪  Image.open( archivo_imagen ))

            # Se escribe el texto obtenido del OCR en
            ↪  el fichero de texto
            fichero.write(texto)
            fichero.close()
    ...
```

Código 9.6: Fragmento de la Clase *ConvertirPDF*.

3) **Clase NER:** Esta clase se encarga de cargar tanto el modelo NER desarrollado, como el componente correspondiente a las expresiones regulares. Seguidamente, este modelo es aplicado sobre el contenido de un fichero de texto plano, generando una vista en formato HTML con las predicciones del modelo. Esta vista será la que se utilice en la página principal para visualizar las predicciones del modelo.

```
import spacy
from spacy.tokens import Doc, Span
from src.RegexNER import REGEXComponent
from spacy import displacy
...

class NER():
    def __init__(self):
        self.utils = Utils()
        self.configuracion =
        ↪  self.utils.cargarConfiguracion()
        self.ruta_tmp =
        ↪  self.configuracion["META"]["tmp"]
        ...
    def obtenerPrediccionesNER(self):
        self.aplicarNER()
        self.exportarDocumentoYPredicciones()
    ...
```

Código 9.7: Fragmento de la Clase *NER*.

4) **Clase Anonimizador:** Esta clase se encarga de orquestar el proceso de anonimización de la herramienta. Para ello inicializa el módulo de grafos y los diccionarios, y procesa cada una de las menciones detectadas en el documento en el orden establecido (menciones relacionadas → menciones no relacionadas). Una vez ha procesado todo, se encarga de generar el nuevo documento con las menciones anonimizadas.

```
import spacy
from spacy.tokens import Doc, Span
from src.Grafos import Grafos
...

class Anonimizador():
```

```
    def __init__(self):
        self.utils = Utils()
        self.configuracion =
        ↪  self.utils.cargarConfiguracion()
        ...

    def AnonimizarDocumento (self):
        self.MODULO_GRAFOS.GenerarReemplazos()
        self.reemplazos_grafo =
        ↪  self.MODULO_GRAFOS.reemplazos_entidades

        self.obtenerReemplazos()

        self.validarReemplazos(self.lista_reemplazos)

        self.generarNuevoDoc()

        return self.generarVisualizacionDocumentos()
    ...
```

Código 9.8: Fragmento de la Clase *Anonimizacion*.

5) **Clase Grafos:** Esta clase se encarga de inicializar el grafo general de reemplazos, obtener las relaciones existentes entre las entidades del documento y generar los reemplazos para este tipo de entidades.

```
import networkx as nx
import Levenshtein as lev
...

class Grafos():
    def __init__(self, doc):
        self.doc = doc
        self.reemplazos_entidades = {}
        ...

    def GenerarReemplazos(self):
        self.obtenerGrafoDocumento(self.doc)
        self.encontrarReemplazosEntidades()
    ...
```

Código 9.9: Fragmento de la Clase *Grafos*.

6) **Clase ExportarPDF:** Esta clase toma el documento anonimizado que se ha generado previamente, y crea un nuevo documento en formato PDF con el contenido de este.

```
from fpdf import FPDF
from datetime import datetime
...
class PDF(FPDF):
    def header(self):
        # Formato del encabezado del PDF
        ...
class ExportarPDF():
    ...
    def exportarPDF():
        pdf = PDF() # Crea el fichero PDF
        pdf.add_page()

        # Añade una celda con el texto especificado
        pdf.cell(0, 25, txt = "Documento Generado:", ln
        ↪  = 5, align = "C")

        # Añade una celda múltiple (varias líneas) con
        ↪  el texto del documento anonimizado
        pdf.multi_cell(0, 6, doc.text, align = "J")
        # Se obtiene la fecha actual para utilizar como
        ↪  sufijo en el nombre del PDF
        sufijo =
        ↪  datetime.now().strftime("%d_%m_%Y_%H_%M_%S")
        # Se guarda el PDF generado
        pdf.output(f"Documento {sufijo}.pdf")
    ...
```

Código 9.10: Fragmento de la Clase *ExportarPDF*.

El código completo de la herramienta se encuentra añadido en el contenido adjunto (ver Apéndice A).

9.5. Pruebas de Caja Negra

En esta sección se describe la realización de distintas pruebas de caja negra para comprobar que el comportamiento de la herramienta es el adecuado en distintos puntos críticos durante la ejecución. En Tablas 9.17 - 9.20 se muestran las distintas pruebas de caja negra que se han realizado sobre la herramienta.

CN-01	Comprobación del formato del documento a anonimizar.
Propósito	Evitar que la aplicación deje de funcionar debido a que el formato del documento a anonimizar no es el esperado por la herramienta.
Prerrequisitos	El usuario solicita cargar un documento para anonimizarlo.
Datos de Entrada	Ruta del documento.
Resultado Esperado	Mensaje de error indicando que no se ha podido procesar el documento.
Resultado Obtenido	Mensaje de error debido a que el formato del documento no es el adecuado.

Tabla 9.17: Prueba de Caja Negra: *CN-01: Comprobación del formato del documento a anonimizar.*

CN-02	Comprobación del guardado de las estructuras de datos de reemplazo.
Propósito	Comprobar que la aplicación guarda correctamente las estructuras de datos para generar los reemplazos, evitando así que estas sean siempre generadas desde cero.
Prerrequisitos	El usuario solicita anonimizar un documento.
Datos de Entrada	Estructuras de datos (diccionario y grafo).
Resultado Esperado	Creación de un fichero binario con la estructura serializada.
Resultado Obtenido	Creación de los ficheros binarios correspondientes a las estructuras serializadas.

Tabla 9.18: Prueba de Caja Negra: *CN-02: Comprobación del guardado de las estructuras de datos de reemplazo.*

CN-03	Comprobación de la existencia de un documento cargado antes de realizar la anonimización.
Propósito	Comprobar que la aplicación dispone de un documento cargado previamente para poder iniciar su anonimización.
Prerrequisitos	El usuario solicita anonimizar un documento.
Datos de Entrada	Ninguno.
Resultado Esperado	Mensaje de error en el caso de que se haya cargado previamente un documento.
Resultado Obtenido	Mensaje de error al no haber cargado un documento antes de tratar de anonimizarlo.

Tabla 9.19: Prueba de Caja Negra: *CU-03: Comprobación de la existencia de un documento cargado antes de realizar la anonimización.*

CN-04	Las entidades que no pueden ser anonimizadas se mantienen en el documento final.
Propósito	Comprobar que el proceso de anonimización finaliza a pesar de la existencia de entidades para las que no se dispone de un reemplazo válido.
Prerrequisitos	El usuario solicita anonimizar un documento.
Datos de Entrada	Documento a anonimizar con las menciones detectadas.
Resultado Esperado	Las menciones que no han podido ser anonimizadas se mantienen en el documento final.
Resultado Obtenido	El documento final contiene menciones no anonimizadas al no disponer de reemplazos válidos en la base de conocimiento.

Tabla 9.20: Prueba de Caja Negra: *CU-04: Las entidades que no pueden ser anonimizadas se mantienen en el documento final.*

9.6. Evaluación del Rendimiento

Dado que en el *pipeline* implementado para llevar a cabo la anonimización de documentos es un *pipeline* en el que intervienen bastantes etapas (extracción del texto a partir de un PDF, detección de entidades mediante el modelo NER, generar las nuevas entidades, y exportar el documento), se va a realizar una evaluación del tiempo empleado en cada una de estas etapas. Para efectuar esta evaluación, se va a aplicar el *pipeline* desarrollado a distintos documentos de diversas características (distinto número de páginas, número de entidades, cantidad de palabras, etc). Además de observar el desempeño de la herramienta, esta prueba permite comprobar qué tan bien se desenvuelve en situaciones

diversas (con documentos del mismo tipo pero con características distintas). Del mismo modo, esta prueba permite detectar los puntos débiles del *pipeline*, así como obtener los componentes que hacen que el desempeño global empeore. Un punto importante que destacar son las características hardware del equipo en el que se han realizado las pruebas de ejecución, ya que dependiendo de las características de este, los tiempos pueden variar considerablemente. La ejecución se ha realizado en un ordenador con sistema un operativo macOS Big Sur en su versión 11.5.1. Del mismo modo, las especificaciones técnicas del equipo son: Procesador I7-9850H, 64 bits, 16GB de RAM y 500GB SSD.

En la siguiente tabla (ver Tabla 9.21) se muestra la comparativa del tiempo de procesamiento empleado (en segundos) para anonimizar distintos documentos, así como las características de estos:

	Test1	**Test2**	**Test3**	**Test4**	**Test5**	**Test6**	**Test7**
Paginas	1	2	3	5	7	11	15
Palabras	685	1.246	1.739	2.998	3.974	6.597	8.287
Entidades	25	50	78	159	237	636	758
Leer PDF	0,05	0,86	0,99	2,8	3,22	5,78	7,91
PDF a Imagen	0,03	0,65	1,004	1,65	2,20	3,64	4,79
Imagen a Texto	9,56	17,88	25,93	46,32	61,95	98,56	126,17
NER	0,16	0,19	0,18	0,25	0,34	0,51	0,58
Reemplazos (Grafo)	0,094	0,108	0,109	0,11	0,113	0,117	0,19
Reemplazos (Diccionario)	0	0	0	0,001	0,003	0,008	0,014
Crear PDF	0,035	0,04	0,037	0,04	0,048	0,053	0,055
Exportar PDF	0,307	0,347	0,392	0,496	0,512	0,514	0,586
TOTAL	10,24	20,08	28,64	51,67	68,36	109,19	140,29
TOTAL (PDF)	9,64	19,397	27,93	50,78	67,36	107,98	138,87
TOTAL Reemplazos	0,25	0,29	0,28	0,35	0,44	0,64	0,78

Tabla 9.21: Tiempos de ejecución de los componentes del *pipeline* desarrollado (en segundos) para distintos documentos.

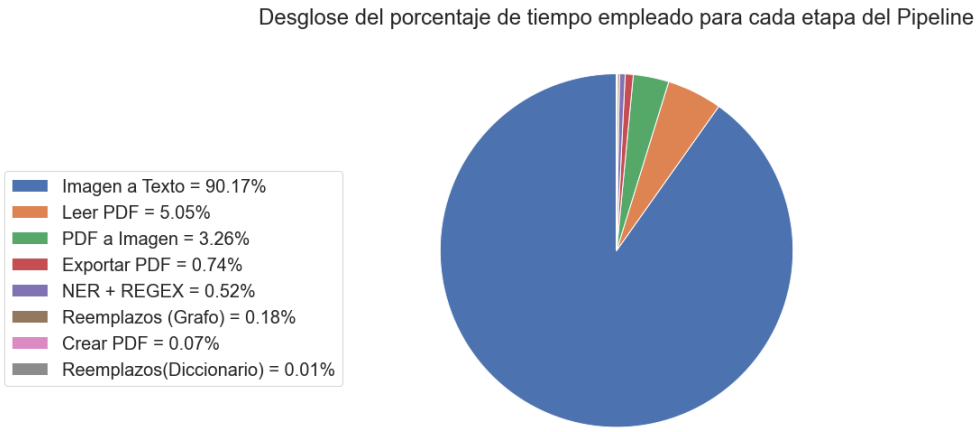


Figura 9.3: Desglose del tiempo de ejecución entre las distintas etapas del *pipeline*.

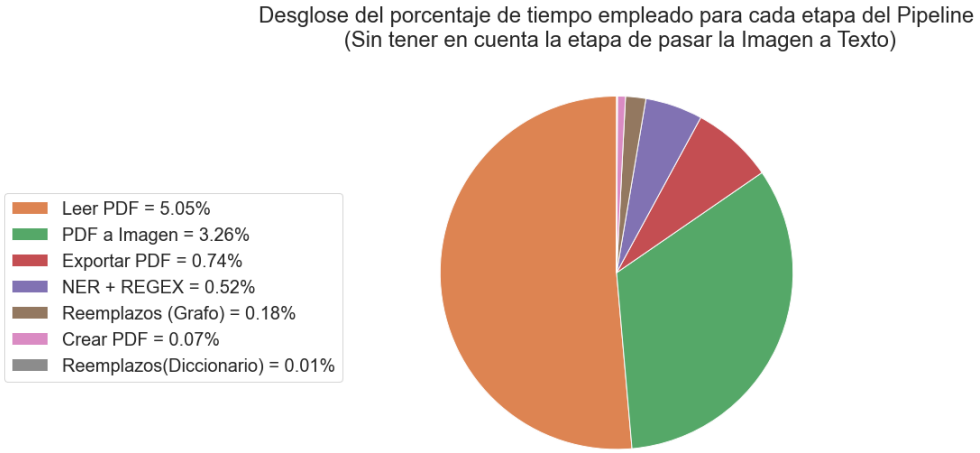


Figura 9.4: Desglose del tiempo de ejecución entre las distintas etapas del *pipeline* (sin tener en cuenta el proceso de OCR).

Conclusiones

Tomando como referencia los resultados presentados en la Tabla 9.21, así como en la Figura 9.5, el tiempo de ejecución requerido para anonimizar los documentos aumenta rápidamente a medida que aumenta el número de páginas del documento. Sin embargo, si se profundiza más en los tiempos de las distintas etapas del proceso (ver Figuras 9.3 y 9.4), este aumento del tiempo viene dado por etapas no desarrolladas en el presente proyecto. Dichas etapas

están relacionadas con la lectura de los archivos PDF, su conversión a imagen, y la extracción del texto en dichas imágenes.

En términos medios, la consecución de las etapas de la herramienta no desarrolladas en este trabajo se corresponde con el 97,61 % del tiempo total utilizado (frente al 2,39 % de las etapas implementadas). Teniendo esto en cuenta, y enfocando el análisis únicamente a las etapas desarrolladas, la comparativa se va a llevar a cabo en base al número de entidades a anonimizar en lugar de al número de páginas del documento. Esto se debe a que, una vez el PDF haya sido procesado, la generación de los reemplazos se aplica únicamente a las entidades, por lo que si un documento tiene 15 páginas y solamente se dispone de una entidad, el proceso de anonimización se realiza muy rápido, y los resultados obtenidos haciendo el análisis de ese modo estarían sesgados.

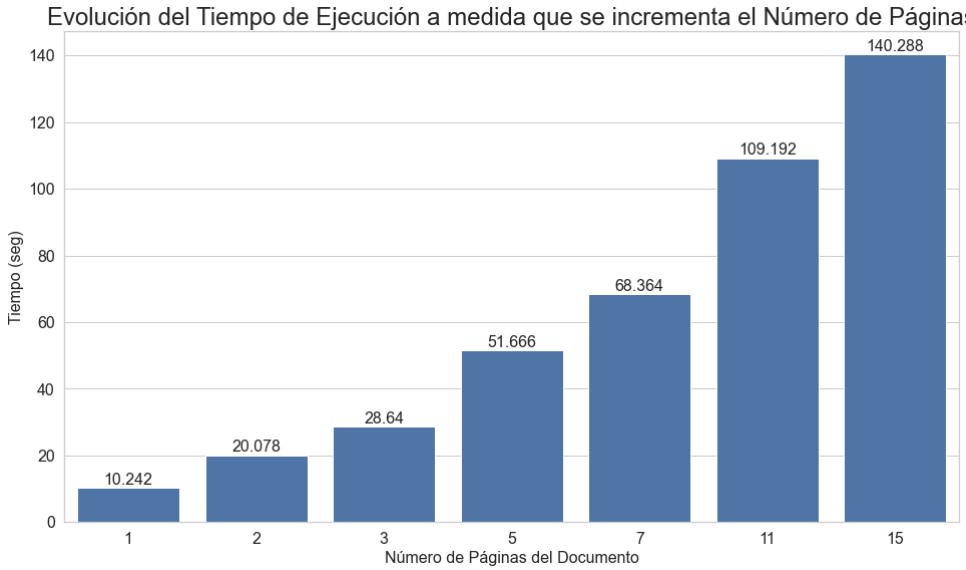


Figura 9.5: Incremento del tiempo de ejecución del *pipeline* en función del número de páginas del documento.

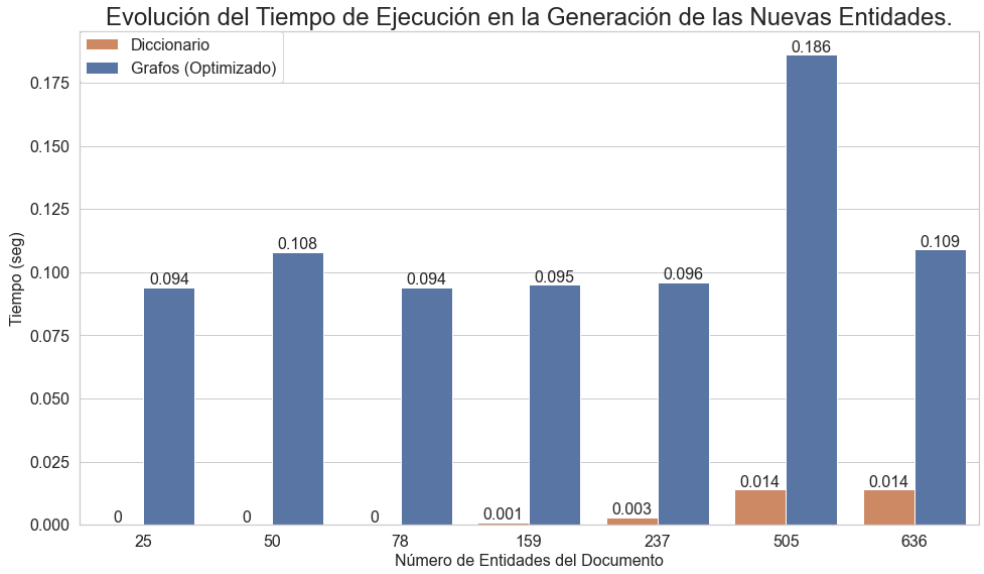


Figura 9.6: Incremento del tiempo de ejecución en la generación de los reemplazos en función del número de entidades del documento.

Como se puede ver en la Figura 9.6, si se atiende únicamente a la relación entre el tiempo, y la aplicación de las fases de generación de reemplazos (tanto por grafos como por diccionarios), los tiempos son similares independientemente del número de entidades. En especial, para el caso de los reemplazos mediante grafos, puede deberse a diversos motivos: 1) A pesar de que las entidades en el documento hayan aumentado, el número de entidades relacionadas (cuyos reemplazos se generan mediante grafos), no lo han hecho; 2) Debido a factores externos (como la dimensión de la base del conocimiento), se ha logrado relacionar un menor número de entidades entre sí, por lo que en términos prácticos se ha materializado en la generación de un grafo con un gran número de componentes aisladas; y 3) Debido a los mecanismos generados para reducir el tiempo de búsqueda de subgrafos isomorfos (ver Apartado 8.1.2). Por otra parte, desde el punto de vista de la obtención de reemplazos mediante diccionarios, los tiempos obtenidos son muy positivos ya que, a pesar del aumento en el número de entidades, el tiempo empleado para llevar a cabo esta etapa se mantiene en el orden de milisegundos. Esto se debe a la gran eficiencia con la que se generan este tipo de reemplazos.

Finalmente, desde el punto de vista del procesamiento del texto, y la detección de entidades por parte del modelo NER y las expresiones regulares (ver Figura 9.7), se puede ver como los tiempos de detección son muy similares a pesar del aumento en el número de palabras de este. Esto, unido a lo ya mencionado anteriormente sobre la generación de los reemplazos, se traduce en que, en términos de escalabilidad, el principal cuello de botella que presenta la herramienta se encuentra asociado al manejo de los PDF's (lectura, conversión

a imagen, y aplicación del OCR). Del mismo modo, teniendo únicamente en cuenta los resultados de las etapas desarrolladas en el proyecto, se puede concluir en que este ofrece un buen rendimiento en términos generales, manteniendo o aumentando ligeramente el tiempo de ejecución a medida que el número de entidades va aumentando.

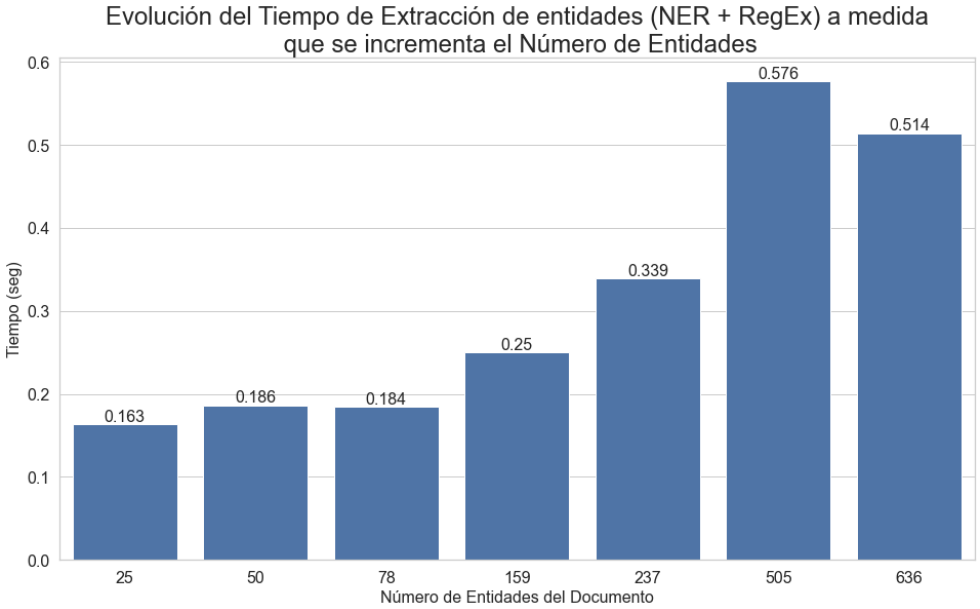


Figura 9.7: Incremento del tiempo de ejecución en la detección de entidades (NER + RegEx) en función del número de entidades del documento.

Parte III

Conclusiones Finales

Capítulo 10

Conclusiones y Trabajo Futuro

10.1. Conclusiones

A lo largo de esta memoria se ha llevado a cabo la documentación técnica del proyecto desarrollado como Trabajo Fin de Máster. Este trabajo ha consistido en la implementación de un *pipeline* de anonimización que permitiese la detección y anonimización de menciones personales en documentos administrativos. Para llevar esto a cabo, se han fijado una serie de objetivos al comienzo de la memoria, cuya finalización nos permitiese alcanzar esta meta. Por esta razón, se va a llevar a cabo un repaso sobre dichos objetivos para evaluar su grado de cumplimiento.

En primer lugar se realizó la obtención del corpus de documentos necesarios para entrenar, validar y probar el desempeño del modelo. Este corpus se encuentra conformado por documentos administrativos de distinta índole (permisos del ayuntamiento, multas, incidencias, catastros, etc). Para generar el conjunto de datos con el que crear el modelo, se realizó un etiquetado manual de las menciones que aparecían en los textos, obteniendo así un conjunto lo suficientemente representativo como para generar un buen modelo. Tras este proceso de etiquetado, a la hora de realizar la división entre los distintos conjuntos, se observó que existía un sesgo en cuanto a las menciones, por lo que la división de los documentos etiquetados entre los distintos conjuntos de datos se realizó en base a menciones (en lugar de a documentos). Gracias a esto se consiguió dividir de una forma correcta los datos y evitar la creación de un modelo sesgado.

Tras la división de los documentos entre los distintos subconjuntos, se llevó a cabo el entrenamiento del modelo, probando su desempeño con distintas configuraciones en sus hiperpárametros. Los resultados obtenidos fueron muy positivos, logrando obtener un modelo capaz de detectar correctamente la mayoría de las menciones de los documentos (en torno a un 94 % de detección).

Sin embargo, la obtención de unos resultados bajos en algunos tipos de menciones (cuentas bancarias, referencias catastrales, matrículas, etc), unido a la estructura-formato de estas, motivó la creación de un componente de detección basado en expresiones regulares. Este componente es el encargado de detectar este tipo de menciones, las cuales el modelo no era capaz de localizar con solvencia. En cuanto a los resultados obtenidos, estos mejoraron ligeramente en términos globales (en torno a un 0,3 %), y en gran medida en términos de entidad (en torno a un 57,79 % de media).

Una vez creado el modelo, se continuó con la implementación de mecanismos de anonimización. Para ello se optó por un enfoque híbrido basado en grafos y diccionarios. El enfoque basado en grafos es el encargado de generar un grafo de relaciones entre las entidades del texto, y obtener un subgrafo con la misma estructura dentro del grafo general (grafo que contiene todos los reemplazos disponibles para sustituir). No obstante, la obtención de un subgrafo con una determinada estructura dentro de otro grafo no era eficiente para estructuras complejas (ya que esta búsqueda es uno de los problemas NP-completos dentro de la Teoría de Grafos). Para solucionar esto, se implementaron distintas soluciones a nivel de código, consiguiendo mejorar los tiempos de búsqueda, a costa de una reducción de las relaciones entre entidades. Por otro lado, la generación de reemplazos de las entidades que no guardan relación entre sí fue abordada mediante un mecanismo de diccionarios (obteniendo reemplazos en tiempos de ejecución constantes). Finalmente, todos estos componentes se combinaron en un *pipeline*, y se les añadió una vista por encima, de tal forma que se pudiese interactuar con ella desde una interfaz visual, dando lugar a la herramienta final desarrollada.

Como resumen de estas conclusiones, destacar que se han cumplido de forma satisfactoria los objetivos fijados al inicio del proyecto, logrando implementar un *pipeline* de anonimización genérico, junto con un modelo, y una herramienta que permite la interacción del usuario con él. Si bien es cierto que el *pipeline* tiene sus limitaciones (como el establecimiento de las conexiones entre los nodos del grafo del documento, la disposición de un espacio limitado de reemplazos, acotación al lenguaje y geografía española, detección de entidades para un tipo concreto de documentos, etc), la propuesta es lo suficientemente genérica como para solventar estos problemas y ser utilizada en cualquier caso de uso con unas características similares. Por lo tanto, en el caso de ser necesaria la detección de otras menciones en otro tipo de documentos, es suficiente con reemplazar el modelo por otro entrenado para el nuevo tipo de documentos, y adaptar las expresiones regulares (en el caso de que fuese necesario). Por otro lado, en el caso de necesitar de reemplazos más específicos o variados, únicamente habría que actualizar la base de conocimiento, generándose automáticamente nuevos grafos y diccionarios actualizados con los nuevos datos. Es por esto por lo que la propuesta de *pipeline* presentada en este Trabajo Fin de Máster se considera

lo suficientemente genérica para abordar otro tipo de casos de uso similares al tratado en este proyecto.

Por otra parte, en cuanto a las métricas se refiere, cabe destacar los resultados obtenidos por el modelo, en combinación con las expresiones regulares, consiguiendo detectar la gran mayoría de las menciones que aparecen en los documentos (94,20 % de detección con el modelo, y 94,50 % de detección con la combinación del modelo y las expresiones regulares). Otro resultado a destacar viene dado por la escalabilidad del mecanismo desarrollado para obtener los reemplazos, cuyos tiempos de ejecución se mantienen constantes, o aumentan ligeramente a medida que se incrementa el número de entidades a anonimizar, lo cuál consolida las soluciones y técnicas presentadas para realizar esta tarea (0,19 segundos de media para documentos entre 25 y 630 menciones). No obstante, en relación a la gestión de los documentos PDF (procesamiento y generación), los resultados no son tan positivos, y la etapa de procesamiento del PDF es la que ocupa el mayor tiempo de la ejecución de la herramienta, lo cual genera un cuello de botella a considerar desde el punto de vista de la utilización de la herramienta con documentos de un mayor tamaño. Sin embargo, esta gestión de los PDF no ha sido íntegramente desarrollada en este proyecto, por lo que en un futuro podría ser sustituida por otra más eficiente sin afectar al desempeño de la herramienta (debido a la estructuración de la funcionalidad en componentes aislados). Esto, unido a la obtención de subgrafos de forma eficiente, abre una línea de investigación futura con la que mejorar las bases que se han consolidado en este proyecto.

10.2. Trabajo Futuro

A pesar de que la entrega de este trabajo incluye tanto un modelo capaz de abordar la problemática planteada, como una herramienta software totalmente funcional para su uso, existen distintas líneas de trabajo futuro que por motivos de profundidad y alcance no se han podido llevar a cabo en este trabajo. Estas líneas de trabajo futuro son las siguientes:

1) **Optimización o implementación de un mecanismo de búsquedas en grafos:** Uno de los problemas que se han presentado a la hora de desarrollar el mecanismo de reemplazos mediante grafos, ha sido el excesivo tiempo de ejecución que se requería para situaciones con grafos complejos. A pesar de que las soluciones propuestas han sido efectivas, no dejan de ser medidas que alivian el problema pero no lo resuelven. La creación de mecanismos de búsqueda eficiente de subgrafos manteniendo la estructura del grafo original, así como aprovechando las propiedades internas de los nodos, es una de las líneas de trabajo futuro presentadas para mejorar el desempeño de la generación de los reemplazos.

2) **Implementación de diccionarios dinámicos de reemplazo:** En la propuesta actual los reemplazos generados mediante diccionarios se obtienen a partir de un conjunto de datos estático. Una posible mejora orientada al enriquecimiento de los reemplazos, así como a la adecuación de estos al contexto en el que se aplican, es la utilización de diccionarios dinámicos para obtener los reemplazos. Estos diccionarios partirían de un conjunto de datos reducido, y se irían aumentando a medida que se van anonimizando más menciones.

3) **Aplicación del modelo en otro tipo de documentos administrativos:** Dado que la estructura y forma de los documentos administrativos son similares, y los resultados obtenidos en este proyecto son muy positivos, una línea interesante a profundizar consiste en el análisis del comportamiento de este modelo para otro tipo de documentos administrativos. Este análisis no se centraría únicamente en otro tipo de documentos, si no que abordaría también el comportamiento para documentos de otras entidades públicas, en este caso, otros ayuntamientos.

4) **Mejora de la gestión y manejo de los PDF:** Al igual que ocurre con la generación de grafos, la gestión y el manejo de los PDF son las únicas partes que no han sido desarrolladas íntegramente en este proyecto, por lo que la optimización de estas se encuentra muy limitada. Una posible línea de trabajo futuro es la generación de un sistema que permita leer y extraer el texto de un PDF de una forma más rápida y eficiente, mejorando así los tiempos y la escalabilidad del sistema.

5) **Desarrollo de nuevos componentes en el *pipeline*:** Debido a que la estructura del *pipeline* desarrollado admite la inclusión de nuevas componentes, una posible línea de trabajo futuro puede venir dada por la creación de nuevos componentes que añadan funcionalidades extra en la herramienta. Algunos ejemplos de nuevas características y/o funcionalidades que se pueden añadir es la introducción de metadatos que permitan volver al estado original del documento, crear un componente que cifre los metadatos para que estos no puedan ser recuperados por alguien no autorizado, etc.

Parte IV

Bibliografía

Bibliografía

[1] Nosheen Abid, Adnan Ul Hasan, and Faisal Shafait. DeepParse: A Trainable Postal Address Parser. *2018 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2018*, 2019.

[2] Aitor Gonzalez Agirre, Montserrat Marimon, Ander Intxaurrondo, Obdulia Rabal, Marta Villegas, and Martin Krallinger. Pharmaconer: Pharmacological substances, compounds and proteins named entity recognition track. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 1–10, 2019.

[3] Luk Arbuckle and Khaled El Emam. *Building an Anonymization Pipeline. Creating Safe Data.* 2020.

[4] Masayuki Asahara and Yuji Matsumoto. Japanese Named Entity Extraction with Redundant Morphological Analysis. 1(June):8–15, 2003.

[5] Fernando Berzal. Patrones en Grafos, (Visitado 06-07-2021). https://elvex.ugr.es/idbis/dm/slides/51%20Graph%20Mining%20-%20Patterns.pdf.

[6] Rosario Catelli, Francesco Gargiulo, Valentina Casola, Giuseppe De Pietro, Hamido Fujita, and Massimo Esposito. Crosslingual named entity recognition for clinical de-identification applied to a covid-19 italian data set. *Applied Soft Computing*, 97:106779, 2020.

[7] Michael Chau, Jennifer J Xu, and Hsinchun Chen. Extracting meaningful entities from police narrative reports. 2002.

[8] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from

scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[10] Riddhiman Dasgupta, Balaji Ganesan, Aswin Kannan, Berthold Reinwald, and Arun Kumar. Fine grained classification of personal data entities. *arXiv preprint arXiv:1811.09368*, 2018.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Sánchez Fernández, Carlos Javier, and Víctor Sandonís Consuegra. Reconocimiento óptico de caracteres (ocr). *Universidad Carlo*, 3(7):2008, 2008.

[13] John Hutchins. From first conception to first demonstration: the nascent years of machine translation, 1947–1954. a chronology. *Machine Translation*, 12(3):195–252, 1997.

[14] María Ibáñez de Opacua Lomoschitz. Named entity recognition y topic modeling: metodología y aplicaciones al procesamiento de texto. B.S. thesis, 2019.

[15] Carolin EM Jakob, Florian Kohlmayer, Thierry Meurers, Jörg Janne Vehreschild, and Fabian Prasser. Design and evaluation of a data anonymization pipeline to promote open science on covid-19. *Scientific data*, 7(1):1–10, 2020.

[16] Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, 2018.

[17] Arzoo Katiyar and Claire Cardie. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, 2018.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[19] John Snow LABS. De-identify PDF Documents, (Visitado 15-06-2021). https://demo.johnsnowlabs.com/ocr/DEID_PDF_HIPAA/.

[20] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[21] Elizabeth D. Liddy. Natural language processing. *Syracuse University*, oct 2001.

[22] Nuno Mamede, Jorge Baptista, and Francisco Dias. Automated anonymization of text documents. In *2016 IEEE congress on evolutionary computation (CEC)*, pages 1287–1294. IEEE, 2016.

[23] Einat Minkov, Richard C Wang, and William Cohen. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 443–450, 2005.

[24] A Miranda-Escalada, E Farré, and M Krallinger. Named entity recognition, concept normalization and clinical coding: Overview of the cantemist track for cancer text mining in spanish, corpus, guidelines, methods and results. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020), CEUR Workshop Proceedings*, 2020.

[25] Eriksson Monteiro, Carlos Costa, and José Luis Oliveira. A machine learning methodology for medical imaging anonymization. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1381–1384. IEEE, 2015.

[26] Isabel Moreno, Ester Boldrini, Paloma Moreda, and M Teresa Romá-Ferri. Drugsemantics: a corpus for named entity recognition in spanish summaries of product characteristics. *Journal of biomedical informatics*, 72:8–22, 2017.

[27] View Next. ¿Qué es el PLN (Procesamiento del Lenguaje Natural)?, (Visitado 26-06-2021). https://www.viewnext.com/que-es-pln-procesamiento-lenguaje-natural/.

[28] School of Computer Science of Birmingham. A brief history of Natural Language Processing, (Visitado 26-06-2021). https://www.cs.bham.ac.uk/~pjh/sem1a5/pt1/pt1_history.html.

[29] Irene Perez-Diez, Raul Perez-Moraga, Adolfo Lopez-Cerdan, Jose-Maria Salinas-Serrano, and Maria de la Iglesia-Vaya. De-identifying spanish medical texts-named entity recognition applied to radiology reports. *Journal of Biomedical Semantics*, 12(1):1–13, 2021.

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[31] C. A. Rahman, W. Badawy, and A. Radmanesh. A real time vehicle's license plate recognition system. *Proceedings - IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2003*, (August):163–166, 2003.

[32] Lisa F Rau. Extracting company names from text. In *Proceedings the Seventh IEEE Conference on Artificial Intelligence Application*, pages 29–30. IEEE Computer Society, 1991.

[33] Inés Roldós. Major Challenges of Natural Language Processing (NLP), (Visitado 26-06-2021). https://monkeylearn.com/blog/natural-language-processing-challenges/.

[34] Satoshi Sekine. NYU: Description of the Japanese NE system used for MET-2. *Proceedings of the 7th Message Understanding Conference (MUC-7)*, 1998.

[35] Satoshi Sekine and Chikashi Nobata. Definition, dictionaries and tagger for Extended Named Entity Hierarchy. *Proceedings of the 4th International Conference on Language Resources and Evaluation, LREC 2004*, pages 1977–1980, 2004.

[36] Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. Extended named entity hierarchy. *Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC 2002*, pages 1818–1824, 2002.

[37] Khaled Shaalan. A Survey of Named Entity Recognition and Classification. *Computational Linguistics*, 40(2):469–510, 2014.

[38] Human-Centered Artificial Intelligence Stanford University. 2021 AI Index Report. pages 1–222, 2021. https://aiindex.stanford.edu/wp-content/uploads/2021/03/2021-AI-Index-Report_Master.pdf.

[39] Alejandro Vaca. Transformers en Procesamiento del Lenguaje Natural, (Visitado 12-06-2021). https://www.iic.uam.es/innovacion/transformers-en-procesamiento-del-lenguaje-natural/.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[41] Stefan Wermter, Ellen Riloff, and Gabriele Scheler. Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, 1040(October 2012):1–16, 1996.

[42] Yonghui Wu, Min Jiang, Jianbo Lei, and Hua Xu. Named entity recognition in chinese clinical text using deep neural network. *Studies in health technology and informatics*, 216:624, 2015.

[43] Lin Yao, Hong Liu, Yi Liu, Xinxin Li, and Muhammad Waqas Anwar. Biomedical named entity recognition based on deep neutral network. *Int. J. Hybrid Inf. Technol*, 8(8):279–288, 2015.

Parte V

Apéndices

Apéndice A

Contenido Adjunto

En esta sección del apéndice se detalla el contenido adjunto a este documento en la entrega del Trabajo Fin de Máster. En este contenido se incluyen tanto los ficheros y directorios necesarios para ejecutar la herramienta presentada, como los *scripts* implementados para llevar a cabo el proceso de entrenamiento y evaluación del modelo NER desarrollado (ver Figura A.1).

El contenido adjunto se encuentra organizado según la siguiente estructura de directorios:

- **/app:** En este directorio se encuentra alojado el código de la herramienta software desarrollada (ver Figura A.2). A su vez, este directorio se encuentra estructurado en los siguientes subdirectorios:
  - **/datos:** En este subdirectorio se encuentran los ficheros de datos necesarios para generar los reemplazos (datos de nombres, apellidos, ciudades, provincias, etc). En él también se almacenan las estructuras de datos serializadas que se emplearán para obtener los reemplazos.
  - **/doc:** En este subdirectorio se encuentran los documentos en formato PDF para cargar en la herramienta.
  - **/keys:** En este subdirectorio se encuentran las claves del certificado SSL para establecer la conexión segura con la herramienta.
  - **/models:** En este subdirectorio se encuentra alojado el modelo NER de detección de entidades.
  - **/output:** Subdirectorio en el que se exportan los nuevos documentos una vez estos han sido anonimizados por la herramienta.
  - **/src:** Subdirectorio en el que se almacena los ficheros python (*.py*) con el código fuente para ejecutar cada uno de los componentes que conforman la herramienta.

- **/static:** En este subdirectorio se encuentran los ficheros de estilo que utiliza Flask para generar la parte visual de las vistas (iconos, hoja de estilo *.css*, etc).
- **/templates:** Subdirectorio en el que se almacenan las vistas (en formato *.html*) de la herramienta.
- **/tmp:** En este subdirectorio se almacenan ficheros temporales que necesita la aplicación para su funcionamiento.
- **config.json:** Fichero donde se encuentra la configuración de los parámetros de la herramienta.
- **Utils.py:** Fichero que contiene funciones comunes utilizadas por las distintas clases de la herramienta.
- **app.py:** Fichero de entrada *(main)* a la herramienta software.

- **/Memoria:** Directorio en el que se encuentra la memoria en formato PDF del Trabajo Fin de Máster.
- **/Notebooks:** En este directorio se encuentran los ficheros de Jupyter que se han utilizado para realizar diversos cálculos y transformaciones durante el desarrollo del modelo, así como los datos de las distintas métricas obtenidas durante su implementación.
- **config.cfg:** Fichero de configuración de Spacy con los parámetros necesarios para crear el modelo NER.
- **requirements.txt:** Fichero de texto en el que se encuentran las dependencias necesarias para ejecutar el proyecto.

```
/
├── /app -> Directorio de la Herramienta (ver Figura A.2)
├── /Memoria
│   └── Memoria TFM Jose Manuel Simon Ramos.pdf
├── /Notebooks
│   ├── Calcular Numero de Menciones en Documentos.ipynb
│   ├── Evaluacion de los Modelos.ipynb
│   ├── Graficos de los Resultados.ipynb
│   └── resultados.xls -> Fichero con las metricas obtenidas
├── /Videos
│   └── Explicacion y Demo de la Herramienta TFM-JMSR.mp4
├── config.cfg
└── requirements.txt
```

Figura A.1: Estructura de directorios de la entrega.

```
/app
├── /datos
│   ├── /csv -> Datos para generar el grafo
│   │   └── dataESP.csv
│   ├── /diccionario ->  Datos del diccionario de reemplazos
│   │   ├── DIC_APELLIDO.txt
│   │   ├── DIC_DIRECCION.txt
│   │   ├── DIC_NOMBRE.txt
│   │   └── ...
│   ├── /fonts -> Fuentes para generar el nuevo PDF
│   └── /serializados -> Estructuras de datos serializadas
├── /doc
├── /keys -> Claves para el certificado SSL
│   ├── cert.pem
│   ├── key.pem
│   └── secret_key
├── /models  -> Modelo NER
│   └── /modeloTFM
├── /output
├── /src  -> Código de la Herramienta
│   ├── Anonimizador.py
│   ├── ConversorPDF.py
│   ├── ExportarPDF.py
│   ├── Grafos.py
│   ├── NER.py
│   └── RegexNER.py
├── /static
│   ├── favicon.ico
│   ├── logo.png
│   └── style.css
├── /templates
│   └── index.html
├── /tmp
├── app.py
├── config.json
└── Utils.py
```

Figura A.2: Estructura de subdirectorios del directorio: */app*.

Apéndice B

Instalación y Versiones

B.1. Versiones de las herramientas utilizadas

En esta sección del apéndice se especifican las librerías utilizadas para el desarrollo del proyecto y de la herramienta, así como la versión que se ha utilizado (ver Tabla B.1). Un punto importante que destacar es que todos los ficheros (tanto *.py*, como *.ipynb*), así como el entrenamiento y validación de los modelos, han sido ejecutados bajo el lenguaje de programación Python en su versión 3.7.10. El correcto funcionamiento de estos en versiones anteriores como la 2.x, o diferentes versiones dentro de la misma rama (3.9 o 3.6) no está 100 % garantizada.

Herramienta / Librería Utilizada	**Versión**	**Herramienta / Librería Utilizada**	**Versión**
Anaconda	4.10.1	Pdf2image	1.15.1
Flask	1.1.2	Pillow	8.2.0
Fpdf	1.7.2	Prettytable	2.1.0
Git	2.30.1	Pytesseract	0.3.7
Jupyter Notebook	6.4.0	PyOpenSSL	1.1.1
Matplotlib	3.3.4	Python	3.7.10
Networkx	2.5.1	Python-Levensthein	0.12.2
Numpy	1.20.2	Spacy	3.0.6
Pandas	1.2.4	Seaborn	0.11.1

Tabla B.1: Versiones de las herramientas y librerías utilizadas.

B.2. Instalación de la herramienta

En esta sección del apéndice se presentan las instrucciones a seguir para llevar a cabo la instalación y ejecución de la herramienta desarrollada. Los pasos que seguir son los siguientes:

1) **Instalación de Python:** Para llevar a cabo la instalación del lenguaje de programación Python, hay que instalar el ejecutable correspondiente a la versión utilizada en el proyecto: https://www.python.org/downloads/release/python-3710/ y seguir las indicaciones proporcionadas en el instalador (durante el proceso de instalación es muy recomendable marcar la opción de **"Añadir Python a la variable PATH"** y la opción **"Instalar pip"**).

2) **Instalación de las librerías necesarias:** Tras la instalación del lenguaje de programación Python, el siguiente paso consiste en instalar las librerías necesarias para ejecutar correctamente la herramienta. Para ello se hará uso del gestor de paquetes *pip* (instalado durante la instalación del lenguaje), y el fichero *requirements.txt* proporcionado en la entrega. Para instalar todas las dependencias en la versión correspondiente hay que abrir una terminal en el mismo directorio donde se encuentre el fichero *.txt* de los requisitos y ejecutar el siguiente comando:

```
pip install -r ./requirements.txt
```

   Una vez finalice la ejecución ya se habrá completado la instalación de todas las dependencias necesarias de la herramienta.

3) **Ejecución de la herramienta:** Para ejecutar la herramienta hay que abrir una terminal y acceder al directorio */app* adjunto a la entrega (directorio en el que se encuentra el código fuente de la herramienta). Una vez allí se procede al despliegue de esta, ejecutando el siguiente comando:

```
python app.py
```

   Finalmente, tras abrir el navegador y acceder a la ruta: *https://localhost:5000/*, debería aparecer la vista principal de la herramienta (ver Figura B.1).

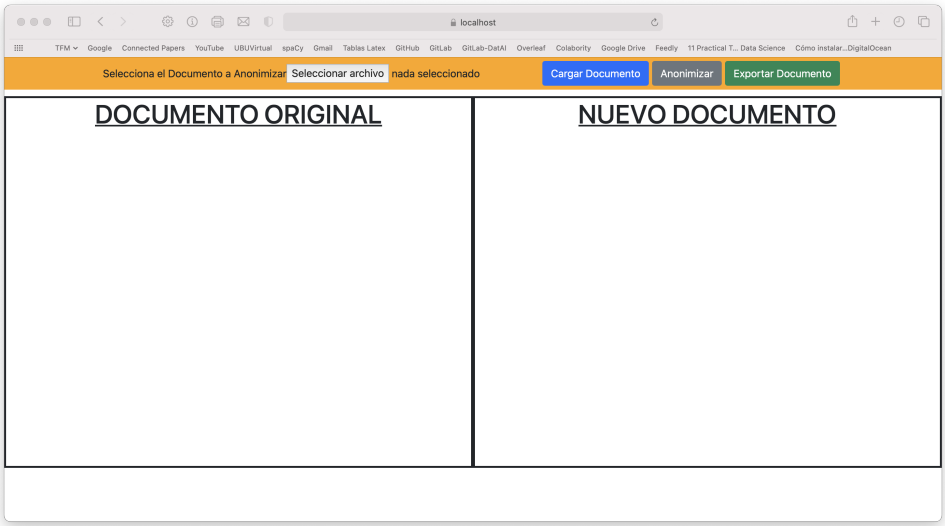


Figura B.1: Vista de Inicio de la Herramienta.

Además, la ventana de la terminal muestra que el despliegue se ha realizado correctamente (ver Figura B.2).

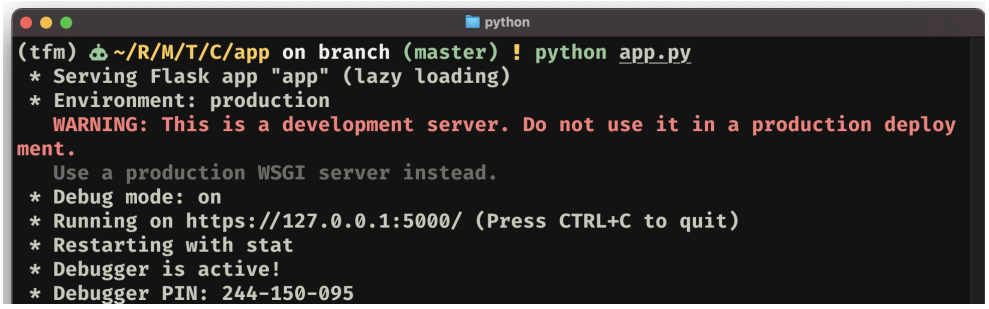


Figura B.2: Estado de la ventana de la terminal tras el despliegue de la herramienta.

Apéndice C

Manual de Usuario

En esta sección del apéndice se presenta un manual de usuario en el que se detallan las funcionalidades de la herramienta, así como la ejecución de las distintas acciones que esta permite realizar. El acceso a la herramienta se lleva a cabo desde un navegador web, una vez esta haya sido desplegada desde una terminal. Una consideración importante a tener en cuenta es que, dado que el objetivo de esta herramienta es que pueda ser utilizada por cualquier usuario, no se ha implementado ningún mecanismo de autenticación para poder acceder a la misma. Esto implica que nada más acceder a la herramienta desde el navegador, se podrá acceder a toda la funcionalidad de esta sin ninguna restricción. Desde el punto de vista de la funcionalidad, esta se encuentra dividida en tres botones, los cuáles lanzan diversos componentes del *pipeline* en función de la acción a llevar a cabo. Estos botones junto con los componentes del *pipeline* que ejecutan son los siguientes:

- **Examinar:** Este apartado de la interfaz permite seleccionar un documento dentro del directorio */docs* para cargarlo en la herramienta y anonimizarlo. Una vez este haya sido seleccionado (ver Figura C.1), ya se puede llevar a cabo el resto de las operaciones del *pipeline* sobre él.

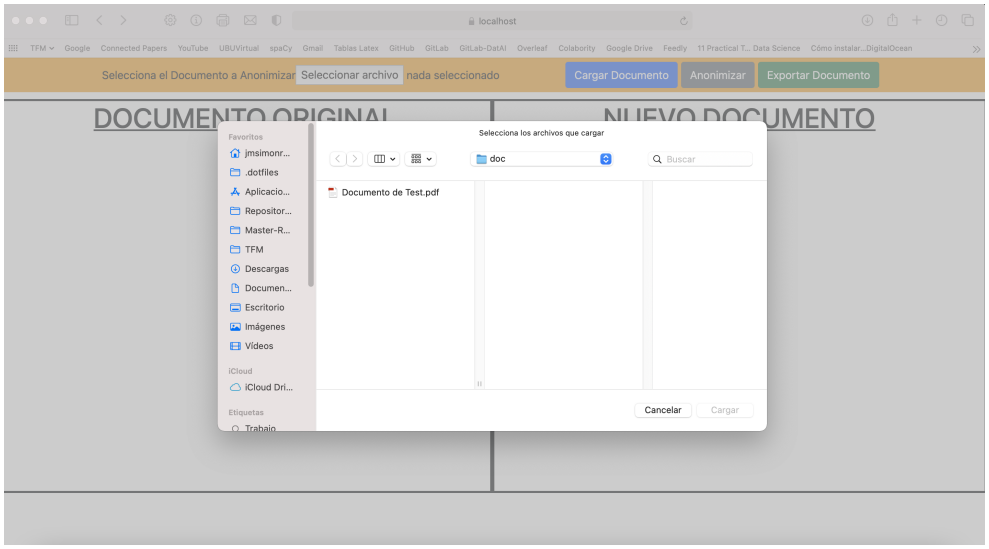


Figura C.1: Herramienta: Examinar Documento para anonimizar.

- **Cargar Documento:** Este botón es el encargado de realizar todo el procesamiento del PDF original y aplicar sobre este el modelo NER desarrollado. Los componentes del *pipeline* que se ejecutan al invocar esta acción son: "Extracción de texto (OCR)" y "Detección de menciones". Una vez se ha procesado el PDF y se ha aplicado el modelo NER, se muestra por pantalla el resultado de las predicciones realizadas (ver Figura C.2), para su posterior anonimización.

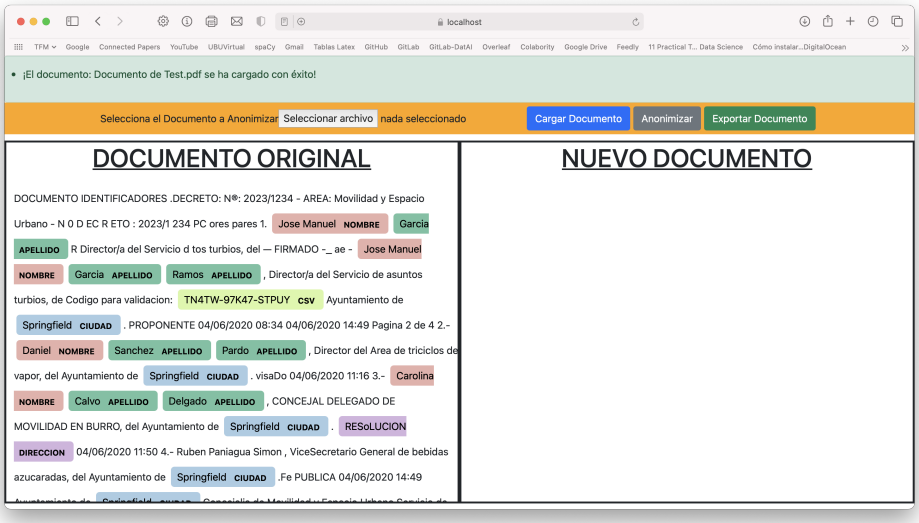


Figura C.2: Herramienta: Cargar Documento y aplicar el modelo NER.

- **Anonimizar:** Este apartado de la interfaz se encarga de tomar el documento con las predicciones del modelo y aplicar sobre este los mecanismos de reemplazo desarrollados para anonimizar el documento. El componente del *pipeline* que se ejecuta al invocar esta acción es: "Anonimización". Tras la generación de los reemplazos se muestra por pantalla el resultado de la anonimización del documento (se muestra el texto original con los reemplazos correspondientes realizados) (ver Figura C.3). Una vez finalizada esta etapa, el resultado puede ser exportado como un nuevo documento PDF.

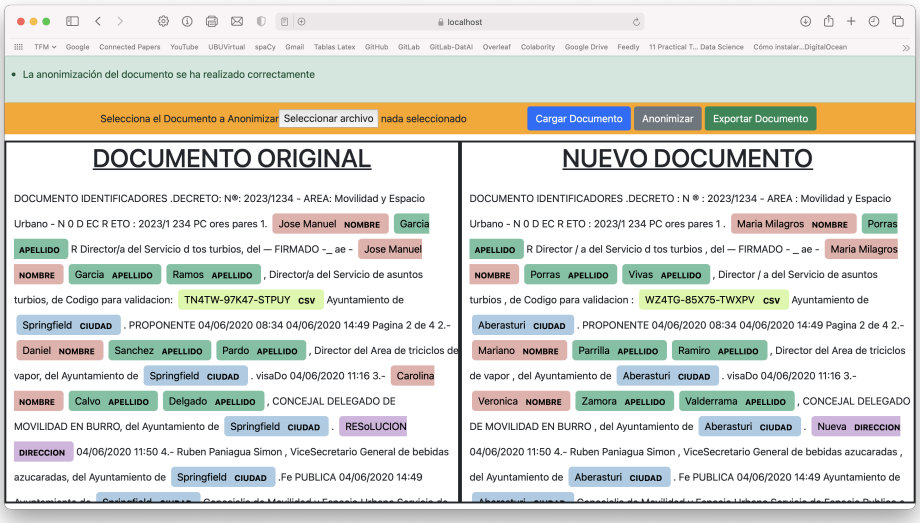


Figura C.3: Herramienta: Anonimizar Documento.

- **Exportar Documento:** Este botón es el encargado de ejecutar la tarea de exportación del documento anonimizado a PDF. Los componentes del *pipeline* que se ejecutan son: "Normalizar Menciones" y "Generador PDF". Una vez finaliza la exportación, se muestra por pantalla un mensaje de informativo del estado de esta y se finaliza el procesamiento de ese documento (ver Figura C.4). Este proceso puede ser repetido para cualquier otro PDF siguiendo los pasos descritos.

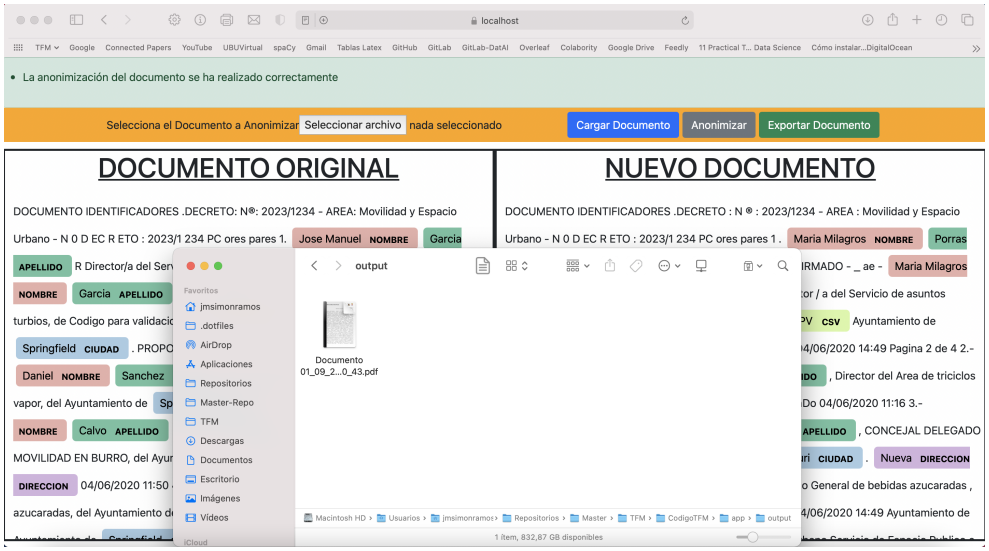
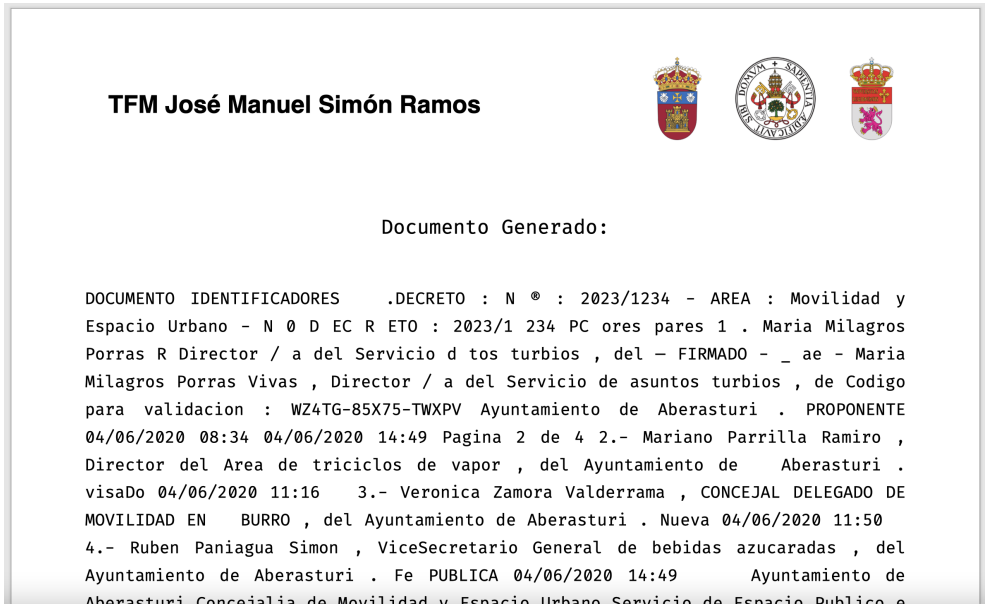


Figura C.4: Herramienta: Exportar Documento anonimizado.

Para cada uno de los PDF anonimizados se genera un nuevo documento PDF similares al presentado en la siguiente figura (ver Figura C.5).



TFM José Manuel Simón Ramos

Documento Generado:

DOCUMENTO IDENTIFICADORES .DECRETO : N ® : 2023/1234 - AREA : Movilidad y Espacio Urbano - N 0 D EC R ETO : 2023/1 234 PC ores pares 1 . Maria Milagros Porras R Director / a del Servicio d tos turbios , del — FIRMADO - _ ae - Maria Milagros Porras Vivas , Director / a del Servicio de asuntos turbios , de Codigo para validacion : WZ4TG-85X75-TWXPV Ayuntamiento de Aberasturi . PROPONENTE 04/06/2020 08:34 04/06/2020 14:49 Pagina 2 de 4 2.- Mariano Parrilla Ramiro , Director del Area de triciclos de vapor , del Ayuntamiento de Aberasturi . visaDo 04/06/2020 11:16 3.- Veronica Zamora Valderrama , CONCEJAL DELEGADO DE MOVILIDAD EN BURRO , del Ayuntamiento de Aberasturi . Nueva 04/06/2020 11:50 4.- Ruben Paniagua Simon , ViceSecretario General de bebidas azucaradas , del Ayuntamiento de Aberasturi . Fe PUBLICA 04/06/2020 14:49 Ayuntamiento de

Figura C.5: Herramienta: Ejemplo del PDF anonimizado generado.

Apéndice D

Configuración de la Herramienta

Para facilitar el uso y mantenimiento de la herramienta se han extraído todos los parámetros de configuración a un único fichero (config.json). De este modo, si en algún momento fuese necesario realizar algún cambio en la estructura de directorios, o en el nombre de algún fichero, bastaría con realizarlo en ese fichero *.json*, en lugar de hacerlo en todas las partes de código que se ven afectadas. El objetivo de esta sección del anexo consiste en presentar la estructura interna del fichero *.json*, así como los distintos parámetros que conforman la configuración de la herramienta, y los valores que tienen por defecto. Desde el punto de vista de la estructura, el fichero se encuentra estructurado en los siguientes cuatro grupos:

- **PDF:** Contiene la configuración relativa a la generación del nuevo PDF con el texto anonimizado (fuente, título, ruta, márgenes, etc). En la siguiente tabla (ver Tabla D.1) se muestran los parámetros correspondientes a esta categoría.

Parámetro	Descripción
titulo	Título utilizado en los encabezados del PDF.
fuente	Ruta de la fuente utilizada (formato *.ttf*).
margen	Márgenes laterales.
logo_path	Ruta del icono con los escudos de las universidades.
output_path	Ruta donde se guardará el PDF de salida.

Tabla D.1: Parámetros de configuración de la categoría *"PDF"*.

- **Anonimización:** Contiene los parámetros utilizados durante el proceso de anonimización (expresiones regulares utilizadas, nombre de las entidades a detectar, ratio de coincidencia por el cuál dos entidades se

interpretan como la misma, etc). En la siguiente tabla (ver Tabla D.2) se muestran los parámetros correspondientes a esta categoría.

Parámetro	**Descripción**
JERARQUIA	Orden jerárquico de las entidades relacionadas.
umbral_coincidencia	Umbral por el cuál dos entidades distintas se interpretan como la misma.
intentos	Número máximo de intentos fallidos para encontrar un reemplazo válido en el diccionario.
intentos_antes_dividir	Número máximo de intentos antes de eliminar aristas del grafo del documento.
intentos_despues_dividir	Número máximo de intentos antes de volver a eliminar aristas del grafo del documento.
ENTIDADES	Entidades a detectar.
EXPRESIONES _REGULARES	Expresiones regulares para las entidades.

Tabla D.2: Parámetros de configuración de la categoría *"Anonimización"*.

- **META:** En esta categoría se encuentran las distintas rutas dentro de la estructura de directorios de la herramienta donde se encuentran ficheros importantes para su funcionamiento. En la siguiente tabla (ver Tabla D.3) se muestran los parámetros correspondientes a esta categoría.

Parámetro	Descripción
path_modelo	Ruta del modelo NER.
path_output_doc_bin	Ruta donde se almacena el documento con sus entidades en formato serializado.
tmp	Ruta del directorio para archivos temporales.
input_docs	Ruta donde se encuentran los documentos originales en PDF.
output_docs	Ruta donde se guardarán los documentos anonimizados en PDF.
path_data_csv	Ruta del fichero .csv de las localizaciones para generar el grafo.
path_data_bin	Ruta del grafo de las localizaciones serializado.
path_diccionario_files	Ruta de los ficheros de texto para generar el diccionario de reemplazos
path_diccionario_bin	Ruta del diccionario de reemplazos serializado.
path_logs	Ruta del fichero de logs de la herramienta.

Tabla D.3: Parámetros de configuración de la categoría *"META"*.

- **Herramienta:** Contiene el esquema de colores utilizado para mostrar las entidades detectadas en los documentos (ver Tabla D.4).

Parámetro	Descripción
COLORES	Paleta de colores para cada una de las entidades.

Tabla D.4: Parámetros de configuración de la categoría *"Herramienta"*.